Metropolis-Hastings simulation of the Ising model

April 29, 2018

Abstract

The Ising model of magnetism was analyzed using the Metropolis algorithm. Critical transitions in heat capacity, magnetisation, magnetic susceptibility and autocorrelation were observed at various critical temperatures T_C, and their scaling with the lattice size was investigated using magnetisation and specific heat. The hypothesis that $T_c(N) = T_C(\infty) + aN^{-1/\nu}$, where $T_C(\text{inf}) = \frac{2}{\ln(1+\sqrt{2})} \approx 2.269$ and ν is the appropriate critical exponent, was investigated and shown to be consistent with measurements. Effects of external magnetic field were considered, and a change in critical behaviour was observed

1 Introduction

The Ising model[1] is a microscopic model of magnetic materials. It consists of an N-dimensional lattice, with periodic topology, and often hyper-cubic distribution of molecular magnetic moments. For our analysis, we shall only consider a two-dimensional Ising model, with equal coupling between nearest neighbours (Four in total), and ignoring all other interactions. Furthermore, we shall restrict the molecular magnetic moments (also referred to as spins) to only two states – *up* and *down*.

This model had been solved Analytically by Lars Onsager in 1944[2, 3], for a special case, of no external magnetic field. We shall be comparing the results of our simulation to Onsager's analytical predictions.

A brief explanation of the algorithm used for the simulation will be given in the following section, after which we shall discuss the particular investigations carried out, and discuss the results.

Our investigation consists of three parts.

- Initial run. We look at time evolution and some low-level phenomenology, to sanity check the simulation. We then analyse autocorrelation, to find out how much time the system needs to equilibrate.
- Critical temperature analysis. Here we analyze critical transitions in energy, magnetisation, and heat capacity, and investigate finite-size scaling (more on that later).

- External field analysis. Here we analyse the system's magnetic susceptibility (dynamic and otherwise), and analyse the critical phenomena's scaling on $H \neq 0$.

The entire working code is attached in the appendix, details of the current implementation – discussed in Section 2.1.

2 Algorithm

We shall restrict ourselves to square lattices of size $N \times N$. Each lattice site has an associated spin $S_{(i,j)}$. Nearest neighbours[1] (i.e. the four[2] closest spins in the lattice) have an associated pair-interaction energy $J \in \mathbb{R}$, also referred to as *exchange energy*. Throughout our analysis we shall set $J = 1$, which corresponds to *ferromagnetic* materials.

Additionally the entire lattice can be subjected to an external magnetic field H which we assume is spatially uniform, i.e. the same across all lattice sites.

In this arrangement a spin $S_{(i,j)}$ shall have an associated *point-energy*

$$E_{(i,j)} = S_{(i,j)} \cdot \left[-J \sum_{m,n}^{\text{nearest neighbours}} S_{(m,n)} - JH \right] \tag{1}$$

and summing over all $N \times N$ pairs of lattice points yields us the total energy associated with the lattice.

One can show that this system shall (given $J \geq 0$) tend to a state where all spins are aligned with each-other and the external field. If the system has a finite temperature $T > 0$ then there will be a competing thermal process that seeks the state with maximum entropy.

We can simulate both processes by means of a simple *Metropolis-Hastings* algorithm:[4]

Algorithm 1 Sweep

```
procedure SWEEP(lattice L)
    for (i, j) ∈ L do
        ΔE_(i,j) ← (L.flip(i, j)).E_total − L.E_total
        r ← random uniform(0, 1)
        if ΔE_(i,j) ≤ 0 ∨ exp[−ΔE_(i,j) / (k_B T)] > r then
            L := L.flip(i, j)
```

In other words, for each lattice site (i, j), compute the energy needed to flip one spin. If by flipping that spin energy can be released ($\Delta E < 0$), flip it. If the associated Boltzmann factor $\exp\left[-\frac{\Delta E}{k_B T}\right]$ is greater than a (random) probability r from a uniform-distribution, flip the spin.

[1] From here onwards we shall ignore all other possible next-nearest neighbour interactions

[2] in D dimensions it would be $2 \times D$ spins

The last step emulates the thermodynamics of the system[5]. We don't know the exact microscopic conditions under which a spin might be put in an energetically unfavourable state, but we anticipate that at time $t \gg 1$ the system will obey Boltzmann statistics. Thus to achieve the same statistical effect as real thermal fluctuations would, we can pick any set of microscopic rules that produces the Boltzmann statistics in the long run.

One can also show using simple algebra, that $\Delta E_{(i,j)} = -2E_{(i,j)}$.

2.1 Implementation.

I elected to separate the program into two pieces: a high performance simulation written in `C++`, and a `python` script that analyses the data. The two can communicate either through temporary files in the `data/` directory, or directly via a `Unix pipe`.

Hence, we benefit from both the speed of a compiled program, and python's flexibility.

2.1.1 C++ program.

The binary `main`, when given the optional arguments `-j`, `-H`, `-n` and `-t` produces and simulates a lattice of size $N \times N$, with exchange energy J, etc., and writes statistics data either to a file (if given an `-f filename.csv` optional argument) or directly to `stdout`, which can be 'piped' to another program, e.g. the python script. To control the number of Monte-Carlo steps to simulate, specify the `-d` ***`duration`*** argument.

2.1.2 Python script.

This script performs all the data analysis in this project. By default it will communicate with `main` via a 'Unix pipe', which can be changed to output simulation data into the `data/` directory, by setting `USE_DISK = True`, in `AnalysisTools.py`.

2.2 Performance.

2.2.1 Optimisations.

The bulk of optimisations in the code are contained in the `C++` program, and is done by the compiler. Some of these optimisations cannot be efficiently done by hand: the compiler is able to access CPU registers, while C++ code acts more like a guideline: the compiler is free to replace any loop with a single mathematical operation, and any manual optimisation that assumes such behaviour can make the code run slower.

Indeed, upon attempting to replace the call to $\exp(-\frac{\Delta E}{T})$ with a lookup table, one may find that the code runs far slower, due to the efficiency of the compilers common subexpression elimination. Another example, the code, runs equally fast, when the division by T is replaced with the inverse temperature, because the compiler has already done that implicitly.

We can however, guide the compiler to attempt multithreading and *Single-instruction multiple data* (SIMD) optimisations, to speed the code up as much as possible. Even then, the overhead of creating a thread pool for smaller tasks at low level can negatively impact

the performance of the program (the data needs to be copied, merged and sometimes this is too inefficient for a small core count).

In `python`, the chief optimisation is to avoid expensive interpreter operations: use vectorized numpy functions, convert arrays into lists, etc. Since python is an interpreted language, few automatic optimisations can be done (unless using `pypy`, which is a compiler). It is useful at this level to make use of multiprocessing. A specialised function `parmap` was specifically implemented to allow high bandwidth data intechange between different processes, which allows for almost perfect scaling across multiple CPU cores. This renders any attempt at multithreading the C++ program ineffective, as additional overhead from updating the lattice in parallel, is greater than the overhead of multiple instances of`subprocess.call`. Of course, the Unix pipe does introduce additional overhead, but it is negligible compared to running each individual simulation.

A notable optimisation is equilibrating the system for only as long as it needs to be equilibrated for. In Section 3.3, we shall investigate the profile of temporal coherence's scaling with temperature. From there we can predict how much longer does a system need to equilibrate at any given temperature and mimicking this profile, reduce the program's overall run-time.

2.2.2 Metrics.

The performance metrics for my system are given in Table1.

Table 1: Performance metric for the Ising model simulation, runtime of each section of investigations of the functions (measured externally)

Function	run-time/s
sanity_checks()	3.254
fss_magnetisation()	1135.205
fss_specific_heat()	839.750
external_field()	215.311

The performance of this program highly depends on the CPU core count and architecture.

To account for this, assume that the overhead scales linearly with the performance of a single simulation ran on a single core (which ignores thermal throttling, CPU boost etc.). On my machine (a laptop), the command

```
time ./main -t 2 -f data/data.csv
-n 200 -d 1000 -p 100
```

runs for roughly `1.287s`. So if on the target system this evaluates in `x` seconds, a reasonable estimate of the following runtimes can be achieved by scaling results by `x/1.287`.

The performance numbers can be further improved if we sacrifice precision and accuracy.

3 Method

In this section we shall discuss what the program will do, how.

3.1 Time evolution.

Time evolution can be measured directly. Some variety of initial conditions can be attained by considering a chequerboard pattern and setting the lattice at random.

3.2 Critical transitions.

At any temperature there are two competing tendencies in a system. To minimise interaction energy, spins align and $M \approx 1$. Thermal effects on the other hand, will maximise the system's entropy.

Temperature determines which response dominates: at high temperatures, the system tends to a disordered phase $M \approx 0$, and vice versa. Hence there is a temperature T_C above which the system will always eventually reach a maximum entropy state, i.e. the system will transition from an ordered phase to a disordered one. Hence the graph of a long run average of the magnetisation vs. temperature would be a 'step-down,' while for long run average mean-energy would be a 'step-up.'

An alternative explanation would combine the two competing responses into the Helmholtz Free energy F. We then simply expect that the transition occurs when $\Delta F = 0$, where the change ΔF is across the infinitesimally small region around T_C.[6]

Near critical temperature the system takes longer to suppress fluctuations, which is known as the *critical slowdown*[7]. It manifests as noise in both M and $\langle E \rangle$ plots near critical temperature. There are many ways to mitigate this noise: e.g. the *Wolff algorithm*[7]. However, since we only need to find $T_C(N)$, we can run the simulation several times (for a sufficiently long number of time steps), take the mean and the standard error of M each value of T, and then do a weighted nonlinear fit with step-like function, such as

$$f(T, a, b) = \frac{1}{\exp\left[\frac{(T-a)}{b}\right] - 1}$$

. Of course, while we could also fit Onsager's analytical solution:

$$M = [1 - \sinh^{-4}(2J/k_B T)]^{\frac{1}{8}} \tag{2}$$

this would not provide us with the critical temperature as easily.

3.3 Auto-correlation.

The autocorrelation function for lag time τ of magnetisation $M(t)$ is given by

$$a(\tau) = \frac{\langle \delta M(t) \delta M(t+\tau) \rangle_t}{\langle (\delta M(t))^2 \rangle} \tag{3}$$

Where $\delta M(t) \triangleq M(t) - \langle M(t) \rangle_t$ and angle brackets denote time averaging over a 'long' time. As the name suggests, this quantifies the 'memory' of the system. We expect for $a(\tau)$ to decay exponentially fast, characterised by coherence lifetime τ_e.

Finding τ_e is complicated, since the correlation function fluctuates significantly, rendering any attempt to fit an exponential to the data ineffective, despite averaging over multiple runs. Another option is to do *Brent's method root finding*[8] with *spline interpolation.*

From then on, we shall investigate scaling of the coherence lifetime with respect to both lattice size and temperature. This allows us to easily predict how much time is sufficient for a system to reach thermodynamic equilibrium.

3.4 Heat capacity.

The *fluctuation-dissipation* theorem[9] states that

$$C_v = \frac{\sigma_E^2}{(k_B T)^2} \tag{4}$$

Where $\sigma_E = \sqrt{\langle (E - \langle E \rangle)^2 \rangle}$, i.e the standard deviation of the mean energy per unit area.

Thus heat capacity, reaches a maximum nearing the critical temperature, due to critical slowdown[3]. We can find this peak to high precision using spline interpolation.

However, critical slowdown complicates obtaining data near critical temperature. We need to let the system run long-enough to reach equilibrium, and to do this efficiently, we shall mimic the temperature scaling from sec:3.3.

3.5 Finite size scaling.

In the limit $N \to \inf$, Onsager showed[2] that the critical temperature is

$$T_C(\infty) = \frac{2}{\ln(1+\sqrt{2})} \tag{5}$$

We expect this to scale with the lattice size[10] according to

$$T_c(N) = T_C(\infty) + aN^{-1/\nu} \tag{6}$$

where a is a constant, ν is the critical exponent for the desired quantity[11], $T_c(\infty)$ is our estimate of Onsager's result.

3.6 External field coupling.

Materials are often convenient to describe using material 'constants',[4] such as magnetic susceptibility, permeability etc. To avoid issues with hysteresis let us define the *dynamic susceptibility* (of the lattice) as

[3] i.e. because the system is less able to suppress fluctuations.

[4] Insofar, as they depend on the material's internal state.

$$dM = \chi dH \tag{7}$$

So far, a thermally dominated system could only converge to a state of maximum disorder, when $M = 0$ due to symmetry. Indeed, keeping all external conditions the same by symmetry of the problem, and inverting the system would invert $M \to -M$. But by assumption in maximum disorder, the distribution of spins in the lattice should be uniform, therefore invariant under the inversion. Hence $M = -M$. The external field, however breaks the symmetry of external conditions, hence the lattice can now be in equilibrium at magnetisations other than $M = 0, 1$.

Dynamic magnetic susceptibility χ:

$$\chi = \frac{\partial M}{\partial H} \tag{8}$$

thus, should also show a critical transition, most likely at a higher temperature $T_C = T_C(H)$.

First of all, we shall check that the linear approximation of magnetic response is valid. Thenceforth, by doing linear regression for $M = M(H)$, obtain dynamic susceptibility, and plot it against different temperatures.

Of course this is a crude model, and one might achieve better precision by using the fluctuation-dissipation theorem[9]:

$$\chi \propto \frac{\sigma_M^2}{(k_B T)} \tag{9}$$

and essentially repeating the analysis from Section 3.4, with H in the role of N.

4 Discussion

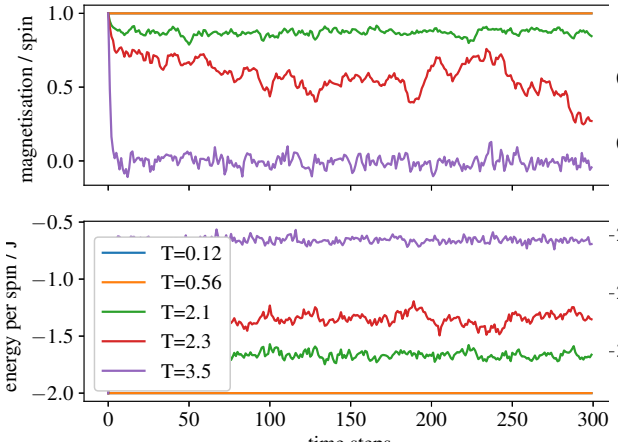
4.1 Time evolution.

Figure 1 shows, that given a totally ordered initial state the system evolves along two paths:

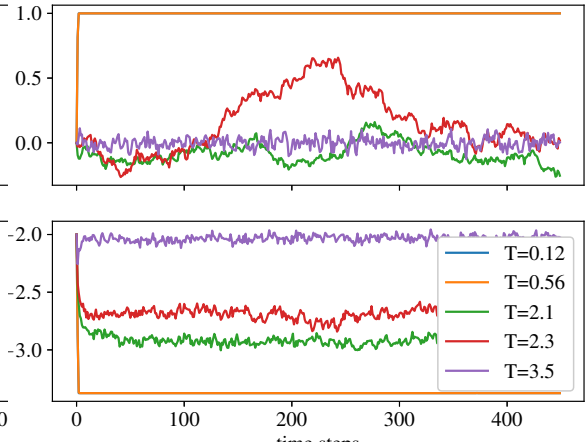
1. System remains ordered. True for sub-critical temperatures. Here any fluctuation about a completely ordered state is quickly suppressed by spin-spin interactions.

2. System decays to a disordered state. This occurs for super-critical temperatures. Here fluctuations are being suppressed by random spin flipping.

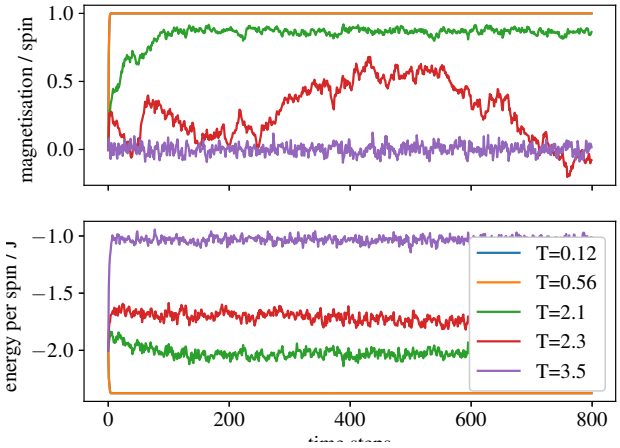
Other initial configurations, e.g. a chequerboard pattern, where spins are grouped into uniform square *domains*, show the same overall behaviour. In some cases, given sub-critical temperature, such a system may lose some but not all of its domains, resulting in a sustained equilibrium magnetisation other than unity. This phenomenon is called domain wall pinning.



(a) Evolution from completely ordered initial state



(b) Evolution from chequerboard pattern with cell size of 3



(c) Evolution from chequerboard pattern with cell size of 10

Figure 1: Time evolution of magnetisation and mean energy per spin, for different starting conditions.

One can also notice that for temperatures approaching the critical value, reaching equilibrium takes more Monte-Carlo steps. This is called the *critical slowdown*, and it's caused by the two competing tendencies: to maximum entropy and minimising spin-spin interactions being comparable, near critical temperature.

4.2 Autocorrelation.

Figure 2 shows the autocorrelation dependence on lag time. Each data point with the corresponding error-bar, is the result of five runs. Interestingly, multiple re-runs of the simulation do not yield noise reduction, and the error in each datum is negligible. This suggests that the autocorrelation is a deterministic function of the process, and not the state.

Next, consider fig. 4, the size-averaged[5] temperature dependence of the coherence lifetime τ_e. We can see manifestations of critical slowdown: the central peak. We also know that temporal coherence of the spin lattice is dominated by temperature, but also scales with size. However, the nature of that scaling is hard to deduce: on average, size scaling is negligible compared to temperature scaling.

Thus, to accurately mimic the profile of equilibration timescales, one would fit a Lorentzian to the coherence lifetime profile and set the simulation durations to a large multiple of the best fit. [6]. Accounting for the large errors in the data, we can also estimate $T_C(\text{inf}) = 2.39 \pm 0.13$, which is within one standard error of Onsager's result.

4.3 Critical transitions.

4.3.1 Magnetisation and energy transitions.

The expected step-like transitions are given in plot 5. Small error-bars indicate that we have sufficiently equilibrated the system.

4.3.2 Heat capacity.

The data are shown in figure 6. One can see peaks in C_V, getting narrower as they converge upon Onsager's prediction for an infinite lattice: $\propto \log(T - T_C)$. Again notice that the error-bars are negligible, indicating that the simulation was run for sufficiently long a time.

4.3.3 Magnetic fluctuations.

Much like the previous case, the critical temperatures are in Table 2. Notice that since we have applied an external field of $H = 1.0$, the critical temperatures have shifted upwards

[5] Arguably, it would have been better to plot different series, for each lattice size, however, that plot would not have given useful information due to noise, which as mentioned previously I was not able to mitigate.

[6] However, this often fails (noisy data), so I've hard-coded the parameters of the last successful fit in `smart_duration` (see sec.6.1.2)

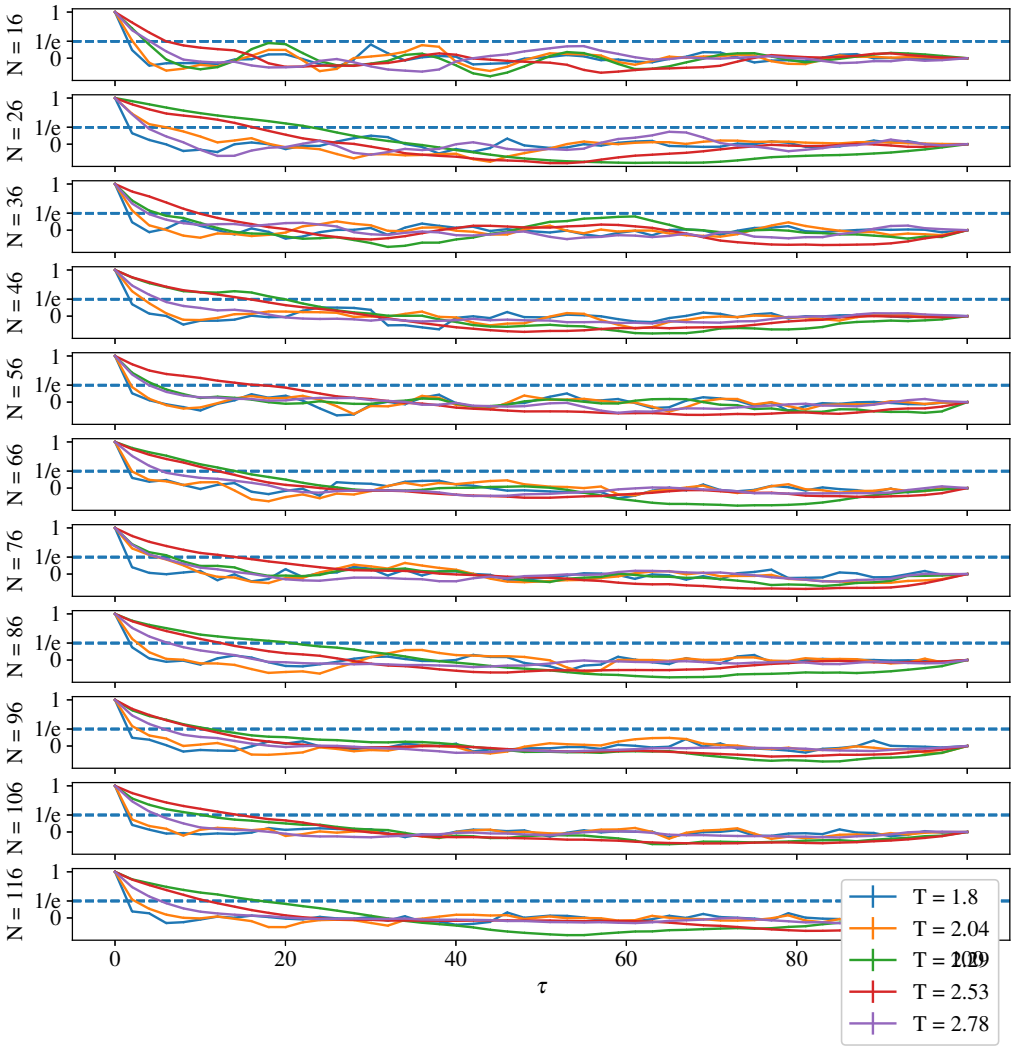


Figure 2: Autocorrelation as a function of time.

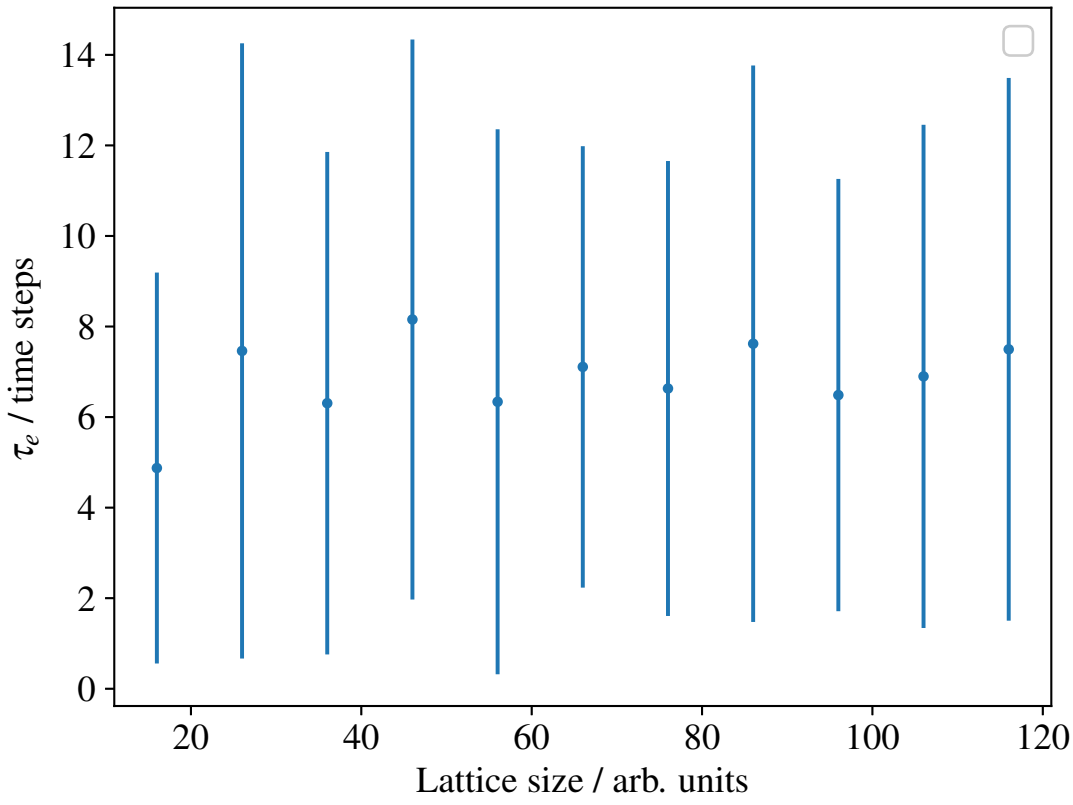


Figure 3: Coherence lifetime as a function of lattice size.

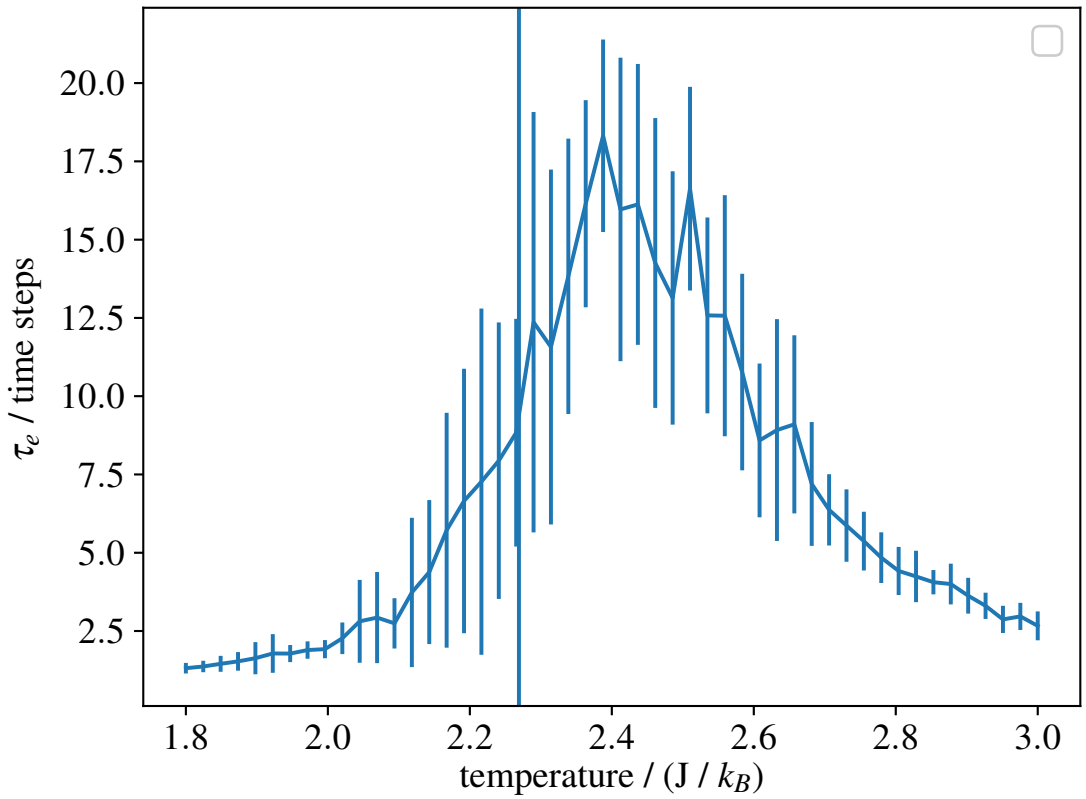


Figure 4: Coherence lifetime as a function of temperature.

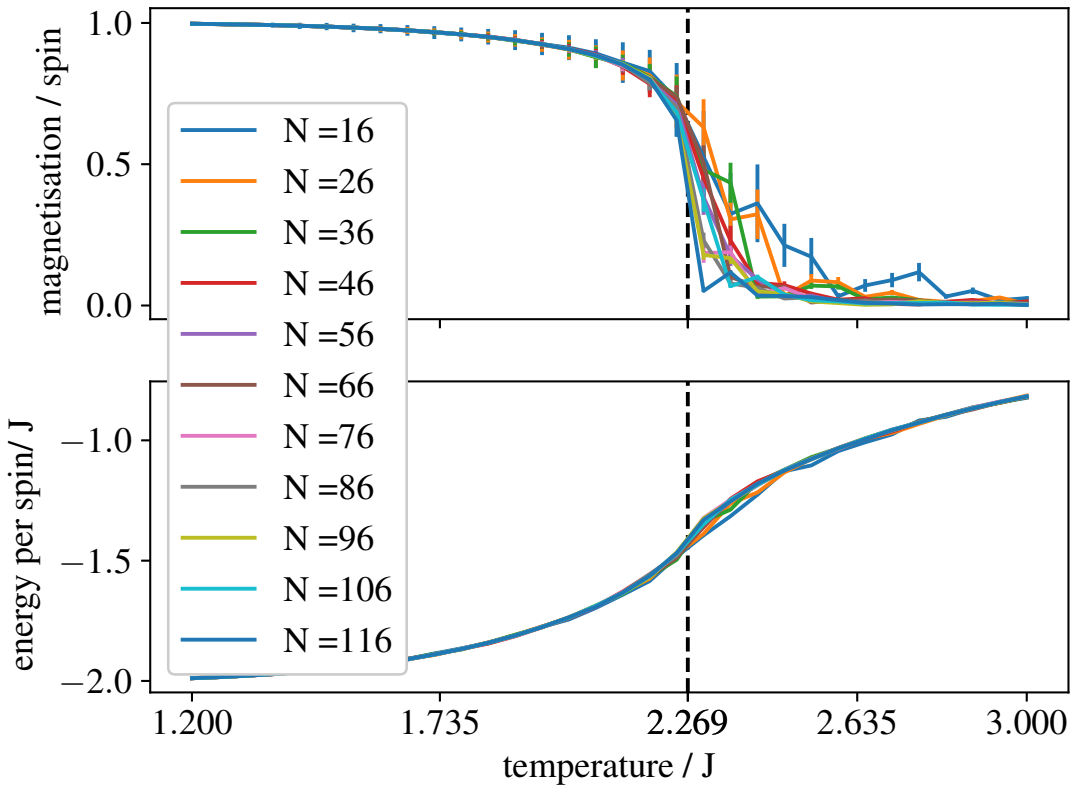


Figure 5: First order phase transitions. The transition in energy is barely noticeable. Magnetisation shows a pronounced step. Large error-bars would indicate that the system hasn't reached equilibrium.

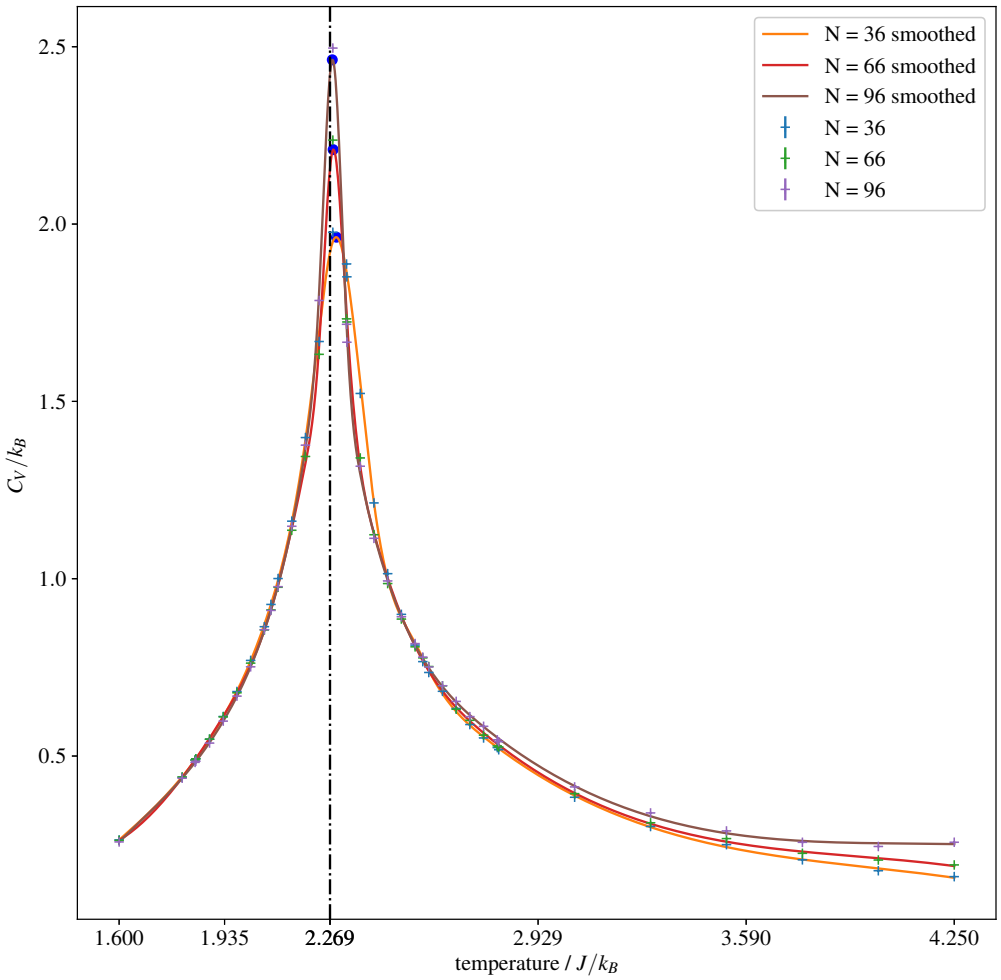


Figure 6: Heat Capacity data. The dashed vertical line indicates Onsager's result for $T_C(\text{inf})$. The estimated location of the critical temperature for each lattice size is indicated by a blue dot.

4.3.4 Finite size scaling.

Table 2: Critical temperatures from magnetic fluctuations

N	T_C	σT_C
16	2.69	0.36
46	2.61	0.28
76	2.58	0.25
106	2.57	0.31

The results of the nonlinear fit can be seen in figure 7a for heat capacity and figure 7b for magnetisation. The estimated values for $T_C(\text{inf})$ are $T_C^M(\text{inf})3.4 \pm 0.7$ for magnetisation and $T_C^C(\text{inf}) = 2.26 \pm 0.01$ for heat capacity, both within one standard error of T_C.

As theory suggests[12], we would also expect for ν for each case is the critical exponent for the respective order parameter: for heat capacity it should be $\nu = 1$, measured – $\nu = 1.44 \pm 0.52$, and for magnetisation it should be $\nu = 7/4$, measured – $\nu = 1.88 \pm 0.7$. Hence our data show remarkable agreement with Theory. The large errors and the few

4.3.5 Dynamic susceptibility.

The system's magnetic response to an applied external field H is given in fig. 8. We can see that the linear response approximation is valid.

Next, we obtain the dependence differential susceptibility on the temperature, and although we an show that a critical transition does occur, it takes place far from T_C. Moreover, it seems to be unaffected by different lattice sizes.

5 Conclusions

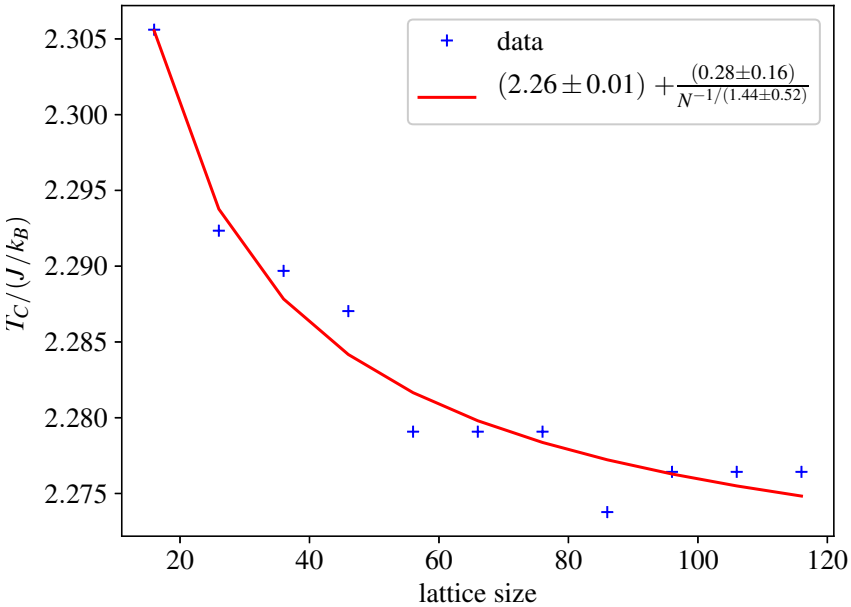
We have, thus investigated the basic results of Onsager's solution to the 2D Ising model. We have evaluated the critical temperatures (TODO) and extrapolated the results for an infinite lattice, to within one standard error and four standard errors in heat capacity and magnetisation order paramters respectively. We have also shown that the scaling relations predict the same scaling exponents for the aforementioned cases.

We have shown that the 2D lattice behaves much like a Ferromagnetic material, and that it shows a critical transition in magnetic susceptibility.

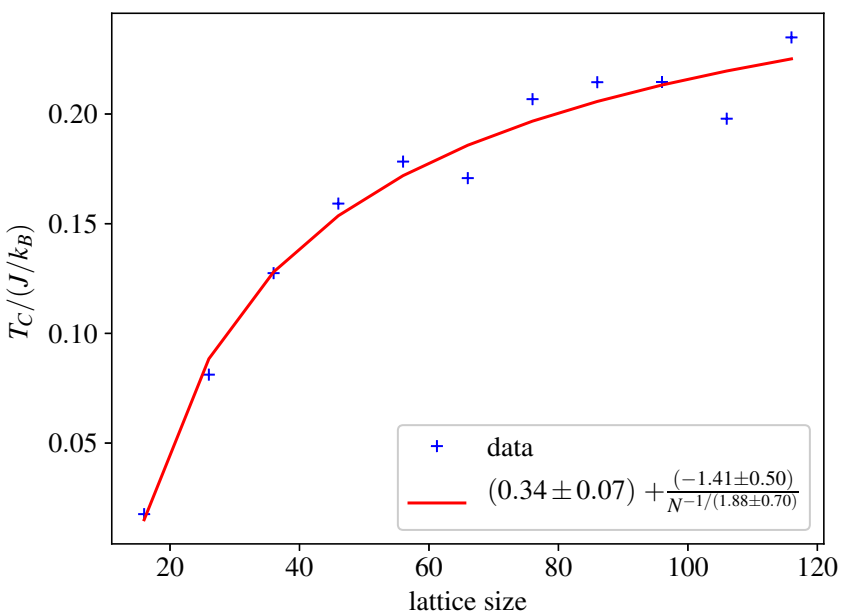
We have encountered persistent noise that deterred us from performing the full analysis of critical exponents, in sec. 3.3, however, we have managed to obtain a $T_C(\text{inf}) = 2.39 \pm 0.13$result consistent with Onsager's

To further explore the Ising Model, we could generalise the simulation to higher dimensions. Although no analytical result is known to exist in more than two dimensions, one can compare the results of this simulation to the slew of experimental data. Another suggestion, to improve the efficiency of the simulation is to use the Wolff algorithm, which improves the performance of the simulation near critical temperatures.

Another unexplored area is the convergence of the scaling relations to the asymptotic solutions.



(a) Finite size scaling: specific heat and best fit.



(b) Finite size scaling: magnetisation and best fit. Note that the temperature is scaled down by a factor of 10, to help the numerical routine with the fit.

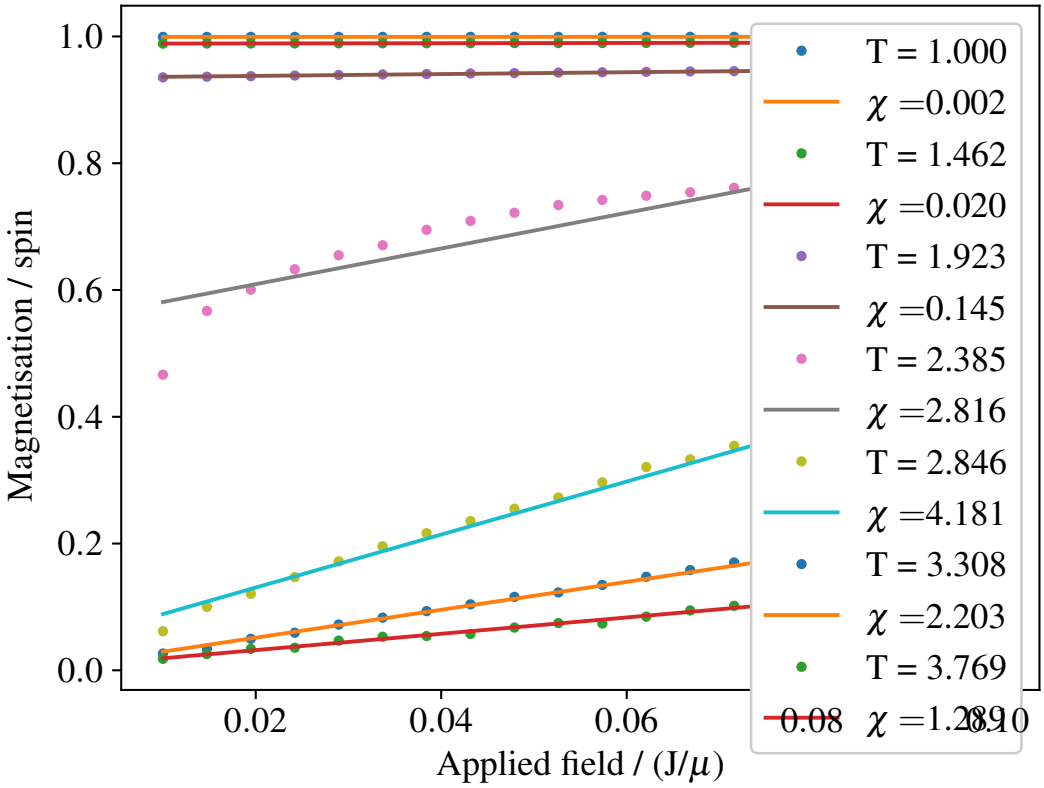


Figure 8: System's response to external field, and linear fit.

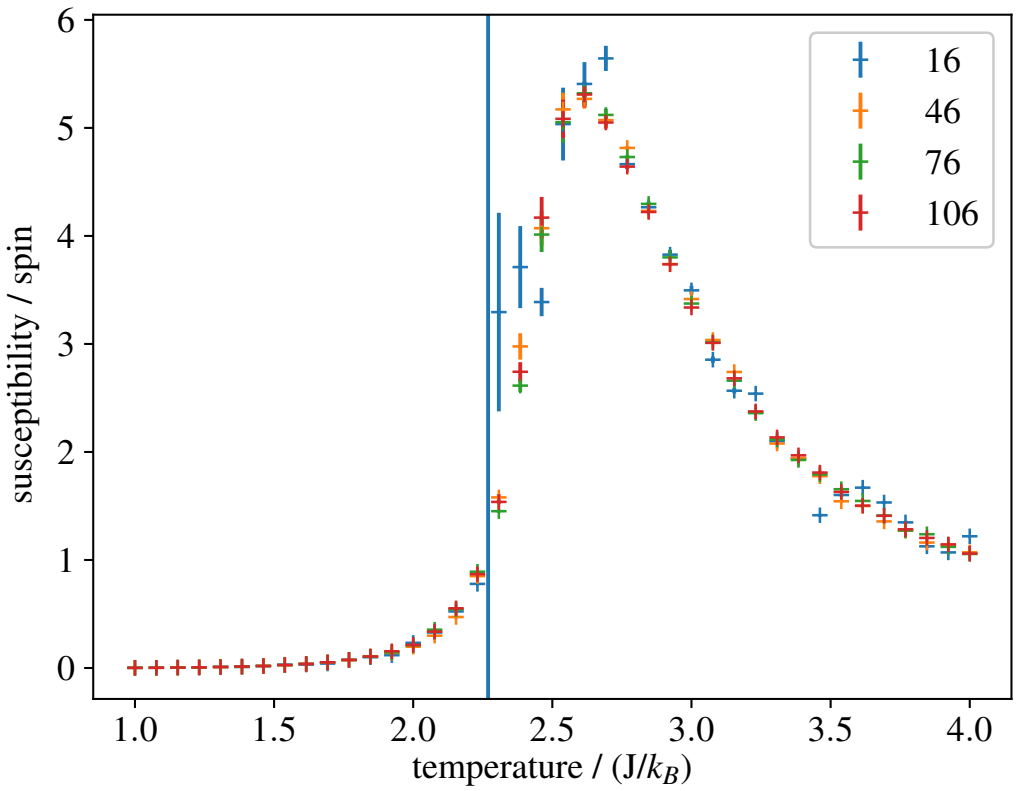


Figure 9: Plot of differential susceptibility.

6 Appendices

6.1 Python Script.

6.1.1 investigator.py

This is the entry point to our simulation and the only program that needs to be run. It contatins the code for all main investigations, and handles output.

6.1.2 AnalysisTools.py

This is the library containting common helper functions It contains the implementation of `parmap`.

6.2 C++ program.

6.2.1 makefile

The `gnu make` script to compile the program. By default it uses `clang++`, due to its superior debugging features.

6.2.2 interface.cpp

This is the part of the program that handles user interactions. It parses the command-line arguments, and runs the simulation.

6.2.3 simulation.cpp

This file contains the logic of the simulation: it implements the advance method and keeps track of internal state.

6.2.4 lattice.cpp

This is the implementation of the lattice class, defines the periodic boundary conditions etc. The lattice is internally represented by a one dimensional `std::vector` of short integers which improves both storage efficiency and speed.

6.2.5 include/simulation.h

Header file declating the class `simulation`. Contains inline functions, getters and setters.

6.2.6 include/lattice.h

Declares the class `lattice`.

6.2.7 include/rng.h

A wrapper for the Gnu scientific library random number generator. It maps the procedural library functions onto object-oriented programming, making it easier to avoid memory leaks.

References

[1] E. Ising, "Beitrag zur theorie des ferromagnetismus," *Zeitschrift für Physik*, vol. 31, pp. 253–258, feb 1925.

[2] L. Onsager, "Crystal statistics. i. a two-dimensional model with an order-disorder transition," *Phys. Rev.*, vol. 65, pp. 117–149, Feb 1944.

[3] S. G. Brush, "History of the lenz-ising model," *Reviews of Modern Physics*, vol. 39, pp. 883–893, oct 1967.

[4] W. K. Hastings, "Monte carlo sampling methods using markov chains and their applications," *Biometrika*, vol. 57, pp. 97–109, apr 1970.

[5] B. McCoy and T. T. Wu, *The Two-Dimensional Ising Model*. Harvard University Press, 1973.

[6] F. Schwabl, *Statistische Mechanik*. Springer-Lehrbuch, Physica-Verlag, 2006.

[7] U. Wolff, "Critical slowing down," *Nuclear Physics B - Proceedings Supplements*, vol. 17, pp. 93 – 102, 1990.

[8] R. P. Brent, *Algorithms for Minimisation without Derivatives (Automatic Computation)*. Prentice Hall, 1972.

[9] D. Chandler, *Introduction to Modern Statistical Mechanics*. Oxford University Press, 1987.

[10] T. Oguchi and Y. Taguchi, "Transfer matrix and finite-size scaling for the ising model on two- and three-dimensional lattices," *Progress of Theoretical Physics Supplement*, vol. 87, pp. 23–32, 1986.

[11] J. Cardy, *Scaling and Renormalization in Statistical Physics (Cambridge Lecture Notes in Physics)*. Cambridge University Press, 1996.

[12] J. Cardy, *Scaling and Renormalization in Statistical Physics (Cambridge Lecture Notes in Physics)*. Cambridge University Press, 1996.