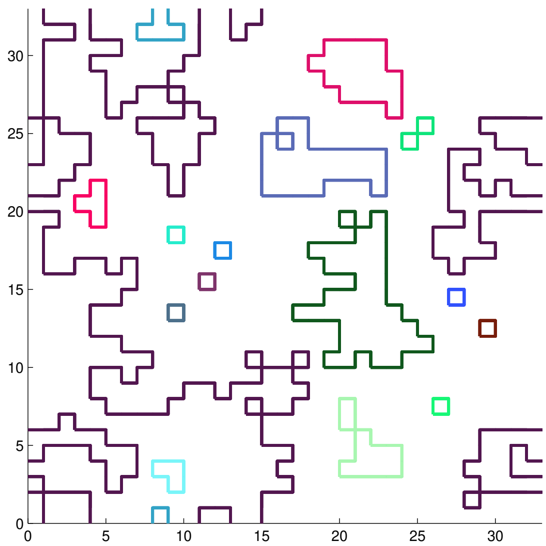
Hans-Marc ERKINGER

A new cluster algorithm for the Ising model

DIPLOMARBEIT

zur Erlangung des akademischen Grades Diplomingenieur
der Studienrichtung Technische Physik

Technische Universität Graz



Betreuer:

Univ.-Ass. Dr. Hans Gerd Evertz und
O. Univ.-Prof. Dr. Wolfgang von der Linden,
Institut für Theoretische Physik

September 2000

Meiner Familie gewidmet.

Ein besonderes Dankeschön für die Unterstützung bei dieser Arbeit an *Univ.-Ass. Dr. Hans Gerd Evertz*, *O. Univ.-Prof. Dr. Wolfgang von der Linden* und *DI Alex Prüll*.

Contents

Chapter 1

Monte Carlo simulations in statistical physics

1.1 The reasons for employing computer simulations

Many-particle systems have so many degrees of freedom that an exact treatment with either classical or quantum mechanics often is not practicable. It has been found that a statistical approach works very well since the number of degrees of freedom is extremely large. This approach is called statistical mechanics and has been well established in physics for a long time. However, only very few models in statistical mechanics are exactly solvable and solutions of exactly treatable models can be very long and complicated. Further, results of approximate solutions are often not easily systematically improvable.
With the invention of the computer and ever more increasing computational power in the second half of the 20^{th} century a new field of research treating many-particle systems emerged: *Computer Simulations in Statistical Physics*. In such simulations a well-defined model system (characterized by its Hamiltonian $\mathcal{H}$) is set up in a computer. By doing this, the thermodynamic behaviour of the system is simulated. Especially, we are interested in the thermal averages of physical quantities and since measuring them means the evaluation of large sums, Monte Carlo methods are the tool of our choice. By employing this method, physical quantities can be computed up to any desired accuracy. Their errors can be estimated and systematically improved by sampling more configurations. In order to obtain this desired accuracy, a sufficient amount of computer power is needed. Further, simulating a system by Monte Carlo does not mean that there is only one way or algorithm for doing it. The aim in writing this thesis was to create and analyze a new algorithm for the Ising model and investigate its performance in terms of computer power when observing the model's critical point.

1.2 Thermal averages

One of the key questions in statistical mechanics is: What is the thermal average of some quantity Q which depends on the system's configuration $\mathbf{x}$ in thermal equilibrium? $\mathbf{x}$ is a vector in phase space and stands symbolically for the considered degrees of freedom of the system, e.g. $\mathbf{x} = (s_1, s_2, \ldots, s_N)$ for the spin values of N atoms in a crystal. Q could be a quantity like the average energy E or the average magnetization M per degree of freedom,

$$E = \frac{\langle \mathcal{H} \rangle_T}{N}, \; M = \frac{\langle \sum_i s_i \rangle_T}{N} \tag{1.1}$$

Here the thermal average of an observable $Q(\mathbf{x})$ is defined as

$$\langle Q(\mathbf{x}) \rangle_T = \frac{1}{Z} \sum_l e^{-\beta\, \mathcal{H}(\mathbf{x}_l)}\, Q(\mathbf{x}_l), \tag{1.2}$$

$$Z = \sum_l e^{-\beta\, \mathcal{H}(\mathbf{x}_l)}, \tag{1.3}$$

where Z is the partition function and $\mathbf{x}_l$ labels the different configurations of the system. The normalized Boltzmann factor

$$p(\mathbf{x}_l) = \frac{1}{Z}\, e^{-\beta\, \mathcal{H}(\mathbf{x}_l)} \tag{1.4}$$

plays the role of a probability density describing the statistical weight with which the configuration $\mathbf{x}_l$ occurs in thermal equilibrium.

How can Eq. (1.2) be evaluated? The first possibility which comes to our mind is the brute-force method of exactly evaluating the sum. Let us do this for the model treated in this thesis, the Ising model. A detailed introduction into the model is given in Ch. 2, but right now it is sufficient to know that it consists of atoms with spins which can only be oriented in 2 different directions. We study the case, where the atoms are arranged in a (hyper)cubic lattice of a desired dimension d. Assume we have N atoms. Then there exist 2^N different configurations of the system and that is exactly the number of terms contributing to our sum in Eq. (1.2). Further, let us assume to simulate a 32×32 lattice in 2 dimensions, which in terms of Monte Carlo simulations is a rather small system. The number of different configurations and terms in our sum is

$$2^{32\times 32} \approx 10^{300}.$$

Let us assume that we compute this sum with one of today's (September 2000) supercomputers which are able to do 10^{12} floating point operations per second. Let us further make the idealized assumption that we only need 1 FLOP for creating the configuration, evaluating the product $e^{-\beta\, \mathcal{H}(\mathbf{x}_l)}\, Q(\mathbf{x}_l)$ and add it to the sum. If our computer was calculating for the age of the universe ($\approx 10^{18}$ sec.), it would be able to sum up

$$10^{18} \times 10^{12} = 10^{30}$$

configurations. We can clearly see that this brute-force method is not the right way to tackle this problem.

1.3 Sampling configurations

We are facing the problem that the number of possible configurations grows exponentially with the system's size and as we have seen, there is no chance to sum up the contributions of every configuration. The *exact diagonalization* of the transfer matrix allows to numerically solve systems, where one dimension is infinite. However, the product of the other dimensions can only be ≈ 30 (simulation of larger lattices is limited by computational power). Another way of tackling the problem is approximating the thermal average by sampling only a very small fraction of configurations in phase space[1].

1.3.1 Simple sampling

How can we sample only a small number of states and still hope to get a reasonable approximation for a thermal average? The simplest way coming to our mind is selecting states **completely at random** and using the subset of phase space $\{\mathbf{x}_1, \mathbf{x}_2, \ldots, \mathbf{x}_M\}$ as a statistical sample. This method is called *simple sampling*. The thermal average $\langle Q \rangle_T$ is approximated by the discrete sum

$$\overline{Q} = \frac{\sum_{l=1}^{M} e^{-\beta\,\mathcal{H}(\mathbf{x}_l)}\, Q(\mathbf{x}_l)}{\sum_{l=1}^{M} e^{-\beta\,\mathcal{H}(\mathbf{x}_l)}} \tag{1.5}$$

The problem is that not every configuration contributes the same amount to the partition function. In fact, there are a lot of configurations in phase space which yield nearly nothing when evaluating (1.5). With our simple sampling strategy we are totally neglecting this fact and hence are using a rather inefficient method. In order to illustrate this, we observe a 4×4 Ising-system with periodic boundary conditions. We see that the variation of the product $e^{-\beta\,\mathcal{H}(\mathbf{x}_l)}\, Q(\mathbf{x}_l)$ for different configuration is mainly caused by the Boltzmann-factor $e^{-\beta\,\mathcal{H}(\mathbf{x}_l)}$. In this example ($\beta$ and the parameter J, which appears in the Ising model, are set to 1), the Boltzmann-factor varies between e^{-32} and e^{+32} whereas the energy only varies in the relatively small region from -32 until $+32$. When computing the contributions to the partition function Z, we find out that at the critical point

[1] By developing the theoretical foundations we will follow [8].

($\beta \approx 0{,}4407$) two configurations alone (namely the configurations with all spins up and all spins down) contribute nearly 48%. 32 other configurations (namely those where all except one spin point into the same direction) yield another 23%. This means that 0,5 ‰ of the configurations make up 71% of the partition function.

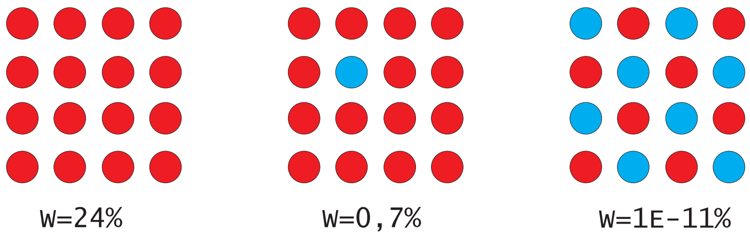


Figure 1.1: 3 of the possible $2^{16} = 65536$ configurations of a 4×4 Ising system and their contribution W to the partition function Z. Different colors are symbolizing the 2 different directions of spin.

From this example we see that we need a more efficient technique that does not sample the configurations $\mathbf{x}_l$ with equal probability, but is more likely to sample those configurations which contribute most to the partition function.

1.3.2 Importance sampling

A way of circumventing the problems above is the method described as follows. Let us consider a process where the phase space points $\mathbf{x}_l$ are selected according to some probability $P(\mathbf{x}_l)$. When choosing this set $\{\mathbf{x}_l\}$ for the estimation of a thermal average, we can replace Eq. (1.5) by

$$\overline{Q} = \frac{\sum_{l=1}^{M} e^{-\beta\,\mathcal{H}(\mathbf{x}_l)}\,Q(\mathbf{x}_l)\,/\,P(\mathbf{x}_l)}{\sum_{l=1}^{M} e^{-\beta\,\mathcal{H}(\mathbf{x}_l)}\,/\,P(\mathbf{x}_l)} \tag{1.6}$$

A simple and natural choice for $P(\mathbf{x}_l)$ would be $P(\mathbf{x}_l) \propto e^{-\beta\,\mathcal{H}(\mathbf{x}_l)}$. Then the Boltzmann factor cancels out and Eq. (1.7) reduces to a simple arithmetic average

$$\overline{Q} = \frac{1}{M} \sum_{l=1}^{M} Q(\mathbf{x}_l). \tag{1.7}$$

We now need to find a procedure which practically realizes this sampling strategy, called **importance sampling**. *Metropolis et. al.* had the idea not to choose the successive states $\{\mathbf{x}_l\}$ indepently of each other, but to construct a so called **Markov process** [2]. During this process a sequence of states, called a **Markov chain**, is created. The state $\mathbf{x}_{l+1}$ is randomly constructed from a state $\mathbf{x}_l$ via a suitable transition probability $W(\mathbf{x}_l \to \mathbf{x}_{l+1})$. The important thing about this probability W is that it depends only on the present state of the system and does not require knowledge of any previous one. It is possible to choose the transition probability W such that in the limit $M \to \infty$ the distribution $P(\mathbf{x}_l)$ of the states generated by this Markov process tends towards the equilibrium distribution

$$P_{eq}(\mathbf{x}_l) = \frac{1}{Z}\, e^{-\beta\, \mathcal{H}(\mathbf{x}_l)} \tag{1.8}$$

as desired. In order to achieve this it is sufficient that the transition probabilities $W(\mathbf{x} \to \mathbf{x}')$ fulfill the following 2 conditions:

1. From any configuration it is possible to reach every other configuration in a finite number of iterations. A Markov chain obeying this rule is called **ergodic**.

2. The transition probabilities fulfill the **detailed balance** condition:

$$P_{eq}(\mathbf{x})\, W(\mathbf{x} \to \mathbf{x}') = P_{eq}(\mathbf{x}')\, W(\mathbf{x}' \to \mathbf{x}) \tag{1.9}$$

Eq. (1.9) implies that the ratio of transition probabilities for a move $\mathbf{x} \to \mathbf{x}'$ and the inverse move $\mathbf{x}' \to \mathbf{x}$ depend only on the energy change $\Delta\mathcal{H} = \mathcal{H}(\mathbf{x}') - \mathcal{H}(\mathbf{x})$,

$$\frac{W(\mathbf{x} \to \mathbf{x}')}{W(\mathbf{x}' \to \mathbf{x})} = e^{-\beta\, \Delta\mathcal{H}} \tag{1.10}$$

If we analyze Eq. (1.10), we find that it does not specify $W(\mathbf{x} \to \mathbf{x}')$ uniquely, and allows us some freedom in the explicit choice of W. A frequently used choice is the Metropolis probability which gives the acceptance probability p_{ac} of a proposed state $\mathbf{x}'$:

$$p_{ac} = \begin{cases} e^{-\beta\, \Delta\mathcal{H}} & \text{if } \Delta\mathcal{H} > 0, \\ 1 & \text{otherwise.} \end{cases} \tag{1.11}$$

Another commonly used choice for accepting a proposed configuration is the heat bath probability. We list it here for the special case of deciding, if a single spin of the Ising system is to point up or down while all the other spins are fixed. The probability of accepting the configuration where the particular spin s_i points upwards is given by

$$p_{ac}(s_i^{\uparrow}) = \frac{e^{-\beta\, \mathcal{H}(s_i^{\uparrow})}}{e^{-\beta\, \mathcal{H}(s_i^{\uparrow})} + e^{-\beta\, \mathcal{H}(s_i^{\downarrow})}}. \tag{1.12}$$

Here $\mathcal{H}(s_i^{\uparrow})$ denotes the energy of the configuration, if the particular spin pointed upwards and similarly $\mathcal{H}(s_i^{\downarrow})$ denotes the energy of the configuration, if the spin pointed downwards while all other spins are fixed. Analogously, $p_{ac}(s_i^{\downarrow})$ is given by:

$$p_{ac}(s_i^{\downarrow}) = \frac{e^{-\beta\, \mathcal{H}(s_i^{\downarrow})}}{e^{-\beta\, \mathcal{H}(s_i^{\uparrow})} + e^{-\beta\, \mathcal{H}(s_i^{\downarrow})}}. \tag{1.13}$$

We conclude: A **Monte Carlo simulation** is a path through phase space. This path is a sequence of configurations of the system and in nearly all Monte Carlo simulations this sequence is constructed via a Markov process. The notion *Monte Carlo* simulation stems from the fact that the path through phase space is created by an element of chance.

1.4 Problems, subtleties and tricks of the trade

1.4.1 Thermalization

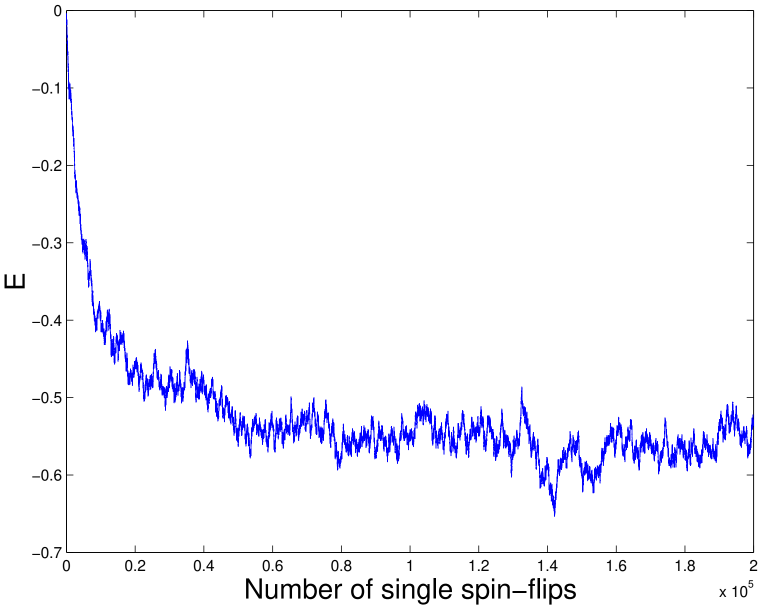


Figure 1.2: Illustration of *thermalization* and *autocorrelation* effects. Monte Carlo measurements of the energy of a 64 $\times$ 64 Ising system simulated via the Metropolis-algorithm.

When we create a series of states in a Monte Carlo simulation, we are not allowed to take measurements right from the beginning of the simulation. We must be sure that the Markov process has *thermalized* and reached equilibrium. This is necessary because we can only perform a finite number of Monte Carlo steps, whereas the Markov process tends towards the equilibrium distribution (Eq. (1.8)) only in the limit $M \to \infty$. This behaviour is illustrated in Fig. 1.2. In practice, we could face the situation that our arbitrary starting configuration lies in a rather improbable region of phase space and due to autocorrelations (see Sec. 1.4.2) is locked in this region for many Monte Carlo steps. By starting the measurements right from the beginning of the simulation, we would obtain wrong

results.
It is not easy to decide when a Markov chain has reached equilibrium. As a general rule of thumb one should perform a Monte Carlo measurement, measure the autocorrelation time τ_{exp} (see Eq. (1.15) for its definition) over the whole simulation and then throw away the data for at least the first $20\,\tau_{exp}$ sweeps. As *Sokal* puts it in [14], deciding if enough thermalization sweeps have been done is only reasonable when one knows the autocorrelation time of the algorithm. Another way of deciding if the Markov process has thermalized, is starting the simulation with different initial configurations and verifying if the measured quantities are identical within statistical errors. A third way of verifying thermalization is plotting the measured quantity as a function of Monte Carlo time and observing when the transient region in the beginning ends. However, all these methods involve the danger that the system has reached a *metastable* region in phase space which could possibly be very far from equilibrium.

1.4.2 Autocorrelations

When we perform a Monte Carlo simulation, we find out that successive configurations are more or less correlated with each other. The amount of correlation depends on the algorithm. This means that reaching a certain configuration in the next step strongly depends on the present configuration. In the local Metropolis algorithm for example (described in detail in Sec. 3.1) only single spin flips are carried out. This means we can only reach N of the 2^N possible states. In our 4×4 Ising model this corresponds to less than 0,3 ‰ of the possible configurations and this ratio becomes smaller and smaller with larger lattices. Because of this correlations it makes no sense to measure quantities in every Monte Carlo step; successive configurations **are not statistically independent**. An example for autocorrelation effects is depicted in Fig. 1.2.
A way of quantifying the correlations of measurement data Q_i (i labels the Monte Carlo steps) is calculating the autocorrelation function $C(t)$

$$C(t) = \frac{\langle Q_i\, Q_{i+t}\rangle - \langle Q_i\rangle\,\langle Q_{i+t}\rangle}{\langle Q_i^2\rangle - \langle Q_i\rangle^2} \tag{1.14}$$

This function is normalized ($C(0) = 1$) and for sufficiently large lags t it decreases like

$$C(t) \propto e^{-\frac{t}{\tau_{exp}}}. \tag{1.15}$$

Here we have introduced the exponential autocorrelation time τ_{exp}. In addition, one defines the integrated autocorrelation time τ_{int} [2]

[2]The term of $\frac{1}{2}$ is a matter of convention.

$$\tau_{int} = \frac{1}{2} + \sum_{t=1}^{\infty} C(t). \qquad (1.16)$$

τ_{int} directly influences the variance of the sample mean $\overline{Q}$. It is a factor $2\,\tau_{int}$ higher than it would be if we could sample configurations without correlations (a proof can be found in [14]). τ_{int} tells us that a Monte Carlo sample of length k contains only $k/(2\,\tau_{int})$ statistically independent data points.

1.4.3 Finite size scaling

When doing Monte Carlo simulations of many-particle systems we are naturally limited to simulate finite systems. But often we are interested in obtaining certain parameters which describe the behaviour of physical quantities of infinite systems at criticality. A phase transition is characterized by a singularity of some physical quantity or one of its derivatives. The singularity can be described by a critical exponent. When simulating finite systems, these singularities are rounded and shifted. By simulating a number of large systems, one is still able to extract the values of the critical exponents by plotting the quantities for different values of L (= linear dimension of the lattice) and fitting these plots with functions that contain the critical exponents as fit parameters. In the instance of a second order phase transition these so called *finite size scaling* relations are known and listed in Sec. 2.2.

Chapter 2

The Ising model

In order to simulate many-particle systems, appropriate models which describe the physics of these systems have to be found. The Ising model is one of the oldest and best studied models of this kind. Although it seems to be a rather simple model, it shows the main characteristics observed in a real-life many-particle system (e.g. a phase transition). Initially it was invented as a simple model of a ferromagnet (by W. Lenz in 1920), but it turned out that its applications reach into other areas of physics, chemistry, biology and even sociology. [1]

2.1 Description of the model

The Ising model views the atoms of a crystal as magnetic dipoles which can only be aligned in 2 opposite directions. Further, it is assumed that only nearest neighbour atoms interact significantly with each other and only their contributions are taken into account in the Hamiltonian. The atoms are usually assumed to be arranged on an evenly spaced lattice in d dimensions. The mathematical description works by assigning every atom i a spin-variable s_i which can only take the discrete values +1 or -1. The Hamiltonian $\mathcal{H}$ of the Ising model is given by

$$\mathcal{H} = -\sum_{\langle ij \rangle} J\, s_i s_j - H \sum_i s_i \tag{2.1}$$

Here $\sum_{\langle ij \rangle}$ means that we sum over all nearest-neighbour pairs in such a manner, that each pair appears exactly once in the sum. H is an externally applied magnetic field and J is a parameter (also called the *exchange integral*) which describes the strength of the interactions. J is normally assumed to be

[1] To mention just a few examples, the Ising model is used in modelling binary alloys, the adsorption of O_2 on haemoglobine, neural networks, protein folding, biological membranes, social imitation and social impact in human societies.

constant, but it is possible to work with spatially varying values of J which are then denoted by J_{ij}.[2]
The Hamiltonian in Eq. (2.1) tells us that a nearest-neighbour pair consisting of equal spins yields a contribution of $-J$ to the energy. Similarly, a pair of unequal spins yields a contribution of $+J$. If $J > 0$, a parallel alignment of the spins is favoured and hence the model describes a *ferromagnet*. If $J < 0$, an anti-parallel alignment is favoured and the model describes an *antiferromagnet*. In all our Monte Carlo simulations we will set J equal to $+1$ which means that we are describing a ferromagnet[3].

2.2 Behaviour at criticality

.

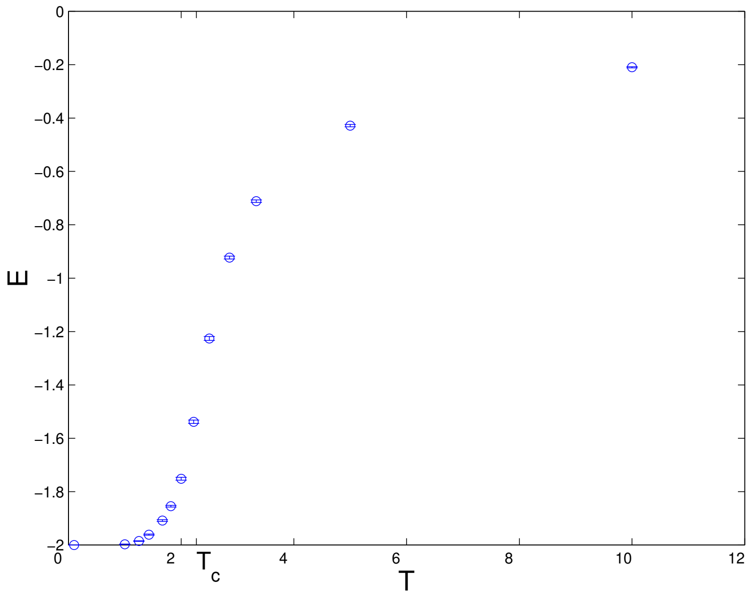


Figure 2.1: Phase transition of the Ising model. Energy vs. temperature, computed for an 8 × 8 lattice with *Swendsen-Wang* Monte Carlo algorithm (see Sec. 3.2).

Exact solutions of the Ising model only exist for the cases $d = 1$ and $d = 2$ (here only in the absence of a magentic field H). The $d = 1$ case shows no phase transition, which is the reason why we will not study it in our simulations. For the case $d = 2$, $H = 0$ *Onsager* was able to find the exact solution. In 2

[2]If the values of J_{ij} are randomly distributed with varying signs, the Hamiltonian models *spin glasses*.

[3]This is equivalent to describing an antiferromagnet on bipartide lattices.

dimensions the model shows a phase transition of second order at the critical point (temperature) T_c, which is given by

$$\frac{J}{k_B T_c} = \frac{1}{2} \ln\left(1+\sqrt{2}\right) \approx 0,44069. \tag{2.2}$$

T_c corresponds to the well known *Curie temperature* found in ferromagnetic materials. As we will learn later (in Ch. 3), one is especially interested in the dynamic behaviour of Monte Carlo algorithms at criticality. Therefore it is necessary to know the value for $\beta_c = \frac{1}{T_C}$ for the Ising model as accurately as possible. We list the values for $2d$, $3d$ and $4d$ in Tab. 2.1.

dimension d	β_C
2	0.44068679350977
3	0.2216595 ± 0.0000026
4	0.149694 ± 0.000002

Table 2.1: Values for β_c of the Ising model. The $2d$ value is analytically exact. $3d$ value taken from [9], $4d$ value from [10]

We now list the behaviour of important quantities at criticality:

- The specific heat c shows a critical behaviour of

  $$c \propto |T - T_c|^{-\alpha}. \tag{2.3}$$

  In 2 dimensions c shows a logarithmic dependence and is formally assigned the critical exponent $\alpha = 0$.

- For $T \geq T_c$ the magnetization in an infinite system disappears, whereas for $T < T_c$ a spontaneous magnetization can be observed, which shows the following behaviour:

  $$m \propto (T_c - T)^{\beta}. \tag{2.4}$$

  Here, β denotes the critical exponent and should not be mixed up with the inverse temperature. m is the magnetic moment per spin:

  $$m = \frac{M}{N} = \lim_{H\to 0+} \lim_{N\to\infty} \frac{1}{N}\langle \sum_i s_i \rangle. \tag{2.5}$$

  M denotes the total magnetic moment, N the number of lattice-sites and H the magnetic field.

- The magnetic susceptibility χ is defined as

$$\chi = \frac{\partial m}{\partial H}. \tag{2.6}$$

It show a critical behaviour, too

$$\chi \propto |T - T_c|^{-\gamma}. \tag{2.7}$$

- Another interesting quantity is the spatial correlation length ξ. We define the spatial correlation function

$$\Gamma(n) = \left\langle \frac{1}{N} \sum_i s_i s_{i+n} \right\rangle - \left\langle \frac{1}{N} \sum_i s_i \right\rangle^2. \tag{2.8}$$

ξ is now obtained via a fit of the asymptotic behaviour of $\Gamma(n)$ which depends on ξ:

$$\Gamma(n) \propto e^{-n/\xi} \tag{2.9}$$

Finally, the critical behaviour of ξ is given by

$$\xi \propto |T - T_c|^{-\nu}. \tag{2.10}$$

The exact results for the critical exponents of the $2d$ Ising model are useful for testing one's Monte Carlo simulations. Note that the values of α, β, γ and ν depend on the spatial dimension d of the system. Because no exact solutions of the Ising model are known for $3d$ and higher dimensions, Monte Carlo simulations yield a precious way of understanding these systems.

In Sec. 1.4.3 we have mentioned that singularities of the above kind are rounded and shifted when simulating *finite* systems. However, it is still possible to extract values for the critical exponents by simulating systems with different lattice sizes (preferrably as large as possible). The values for α, β, γ and ν are then obtained by "Finite Size Scaling", i.e. via a fit where we plot a quantity (e.g. the height of the specific heat peak) against a suitable power of the linear size L of the system. For a second order phase transition the dependences of some quantities on the critical exponents are listed below[4].

$$c_{\max}(L) \propto L^{\alpha/\nu} \qquad c_{\max}(L) = \text{height of specific heat peak} \tag{2.11}$$

$$|T_c(L) - T_c| \propto L^{-1/\nu} \qquad T_c(L) = \text{position of } c_{\max}(L) \tag{2.12}$$

$$\left\langle \left| \frac{1}{V} \sum_i s_i \right| \right\rangle \propto L^{-\beta/\nu} \tag{2.13}$$

$$\chi(T_c) \propto L^{\gamma/\nu} \tag{2.14}$$

In these equations T_c denotes the critical temperature of the infinite system.

[4]Relations taken from [5], p. 201 and p. 289.

2.3 Solving the Ising model

For $d = 1$ the Ising model can be solved exactly. By introducing periodic boundary conditions (the last spin in the chain of spins becomes a nearest neighbour of the first spin) a so called Ising-ring is formed. The partition function Z can be evaluated by calculating the eigenvalues of a special matrix, called *transfer matrix*. Further details can be found in [3]. The solution shows that the $1d$ Ising model has no phase transition. By extending the transfer-matrix method, Onsager was able to calculate the partition function for the $2d$ Ising model. His work is frequently called a mathematical *"tour de force"* and created a sensation when it appeared in 1944. Details about his solution can be found in [18]. As we have mentioned, the exact solution of the Ising model in 2 dimensions is rather lengthy and complicated and for $d \geq 3$ no exact solution has been found yet.

Fortunately, there exists a method which allows precious insights into a model without having to evaluate the partition function Z exactly: **series expansions**. With this method, it has been possible to obtain valuable information about the system's behaviour at criticality. A proof of its efficiency is the fact that critical exponents could be evaluated up to the 3^{rd} decimal place.
The idea behind a series expansion is to expand the partition function in powers of some parameter. If this parameter vanishes in the limit $\beta \to 0 (T \to \infty)$ we have a **high-temperature expansion**. We will study this kind of expansion in more detail because it forms the heart of the new Monte Carlo algorithm presented in this thesis. Nevertheless, it should be mentioned that this is not the only possibility for performing a series expansion. Another way of creating a series expansion is choosing an expansion parameter that vanishes in the limit $\beta \to \infty (T \to 0)$. This is called the low-temperature expansion.

2.3.1 High-temperature expansion

We start[5] with the partition function of the Ising model and consider the case $H = 0$:

$$Z = \sum_{\{s\}} e^{-\beta\,\mathcal{H}(s)} = \sum_{\{s\}} \prod_{\langle ij \rangle} e^{\beta\,J\,s_i s_j} \tag{2.15}$$

Here, $\sum_{\{s\}}$ means that we sum over all possible spin-configurations. We observe that the product $s_i s_j$ can only take 2 different values, $s_i s_j = \pm 1$, which allows us to write

$$\begin{aligned} e^{\beta\,J\,s_i s_j} &= \cosh(\beta J) + s_i s_j \sinh(\beta J) \\ &= \cosh(\beta J)\,(1 + s_i s_j v), \text{ where } v = \tanh \beta J. \end{aligned} \tag{2.16}$$

[5]We closely follow [4].

Here v plays the role of the expansion parameter and we see that $v \to 0$ as $T \to \infty$.

By using Eq. (2.16), we can write for the partition function

$$\begin{aligned} Z &= \sum_{\{s\}} \prod_{\langle ij \rangle} e^{\beta\, J\, s_i s_j} \\ &= (\cosh \beta J)^{N_{nn}} \sum_{\{s\}} \prod_{\langle ij \rangle} (1 + s_i s_j v) \\ &= (\cosh \beta J)^{N_{nn}} \sum_{\{s\}} \left(1 + v \sum_{\langle ij \rangle} s_i s_j + v^2 \sum_{\langle ij \rangle, \langle kl \rangle} s_i s_j s_k s_l + \dots \right). \end{aligned} \tag{2.17}$$

N_{nn} denotes the number of nearest neighbour pairs on the lattice. How can we evaluate Eq. (2.17)? A very helpful method for doing this is the visualization of the spin products $s_i s_j, s_i s_j s_k s_l, \dots$. We do this by assigning each pair of spins $s_i s_j$ a *bond* on the lattice which connects those 2 spins. Contributions to terms of order v are then represented by a single bond lying anywhere on the lattice. Similarly, terms of order v^2 are represented by 2 bonds (which may touch or not) and terms of order v^n are visualized by a pattern of n bonds on the lattice. Fortunately, we do not have to consider *all* possible bond patterns when evaluating the partition function. This becomes clear when we observe a single spin-product and remember that $s_i = \pm 1$ only:

$$\sum_{\{s\}} \left(s_i^{n_i} s_j^{n_j} s_k^{n_k} s_l^{n_l} \dots \right) = \begin{cases} 2^N, & \text{all } n_i \text{ even} \\ 0, & \text{otherwise} \end{cases} \tag{2.18}$$

where N denotes the total number of spins on the lattice. Eq. (2.18) expresses an important fact: Only those configurations of bonds which fulfill the constraint that the number of bonds radiating from each site is even, contribute to Z. This constraint is equal to the demand that the configuration consists of **closed loops only**. In future we will denote these configurations as *allowed configurations*. Every such configuration contributes the same weight to Z, all other configurations contribute nothing. The evaluation of the sum in Eq. (2.17) has now been reduced to counting the number of different configurations which consist of closed loops of $0, 2, 4, 6, 8, 10, \dots$ bonds (The case of 2 bonds forming a closed loop is only possible when at least one dimension of the lattice is 2 and periodic boundary conditions are used. The loop then winds around the lattice). The solution of this combinatorical task yielded valuable results for the values of T_c and the critical exponents.[6]

[6] In 2 dimensions the number of different closed loops with a certain number of bonds has been derived exactly, in greater dimensions at least approximate results by evaluating the contributions of as much loops as possible could be obtained.

By knowing that only closed loops contribute to Z, we will construct our new Monte Carlo algorithm right from the beginning in such a way, that we only create the contributing configurations and omit the non-contributing ones completely.

Measurement of the energy in the HT-representation

When observing Eq. (2.17) we recognize that it is possible to express the partition function Z as a sum over all allowed configurations of bonds rather than a sum over all possible spin-configurations. We further find that the weight of each bond-configuration only depends on the number n_b of bonds on the lattice (as usual, we have set the parameter J equal to 1) [7]:

$$Z = (\cosh\beta)^{N_{nn}}\, 2^N \sum_{\{b\}} v^{n_b} \tag{2.19}$$

Here $\sum_{\{b\}}$ denotes the sum over all allowed bond-configurations on the lattice. This representation of Z allows us to derive an expression for the thermal average of the internal energy E per site:

$$\begin{aligned}
\left\langle \frac{E}{N} \right\rangle &= -\frac{1}{N}\,\frac{\partial \ln Z}{\partial\beta} = -\frac{1}{N}\,\frac{1}{Z}\,\frac{\partial Z}{\partial\beta} \\
&= -\frac{1}{N}\,\frac{1}{Z}\,\Bigg(N_{nn}\,\sinh\beta\,(\cosh\beta)^{N_{nn}-1}\,2^N \sum_{\{b\}} v^{n_b} + \\
&\qquad\qquad (\cosh\beta)^{N_{nn}}\,2^N \sum_{\{b\}} \frac{n_b}{\cosh^2\beta}\,v^{n_b-1} \Bigg) \\
&= -\frac{1}{N}\left(N_{nn}\,v + \frac{1}{Z}\,(\cosh\beta)^{N_{nn}}\,2^N \sum_{\{b\}} \frac{n_b}{\sinh\beta\,\cosh\beta}\,v^{n_b} \right) \\
&= -\frac{1}{N}\left(N_{nn}\,v + \left\langle \frac{n_b}{\sinh\beta\,\cosh\beta} \right\rangle \right)
\end{aligned} \tag{2.20}$$

Since the number of nearest-neighbour pairs in a single-cubic lattice with periodic boundary conditions and of spatial dimension d is given by $N_{nn} = N \cdot d$, Eq. (2.20) becomes

$$\left\langle \frac{E}{N} \right\rangle = -\left(d\,v + \frac{\langle n_b \rangle}{N\,\sinh\beta\,\cosh\beta} \right). \tag{2.21}$$

[7] We will set J to 1 from now on without explicitly mentioning it.

Chapter 3

Existing Monte Carlo algorithms for the Ising model

3.1 The Metropolis algorithm

An early algorithm which realized the Markov process described in Sec. 1.3.2 was invented by *Metropolis et. al.* in 1953. In this algorithm, the probability of accepting a proposed state $\mathbf{x}'$ when the system is in the state $\mathbf{x}$, depends on the difference of energies between those 2 configurations only ($\Delta E = E(\mathbf{x}') - E(\mathbf{x})$):

- if $\Delta E \leq 0$: accept new configuration with $p = 1$
- if $\Delta E \geq 0$: accept new configuration with $p = e^{-\beta\,\Delta E}$

In other terms, the probability P for accepting the trial state $\mathbf{x}'$ is given by

$$P(\mathbf{x} \to \mathbf{x}') = \min\left(1, e^{-\beta\,\Delta E}\right) \tag{3.1}$$

The Metropolis algorithm fulfills detailed balance, provided the probability of proposal $\mathbf{x}'$ given $\mathbf{x}$ is the same as the probability for the proposal $\mathbf{x}$ given $\mathbf{x}'$. In the case of single spin-flip trials for the Ising model the created Markov chain is ergodic. We can understand this by observing the fact that every spin has a finite probability for being flipped. Hence, it is possible to reach from a given configuration any other configuration in a finite number of steps.

3.1.1 Monte Carlo algorithm

A Monte Carlo simulation of the Ising model using the Metropolis algorithm works as follows:

1. Create any initial spin configuration $\{s\}$ you like (e.g. by running through all spins of the lattice and deciding with a random number generator if a spin is to point up or down).

2. Randomly choose a site k with equal probability and propose to flip this spin: $s'_k = -s_k$. The proposed new state $\mathbf{x}'$ looks just like the old one but with this one spin k flipped.

3. Now calculate the energy difference:

$$\Delta E = E(\mathbf{x}') - E(\mathbf{x}) = 2\, s_k \sum_{\substack{\text{neighbors} \\ \text{of } k}} s_i \tag{3.2}$$

4. Draw a random number r from a uniform distribution between 0 and 1. Accept the proposed state if

$$r < e^{-\beta\,\Delta E}.$$

Otherwise retain the old state.

5. Measure the physical quantities you desire. In order to proceed, go to 2.

By definition, one Monte Carlo sweep consists of N Monte Carlo steps described above. After one Monte Carlo sweep, on average each spin has been suggested to be flipped. The practical implementation is rather easy and straightforward, which was one of the reasons for its great success in the early years of Monte Carlo simulations. However, the method bears a **major drawback**.

3.1.2 Critical slowing down

The Metropolis algorithm makes local changes one site at a time. Therefore successive configurations are strongly correlated and not statistically independent. As we have learned in Sec. 1.4.2, a measure for the correlation of configurations is the exponential autocorrelation time τ_{exp}. Since configurations are not statistically independent within a time interval of order τ_{exp}, this greatly increases the statistical error for observables if τ_{exp} becomes large. When we approach the critical point (critical temperature), τ_{exp} diverges as

$$\tau_{exp} \sim \xi^z \tag{3.3}$$

where ξ is the spatial correlation length and z is called dynamic critical exponent. This relation has been found *empirically*. For finite systems used in computer simulations the correlation length does not diverge at the critical point (this is a typical *finite size effect*), but it does increase with the system size as

$$\tau_{exp} \sim L^z \tag{3.4}$$

where L is the linear dimension of the system. The dynamic critical exponent z strongly depends on the kind of algorithm used to perform the Monte Carlo simulation and on the spatial dimension d. For local update algorithms like the Metropolis algorithm $z \approx 2$. This tells us that τ_{exp} grows rapidly for large systems. However, for extracting critical exponents or determining T_c it is necessary to simulate systems as large as possible and hence it is essential to find more advanced algorithms which show a lower value for z. The measured values of z for the Metropolis algorithm are listed in Tab. 3.1.

dimension d	z
2	2.167 ± 0.001
3	2.02 ± 0.02
4	≈ 2

Table 3.1: Dynamic critical exponents z for the Metropolis algorithm. Values for $2d$ and $3d$ taken from [17], for $4d$ apparently no value has been measured.

The large increase of τ_{exp} near the critical point is a phenomenon called **critical slowing down**. Let us try to find a reasonable physical interpretation for this behaviour: Eq. (2.10) tells us that the approach of the critical point is accompanied by a large increase in the spatial correlation length ξ. This means that the system contains larger and larger clusters (= areas with equal alignment of spins). If we want to create statistically independent configurations, we need to have a reasonable probability for flipping all spins in a cluster. However, the Metropolis algorithm just allows us to flip spins one by one. Since neighbouring spins in a cluster are aligned, flipping a spin means creating an energetically unfavourable configuration and hence such moves have a small probability to be accepted (remember, that things get worse because we have to flip all the spins in a cluster).

The way out of this problem now nearly suggests itself. We need to find an algorithm which flips many spins simultaneously. When creating this algorithm, it is important to maintain the 2 conditions detailed balance and ergodicity. In 1987, *Swendsen* and *Wang* presented the first successful algorithm of this kind.

3.2 The Swendsen-Wang algorithm

3.2.1 Monte Carlo algorithm

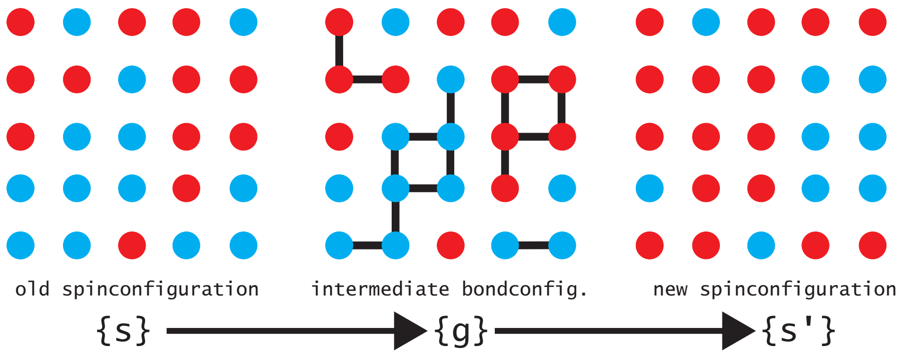


Figure 3.1: Illustration of the Swendsen-Wang algorithm.

The rules for the Swendsen-Wang algorithm are as follows:

1. Start with *any* spin configuration $\{s\}$ you like.

2. Now go through each nearest neighbour pair of the lattice. Create a *bond* between the two neighbouring sites i and j with probability p_{SW}:

$$p_{SW} = \begin{cases} 1 - e^{-2\beta} & \text{if } s_i = s_j, \\ 0 & \text{if } s_i \neq s_j. \end{cases} \tag{3.5}$$

   This means that we are setting a bond between two neighbouring sites with a certain probability if the spin values are the same. However, we *never* put a bond between two sites if the spin values are different. By applying the rules above we create a configuration of bonds $\{g\}$.

3. We now identify clusters as a set of sites connected by bonds. An isolated site is also called a cluster. With this definition, each site belongs to a cluster. To proceed, there are 2 slightly different possibilities:

   - In the **single-cluster algorithm** one cluster is selected at random and all the spins in the cluster are flipped with probability 1. Note that only this cluster has to be constructed.
   - In the **multi-cluster algorithm** the clusters are scanned sequentially and each cluster is assigned a spin value of +1 or -1 with equal probability. This means that the clusters are flipped with probability 1/2.

   At this point we have created a new spin configuration $\{s'\}$ and the old spin values can now be "forgotten".

4. We can finally carry out the measurements for the desired observables. By doing so we can take advantage of the clusters using *improved estimators*. This technique is described in detail below. One Monte Carlo step is now finished. Goto 2 for the next step.

3.2.2 Theoretical background of the Swendsen-Wang algorithm

We start with a special way of writing the Boltzmann-weight of a configuration which is called the *Fortuin-Kasteleyn representation*. The contribution of a nearest neighbour pair of spins i and j to the weight of a configuration can be written as

$$e^{\beta(s_i s_j - 1)} = p + (1-p)\,\delta_{s_i s_j}, \text{ where } p = e^{-2\beta} \tag{3.6}$$

Note, that we have introduced an additional constant factor $e^{-\beta}$ which cancels out when evaluating thermal averages. The identity in Eq. (3.6) can be easily verified by observing that the product $s_i s_j$ can only take the values +1 (if $s_i = s_j$) or -1 (if $s_i \neq s_j$). We now introduce a new set of variables g which can take the values 0 or 1. These variables describe the Swendsen-Wang bonds. The right hand side of Eq. (3.6) can now be written as

$$p + (1-p)\,\delta_{s_i s_j} = \sum_{g=0,1} \underbrace{\delta_{g,0}\,p + \delta_{g,1}\,(1-p)\,\delta_{s_i s_j}}_{=W_{ij}(s,g)} \tag{3.7}$$

This representation describes the configuration depicted in the middle of Fig. 3.1, which consists of two set of variables, namely spins s and bonds b. We can therefore write for the partition function Z:

$$Z \sim \sum_{\{s\}} \sum_{\{g\}} \prod_{\langle ij \rangle} W_{ij}(s,g) \tag{3.8}$$

Eq. (3.7) tells us the probability for setting a bond g, if the values of the spins s_i and s_j are given

Case 1: $s_i = s_j$

$\Rightarrow W_{ij} = \delta_{g,0}\,p + \delta_{g,1}\,(1-p)$

$\Rightarrow g_{ij} = 1$

$$\text{with heat bath probability } \frac{(1-p)}{p+(1-p)} = 1 - p = 1 - e^{-2\beta} := p_{SW} \tag{3.9}$$

Here p_{SW} is the Swendsen-Wang probability of setting a bond between spins of equal value.

Case 2: $s_i \neq s_j$

$$\Rightarrow W_{ij} = \delta_{g,0}\, p$$
$$\Rightarrow g_{ij} = 0 \text{ with probability } \frac{p}{p} = 1 \tag{3.10}$$

This result tells us that spins with different values are never connected via a bond.

Next we ask for the conditional probabilities of getting new values for the spins s_i and s_j when the Swendsen-Wang bonds g_{ij} are given. These probabilities determine the transition from the middle configuration to the right configuration in Fig. 3.1.

Case 1: $g_{ij} = 1$

$$\Rightarrow W_{ij} = \delta_{g,1}\,(1-p)\,\delta_{s_i s_j}$$
$$\Rightarrow s_i = s_j = \pm 1 \text{ with probability } \frac{(1-p)}{(1-p)} = 1 \tag{3.11}$$

This is in accordance with the Swendsen-Wang algorithm, where all spins in the same cluster need to have the same values and hence we have to flip whole clusters to obtain a new spin-configuration $\{s'\}$.

Case 2: $g_{ij} = 0$

$$\Rightarrow W_{ij} = \delta_{g,0}\, p$$

We see that s_i and s_j do not appear in this equation and therefore s_i and s_j may be chosen freely. Again, this result is in accordance with the Swendsen-Wang algorithm, which imposes no restrictions on the spin values in neighbouring clusters.

3.2.3 Improved estimators

The expectation value of an observable $\mathcal{O}$ can be written as

$$\langle \mathcal{O} \rangle = \frac{1}{Z} \sum_g \sum_s \mathcal{O}(s)\, W(g,s) \tag{3.12}$$
$$= \frac{1}{Z} \sum_g \sum_s \mathcal{O}(s)\, \frac{W(g,s)}{W(g)}\, W(g) \tag{3.13}$$
$$= \frac{1}{Z} \sum_g \langle \mathcal{O} \rangle_g\, W(g) \tag{3.14}$$
$$\text{with } \langle \mathcal{O} \rangle_g \equiv \sum_s \mathcal{O}(s)\, \underbrace{\frac{W(g,s)}{W(g)}}_{p(s|g)} \tag{3.15}$$
$$\text{and } W(g) \equiv \sum_s W(g,s). \tag{3.16}$$

Note that all spins within a cluster must be the same and can only take 2 different values. Therefore

$$p(s|g) = \frac{1}{2^{\#clusters}}. \tag{3.17}$$

This result tells us that it is possible to measure an observable either in the spin- or graph representation. A measurement of an observable in the graph representation is preferrable to a measurement in the spin representation because it shows a reduced standard deviation of the sample mean. If this is the case, we speak of **improved estimators**. In order to illustrate this, we will study an example.

Measurement of spatial correlation functions

Let us calculate the thermal average of spatial correlation functions $\langle s_i s_j \rangle$. In the graph representation we can distinguish 2 different cases:

1. Site i and site j belong to different clusters. The average over the 4 possible combinations of the spin values $s_i, s_j = \pm 1$ yields

$$\langle s_i s_j \rangle_g = \sum_s p(s|g)\, s_i\, s_j = \frac{1}{4} \sum_{s_i=\pm 1} \sum_{s_j=\pm 1} s_i\, s_j = 0. \tag{3.18}$$

2. Site i and site j belong to the same cluster. Because all spins in the same cluster need to have the same spin value, $s_i = s_j$ and $s_i\, s_j = 1$. Now the average yields

$$\langle s_i s_j \rangle_g = \sum_s p(s|g)\, s_i\, s_j = \frac{1}{2} \sum_{s_i=s_j=\pm 1} s_i\, s_j = 1. \tag{3.19}$$

By combining the results above we obtain the rule for measuring an improved estimator in the graph representation:

$$\langle s_i s_j \rangle_g = \delta_{i \text{ and } j \text{ belong to the same cluster}}. \tag{3.20}$$

The advantage of measuring $\langle s_i s_j \rangle$ via an improved estimator becomes clear when we calculate the variance $\sigma^2 = \langle (s_i s_j)^2 \rangle - \langle s_i s_j \rangle^2$ of the average:

1. Conventional method of measuring $\langle s_i s_j \rangle$:

$$\begin{aligned} \sigma^2 &= \sum_s \underbrace{(s_i s_j)^2}_{=1} \frac{W(s)}{Z} - \left(\sum_s s_i s_j \frac{W(s)}{Z} \right)^2 \\ &= \underbrace{\sum_s p(s)}_{=1} - \langle s_i s_j \rangle^2 \end{aligned} \tag{3.21}$$

Since $\langle s_i s_j \rangle \propto e^{-\frac{|i-j|}{\xi}} \to 0$ for large $|i-j|$ and $T \geq T_c$ (cf. Eq. (2.9)), we find

$$\sigma^2 \approx 1 - 0 \approx 1 \text{ for large } |i-j|. \tag{3.22}$$

Note that this value is further reduced by up to a factor of $O(\frac{1}{N})$ when averaging over different sites i for a correlation function $\Gamma(r)$.

2. Improved estimator method of measuring $\langle s_i s_j \rangle$:

$$\begin{aligned}
\sigma_{IE}^2 &= \sum_g \underbrace{(\langle s_i s_j \rangle_g)^2}_{=\langle s_i s_j \rangle_g} \frac{W(g)}{Z} - \left(\sum_g \langle s_i s_j \rangle_g \frac{W(g)}{Z} \right)^2 \\
&= \langle s_i s_j \rangle - \langle s_i s_j \rangle^2 \\
&= e^{-\frac{|i-j|}{\xi}} - e^{-\frac{2\,|i-j|}{\xi}} \\
&\approx e^{-\frac{|i-j|}{\xi}} \xrightarrow{\text{for large } |i-j|} 0.
\end{aligned} \tag{3.23}$$

One is especially interested in the relative Monte Carlo error ε of the average $\langle Q \rangle$ when sampling N_{meas} measurement values of a quantity Q:

$$\varepsilon = \frac{\sigma}{\langle Q \rangle \sqrt{N_{meas}}}. \tag{3.24}$$

Hence, for the conventional method of measuring $\langle s_i s_j \rangle$ we obtain

$$\varepsilon = \frac{1}{\sqrt{N_{meas}}} e^{+\frac{|i-j|}{\xi}}. \tag{3.25}$$

Similarly, the improved estimator yields

$$\varepsilon_{IE} = \frac{1}{\sqrt{N_{meas}}} e^{+\frac{|i-j|}{2\,\xi}}. \tag{3.26}$$

Measurement of $\sum_{ij} s_i s_j = (\sum_i s_i)^2$

By applying the improved estimator for measuring a spatial correlation function, we find a rule for measuring $\sum_{ij} s_i s_j$:

$$\begin{aligned}
\left\langle \sum_{ij} s_i s_j \right\rangle_g &= \sum_{ij} \delta_{i \text{ and } j \text{ in same cluster}} \\
&= \sum_{\substack{\text{all} \\ \text{clusters } k}} (\text{size of cluster } k)^2
\end{aligned} \tag{3.27}$$

Measurement of the magnetization

The magnetization M of the system is defined via a thermodynamic limit (H denotes the magnetic field)

$$\frac{M}{N} = \lim_{H\to 0+} \lim_{N\to\infty} \frac{1}{N} \langle \sum_i s_i \rangle \tag{3.28}$$

In this equation it is important not to exchange the limits. In an infinite Ising system M equals the fraction of sites in the infinite cluster (it has been found that there exists only 1 infinite cluster) [19]. Therefore an improved estimator with the correct thermodynamic limit for measuring the magnetization is to measure the size of the **largest** cluster on the lattice.

Measurement of the susceptibility

By using Eq. (2.6) we find

$$\chi_N = \langle \sum_{ij}^{N} s_i s_j \rangle - M^2 \tag{3.29}$$

which approaches χ in the thermodynamic limit $N \to \infty$. Since we have given rules both for measuring $\sum_{ij} s_i s_j$ and M, χ can be obtained immediately.

3.2.4 Autocorrelation results

The performance of the Swendsen-Wang algorithm in terms of correlation times is remarkable. The dynamical critical exponent z for $2d$, $3d$ and $4d$ can be found in Tab.3.2.

dimension d	z
2	0.25 ± 0.01
3	0.54 ± 0.02
4	0.86 ± 0.02

Table 3.2: Dynamic critical exponents z for the Swendsen-Wang algorithm. Values taken from [17].

An autocorrelation analysis for the multicluster Swendsen-Wang algorithm for 2,3 and 4 dimensions was performed[1]. The values for z obtained with this analysis are listed in Tab. 3.3. The number of Monte Carlo measurements taken for each lattice-size can be found in Sec. A.4.

[1]For details on how the analysis was carried out see Sec.A.2.

dimension d	z
2	0.32 ± 0.01
3	0.60 ± 0.01
4	0.81 ± 0.01

Table 3.3: Dynamic critical exponents z for the Swendsen-Wang algorithm. Values obtained with own autocorrelation analysis.

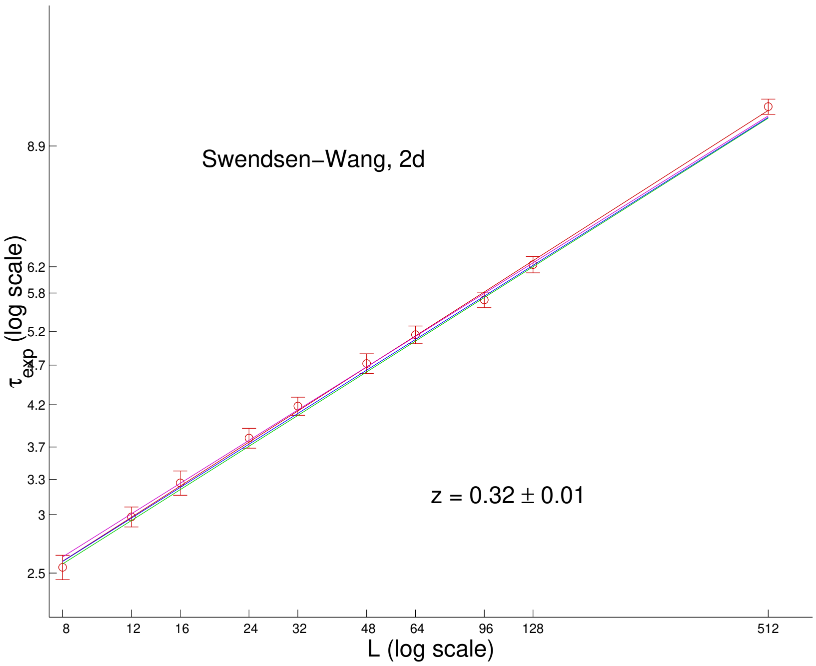


Figure 3.2: Autocorrelation analysis of 2d Swendsen-Wang multicluster data. τ_{exp} vs. linear dimension L at β_c.

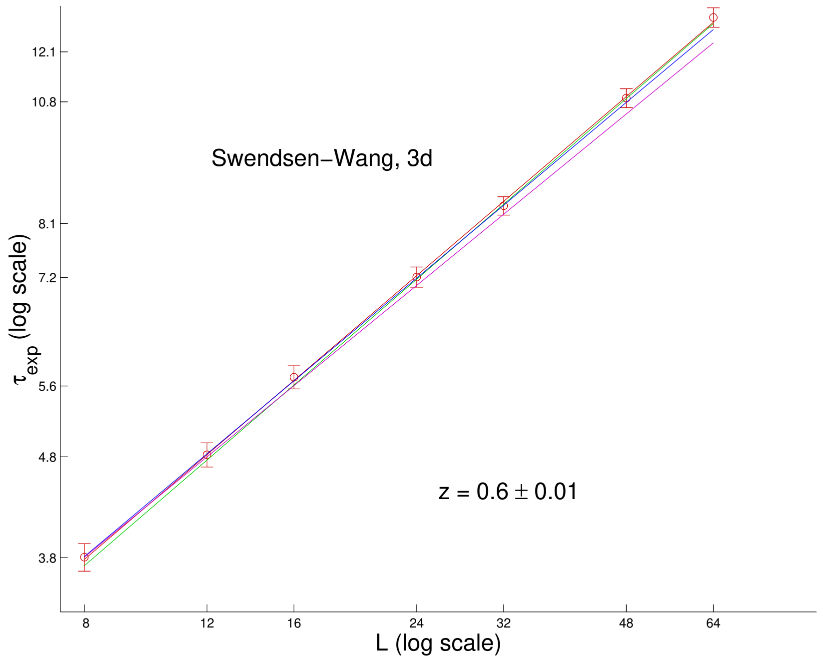


Figure 3.3: Autocorrelation analysis of 3d Swendsen-Wang multicluster data. τ_{exp} vs. linear dimension L at β_c.

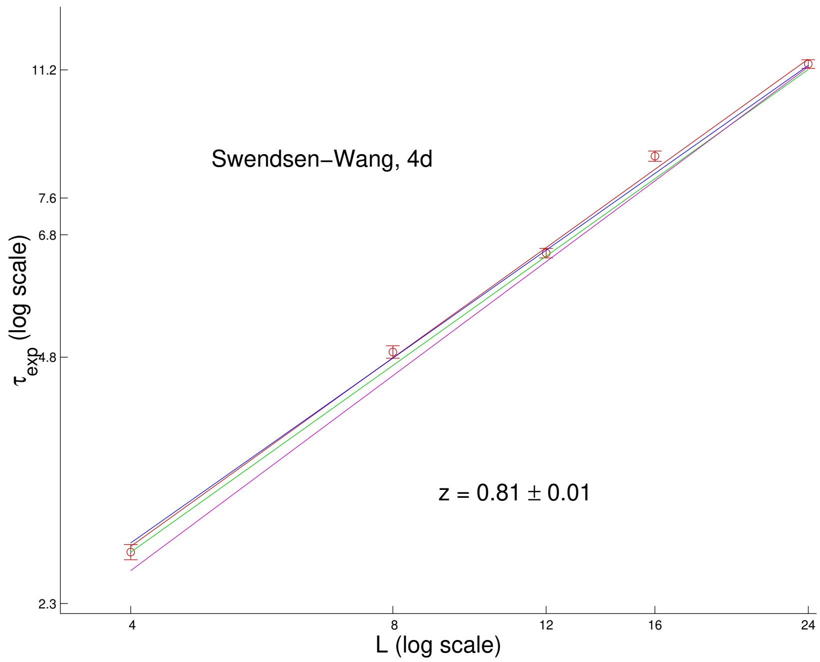


Figure 3.4: Autocorrelation analysis of 4d Swendsen-Wang multicluster data. τ_{exp} vs. linear dimension L at β_c.

Chapter 4

A new cluster algorithm for the Ising model

In Sec. 2.3.1 we have presented the high-temperature expansion as one way of representing the partition function of the Ising model. We found out that we can visualize spin-products, which make up the partition function, as bonds on a lattice. At first these bonds just seem to be a useful tool for handling these products. They seem to have no connection to the Swendsen-Wang bonds, which are physical bonds. However, in Sec. 4.2.2 it will turn out that there exists a remarkable relation to the Swendsen-Wang bonds which is not at all obvious in the beginning.

4.1 A local algorithm based on the HT-expansion

To obtain the allowed configurations, which consist of closed loops only, already a very simple local update algorithm does the job.[1] The algorithm works as follows.

4.1.1 Monte Carlo algorithm

1. Start with an allowed bond-configuration with winding number[2] zero in the HT-representation[3] (= all configurations consisting of closed loops only). For example, you could start with an empty lattice (this means setting the values of each bond to 0, if we define an existing bond to have the value 1).

2. We define a so called **plaquette** to consist of the edges connecting 4 lattice sites which are arranged in a square. This is the second simplest kind of a closed loop on the lattice (with no loop at all being the simplest kind).

[1] Only in a subspace of constant winding number, see discussion below.

[2] We define the winding number to be the total number of times all loops wind around the lattice when using periodic boundary conditions.

[3] Because the state of lowest energy is contained in this subspace.

Choose any of the plaquettes on the lattice with equal probability $p = 1/N_{pl}$. Here, N_{pl} denotes the number of different plaquettes on the lattice. An analytical expression for N_{pl} is given in Eq. (4.6).

3. Now we make a proposal for a new configuration. The new configuration looks exactly like the old one except that all bonds of the selected plaquette are changed. This means bonds with values of 0 are set to 1 and vice versa. Examples for these proposals are depicted in Fig. 4.1 and Fig. 4.2.

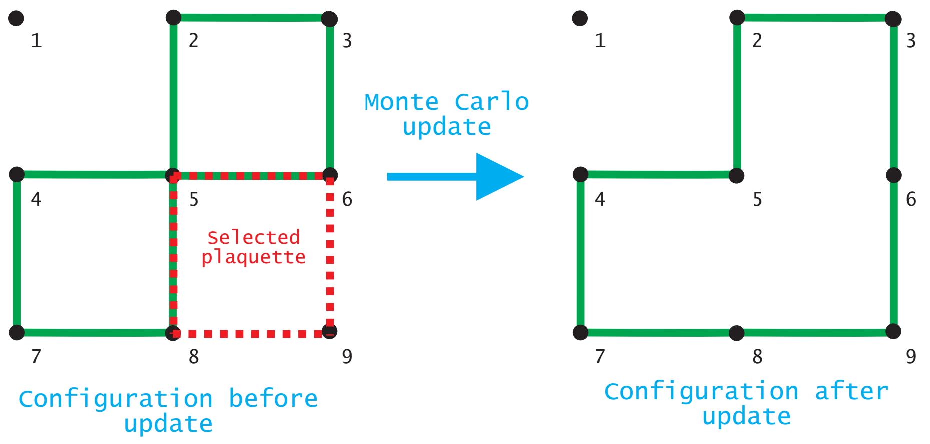


Figure 4.1: An example for a possible MC-update. Here $n_{b,\text{new}} = n_{b,\text{old}}$ and hence we accept the move with probability $p = 1$

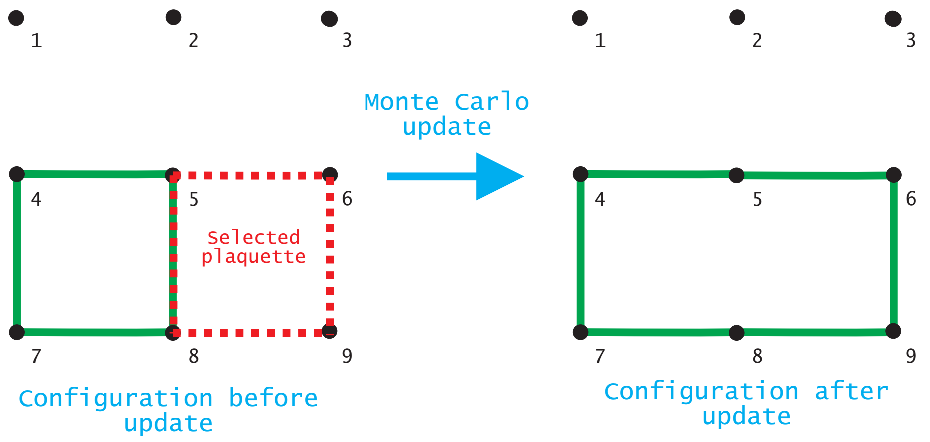


Figure 4.2: Another example for a possible MC-update. Here $n_{b,\text{new}} = 6, n_{b,\text{old}} = 4$. We therefore accept the move with probability $p = v^2$

We have to calculate the number of existing bonds in the old configuration $n_{b,\text{old}}$ and the number of existing bonds in the new configuration $n_{b,\text{new}}$.

$n_{b,\text{new}}$ can be calculated by counting the number of bonds b_{count} surrounding the selected plaquette in the old configuration:

$$n_{b,\text{new}} = n_{b,\text{old}} + 4 - 2\, b_{\text{count}} \tag{4.1}$$

We accept the proposed update of the configuration with Metropolis acceptance probability

$$p_{ac} = \min\left(1,\, \frac{v^{n_{b,\text{new}}}}{v^{n_{b,\text{old}}}}\right) \tag{4.2}$$

The weight of a configuration consisting of n_b bonds is proportional to v^{n_b} as we have seen in Eq. (2.19).

4. Now we are able to measure the energy, but *not* the magnetization, susceptibility or spatial correlation functions. In Eq. (2.21) we have learned that for measuring the energy per site only the number of bonds n_b of a configuration needs to be known:

$$\frac{E}{N} = -\left(d\, v + \frac{n_b}{N \sinh\beta \cosh\beta}\right) \tag{4.3}$$

5. After this go to 2 and start the next Monte Carlo step. Because this algorithm is inherently local, it is (similar to the Metropolis algorithm) only reasonable to measure a quantity every N^{th} Monte Carlo step.

When the algorithm described above is used for calculating thermal averages, we find that the results for small lattices do not agree (within statistical errors) with results of other simulations, e.g. Swendsen-Wang. The reason for this is that the algorithm **is not able to produce all possible configurations**. The configurations that can not be created are loops which wind around the lattice (winding around the lattice is possible, because we use periodic boundary conditions). At large lattices, such configurations only have a very small weight and therefore the thermal averages are correct within statistical errors. However, when simulating small lattices, such configurations have a higher weight and hence the averages are incorrect. Two of those configurations, which can not be created, are displayed in Fig. 4.3.

4.1.2 Practical implementation

To implement the algorithm described above we need to label the different plaquettes. Therefore it is useful to assign plaquettes to lattice sites. In 2 dimensions

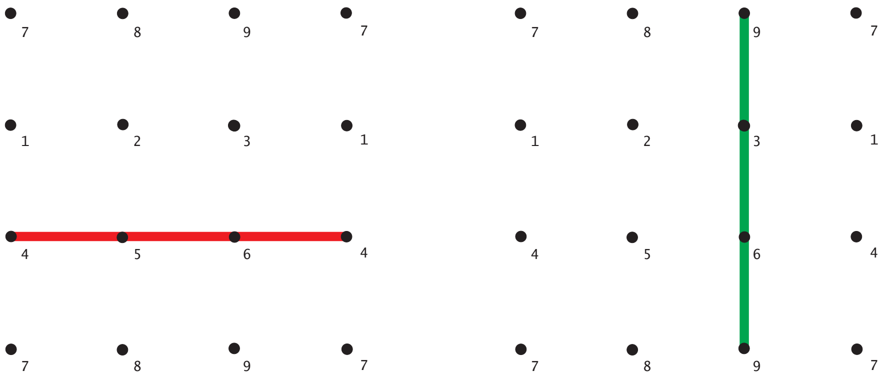


Figure 4.3: Two of the configurations which can not be created by the local algorithm based on the HT-representation.

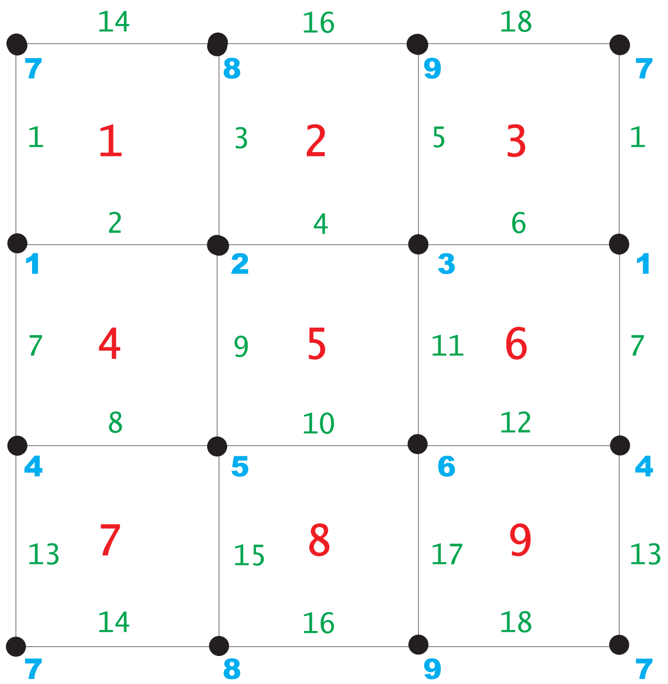


Figure 4.4: Illustration of plaquette labeling and bond labeling. Red numbers label plaquettes, green numbers label bonds and blue numbers label the lattice sites.

every site is assigned 1 plaquette, in 3 dimensions 3 plaquettes and in 4 dimensions 6 plaquettes. The general rule for the number of plaquettes n assigned to a lattice site in d dimensions is

$$n = \frac{d\,(d-1)}{2} \tag{4.4}$$

This formula can be derived by counting the number of different 2-dimensional orthogonal subspaces in a space of dimension d:

$$n = \binom{d}{2} = \frac{d!}{2!\,(d-2)!} = \frac{d\,(d-1)}{2} \tag{4.5}$$

By using Eq. (4.4), we get for the total number N_{pl} of different plaquettes

$$N_{pl} = N\,\frac{d\,(d-1)}{2} \tag{4.6}$$

In order to change the bonds of a plaquette during the MC-simulation it is necessary to enumerate the bonds and assign each plaquette the numbers of the 4 surrounding bonds. Therefore each lattice site are assigned d bonds, see Fig 4.4. Now the task of assigning 4 bonds to each plaquette is relatively easy and can be done for all dimensions desired. It turned out that the method of finding a general rule for plaquette- and bond labeling works by far the best. It is not advised to use spatial imagination for this task, because at latest in the $4d$ case this method fails to succeed.

4.1.3 Autocorrelation results for the local algorithm

An autocorrelation analysis of the Monte Carlo data produced with the algorithm was performed. We simulated a $2d$ simple-cubic lattice with periodic boundary conditions with linear dimensions L of 4, 8 and 16. We measured the exponential autocorrelation time τ_{exp} for the energy as a function of the linear dimension L. The result is depicted in Fig. 4.5. A fit yields the value for the critical dynamic exponent z:

$$z = 1.98 \pm 0.01$$

This values is very close to the value of $z \approx 2$ of the Metropolis single spin-flip algorithm. This is no surprise since this algorithm performs only local changes.

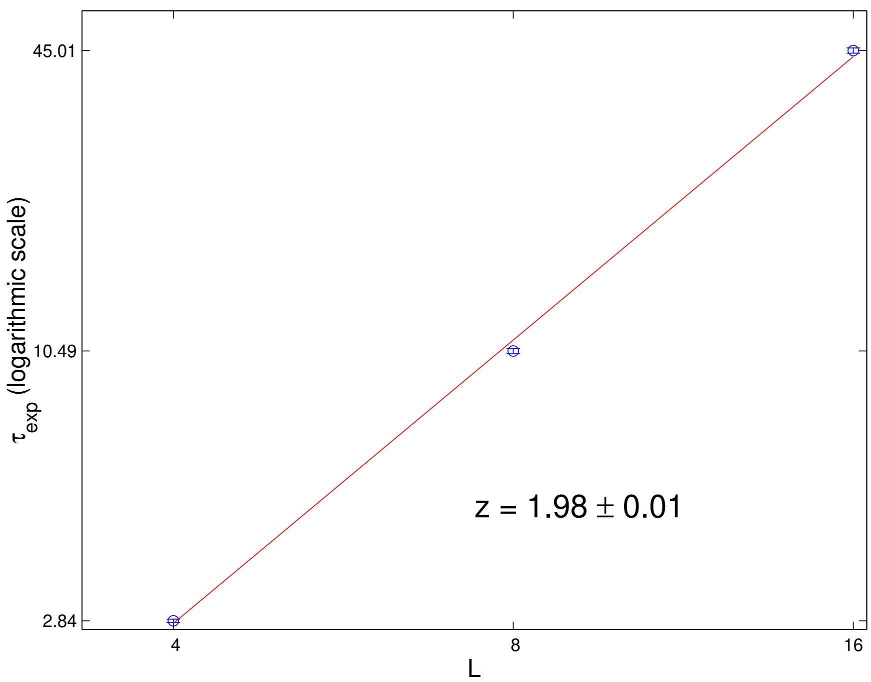


Figure 4.5: Autocorrelation analysis of 2d HT local algorithm data. τ_{exp} (logarithmic scale) vs. linear dimension L at β_c.

4.2 A new cluster algorithm

As we have seen in Sec. 4.1.3, a local update algorithm in the HT-representation shows a very unsatisfying critical exponent of $z \approx 2$. It is obvious, that a *global* update algorithm which operates in the HT-representation is needed. In 1989, *Ben-Av et. al.* presented a global-update algorithm for simulations of the $Z(2)$ lattice gauge theory in 3 dimensions and were able to reduce z from $z > 2$ to $z \approx 0.73$ [7]. Their algorithm actually updates configurations of the dual model, namely the $3d$ Ising model. We adapted the algorithm for simulating the Ising model and found that it allows us to perform global updates in the HT-representation. Further, we are not limited to $d = 3$, but generalized the algorithm for use in any desired dimension.

4.2.1 Theoretical background

In order to develop the algorithm, we start with the high-temperature representation of the partition function:

$$Z = \sum_{\{s\}} e^{-\beta H} = \sum_{\{s\}} (\cosh \beta)^{N d} \prod_{<ij>} (1 + v\, s_i s_j). \tag{4.7}$$

Now we introduce a new set of variables b_{ij}, where $b_{ij} = 0, 1$.

$$Z = (\cosh \beta)^{N d} \sum_{\{s\}} \prod_{<ij>} \sum_{b_{ij}=0,1} (\delta_{b_{ij},0} + \delta_{b_{ij},1}\, v\, s_i s_j) \tag{4.8}$$

$$= (\cosh \beta)^{N d} \sum_{\{b_{ij}\}} \sum_{\{s\}} \prod_{<ij>} (\delta_{b_{ij},0} + \delta_{b_{ij},1}\, v\, s_i s_j). \tag{4.9}$$

Here, $\sum_{\{b_{ij}\}}$ denotes the sum over all possible bond-configurations. We define the weight

$$W_{ij}(b_{ij}, s_i, s_j) = \delta_{b_{ij},0} + \delta_{b_{ij},1}\, v\, s_i s_j. \tag{4.10}$$

Further, we introduce another set of variables g_{ij}. We will see, that these variables will be helpful in constructing the allowed configurations which will be represented by the variables b_{ij}. By introducing the g_{ij} we can write W_{ij} as

$$W_{ij}(b_{ij}, s_i, s_j) = \sum_{g_{ij}=0,1} W_{ij}(b_{ij}, g_{ij}, s_i, s_j) \text{ with}$$

$$W_{ij}(b_{ij}, g_{ij}, s_i, s_j) = \delta_{g_{ij},1}\, v(\delta_{b_{ij},0} + \delta_{b_{ij},1}\, s_i s_j) + \delta_{g_{ij},0}\, \delta_{b_{ij},0}\, (1 - v) \tag{4.11}$$

Now we can do a Monte Carlo simulation in the variables b_{ij} and g_{ij} and analytically perform the sum over the spins s_i.

First we ask for the conditional probability for setting a graph $G = \{g_{ij}\}$, if we have given the bonds $B = \{b_{ij}\}$ of the HT-clusters which make up an allowed

configuration.
Case 1: $b_{ij} = 0$

$$\begin{aligned}
&\Rightarrow W_{ij} = \delta_{g_{ij},1}\, v + \delta_{g_{ij},0}(1-v) \\
&\Rightarrow p(g_{ij} = 1 | b_{ij} = 0, s_i, s_j) = v \qquad (4.12) \\
&\Rightarrow p(g_{ij} = 0 | b_{ij} = 0, s_i, s_j) = 1 - v \qquad (4.13)
\end{aligned}$$

Case 2: $b_{ij} = 1$

$$\begin{aligned}
&\Rightarrow W_{ij} = \delta_{g_{ij},1} v\, s_i\, s_j \\
&\Rightarrow p(g_{ij} = 1 | b_{ij} = 1, s_i, s_j) = 1 \qquad (4.14) \\
&\Rightarrow p(g_{ij} = 0 | b_{ij} = 1, s_i, s_j) = 0 \qquad (4.15)
\end{aligned}$$

Second we ask for the conditional probability for setting a bond B of the HT-clusters, if we have given a graph G.
Case 1: $g_{ij} = 0$

$$\begin{aligned}
&\Rightarrow W_{ij} = \delta_{b_{ij}=0}\,(1-v) \\
&\Rightarrow p(b_{ij} = 1 | g_{ij} = 0, s_i, s_j) = 0 \qquad (4.16) \\
&\Rightarrow p(b_{ij} = 0 | g_{ij} = 0, s_i, s_j) = 1 \qquad (4.17)
\end{aligned}$$

Case 2: $g_{ij} = 1$

$$\Rightarrow W_{ij} = v\,(\delta_{b_{ij},0} + \delta_{b_{ij},1} s_i s_j) \qquad (4.18)$$

Another important result is obtained by the summation of $W_{ij}(b_{ij}, g_{ij}, s_i, s_j)$ over $\{s\}$. We find that $W(b|g) = \sum_s W_{ij}(b_{ij}, g_{ij}, s_i, s_j)$ is different from zero if and only if b obeys the constraint of forming closed loops on those edges where $g_{ij} = 1$.

$$\begin{aligned}
Z &\sim \sum_g \sum_b \sum_s \prod_{<ij>} W_{ij}(b_{ij}, g_{ij}, s_i, s_j) \\
&= \sum_g \sum_b \sum_s \prod_{<ij>} \big(\delta_{g_{ij},1}\, v\,(\delta_{b_{ij},0} + \delta_{b_{ij},1}\, s_i s_j) + \delta_{g_{ij},0}\, \delta_{b_{ij},0}\,(1-v)\big) \\
&= \sum_g \sum_b \sum_s \prod_{\substack{<ij> \\ g_{ij}=1}} v\,(\delta_{b_{ij},0} + \delta_{b_{ij},1}\, s_i s_j) \prod_{\substack{<ij> \\ g_{ij}=0}} \delta_{b_{ij},0}\,(1-v) \\
&= \sum_g v^{\# g_{ij}=1}\,(1-v)^{\# g_{ij}=0} \sum_b \prod_{\substack{<ij> \\ g_{ij}=0 \wedge b_{ij}=0}} 1 \sum_s \prod_{\substack{<ij> \\ g_{ij}=1 \wedge b_{ij}=1}} (s_i\, s_j) \qquad (4.19)
\end{aligned}$$

Remembering Eq. (2.18), we find for the sum over the spins

$$\sum_s \prod_{\substack{<ij> \\ g_{ij}=1 \wedge b_{ij}=1}} (s_i\, s_j) = \begin{cases} 2^N & \text{if } b \text{ forms closed loops,} \\ 0 & \text{otherwise.} \end{cases} \qquad (4.20)$$

We finally get

$$Z \sim 2^N \sum_g v^{\# g_{ij}=1} (1-v)^{\# g_{ij}=0} \sum_{\substack{b,\text{ with}\\ \text{constraint}}} 1^{\# b_{ij}=1}. \tag{4.21}$$

From this result we see that all B-configurations, which fulfill the constraint, are chosen with **equal probability**.

4.2.2 Relation to Swendsen-Wang bonds

As we have mentioned in the introduction of this chapter, the variables in the HT-representation which are assigned to the edges of the lattice merely seem to be a smart way of handling the spin-products appearing in the high-temperature series expansion. However, we have found the *remarkable* result that the elements of the graph G in the HT-representation are identical with the Swendsen-Wang bonds (see Sec. 3.2) which are *physical* bonds. Remember that G does *not* have to fulfill the HT-constraint which is only imposed on the elements of B. The astonishing relation between G and the Swendsen-Wang bonds is derived as follows:

We start with W_{ij}, which is given by

$$W_{ij}(b_{ij}, g_{ij}, s_i, s_j) = \delta_{g_{ij},1}\, v(\delta_{b_{ij},0} + \delta_{b_{ij},1}\, s_i s_j) + \delta_{g_{ij},0}\, \delta_{b_{ij},0}\, (1-v) \tag{4.22}$$

The marginalization over the variables b_{ij} yields

$$\begin{aligned} W_{ij}(g_{ij}, s_i, s_j) &= \delta_{g_{ij},1}\, (1 + s_i s_j)\, v + \delta_{g_{ij},0}\, (1-v) \\ &= \delta_{g_{ij},1}\, (\delta_{s_i s_j} 2v) + \delta_{g_{ij},0}\, (1-v) \end{aligned} \tag{4.23}$$

We compare this result with Eq. (3.7) and see that $1-v$ correponds to p and $2v$ corresponds to $1-p$:

$$\frac{1-v}{2\,v} = \frac{1}{2}\,\frac{\cosh\beta - \sinh\beta}{\sinh\beta} = \frac{e^{-2\beta}}{1-e^{-2\beta}} = \frac{p}{1-p}. \tag{4.24}$$

Thus Eq. (4.23) is a constant multiple of the *Fortuin-Kasteleyn* representation in Eq. (3.6). This result shows that the graph G is equivalent to the Swendsen-Wang clusters. This important result allows us to measure quantities like the magnetization, susceptibility and spatial correlation functions by using the **improved estimators** for the Swendsen-Wang algorithm. The rules for measuring these quantities were listed in Sec. 3.2.3. Note that we merely need $1-v = a\,p$ and $2\,v = a\,(1-p)$, where a is constant. Further, by observing (4.24) and (3.8) we find a relation between the partition functions of the Swendsen-Wang and the HT-cluster algorithm

$$Z^{HT} = a^{N\,d}\, Z^{SW}. \tag{4.25}$$

4.2.3 Monte Carlo algorithm

The algorithm is implemented as follows:

1. We start with a lattice where the dynamic variables consist of the bonds g_{ij}[4]. We are allowed to start with any bond-configuration we like. Note, that this bond configuration does not have to satisfy the HT-constraint (=only closed loops allowed). We will call this bond-pattern the **graph of deletions**[5].

2. Our next task is to map the graph of deletions G to an allowed bond-configuration B in the HT-representation in such a way, that the resulting configurations consist of closed loops only. We call this operation the *Ben-Av-construction*. The construction is described in detail below.

3. After having applied the Ben-Av construction, we are left with a valid configuration B in the HT-representation, which consists of closed loops only. Now we are able to measure the energy, but not the quantities magnetization, susceptibility and spatial correlation functions. For measuring the energy we only need to know the number of bonds on the lattice and can use Eq. (4.3).

4. In order to create a new graph of deletions G', we need a rule which tells us the probability for setting bonds on the lattice. A look at Eq. (4.16) and Eq. (4.18) tells us, that the probability of setting a bond depends on whether a bond in the valid configuration just created in 3 is already set or not. The probabilities p for setting a bond are given as follows:

   - $p = \tanh \beta$, if no bond set in the old configuration.
   - $p = 1$, if a bond is set in the old configuration.

   After having made a decision if to set or not to set a bond for all bonds on the lattice, we are left with a new graph of deletions G'. Now we can measure the magnetization, susceptibility and spatial correlation functions. We can do this by using improved estimators. Finally we can jump to 2.

The Ben-Av-construction

In order to create a valid configuration by reducing the graph of deletions the following algorithm is used. Note, that a graph of deletions G can be reduced to **several** different allowed configurations B. The Ben-Av-construction satisifies the demand that all valid configurations are created with equal probability (see Eq. (4.19)). We start with a graph of deletions. It normally consists of several

[4] We will use the term "bond" both for the elements g_{ij} of G and for the elements b_{ij} of B.
[5] With this terminology we follow *Ben-Av et. al.*

disconnected parts (clusters) and one of those could look like the bond-pattern depicted in Fig. 4.6 (the meaning of different bond colors and line styles is explained below). The following rules then have to be applied to all clusters making up the graph of deletions.

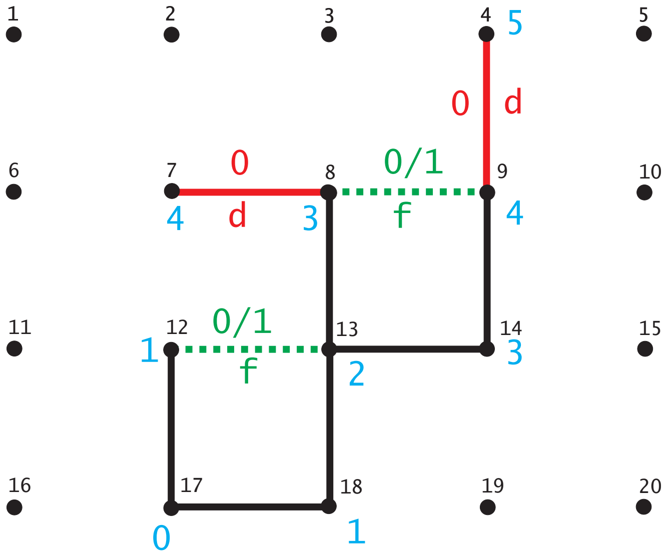


Figure 4.6: A disconnected part (cluster) of the *graph of deletions* containing *free links* (green) and *dangling bonds* (red). The randomly selected root is marked with 0 and the different levels of the tree are labeled from 0 to 5.

1. First we need to define a randomly selected site of the cluster as root of a tree. This site is assigned the level 0. Starting from the root we visit all sites of the cluster and assign numbers (levels) to them. One possible way of realizing this is assigning the minimum number of bonds separating the root and site to this site. We have now constructed a **spanning tree**.

2. Whenever a bond of the graph of deletions closes a loop, we will mark this bond as **free link**. The notion free link stems from the fact, that we can freely choose to set or not to set a bond at this free link while reducing the graph of deletions in order to obtain valid configurations. It turns out that the number of different HT-configurations which can be obtained from a single cluster equals $2^{\#\text{free links}}$.

3. Further, we will normally find some dangling ends. These bonds are called **dangling bonds**.

4. In order to obtain valid configurations from a cluster of the graph of deletions we procede as folows. First, all dangling bonds are assigned the value

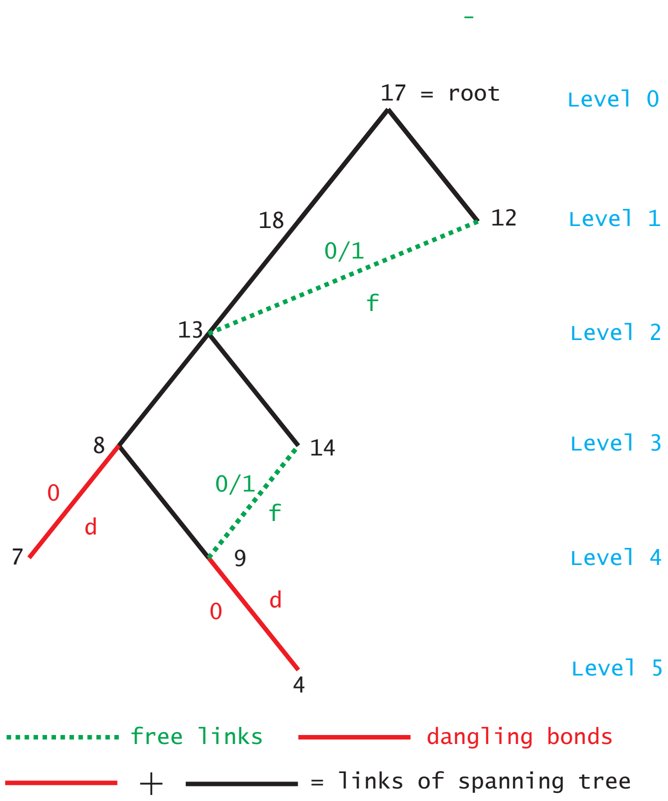


Figure 4.7: The corresponding spanningtree of the cluster in Fig. 4.6. The spanning tree is made up of black and red lines.

0. This means that all dangling bonds are erased. We find this rule to be reasonable because dangling bonds definitely violate the HT-constraint at one of their ends (# of set bonds leading to one of their ends = 1 = *odd*). Further, we assign all free links of the cluster (=disconnected part of the graph of deletions) *independent* random values: 0 or 1. Here 1 means that a bond is set and 0 means that no bond is set, respectively. The number of possible free links-patterns is therefore $2^{\#\text{free links}}$, which explains, why a single graph of deletions can be reduced to $2^{\#\text{free links}}$ different valid bond configurations.

5. Next we apply the HT-constraint (this means, that the number of bonds with value 1 pointing to them must be *even*) at all sites starting from the highest level. By applying the constraint, we see that at each of these sites the bond connecting the site to a site of next lower level is forced to have a particular value. Now we can apply the HT-constraint to all sites of the next lower level and again all bonds connecting these sites to the next lower level are forced to take a definite value: 0 or 1. This procedure can be continued down to the sites of level 1. Contradictions and ambiguities cannot arise, because applying the HT-constraint at sites of level 1 or higher always forces one and only one bond[6].

An example

To illuminate the algorithm described above, we will study an example. Let us take the cluster depicted in Fig. 4.6. Say, we have randomly assigned both free links the value 1. Now we go to the highest level 5 (site #4) and apply the HT-constraint.

The bond attached to site 4 is a dangling bond and hence, according to our rules, automatically assigned the value 0. Next we go to sites of level 4 (sites no. #7 and #9). The bond attached to 7 is a dangling bond and therefore vanishes. The bond linking site 9 and site 14 is assigned the value 1 because the free link between 8 and 9 was given the value 1 and in order to satisfy the HT-constraint we need 2 bonds attached to site 9. For practice, the reader can now easily finish the Ben-Av-construction down to the root. The resulting valid configuration is a double-loop and is depicted in Fig. 4.8 As mentioned above, the resulting configurations depend on the random choice of the values of free links (0 or 1 with equal probability). If we had chosen the lower free link to have the value 0 and the upper free link to have the value 1, we had ended with the configuration in Fig. 4.9 Similarly we obtain the configuration in Fig. 4.10 by setting the lower free link to 1 and the upper free link to 0.

[6]In [7] a proof can be found that with this construction the HT-constraint holds at the root too.

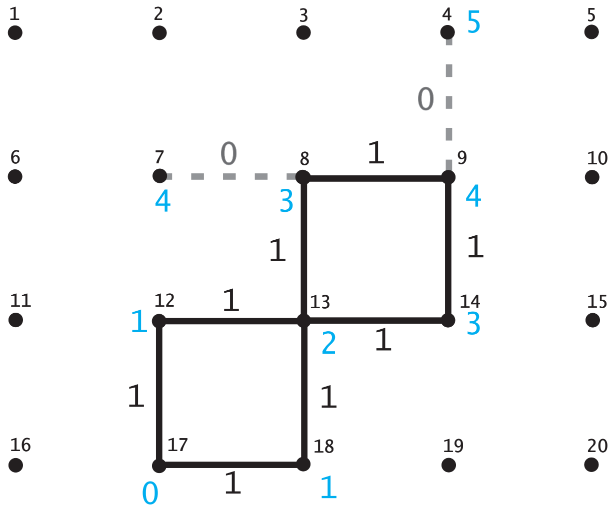


Figure 4.8: The resulting valid configuration when both free links are assigned the value 1.

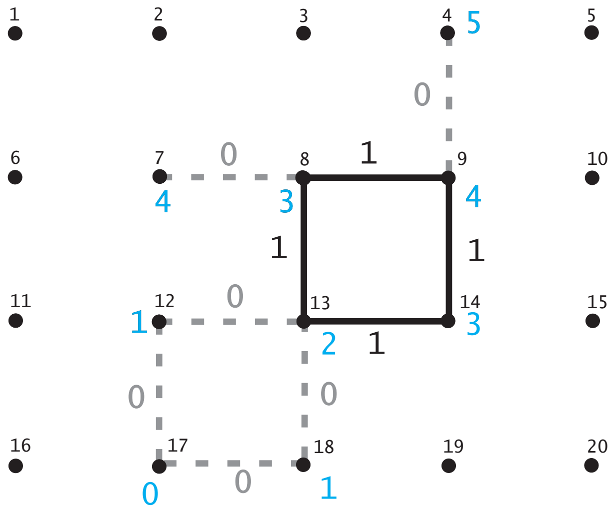


Figure 4.9: The resulting valid configuration when the free link between 8 and 9 is set to 1 and between 12 and 13 set to 0.

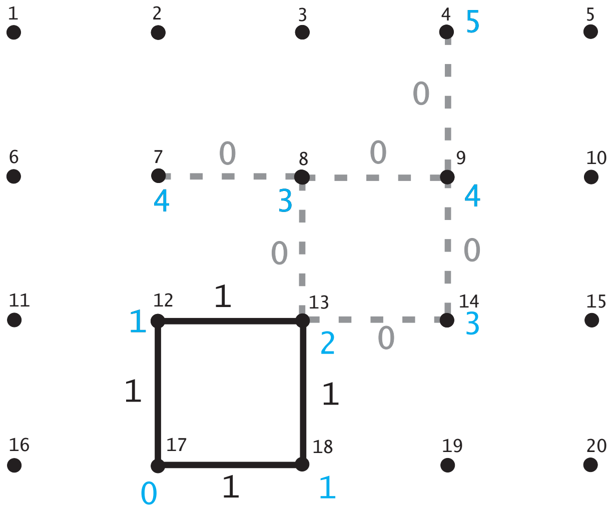


Figure 4.10: The resulting valid configuration when the free link between 8 and 9 is set to 0 and between 12 and 13 set to 1.

If we set both free links to 0, the resulting configuration consists of no bonds at all (Fig. 4.11).

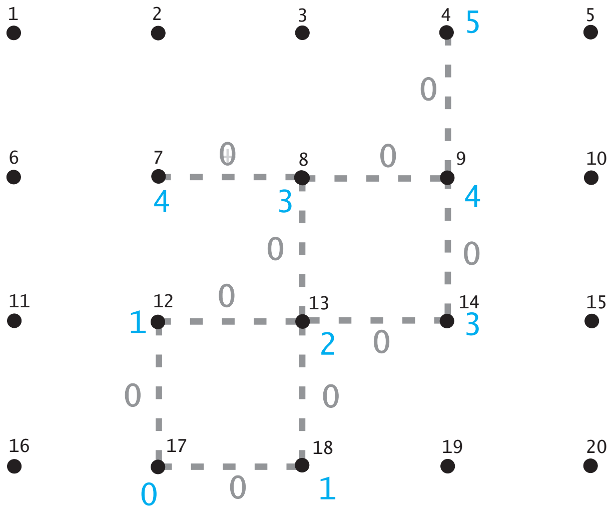


Figure 4.11: The resulting valid, empty configuration when setting both free links to 0.

Chapter 5

Results

We used the HT-cluster algorithm to simulate an Ising system in 2, 3 and 4 dimensions[1]. In all simulations the lattice was taken to be simple cubic and to have periodic boundary conditions. The chosen boundary conditions allow the loops to wind around the lattice. An illustration for this is the picture on the cover of this thesis. In our simulations we measured the energy, magnetization, $\sum_{ij} s_i s_j$ and the spatial correlation function at distance $L/2$. The expectation values of all these quantities agreed within statistical errors with results obtained from Swendsen-Wang simulations. Further, an autocorrelation analysis as described in the appendix was performed. From these results the dynamic critical exponent z for the HT-cluster algorithm was extracted via a fit of the exponential autocorrelation time τ_{exp} vs. the linear dimension L of the lattice. The values for z are listed in Tab. 5.1.

dimension d	z
2	0.31 ± 0.01
3	0.67 ± 0.03
4	1.07 ± 0.05

Table 5.1: Dynamic critical exponents z for the HT-cluster algorithm. Values obtained with own autocorrelation analysis.

In the following sections we list the dependence of τ_{exp} on L for 2d, 3d and 4d. The graphs contain the values for all 4 measured quantities. The energy is depicted in red, magnetization in green, $\sum_{ij} s_i s_j$ in blue and $\Gamma(L/2)$ in magenta. Note that for reasons of clarity only the errorbars of the energy are printed. The values on the log-scaled τ_{exp}-axis are the ones for the energy. Notice that although we have used a *logarithmic* scale, we have plotted the *real* values of τ_{exp} in order to facilitate the comparison with results of other algorithms. Since the values of

[1] The program can be downloaded at [1].

τ_{exp} for the measured quantities are very similar, it may be difficult to distinguish the different quantities by their colors. For the cases $2d$ and $3d$ we have depicted both the HT-cluster and the Swendsen-Wang results in one picture in order to compare the algorithms. Because of the similarity of the τ_{exp} values of the HT-cluster and the Swendsen-Wang algorithm in the $2d$ case, we depict the HT-cluster results only. The values of τ_{exp} were obtained via a fit of the autocorrelation functions. We include these fits for single, medium and large lattices and for all dimensions in the appendix. When doing these fits one has to be especially careful about choosing the correct fit ranges and fitting autocorrelation functions with sufficient small errors. Especially autocorrelations of large lattices have to be treated carefully since the number of measurements for these large systems is naturally limited by computer power and therefore the autocorrelations show large errors. An example for a misfit due to an autocorrelation function with too large errors is presented in Sec. A.2.3.

5.1 2d results

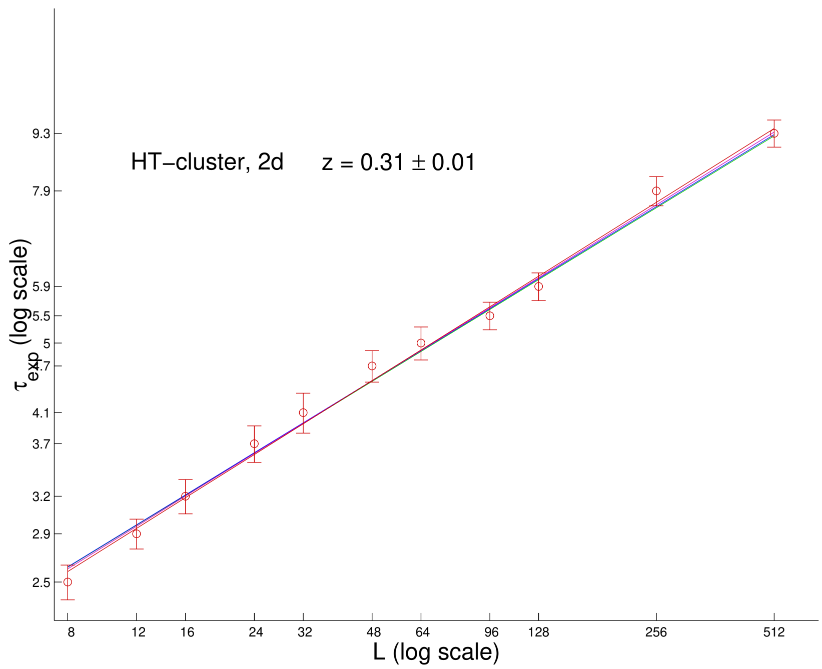


Figure 5.1: Autocorrelation analysis of $2d$ HT-cluster Monte Carlo data. τ_{exp} vs. linear dimension L at β_c.

5.2 3d results

In the $3d$ case we find a slightly higher value of $z \approx 0.65$ compared with the Swendsen-Wang value of $z \approx 0.6$. However, the values of τ_{exp} are significantly higher for the HT-cluster algorithm compared to Swendsen-Wang. This behaviour is depicted in Fig. 5.2.

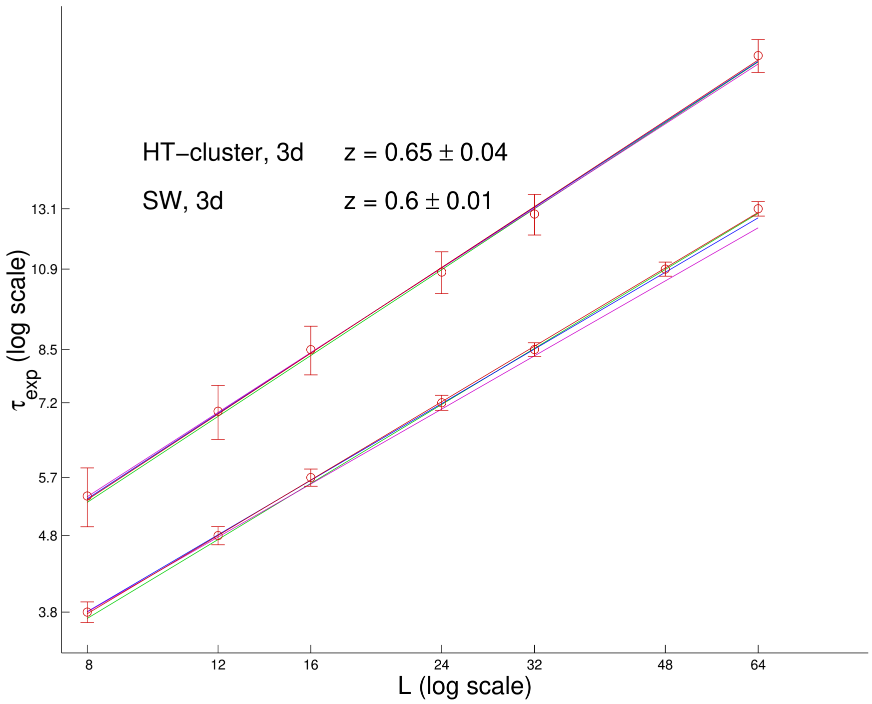


Figure 5.2: Autocorrelation analysis of $3d$ HT-cluster Monte Carlo data and Swendsen-Wang Monte Carlo data. τ_{exp} vs. linear dimension L at β_c.

In order to illustrate the differences in τ_{exp} and the autocorrelation function $C(t)$ between the HT-cluster and Swendsen-Wang algorithm, we show the graphs of the autocorrelation analysis for the energy (Fig. 5.3) and the magnetization (Fig. 5.4) in a 64^3 lattice.

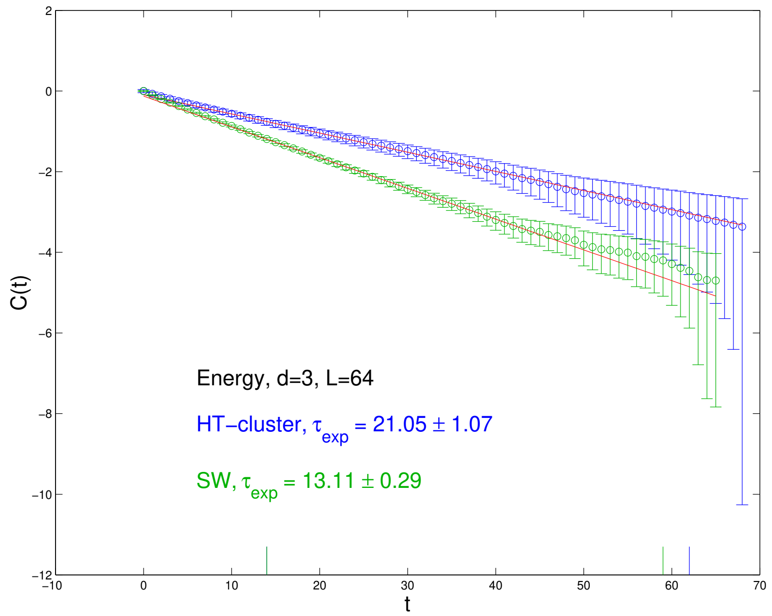


Figure 5.3: Comparison of HT-cluster and Swendsen-Wang autocorrelations for the energy in a 64^3 lattice. Fits plotted in red, fit ranges indicated on t-axis.

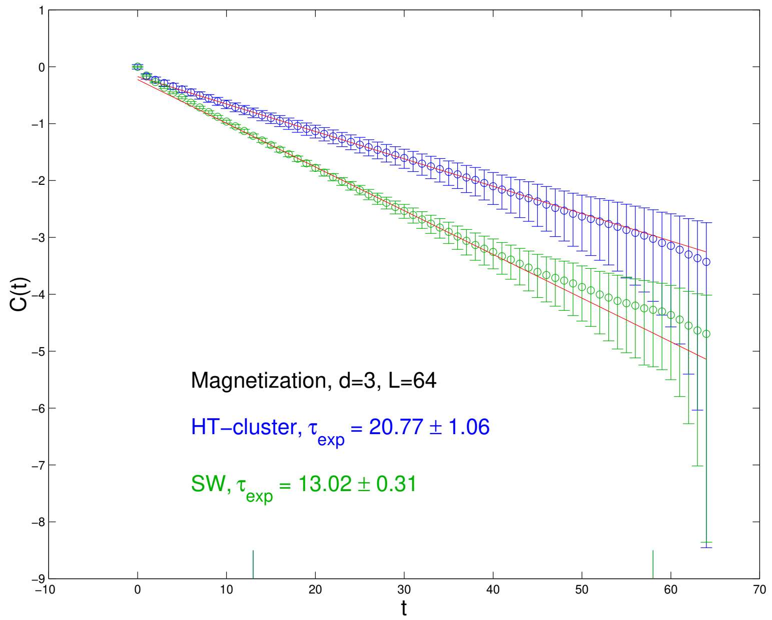


Figure 5.4: Comparison of HT-cluster and Swendsen-Wang autocorrelations for the magnetization in a 64^3 lattice.

5.3 4d results

In the $4d$ case we find a significant difference between the z-values of the HT-cluster and the Swendsen-Wang algorithm. The HT-cluster algorithm yields $z \approx 1.07$ whereas the Swendsen-Wang result is $z \approx 0.81$. The results are depicted in Fig. 5.5.

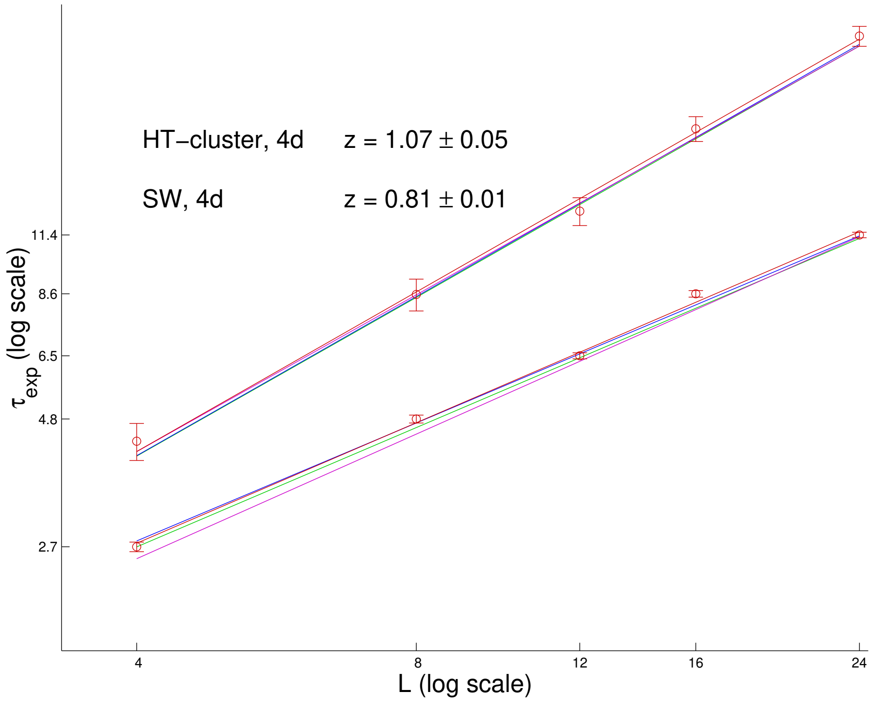


Figure 5.5: Autocorrelation analysis of $4d$ HT-cluster Monte Carlo data and Swendsen-Wang Monte Carlo data. τ_{exp} vs. linear dimension L at β_c.

5.4 Autocorrelations vs. temperature

It is interesting to know the dependence of τ_{exp} on the temperature T and therefore we measured this dependence for an 8×8 lattice. The results are depicted in Fig. 5.6. We see that τ_{exp} does not reach its maximum exactly at T_c but at a slightly higher temperature. This is a *finite size effect*.

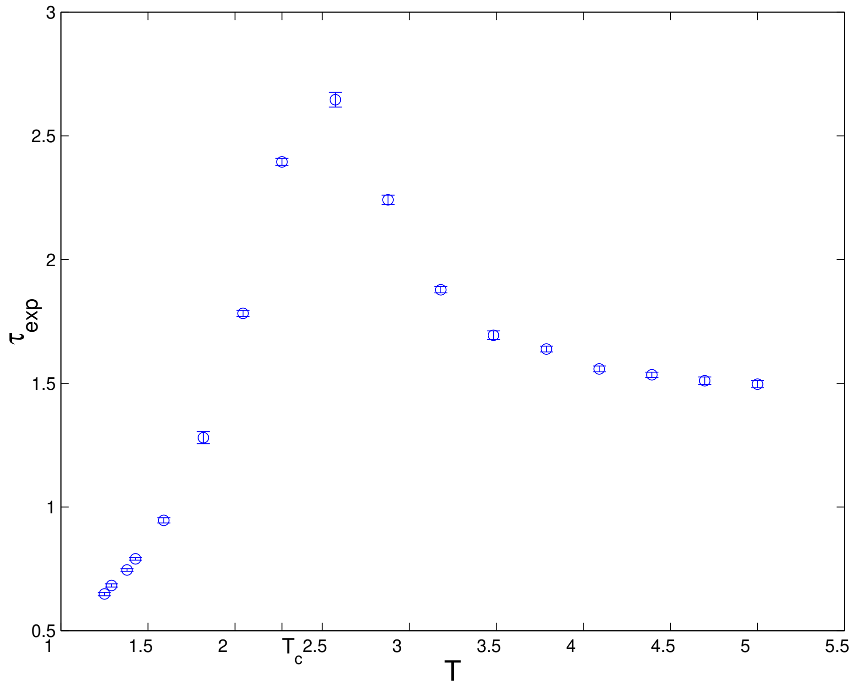


Figure 5.6: Exponential autocorrelation time τ_{exp} vs. temperature T for the HT-cluster algorithm (8×8 lattice)

Chapter 6

Conclusion

In this thesis we have developed and analyzed a new cluster algorithm for the Ising model. We started with the high-temperature expansion of the model's partition function. We analyzed Z in more detail, visualized spin-products by bonds and found that only those configurations consisting of closed loops only yield a contribution to Z.

With this knowledge we first constructed a Monte Carlo algorithm which creates these *allowed configurations* locally. We found a critical dynamic exponent of $z \approx 2$ and we concluded that a global update algorithm is necessary to overcome critical slowing down.

To achieve this we constructed a Monte Carlo algorithm which creates and updates the allowed configurations in a global manner. We reviewed existing Monte Carlo algorithms and compared them to the new one in terms of autocorrelation behaviour. By doing this, we found that in 2 dimensions the new algorithm shows the same value for the dynamic critical exponent as the Swendsen-Wang algorithm. In 3 and 4 dimensions we could observe a somewhat higher value for z compared to Swendsen-Wang.

We think that the algorithm could be valuable for observing systems where Swendsen-Wang fails to succeed, e.g. frustrated systems. Further, it would be interesting to study the algorithm when applying finite magnetic fields to the system.

Appendix A

Appendix

A.1 Breadth-First cluster search algorithm

When using a cluster algorithm it is necessary to have an efficient method which performs the detection of clusters on the lattice. Such an algorithm is presented here. It is called **Breadth-First search algorithm**. The algorithm explores all sites adjacent to the current one and all sites connected to the current site are stored in a list. Then all the adjacent sites of the sites in the list are visited and if connected added to the list. This procedure is performed until the whole cluster extending from the current site is detected. Hence the name *Breadth-First*. The single steps of the algorithm are described below:

1. Go to lattice site 1 and start the list with value 1.

2. Look at all neighbours and check if they are connected to the current site via a bond. If this is the case and if they are not yet in a cluster, add the site to the list.

3. By visiting all sites in the list and performing the same task as in 2 the whole cluster is finally detected. During the detection of the clusters the list grows dynamically. All sites in the list are assigned the same number which is different for every cluster. After a whole cluster has been detected, the list is newly created.

4. After a single cluster is detected, go to the next site of the lattice and check, if this site is already contained in any cluster. If this is the case, then go to the next site of the lattice. If this is not the case, start a new cluster at this site and continue the algorithm at 2.

5. Continue to use the rules described above until the whole lattice is analyzed. Finally, every lattice site has been assigned a number which indicates the cluster it belongs to.

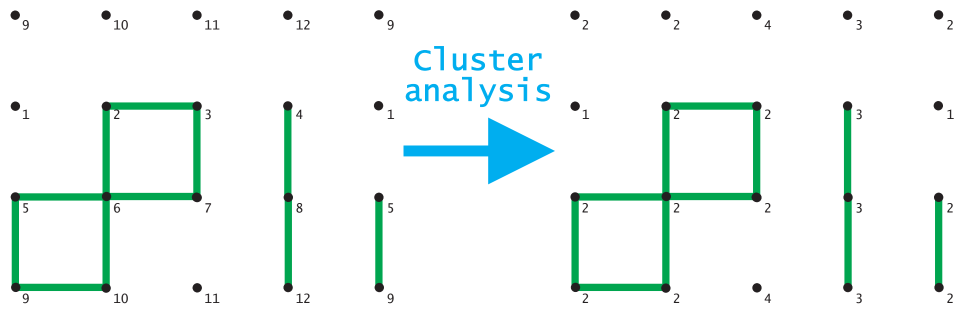


Figure A.1: Example of cluster analysis. Note that in the left panel the site numbers are displayed whereas in the right panel the cluster numbers are displayed.

In order to facilitate the understanding of the algorithm, we will take a look at the following example.

1. The algorithm starts at site 1. This site is assigned to cluster #1. All 4 neighbours are visited, but none is connected to 1. Therefore cluster #1 is finished and we move on to site 2, where cluster #2 is started.

2. We find, that 3 and 6 are connected to 2 and add them to the list (which consists of 2,3 and 6 now). Then we move on to the next site in the list which is 3. We find that only 7 is connected to 3 and add 7 to the list (list=(2,3,6,7)). We move to 6. We see, that 2 and 7 are already a part of cluster #2. Further, 10 and 5 are connected to 6, which are added to the list (list=(2,3,6,7,10,5)). We move on to the next element in the list, which is 7. There we find out, that all connected neighbours (3 and 6) are already a part of cluster #2. Therefore we move to 10 and add 9 to the list (list=(2,3,6,7,10,5,9)). Finally we move to 5 and 9 but find out that all connected neighbours are already a part of cluster #2. Now we have arrived at the end of the list and this cluster is finished.

3. We move on to site 4, start a new cluster (#3) and add 8 to cluster #3 and to the list (list=(4,8)). We move to 8 and add 12 to cluster #3 and to the list (list=(4,8,12)). We see that all connected neighbours of 12 are already a part of a cluster and are finished with cluster #3.

4. We now move to site 5. We find out that it already belongs to cluster #2 and therefore move on to site 6. Here we are in the same situation as at site 5 and we find out that sites 6, 7, 8, 9 and 10 are already part of a cluster either.

5. Site 11 is not part of a cluster and we start a new cluster (#4) there. This cluster consists of 1 site only and we go to site 12. This is already a part of

a cluster and we are finally finished with the cluster analysis of the whole lattice.

The following pseudocode shows the implementation of the algorithm. In the pseudocode, `v` denotes the number of sites, `b(i,j)` contains the bond-

```
allocate(cluster(1:v), list(1:v))
current = 1; endp = 1;
ncluster = 1; cluster = 0;
for i = 1 to v do
  if cluster(i) .ne. 0 then
    cycle
  end if
  current ⇐ 1
  endp ⇐ 1
  cluster(i) ⇐ ncluster
  list(1) ⇐ i
  while current .le. endp do
    for j1 = 1 to 2*d do
      i1 = list(current)
      if b(i1, j1) .eq. -1 then
        if cluster(nn(i1, j1)) .eq. 0 then
          endp ⇐ endp+1
          list(endp) ⇐ nn(i1, j1)
          cluster(nn(i1, j1)) ⇐ ncluster
        end if
      end if
    end for
    current ⇐ current+1
  end while
  ncluster ⇐ ncluster+1
end for
ncluster ⇐ ncluster-1
```

values and `nn(i,j)` contains all next neighbours of site i. Finally, the array called `cluster` contains the number of the cluster each site is belonging to.

A.2 Autocorrelation analysis of MC data

In order to obtain the values of the autocorrelation-function $C(t)$, a fast fourier analysis (FFT) of the Monte Carlo time series of the physical quantities (e.g. energy, magnetization, $\sum_{ij} s_i s_j$, spatial correlation function) was performed. The errors of $C(t)$ were calculated by doing a *jackknife analysis* of the Monte Carlo data. This method is explained below, for further details cf. [11], [12] and [13].

A.2.1 Calculating autocorrelations with FFT

When calculating the autocorrelations of a time series, we need to evaluate sums of the kind below[1]

$$\mathcal{C}(Q,Q)_t \equiv \sum_{i=1}^{n_{sum}} Q(i)\, Q(i+t) \tag{A.1}$$

A fast way of evaluating this sum is based on the *discrete correlation theorem*

$$\mathcal{F}\left(\mathcal{C}(Q,Q)_t\right) = \mathcal{F}\left(Q(1\ldots n_{sum})\right) \cdot \mathcal{F}^*\left(Q(1\ldots n_{sum}+t_{max})\right) \tag{A.2}$$

Here, $\mathcal{F}$ denotes the discrete fourier-transformation and the asterix a complex conjugation. t_{max} is the largest lag for which we want to calculate an autocorrelation. Note that in order to perform an FFT we need to extend the time-series Q by padding it with zeros until it has a length n_{FFT} equal to a power of 2. Thus, to obtain $\mathcal{C}(Q,Q)_t$ we need to

1. Fast fourier-transform $Q(1\ldots n_{sum})$ and $Q(1\ldots n_{sum}+t_{max})$.

2. Multiply $\mathcal{F}\left(Q(1\ldots n_{sum})\right)$ with $\mathcal{F}^*\left(Q(1\ldots n_{sum}+t_{max})\right)$.

3. Now perform an inverse FFT of this product and divide it by n_{FFT}. The result gives the desired values for $\mathcal{C}(Q,Q)_t$ (Note that the first entry in the resulting array equals $\mathcal{C}(Q,Q)_0$).

A program accomplishing the task described above can be downloaded at [1].

[1] We follow closely [20], p.538

A.2.2 Computing the errors of the autocorrelation-function via jackknife

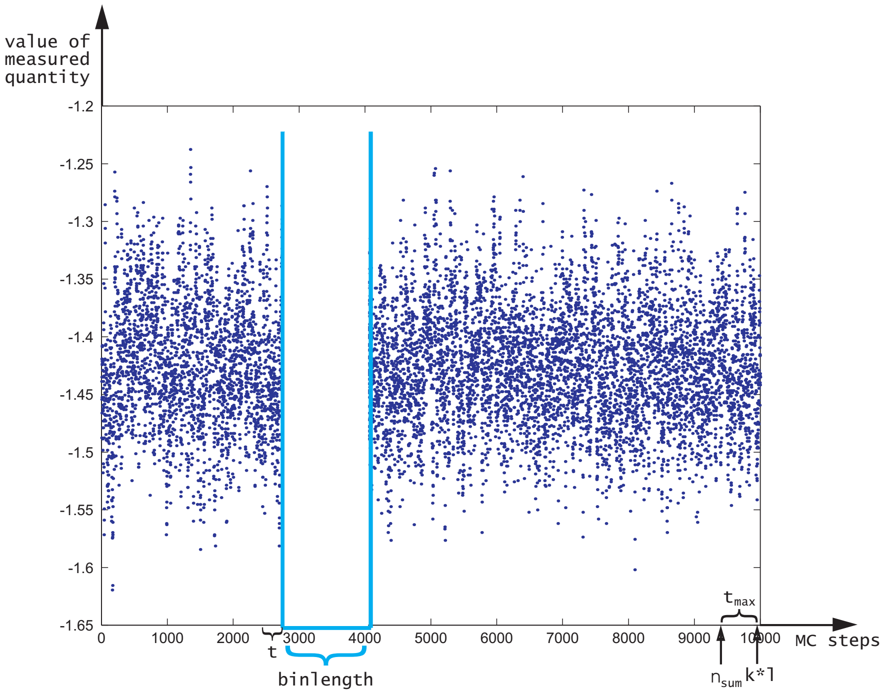


Figure A.2: Illustration of *jackknife* analysis.

In Eq. (1.14) we defined the (normalized) autocorrelation function $C(t)$. We denote the values of the Monte Carlo time-series with $Q(i)$ and its length with l_{MC}. Assume that we want to compute $C(t)$ up to lags $t = t_{max}$. This means that we can sum the product $Q(i)\,Q(i+t)$ only up to $n_{sum} = l_{MC} - t_{max}$. The unnormalized autocorrelation-function is then given by

$$C(t) = \frac{1}{n_{sum}} \sum_{i}^{n_{sum}} Q(i)\,Q(i+t) - \frac{1}{n_{sum}} \sum_{i}^{n_{sum}} Q(i)\,\frac{1}{n_{sum}} \sum_{i}^{n_{sum}} Q(i+t) \qquad \text{(A.3)}$$

The jackknife method is now applied in the following way:

1. First we devide the Monte Carlo data into k bins of length l. We determine l in such a way, that

   $$k \cdot l \leq l_{MC}.$$

   In case the length of data is not an integer multiple of l, the data is simply cut off.

2. The first autocorrelation-analysis is performed with all the data. The result is stored in $C^{(0)}(t)$.

3. Then, for $j = 1, \dots, k$ we perform the autocorrelation-analysis for all the data but we set the data in the j^{th} bin to 0. This analysis yields values $C^{(j)}(t)$. Note that we have to be *careful* with the number of terms n_s contributing to each of the sums in Eq. (A.3). By looking at Fig. A.2 we find for n_s:

$$n_s = \begin{cases} \frac{1}{n_{sum}-l} & \text{if } j = 1 \text{ or } j = k \\ \frac{1}{n_{sum}-l-t} & \text{otherwise} \end{cases} \tag{A.4}$$

All sums in Eq. (A.3) now have to be divided by n_s instead of n_{sum}.

4. The overall result for $C(t)$ is finally given by

$$C(t) = C^{(0)}(t) - B, \text{ where} \tag{A.5}$$

$$\mathrm{B} = (k-1)\left(C^{av}(t) - C^{(0)}(t)\right) \tag{A.6}$$

$$C^{av}(t) = \frac{1}{k}\sum_{j=1}^{k} C^{(j)}(t) \tag{A.7}$$

The statistical error of $C(t)$ is now given by

$$\Delta C(t) = (k-1)^{1/2} \left(\frac{1}{k}\sum_{j=1}^{k} (C^{(j)}(t))^2 - (C^{av}(t))^2 \right)^{1/2} \tag{A.8}$$

In practice it is only reasonable to compute the AC function up to a certain distance t because for larger values there is not enough statistical information and the errors of $C(t)$ become very large. It turned out that in all cases a computation up to values $t \approx 100$ was sufficient.

A.2.3 Fitting the autocorrelation function

Theory [14], describes the typical behaviour of the autocorrelation function $C(t)$:

$$C(t) = \sum_j c_j \; e^{-\frac{t}{\tau_j}} \tag{A.9}$$

We define τ_{exp} to be $\tau_{exp} = \max_j(\tau_j)$. For sufficiently large lags t we use the following simple model function $f(t)$ to fit the data $C(t) \pm \Delta C(t)$:

$$f(t) = c \; e^{-\frac{t}{\tau_{exp}}} \tag{A.10}$$

The fitting was done in `MATLAB`. A problem arose, because of the program's insufficient capabilities to fit nonlinear data *plus* its errors and return the estimated best fit parameters and their errors. Therefore a fit routine which is able to do the task was developed. The program can be downloaded at [1]. It is based on the theory listed below.

1. Define the χ^2 function:

$$\chi^2 = \sum_i \frac{(f_i(c, \tau_{exp}, t_i) - d_i)^2}{{\sigma_i}^2} \tag{A.11}$$

2. This function is minimized with respect to the fit parameters c and τ_{exp}. Here t_i denotes the different lags, d_i the values of $C(t_i)$ and σ_i the values of $\Delta C(t_i)$. This is done via the routine `lsqnonlin` in `MATLAB` (Optimization Toolbox).

3. To obtain the errors of the fit parameters we have to calculate the following:

$$H = \frac{1}{2} \frac{\partial^2 \chi^2}{\partial c \, \partial \tau_{exp}} \tag{A.12}$$

This matrix is called *Hesse matrix*. It has to be inverted and evaluated at the minimum position of χ^2. Finally, the squares of the errors of c and τ_{exp} can be found in the main diagonal of this inverted matrix:

$$H_{ij}^{-1} \,|_{\chi^2_{min}} = \begin{pmatrix} \sigma_c^2 & \langle \Delta c \Delta \tau_{exp} \rangle \\ \langle \Delta c \Delta \tau_{exp} \rangle & \sigma_{\tau_{exp}}^2 \end{pmatrix} \tag{A.13}$$

An example for a misfit

When doing fits one has to be especially vigilant about choosing the correct fit ranges and fitting autocorrelation functions with sufficient small errors. When measuring large lattices we are naturally limited by computer power and therefore limited in the number of measurements. Because of this reason autocorrelations of large lattices show large errors and therefore have to be treated carefully. We simulated a 256×256 lattice and first ($N_{meas} \approx 15000$) found a value of $\tau_{exp} \approx 6.74$, which did not match with a power-law fit of the data (green plot in Fig. A.3). However, by increasing the number of measurements to $N_{meas} \approx 150000$ we were able to reduce the errorbars of the autocorrelation function and found $\tau_{exp} \approx 7.87$ (blue plot in Fig. A.3).

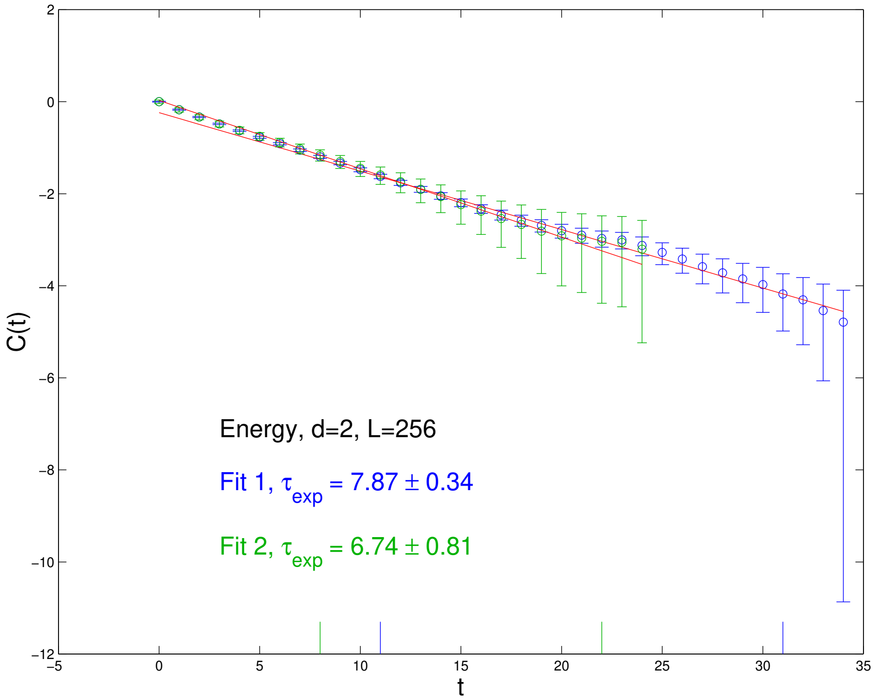
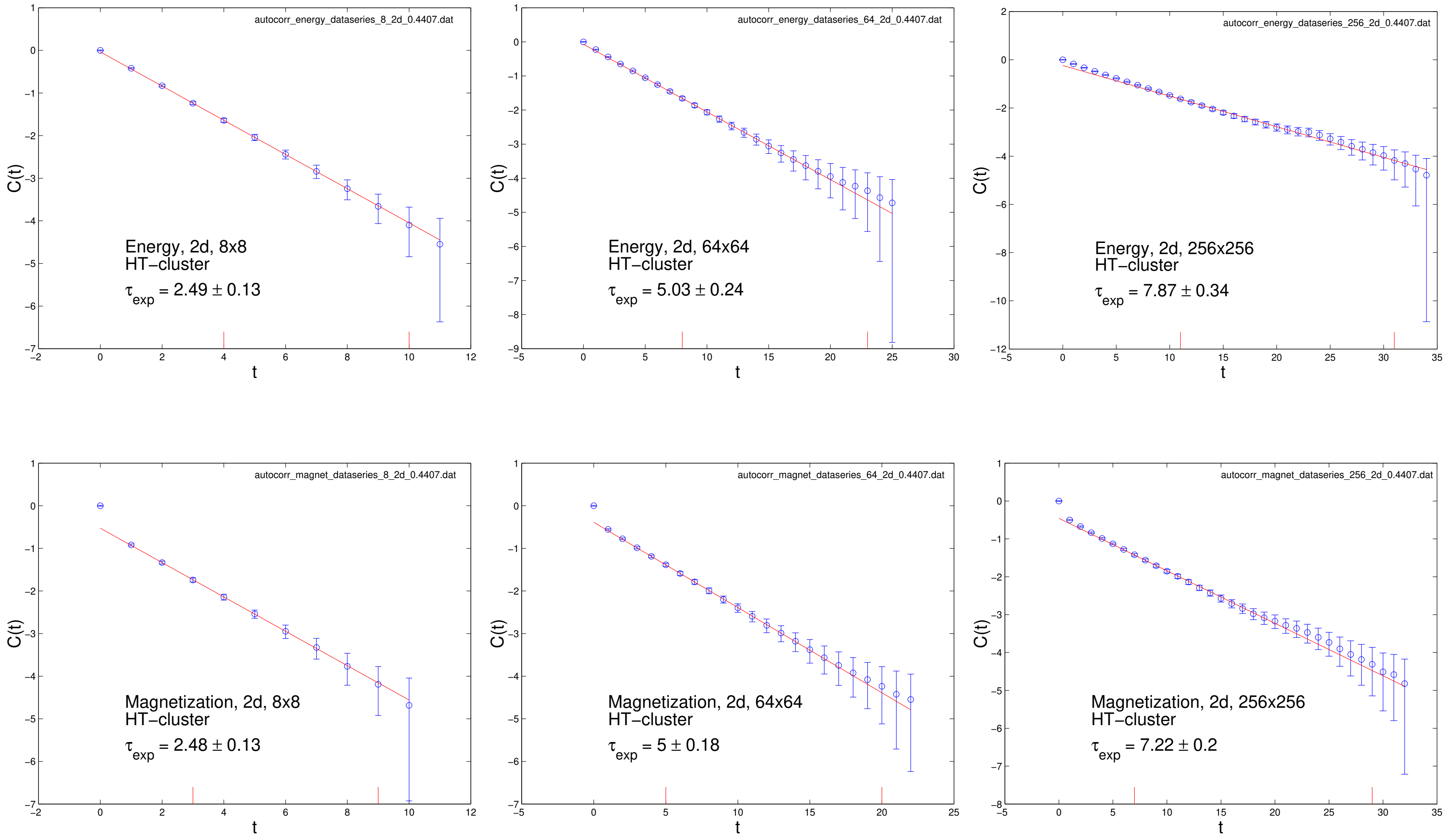
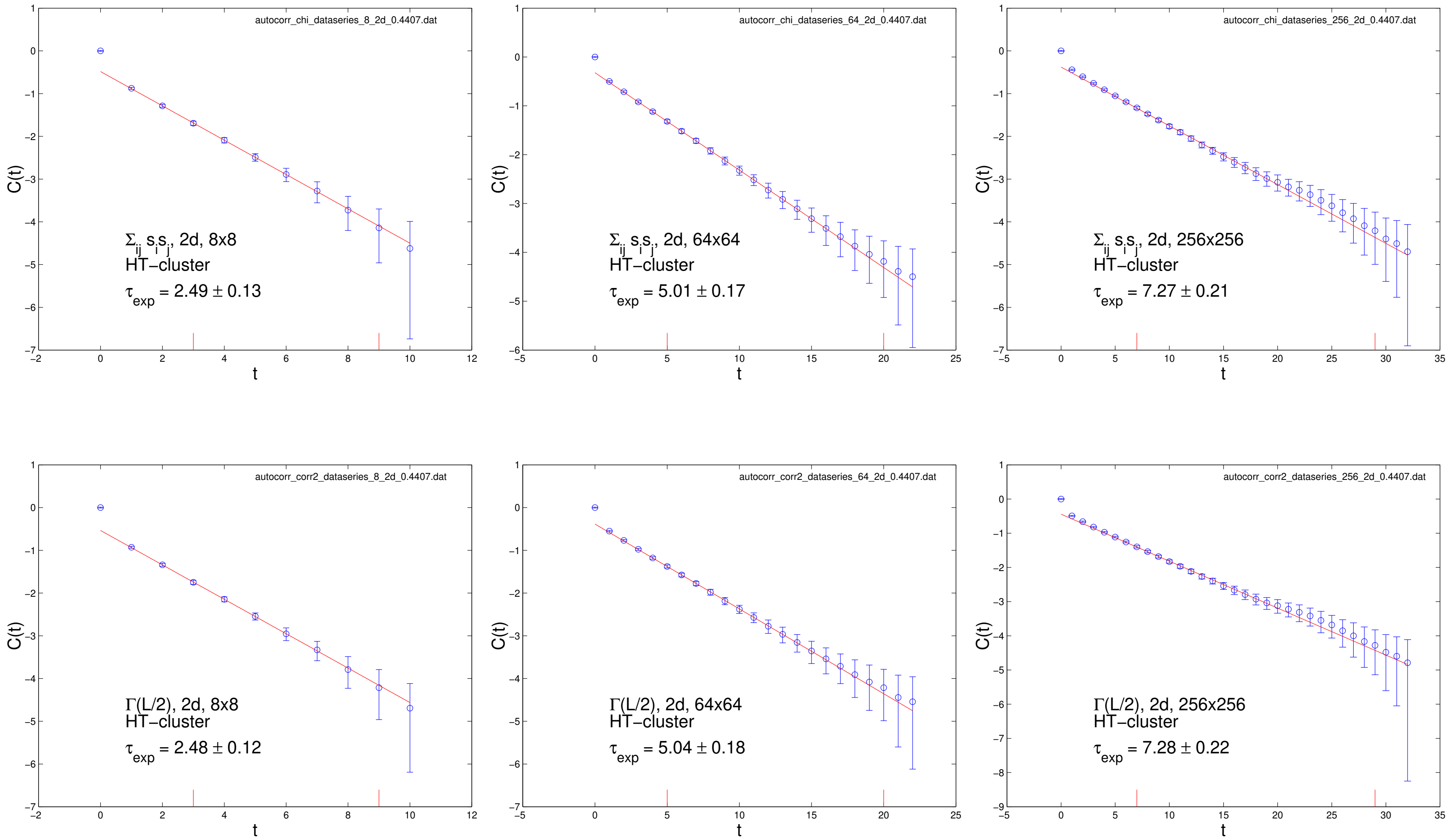


Figure A.3: Example for a misfit. Fit ranges indicated on t-axis.

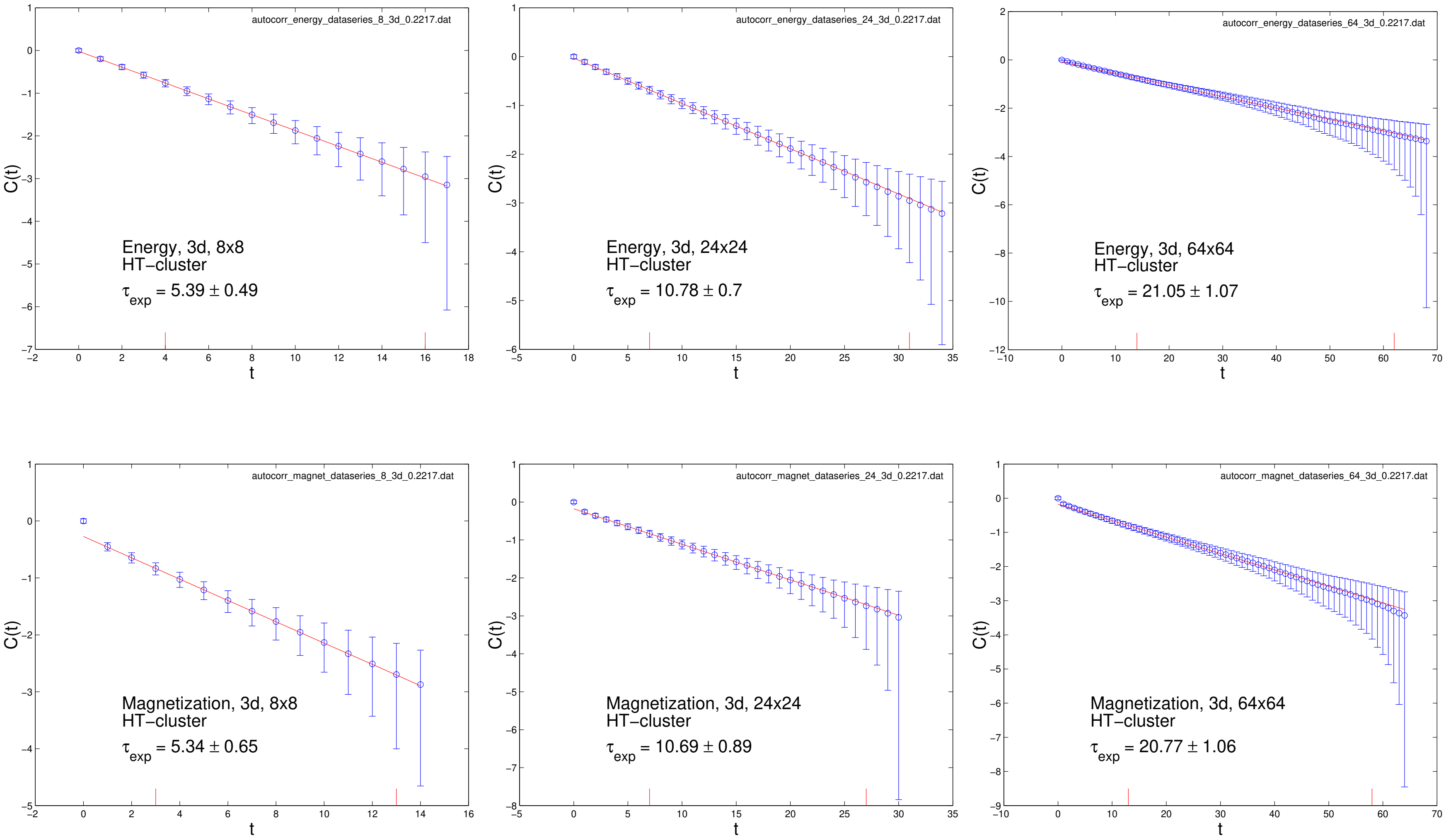
A.3 Detailed results of HT-cluster autocorrelation analysis

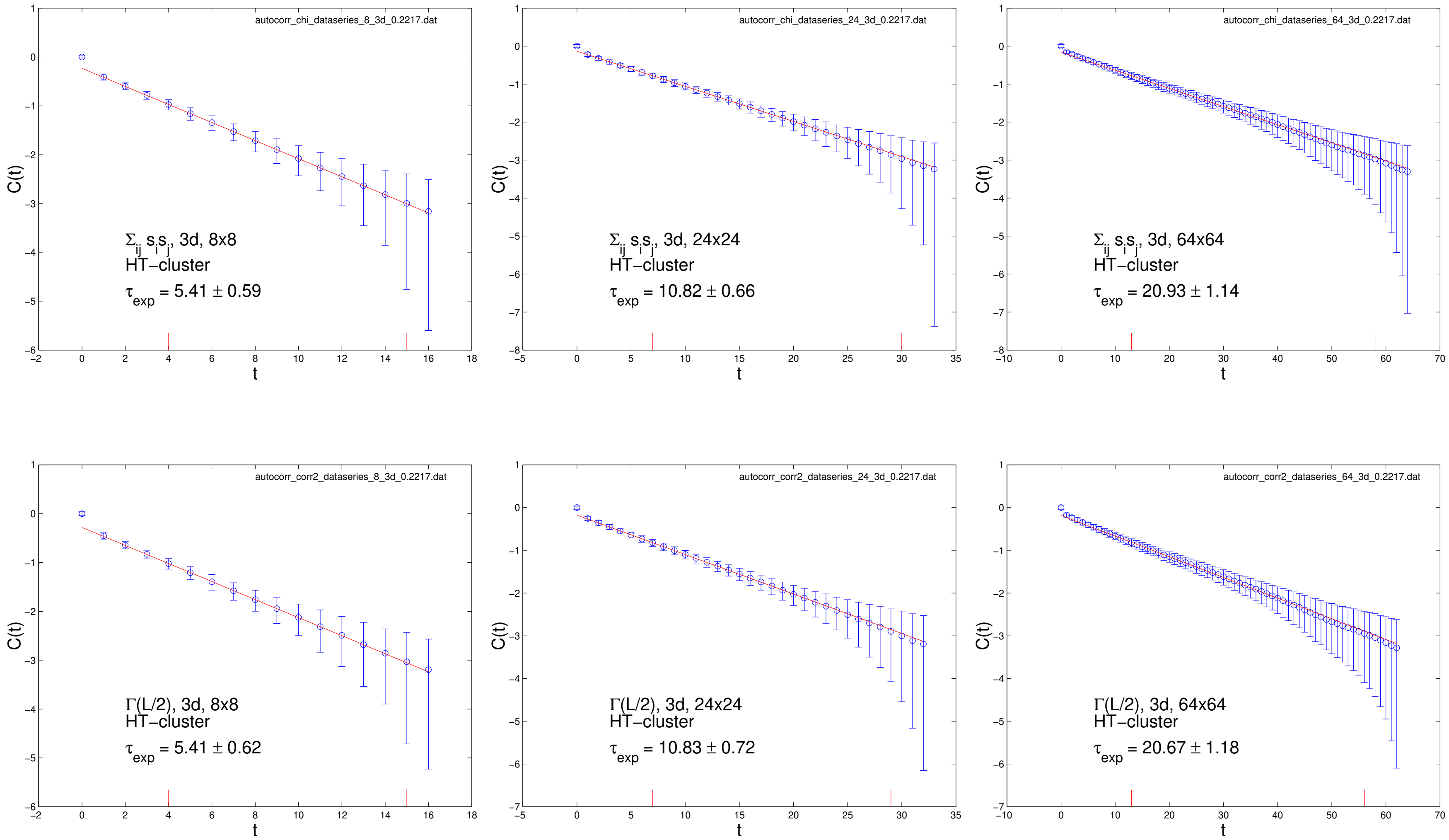
On the following pages we depict the autocorrelation function plus error and the fit to obtain τ_{exp} for small, medium and large lattices and all dimensions and measured quantities. The autocorrelation function is plotted in blue and its fit in red. Further, the red labels on the t-axis indicate the fit-range.

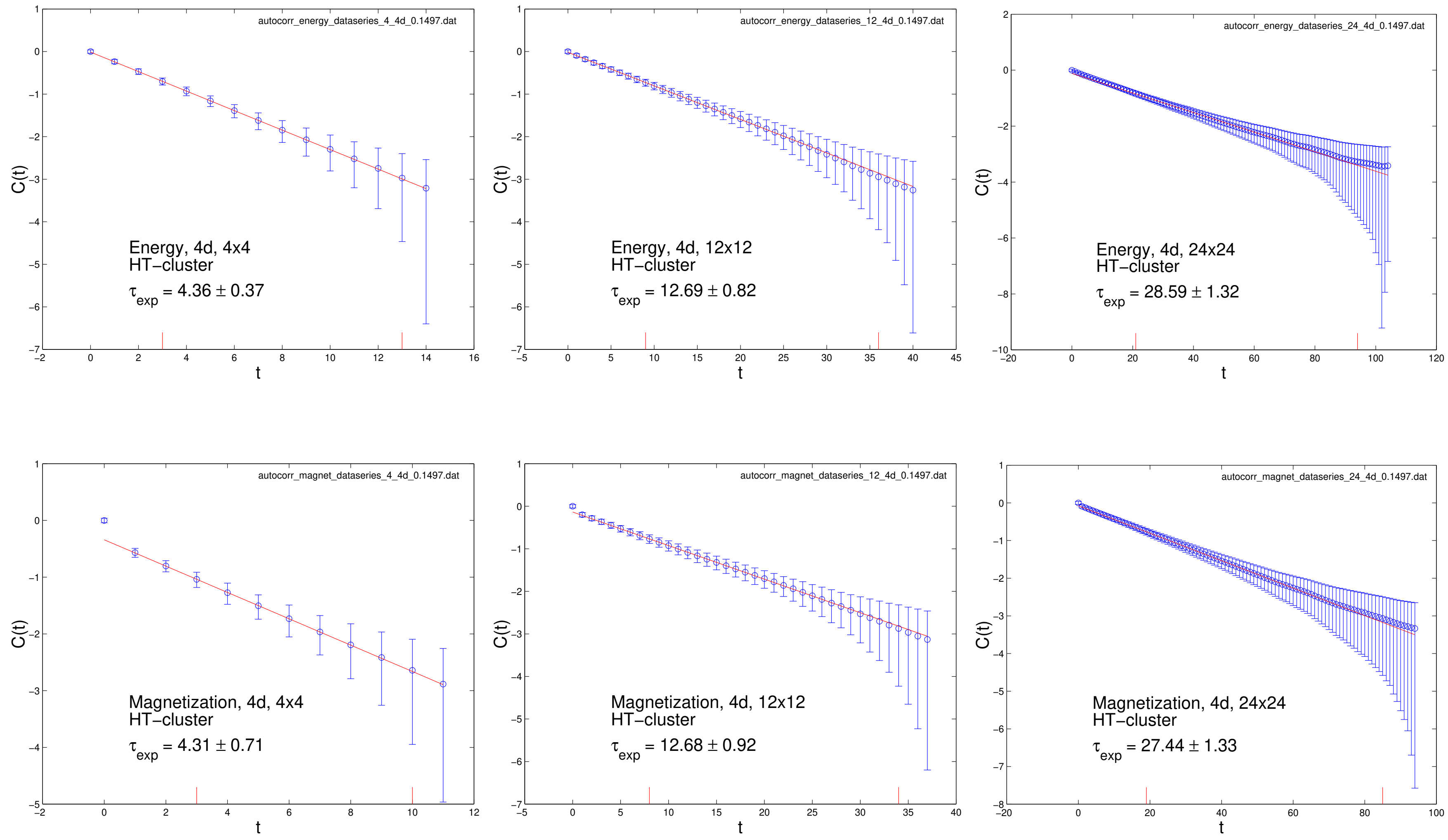

autocorr_energy_dataseries_8_2d_0.4407.dat
Energy, 2d, 8x8
HT−cluster
$\tau_{exp} = 2.49 \pm 0.13$
autocorr_energy_dataseries_64_2d_0.4407.dat
Energy, 2d, 64x64
HT−cluster
$\tau_{exp} = 5.03 \pm 0.24$
autocorr_energy_dataseries_256_2d_0.4407.dat
Energy, 2d, 256x256
HT−cluster
$\tau_{exp} = 7.87 \pm 0.34$
autocorr_magnet_dataseries_8_2d_0.4407.dat
Magnetization, 2d, 8x8
HT−cluster
$\tau_{exp} = 2.48 \pm 0.13$
autocorr_magnet_dataseries_64_2d_0.4407.dat
Magnetization, 2d, 64x64
HT−cluster
$\tau_{exp} = 5 \pm 0.18$
autocorr_magnet_dataseries_256_2d_0.4407.dat
Magnetization, 2d, 256x256
HT−cluster
$\tau_{exp} = 7.22 \pm 0.2$
C(t)
t



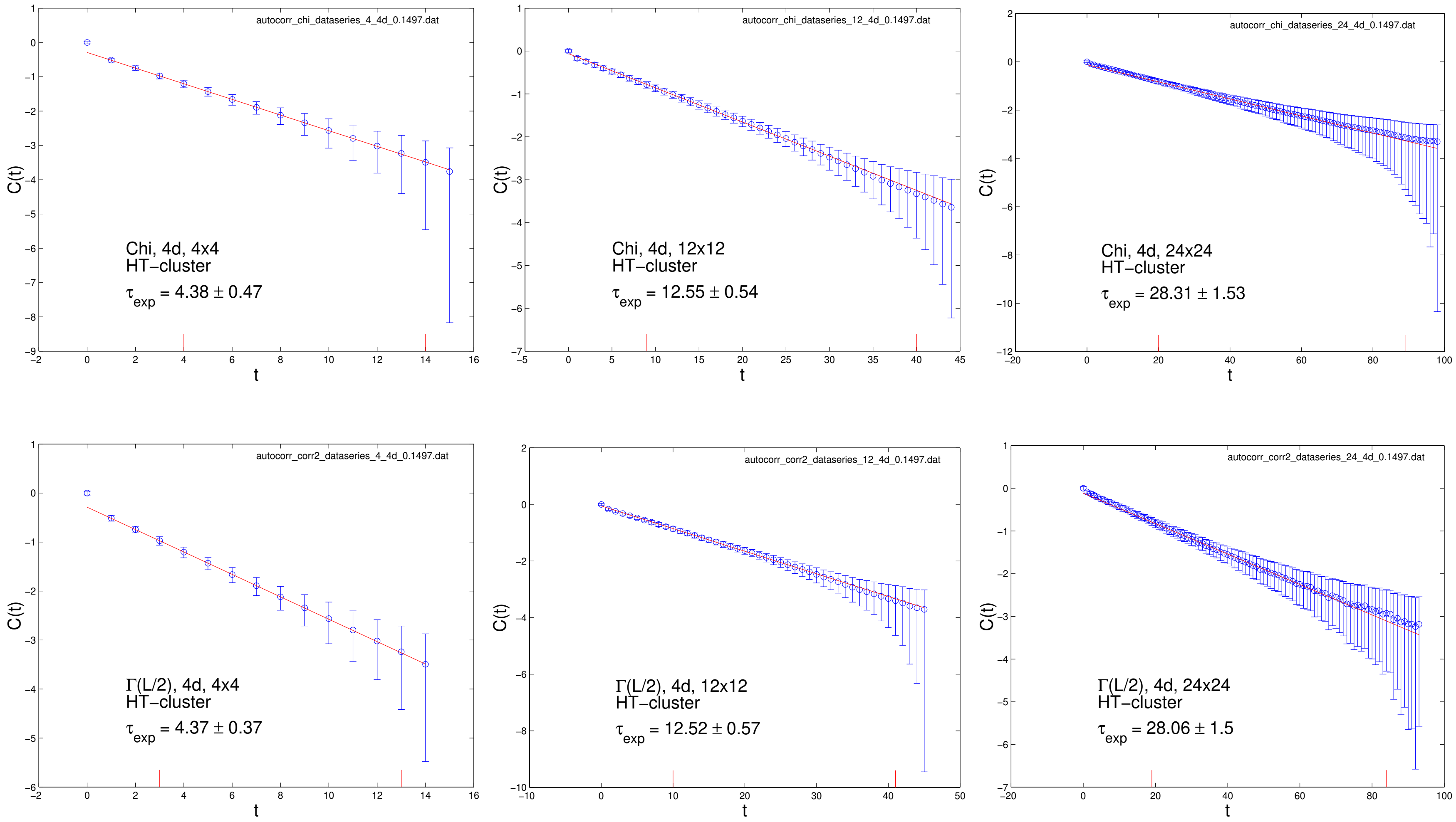
autocorr_chi_dataseries_8_2d_0.4407.dat
Σ_{ij} $s_i s_j$, 2d, 8x8
HT−cluster
τ_{exp} = 2.49 ± 0.13
autocorr_chi_dataseries_64_2d_0.4407.dat
Σ_{ij} $s_i s_j$, 2d, 64x64
HT−cluster
τ_{exp} = 5.01 ± 0.17
autocorr_chi_dataseries_256_2d_0.4407.dat
Σ_{ij} $s_i s_j$, 2d, 256x256
HT−cluster
τ_{exp} = 7.27 ± 0.21
autocorr_corr2_dataseries_8_2d_0.4407.dat
Γ(L/2), 2d, 8x8
HT−cluster
τ_{exp} = 2.48 ± 0.12
autocorr_corr2_dataseries_64_2d_0.4407.dat
Γ(L/2), 2d, 64x64
HT−cluster
τ_{exp} = 5.04 ± 0.18
autocorr_corr2_dataseries_256_2d_0.4407.dat
Γ(L/2), 2d, 256x256
HT−cluster
τ_{exp} = 7.28 ± 0.22
C(t)
t


autocorr_energy_dataseries_8_3d_0.2217.dat
Energy, 3d, 8x8
HT−cluster
τexp = 5.39 ± 0.49
autocorr_energy_dataseries_24_3d_0.2217.dat
Energy, 3d, 24x24
HT−cluster
τexp = 10.78 ± 0.7
autocorr_energy_dataseries_64_3d_0.2217.dat
Energy, 3d, 64x64
HT−cluster
τexp = 21.05 ± 1.07
autocorr_magnet_dataseries_8_3d_0.2217.dat
Magnetization, 3d, 8x8
HT−cluster
τexp = 5.34 ± 0.65
autocorr_magnet_dataseries_24_3d_0.2217.dat
Magnetization, 3d, 24x24
HT−cluster
τexp = 10.69 ± 0.89
autocorr_magnet_dataseries_64_3d_0.2217.dat
Magnetization, 3d, 64x64
HT−cluster
τexp = 20.77 ± 1.06
C(t)
t


autocorr_chi_dataseries_8_3d_0.2217.dat
Σij sisj, 3d, 8x8
HT−cluster
τexp = 5.41 ± 0.59
autocorr_chi_dataseries_24_3d_0.2217.dat
Σij sisj, 3d, 24x24
HT−cluster
τexp = 10.82 ± 0.66
autocorr_chi_dataseries_64_3d_0.2217.dat
Σij sisj, 3d, 64x64
HT−cluster
τexp = 20.93 ± 1.14
autocorr_corr2_dataseries_8_3d_0.2217.dat
Γ(L/2), 3d, 8x8
HT−cluster
τexp = 5.41 ± 0.62
autocorr_corr2_dataseries_24_3d_0.2217.dat
Γ(L/2), 3d, 24x24
HT−cluster
τexp = 10.83 ± 0.72
autocorr_corr2_dataseries_64_3d_0.2217.dat
Γ(L/2), 3d, 64x64
HT−cluster
τexp = 20.67 ± 1.18
C(t)
t



autocorr_energy_dataseries_4_4d_0.1497.dat
Energy, 4d, 4x4
HT−cluster
τ_{exp} = 4.36 ± 0.37
autocorr_energy_dataseries_12_4d_0.1497.dat
Energy, 4d, 12x12
HT−cluster
τ_{exp} = 12.69 ± 0.82
autocorr_energy_dataseries_24_4d_0.1497.dat
Energy, 4d, 24x24
HT−cluster
τ_{exp} = 28.59 ± 1.32
autocorr_magnet_dataseries_4_4d_0.1497.dat
Magnetization, 4d, 4x4
HT−cluster
τ_{exp} = 4.31 ± 0.71
autocorr_magnet_dataseries_12_4d_0.1497.dat
Magnetization, 4d, 12x12
HT−cluster
τ_{exp} = 12.68 ± 0.92
autocorr_magnet_dataseries_24_4d_0.1497.dat
Magnetization, 4d, 24x24
HT−cluster
τ_{exp} = 27.44 ± 1.33
C(t)
t


autocorr_chi_dataseries_4_4d_0.1497.dat
Chi, 4d, 4x4
HT−cluster
τexp = 4.38 ± 0.47
autocorr_chi_dataseries_12_4d_0.1497.dat
Chi, 4d, 12x12
HT−cluster
τexp = 12.55 ± 0.54
autocorr_chi_dataseries_24_4d_0.1497.dat
Chi, 4d, 24x24
HT−cluster
τexp = 28.31 ± 1.53
autocorr_corr2_dataseries_4_4d_0.1497.dat
Γ(L/2), 4d, 4x4
HT−cluster
τexp = 4.37 ± 0.37
autocorr_corr2_dataseries_12_4d_0.1497.dat
Γ(L/2), 4d, 12x12
HT−cluster
τexp = 12.52 ± 0.57
autocorr_corr2_dataseries_24_4d_0.1497.dat
Γ(L/2), 4d, 24x24
HT−cluster
τexp = 28.06 ± 1.5
C(t)
t

A.4 Length of Monte Carlo runs

SW 2d		SW 3d		SW 4d	
L	N_{meas}	L	N_{meas}	L	N_{meas}
8	$2 \cdot 10^6$	8	$1 \cdot 10^6$	4	$1 \cdot 10^6$
12	$2 \cdot 10^6$	12	$1 \cdot 10^6$	8	$1 \cdot 10^6$
16	$2 \cdot 10^6$	16	$1 \cdot 10^6$	12	$1 \cdot 10^6$
24	$2 \cdot 10^6$	24	$5 \cdot 10^5$	16	$1 \cdot 10^6$
32	$1 \cdot 10^6$	32	$5 \cdot 10^5$	24	$6.4 \cdot 10^5$
48	$1 \cdot 10^6$	48	$5 \cdot 10^5$		
64	$1 \cdot 10^6$	64	$5 \cdot 10^5$		
96	$1 \cdot 10^6$				
128	$1 \cdot 10^6$				
256	$6 \cdot 10^5$				
512	$1 \cdot 10^5$				

Table A.1: Length of Monte Carlo runs for the Swendsen-Wang algorithm. Number of thermalization runs = 10% of N_{meas}.

HT-cluster 2d		HT-cluster 3d		HT-cluster 4d	
L	N_{meas}	L	N_{meas}	L	N_{meas}
8	$2 \cdot 10^6$	8	$1 \cdot 10^6$	4	$1 \cdot 10^6$
12	$2 \cdot 10^6$	12	$1 \cdot 10^6$	8	$1 \cdot 10^6$
16	$2 \cdot 10^6$	16	$1 \cdot 10^6$	12	$1 \cdot 10^6$
24	$2 \cdot 10^6$	24	$1 \cdot 10^6$	16	$1 \cdot 10^6$
32	$2 \cdot 10^6$	32	$1 \cdot 10^6$	24	$2.4 \cdot 10^5$
48	$2 \cdot 10^6$	48	$1 \cdot 10^6$		
64	$1 \cdot 10^6$	64	$2 \cdot 10^5$		
96	$1 \cdot 10^6$				
128	$1 \cdot 10^6$				
256	$1 \cdot 10^6$				
512	$1.5 \cdot 10^5$				

Table A.2: Length N_{meas} of Monte Carlo runs for the HT-cluster algorithm. Number of thermalization runs = 10% of N_{meas}.

Bibliography

[1] Online resources concerning my thesis. All programs and documentation available under this URL. *http://www.itp.tu-graz.ac.at/AG/AVP/erkinger/*

[2] N. Metropolis, A.W. Rosenbluth, M.N. Rosenbluth, A.H. Teller, E. Teller, J. Chem. Phys. 21, 1087 (1953)

[3] J.J. Binney, N.J. Dowrick, A.J. Fisher, and M.E.J. Newman, *The Theory of Critical Phenomena*, Oxford Science Publications, 1992.

[4] J.M. Yeomans, *Statistical Mechanics of Phase Transitions*, Oxford Science Publications, 1992.

[5] M. Plischke, B. Bergersen, *Equilibrium Staistical Physics*, World Scientific, 1994.

[6] W. Nolting, *Quantentheorie des Magnetismus 2*, Teubner Studienbücher Physik, 1986.

[7] R. Ben-Av, D. Kandel, E. Katznelson, P.G. Lauwers, and S. Solomon, *Critical Acceleration of Lattice Gauge Simulations*, J.Stat.Phys 58, 125 (1990).

[8] K. Binder, D.W. Heermann, *Monte Carlo Simulation in Statistical Physics*, Springer Series in Solid-State Sciences 80, Springer-Verlag, 1988.

[9] Alan M. Ferrenberg and D.P. Landau, *Critical behaviour of the three-dimensional Ising model: A high-resolution Monte Carlo study*, Phys. Rev. B 44, 5081(1991).

[10] D. Stauffer, J. Adler, *Logarithmic factors, critical temperature, and zero temperature flipping in the 4D kinetic Ising model.*, Int. J. Mod. Phys. C, 263(1997).

[11] Rupert G. Miller, *The jackknife - a review.*, Biometrika 61, 1(1974).

[12] Mark C. K. Yang, David H. Robinson, *Understanding and Learning Statistics by Computer*, World Scientific Series in Computer Sciences, Vol. 4, 1986.

[13] H.G. Evertz, *The Loop Algorithm*, to be published in *Numerical Methods for Lattice Quantum Many Body Problems*, Perseus Books, Frontiers in Physics.

[14] Alan D. Sokal, *Bosonic Algorithms*, published in *Quantum Fields on the Computer (Advanced Series on Directions in High Energy Physics, Vol 11)*

[15] S.G. Brush, *History of the Lenz-Ising model*, Rev. Mod. Phys. 39, 883 (1967).

[16] W. v. d. Linden. Skriptum zur Vorlesung *Numerische Behandlung von Vielteilchenproblemen I, II.* TU Graz, 1999/2000.

[17] G. T. Barkema, M. E. J. Newman, *New Monte Carlo algorithms for classical spin systems*, condmat/9703179.

[18] L. Onsager, *A 2d model with an order-disorder transition*, Phys. Rev. 65, 117 (1944).

[19] M. D'Onorio De Meo, D. W. Heermann, and K. Binder, *Monte Carlo Study of the Ising Model Phase Transition in Terms of the Percolation Transition of "Physical Clusters"*, J. Stat. Phys. 60, 585(1990).

[20] W. H. Press, S. A. Teukolsky, W. T. Vetterling, B. P. Flannery, M. Metcalf, *Numerical Recipes in Fortran 90* Cambridge University Press, 1996.

[21] W. v. d. Linden, A. Prüll, *Wahrscheinlichkeitstheorie, Statistik und Datenanalyse*, TU Graz, 2000.