UiO : Department of Physics
University of Oslo

Project 4: The Ising model

FYS3150 - Computational physics

Anders Johansson

Contents

Abstract

In this project, the Ising model for a two-dimensional ferromagnetic is studied numerically using Monte Carlo simulations with the Metropolis algorithm. A 2×2 system is also solved analytically for comparison, and the numerical results fit well. The stability and convergence rate of the Metropolis algorithm are also tested for a 20×20 system. Heavy simulations confirm a phase transition at approximately $T_C = 2.272$, which is close to the temperature analytically predicted by Lars Onsager.

1 Introduction

The Ising model was developed by the famous physicist Wilhelm Lenz and given to Ernst Ising as a problem for his thesis, in which he found the analytical solution for the one-dimensional situation. Ising's model describes a coupled system in which only the nearest neighbours interact. Aside from describing ferro- and antiferromagnetic materials as in this project, the model has also been used successfully to describe other coupled systems, for instance the development of birdsong dialects[1].

In this project, the Ising model is used to describe a two-dimensional ferromagnetic, chrystalline material, modelled as a grid of spins where each spin can have the values ± 1. For an infinite grid, the analytical solution was found by Lars Onsager, while a 2×2 grid is solved analytically in this project.

The problem with analytical solutions for other sizes is finding the partition function, which is the "holy grail" of statistical mechanics. From this quantity, other thermodynamic quantities can be calculated. The difficulty arises from the formula for the partition function, which includes summing over all possible microstates. With each spin having two possible values, a simple 10×10 grid will have $2^{10 \cdot 10} \approx 10^{30}$ possible microstates. For comparison, only approximately 10^{17} seconds have passed since the creation of the universe. Considering that the UiO computing cluster is capable of approximately 10^{14} calculations per second, it would take this supercomputer at least 10 % of the time since Big Bang to calculate the partition function.

As a consequence, calculating the partition function analytically for a grid of reasonable size is not a viable option, and numerical approximations must once again be applied. Fortunately, the Metropolis algorithm, declared one of the top 10 algorithms of the 20th century[2], fits the problem very well.

This report starts off with a lenghty introduction, before giving an overview of both the mathematical and the physical theory of the problem at hand. The above mentioned Metropolis algorithm is implemented and tested on the 2×2 grid, for which an analytical solution is also developed. The convergence of the Metropolis algorithm is then tested for a 20×20 grid, and the numerically approximated probability distribution is analysed. Finally, the phase transition is thoroughly studied for varying grid sizes, and the critical temperature is found.

2 Physical theory

2.1 The Ising model

As stated in the introduction, the Ising model describes a coupled system where nearest neighbours affect each other. In this project, the Ising model is applied to a two-dimensional magnetic material modelled as a grid or lattice of spins, where each spin s_i can have the values $+1$ or -1, denoted by $\uparrow$ and $\downarrow$ respectively. Assuming that all couplings are of equal magnitude, the coupling in the Ising model leads to the energy

$$E = -J \sum_{\langle ij \rangle} s_i s_j \tag{1}$$

where the sum is over the nearest neighbours. From this expression, we see that if J is positive, the energy is minimised when all spins are aligned in parallel, which is the behaviour of a ferromagnetic material. If, on the other hand, J is negative, a state where all spins are antiparallel to each other has the lowest energy, corresponding to an antiferromagnetic material.

2.1.1 Borders

The treatment of the borders of the lattice is a technicality which shows up when the lattice is not infinite. In the so-called thermodynamic limit, where the lattice is infinite, all spins are surrounded the same amount of other spins. With a finite lattice, however, this is not the case, which causes finite size effects.

One possible workaround is to ignore this problem completely, while a smarter alternative is to use what is commonly referred to as periodic boundary conditions. The idea behind this is to simulate an infinite lattice by assuming e.g. that the spins to the right of the right edge of the grid would be equal to the spins on the left edge. This approach has been implemented in ising.cpp.

2.2 Statistical physics

The basis for the statistical physics of this problem is the probability distribution used for the macrostates, which is the Boltzmann distribution. This distribution states that the probability of a system being in a state with energy E_i is proportional to $\mathrm{e}^{-\beta E_i}$, where $\beta = 1/kT$. This probability distribution needs to be normalised, and with the normalisation factor denoted $1/Z$, we get the equation

$$1 = \sum_i P(E_i) = \sum_i \frac{1}{Z} \mathrm{e}^{-\beta E_i} = \frac{1}{Z} \sum_i \mathrm{e}^{-\beta E_i} \implies Z = \sum_i \mathrm{e}^{-\beta E_i}$$

Z is called the partition function, and is perhaps the most important quantity in statistical physics, as it makes it possible to calculate most other thermodynamic quantities. The energy

can be calculated directly as

$$E = -\frac{\partial \ln(Z)}{\partial \beta}$$

In general, the expectation value of any thermodynamic quantity A can be calculated through

$$\langle A \rangle = \sum_i A_i P(E_i)$$

For the two-dimensional Ising model, we are primarily interested in the expectation values of the energy E and magnetization $|M|$, as well as the heat capacity C_V and magnetic susceptibility χ. The latter quantities can be calculated from the formulas[3]

$$C_V = \frac{1}{kT^2}\left(\langle E^2 \rangle - \langle E \rangle^2\right)$$
$$\chi = \frac{1}{kT}\left(\langle M^2 \rangle - \langle M \rangle^2\right)$$

2.3 Phase transitions

One of the great achievements of the two-dimensional Ising model is its ability to predict a phase transition in ferromagnetic materials, an ability which the one-dimensional version lacks. When a ferromagnetic material is heated, a temperature is reached where the materials suddenly start behaving paramagnetic, i.e. where the magnetic moment drops to zero.

An equally dramatic change, which is easier to analyse numerically, is that both the heat capacity and the magnetic susceptibility diverge when the temperature approaches the critical temperature. As such, the critical temperature can be found numerically as the point where the heat capacity and magnetic susceptibility have their largest value. I have chosen to use the heat capacity, as this settles down more quickly, yielding a smoother function of temperature. One can derive the following expression for the magnetic susceptibility as a function of temperature, which shows the divergence:

$$\chi(T) \propto |T_C - T|^{-\alpha}$$

where α is a positive constant, so $T = T_C$ leads to a division by zero. The heat capacity has a similar expression.

As the analytical expression for the two dimensional Ising model is known for the thermodynamic limit, it is interesting to attempt an approximation of the critical temperature in the thermodynamic limit based on the values found numerically for finite lattices. To achieve this, we have the relation

$$T_C(L) - T_C(L = \infty) = aL^{-1/\nu} \tag{2}$$

where a is some unknown constant and $\nu = 1$ can be calculated analytically. To find an expression for a, note that if we subtract the above equations for two values of L for which T_C has been calculated numerically, $T_C(L = \infty)$ cancels and we get

$$T_C(L_i) - T_C(L_j) = a\left(L_i^{-1/\nu} - L_j^{-1/\nu}\right) \implies a = \frac{T_C(L_i) - T_C(L_j)}{L_i^{-1/\nu} - L_j^{-1/\nu}}$$

The thermodynamic limit can then be calculated using equation (2):

$$T_C(L=\infty) = T_C(L_i) - aL_i^{-1/\nu} = T_C(L_i) - \frac{T_C(L_i) - T_C(L_j)}{L_i^{-1/\nu} - L_j^{-1/\nu}} \cdot L_i^{-1/\nu}$$

for some L_i. Using many, e.g. 4, values L_i and L_j, we should get a good approximation of the thermodynamic limit. This is calculated in findcritical.py, where the mean value is calculated.

3 Computational theory

In this project, the famous Metropolis algorithm is used to do a Monte Carlo simulation of the Ising model. General Monte Carlo methods are based on probabilities and randomness, combined with the law of large numbers, which states that the mean value of a series of experiments will converge towards the expectation value. The Metropolis algorithm is no exception.

As discussed in the introduction, it is not possible to calculate the partition function needed for the Boltzmann distribution. As such, the numerical method has to manage without the partition function. The method used in the Metropolis algorithm chooses to instead look at the ratios of the probabilities, as the partition function will then cancel. Assume that the system is in state i with energy E_i and considers jumping to state j with energy E_j. The ratio of the probabilities of being in these two states then becomes

$$\frac{P(E_j)}{P(E_i)} = \frac{\mathrm{e}^{-\beta E_j}/Z}{\mathrm{e}^{-\beta E_i}/Z} = \mathrm{e}^{-\beta(E_j - E_i)} = \mathrm{e}^{-\beta \Delta E}$$

When the energy decreases, $\Delta E < 0$ and the ratio becomes greater than 1. When the energy increases, the ratio similarly becomes less than 1. The Metropolis algorithm says that if the energy decreases, the proposed change of state should be accepted, otherwise it should be accepted with a probability $\mathrm{e}^{-\beta \Delta E}$. This balances the attractions towards a state with lower energy and a state with higher multiplicity (entropy), thus minimising Helmholtz's free energy, $F = E - TS$.

3.1 The Metropolis algorithm

Choosing to for each step flip one spin, the above discussion can be summarised as

1. Choose one of the spins at random.
2. Calculate what the change in energy will be if the spin is flipped.
3. Decide whether to flip the spin or not.
    - If the change in energy is negative: Flip the spin.
    - Else: Draw a random number in the interval $[0, 1]$. Flip the spin if the number is smaller than $\mathrm{e}^{-\beta \Delta E}$.
4. Update expectation values.

With L rows and L columns in the lattice, one Monte Carlo cycle denotes the process of L^2 attempts at flipping spins.

3.2 Technicalities

While it on paper is straight forward to calculate the energy difference and then $e^{-\beta\Delta E}$, calculating an exponential is an expensive computation compared to the flipping of spins. Fortunately, it can be shown[3] that in a two-dimensional lattice, only five different energy differences are possible, namely $-8J$, $-4J$, 0, $4J$, $8J$. As such, the exponentials can be precalculated. In order to calculate the energy difference itself as efficiently as possible, it can also be shown that when flipping spin j,

$$\Delta E = 2Js_j \sum_{\langle k \rangle} s_k$$

where s_j is the spin *before* the flip and the sum is over the nearest neighbours of spin j. Similarly, the change in magnetic moment can be calculated as

$$\Delta M = -2Js_j$$

where s_j again is the spin *before* the flip. Alternatively, the spin can be flipped before the change in magnetic moment is calculated with the opposite sign in the formula above.

4 Implementation

The Metropolis algorithm discussed above has been implemented fairly straightforwardly in ising.cpp. phases.cpp is a utility to run simulations as in section 5.4, to which the number of simulations, number of temperatures, etc. can be given as command line arguments. findcritical.py can then be used to analyse the resulting data file and find the critical temperatures.

To save runtime, certain parts of the programs are run in parallel. The Metropolis algorithm, like most Monte Carlo methods, are generally well suited for parallel programming, as many threads can work on different parts of the problem at the same time, without communicating.

In order to simply do separate parts of the project in parallel, all C++ programs use the `<thread>` library from the `C++11` standard. The basic syntax is to create a `thread` object, where the function to be called and the arguments to the function are given to the constructor, followed by a call to the thread's `join` method when one wishes to wait for the execution to finish. A simple example from analysis.cpp is

Code snippet 1: Example of parallel programming using `<thread>`.

```
thread hot(ising,hotfile,"random",L,N,dN,hotT,startindex,hotres);
thread cold(ising,coldfile,"random",L,N,dN,coldT,startindex,coldres);
thread orderedhot(ising,orderedhotfile,"o",L,N,dN,hotT,startindex,
    hotres2);
thread orderedcold(ising,orderedcoldfile,"o",L,N,dN,coldT,startindex,
    coldres2);

hot.join();
cold.join();

thread hotpdf(findpdf,"hotanalysis.dat","hotpdf.dat");
thread coldpdf(findpdf,"coldanalysis.dat","coldpdf.dat");

orderedhot.join();
orderedcold.join();
hotpdf.join();
coldpdf.join();
```

In this snippet, four different Monte Carlo simulations are started as separate threads. Then, the program waits for two of them to finish before starting two other threads which analyse the results from these two. Finally, the program waits for all threads to finish.

I was originally planning to use the `OpenMPI` library for parallelisation of the phase transition study, but time constraints have forced me to stick with `<thread>`. As it has not been necessary to split the computations over separate machines, this has not been a problem. It would, however, have enabled the programs to run on several of the computers connected to the HTCondor service at once, which would have been an advantage for the heavy simulations.

The servers at the Department of Informatics as well as their HTCondor service have been used to run the longer simulations in section 5.4.

5 Results

All numerical values are given in units of J and k.

5.1 Test of parallelisation

In timetest.cpp, two Monte Carlo simulations are first run sequentially and then in parallel. If the parallelisation has been successful, the parallel run should take only about half the time of the sequential run, as the two threads can work completely separate of each other.

One technical issue of checking parallelisation is that the normal `clock` method does not work - this counts CPU cycles, which is the same for the sequential and parallel runs. As the main interest in this test is to get a rough estimate rather than exact runtime, I have chosen to simply use the `time` and `difftime` functions, which gives an integer amount of second. The results are

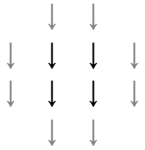
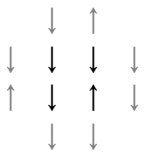
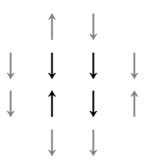
Sequential time: 12 s Parallel time: 6 s

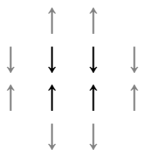
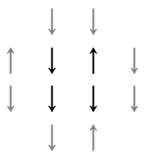
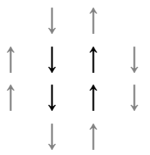
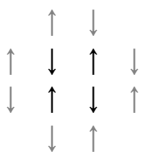
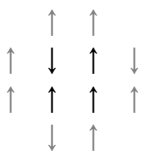
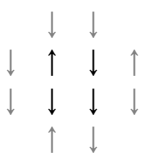
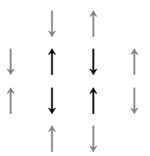
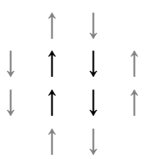
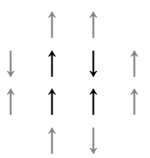
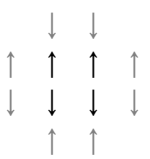
confirming the expected speedup factor of 2.

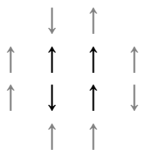
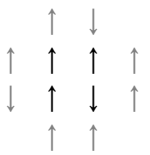
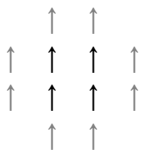
5.2 Numerical and analytical results for a 2×2 lattice

5.2.1 Analytical result

For a 2×2 lattice, the thermodynamic quantities can be found analytically without too much work. To find the partition function, we need to write out all possible microstates and calculate their energies. Using the periodic boundary conditions, we get

Microstate	Energy	Magnetic moment
	$E = -8J$	$M = -4$
	$E = 0$	$M = -2$
	$E = 0$	$M = -2$

Microstate	Energy	Magnetic moment
	$E = 0$	$M = 0$
	$E = 0$	$M = -2$
	$E = 0$	$M = 0$
	$E = 8J$	$M = 0$
	$E = 0$	$M = 2$
	$E = 0$	$M = -2$
	$E = 8J$	$M = 0$
	$E = 0$	$M = 0$
	$E = 0$	$M = 2$
	$E = 0$	$M = 0$

Microstate	Energy	Magnetic moment
	$E = 0$	$M = 2$
	$E = 0$	$M = 2$
	$E = -8J$	$M = 4$

To summarise, we have

Number of ↑	Multiplicity	Energy	Magnetic moment
4	1	$-8J$	4
3	4	0	2
2	2	$8J$	0
2	4	0	0
1	4	0	-2
0	1	$-8J$	-4

Summing over all microstates, we get the partition function

$$Z = \sum_{\substack{\text{all}\\ \text{micro-}\\ \text{states}}} \mathrm{e}^{-\beta E_i} = 2\mathrm{e}^{8J\beta} + 2\mathrm{e}^{-8J\beta} + 12$$

The expectation value of the energy can then be found from

$$\begin{aligned}\langle E\rangle &= -\frac{\partial \ln(Z)}{\partial\beta} = -\frac{\partial}{\partial\beta}\left(\ln\left(2\mathrm{e}^{8J\beta} + 2\mathrm{e}^{-8J\beta} + 12\right)\right)\\ &= -\frac{1}{2\mathrm{e}^{8J\beta} + 2\mathrm{e}^{-8J\beta} + 12}\left(16J\mathrm{e}^{8J\beta} - 16J\mathrm{e}^{-8J\beta}\right)\\ &= -\frac{8J\left(\mathrm{e}^{8J\beta} - \mathrm{e}^{-8J\beta}\right)}{\mathrm{e}^{8J\beta} + \mathrm{e}^{-8J\beta} + 6}\end{aligned}$$

yielding the heat capacity

$$C_V = \frac{\partial}{\partial T}(\langle E\rangle) = \frac{\partial\beta}{\partial T}\frac{\partial}{\partial\beta}(\langle E\rangle) = -\frac{1}{kT^2}\frac{\partial}{\partial\beta}(\langle E\rangle)$$

$$
\begin{aligned}
&= -\frac{1}{kT^2}\frac{\partial}{\partial\beta}\left(-\frac{8J\left(e^{8J\beta}-e^{-8J\beta}\right)}{e^{8J\beta}+e^{-8J\beta}+6}\right)\\
&= \frac{8J}{kT^2}\frac{8J\left(e^{8J\beta}+e^{-8J\beta}\right)(e^{8J\beta}+e^{-8J\beta}+6)-\left(e^{8J\beta}-e^{-8J\beta}\right)8J\left(e^{8J\beta}-e^{-8J\beta}\right)}{\left(e^{8J\beta}+e^{-8J\beta}+6\right)^2}\\
&= \frac{64J^2}{kT^2}\frac{6\left(e^{8J\beta}+e^{-8J\beta}\right)+\left(e^{8J\beta}+e^{-8J\beta}\right)^2-\left(e^{8J\beta}-e^{-8J\beta}\right)^2}{\left(e^{8J\beta}+e^{-8J\beta}+6\right)^2}\\
&= \frac{64J^2}{kT^2}\frac{6\left(e^{8J\beta}+e^{-8J\beta}\right)+e^{16J\beta}+2+e^{-16J\beta}-\left(e^{16J\beta}-2+e^{-16\beta}\right)}{\left(e^{8J\beta}+e^{-8J\beta}+6\right)^2}\\
&= \frac{64J^2}{kT^2}\frac{6\left(e^{8J\beta}+e^{-8J\beta}\right)+4}{\left(e^{8J\beta}+e^{-8J\beta}+6\right)^2}
\end{aligned}
$$

To find the various quantities connected to magnetisation, we use the general formula

$$\langle A\rangle = \frac{1}{Z}\sum_{\substack{\text{all}\\ \text{micro-}\\ \text{states}}} A_i e^{-\beta E_i}$$

Mean magnetic moment(s):

$$
\begin{aligned}
\langle M\rangle &= \frac{1}{2e^{8J\beta}+2e^{-8J\beta}+12}\left(-4e^{8J\beta}+4\cdot(-2)+0+0+4\cdot 2+4e^{8J\beta}\right) = 0\\
\langle M^2\rangle &= \frac{1}{2e^{8J\beta}+2e^{-8J\beta}+12}\left((-4)^2e^{8J\beta}+4\cdot(-2)^2+0+0+4\cdot 2^2+4^2e^{8J\beta}\right)\\
&= \frac{32\left(e^{8J\beta}+1\right)}{2e^{8J\beta}+2e^{-8J\beta}+12} = \frac{16\left(e^{8J\beta}+1\right)}{e^{8J\beta}+e^{-8J\beta}+6}\\
\langle |M|\rangle &= \frac{1}{2e^{8J\beta}+2e^{-8J\beta}+12}\left(|-4|e^{8J\beta}+4\cdot|-2|+0+0+4\cdot 2+4e^{8J\beta}\right)\\
&= \frac{8\left(e^{8J\beta}+2\right)}{2e^{8J\beta}+2e^{-8J\beta}+12} = \frac{4\left(e^{8J\beta}+2\right)}{e^{8J\beta}+e^{-8J\beta}+6}
\end{aligned}
$$

Magnetic susceptibility:

$$\chi = \beta\left(\langle M^2\rangle - \cancel{\langle M\rangle^2}\right) = \frac{16\beta\left(e^{8J\beta}+1\right)}{e^{8J\beta}+e^{-8J\beta}+6}$$

5.2.2 Numerical results

With $J=1$ and $\beta=1$, the above expressions give

$$
\begin{aligned}
\langle E\rangle/L^2 &= -1.99598 & C_V/L^2 &= 0.03208\\
\langle |M|\rangle/L^2 &= \ \ 0.99866 & \chi/L^2 &= \ \ 3.9933
\end{aligned}
$$

Running 100 000 Monte Carlo cycles with the same parameters gives figure 1 on the following page.

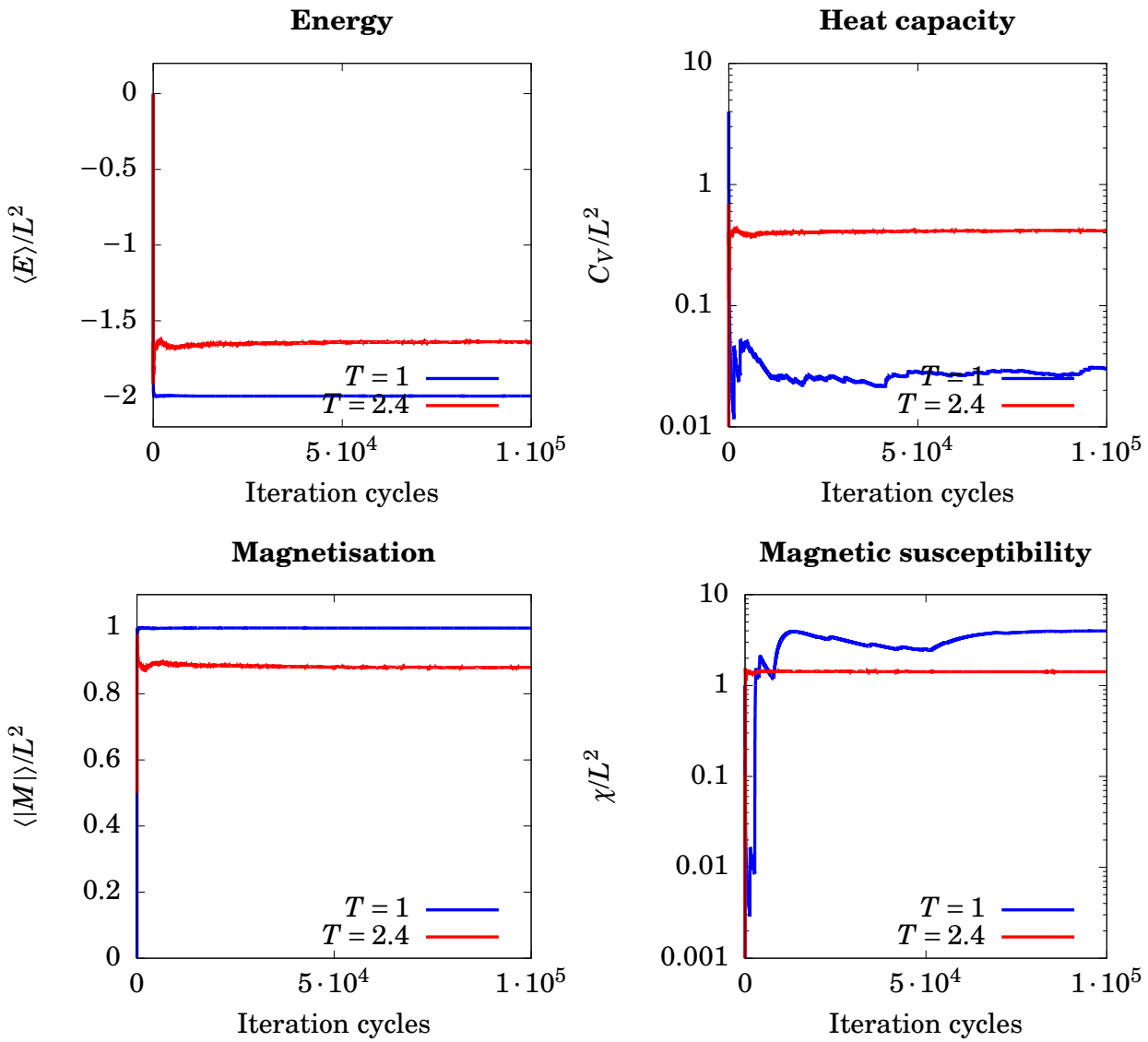


Figure 1: Numerical results from 100 000 Monte Carlo cycles. Generated by 2by2.cpp.

These results fit fairly well with the numbers calculated above, but we see from the graphs that the values for the heat capacity and magnetic susceptibility have yet to settle down completely. Running 10 000 000 Monte Carlo cycles yields

$$\langle E\rangle/L^2 = -1.99588 \qquad C_V/L^2 = 0.0328746 \qquad \langle|M|\rangle/L^2 = 0.998626 \qquad \chi/L^2 = 3.99245$$

These results fit very well with the analytical results.

5.3 Analysis

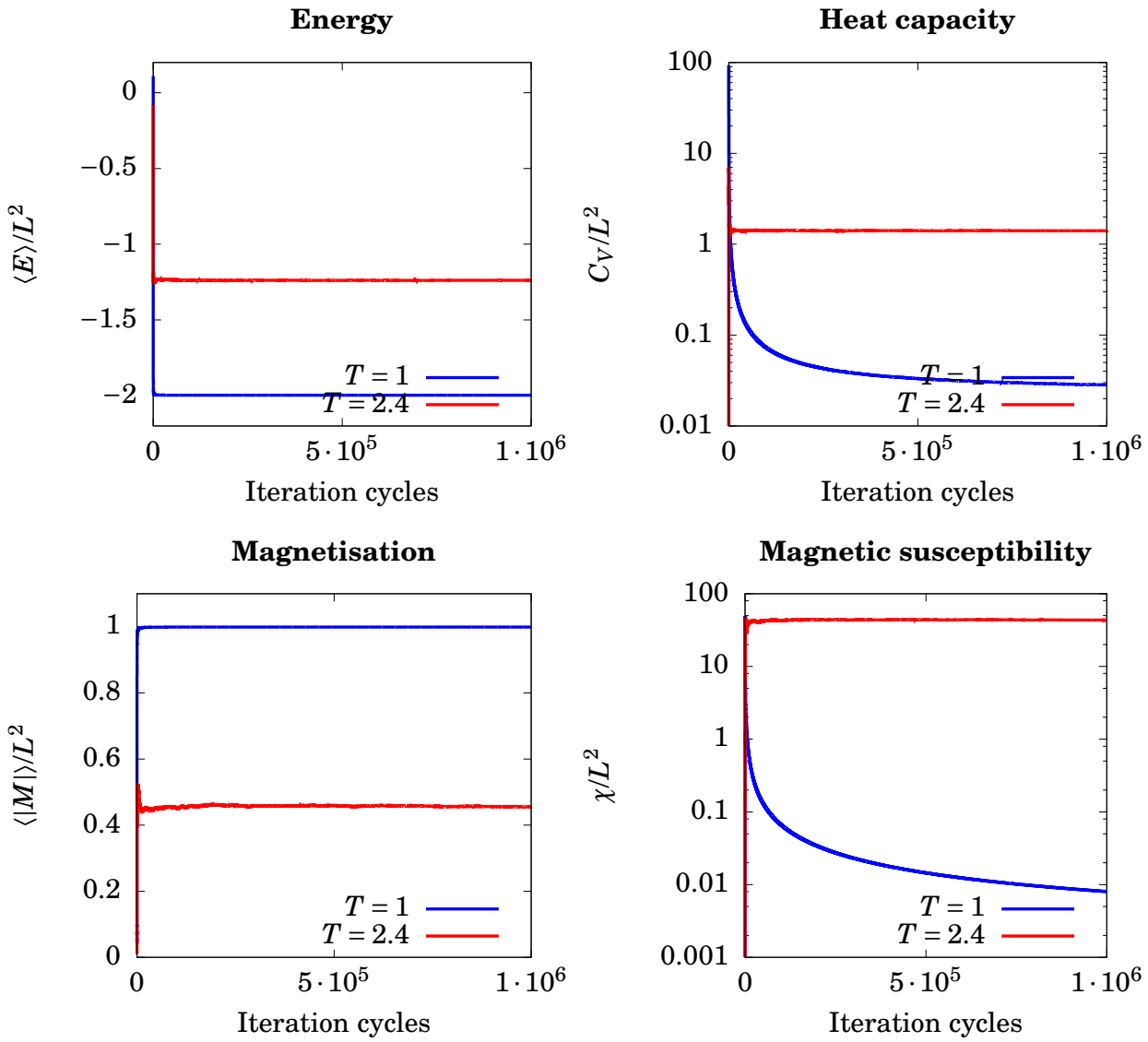


Figure 2: Simulation for 10^6 Monte Carlo cycles with two different temperatures and random initial states. Generated by analysis.cpp.

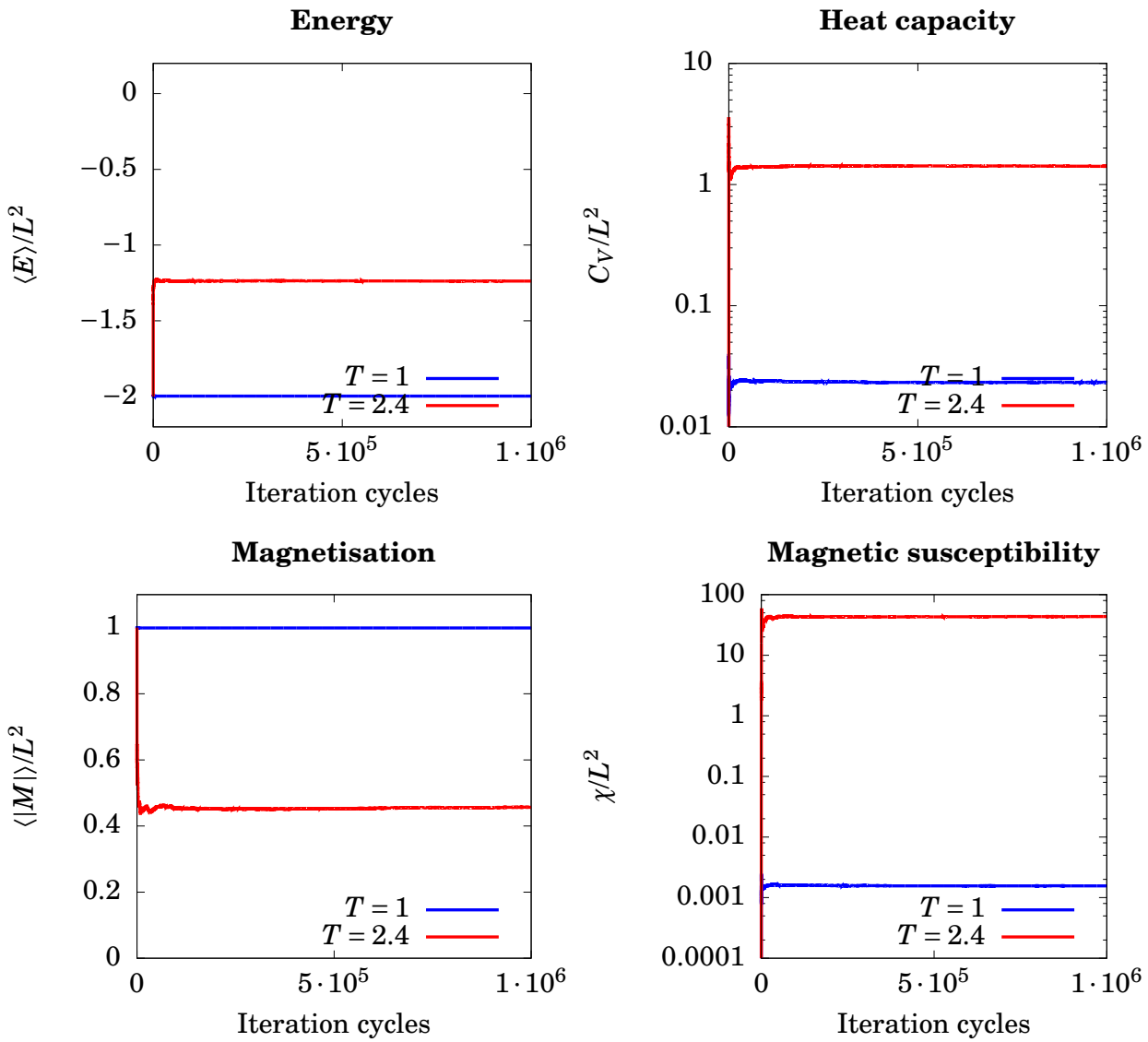


Figure 3: Simulation for 10^6 Monte Carlo cycles with two different temperatures and all spins initially pointed upwards. Generated by analysis.cpp.

Comparing figure 2 and figure 3, we see that the initial state has very little to say for the final values, as well as the time it takes to reach a steady state[1].

[1]Note that the y-axis is logarithmic in the plots for heat capacity and magnetic susceptibility, thus greatly exaggerating the difference in choice of initial state.

5.3.1 Number of accepted flips

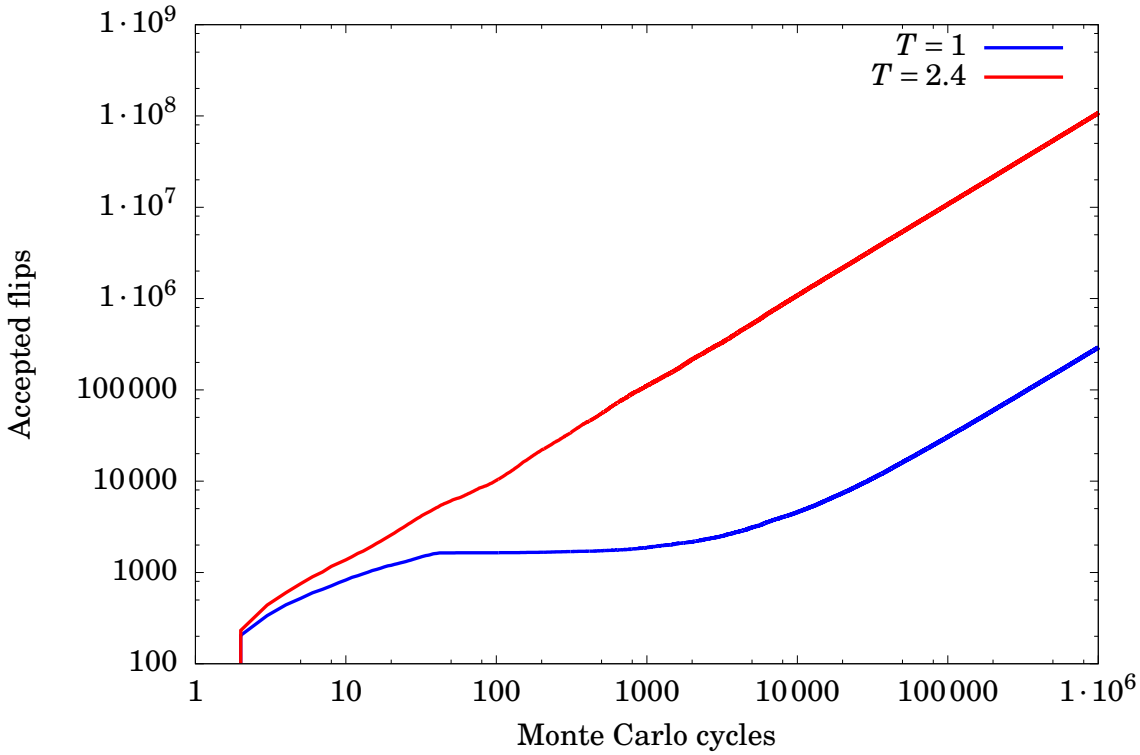


Figure 4: The number of accepted flips as a function of the number of Monte Carlo cycles, on a logarithmic scale. Generated by analysis.cpp and accepted.gpi.

As the number of accepted flips should grow linearly as a function of the number of Monte Carlo cycles when the system is in equilibrium, we see that 10 000 cycles seems sufficient to reach a steady state, as the graphs are approximately linear with a derivative of 1 from this point.

5.3.2 Numerical probability distribution

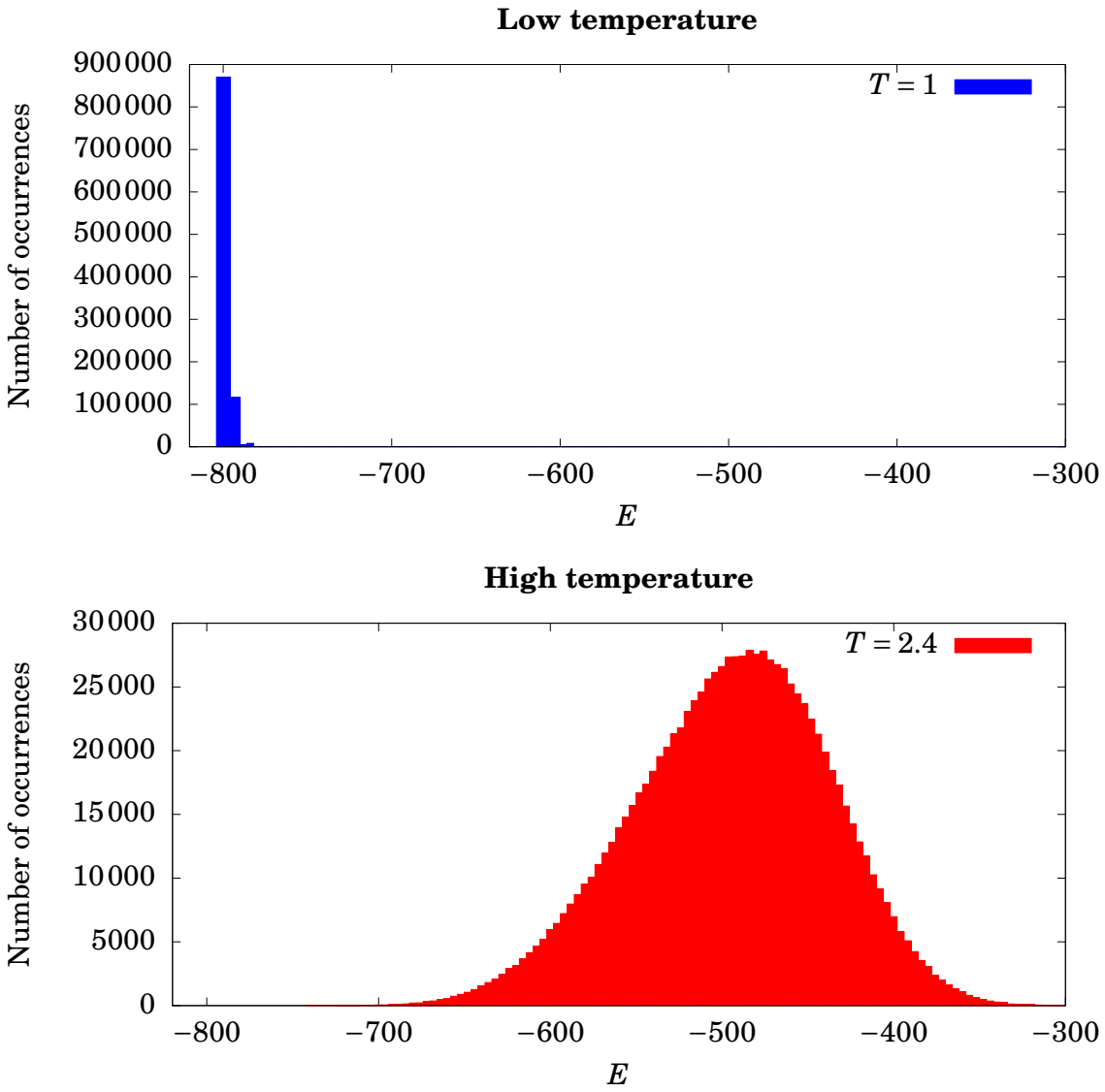


Figure 5: Counting the ocurrences of the different energy levels. For practical reasons, the counting includes all Monte Carlo cycles. This has negligible effect on the resulting histograms. Generated by analysis.cpp and probdist.gpi.

When the temperature is higher, it is expected that the mean energy is higher, and also that the energy distribution is more spread out and more states are visited. This fits very well with the histograms above. The number of visited energy levels also fits with figure 4 on the previous page, where the number of accepted changes of state is much higher when the temperature is higher, giving a larger variance and heat capacity, as seen in figure 2 on page 15.

5.4 Discovering a phase transition

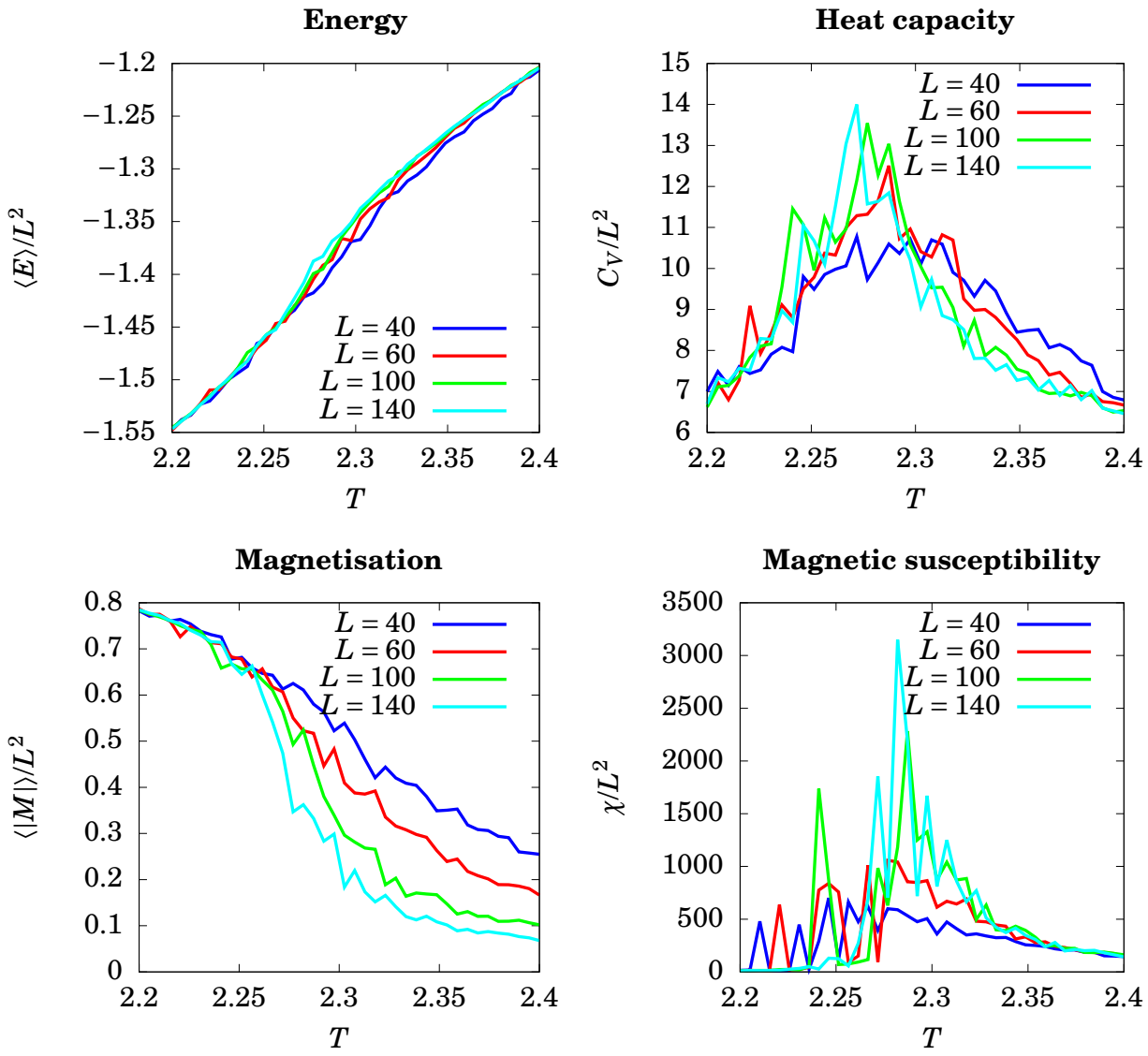


Figure 6: Simulation for 40 different temperatures with 100 000 Monte Carlo cycles (after skipping 20 000). Run by condorquick.cmd on the HTCondor system and generated by phases.cpp.

From the figure, we see that the heat capacity magnetic susceptibility peak at a temperature around 2.3. Unfortunately, it is clear that more data is required in order to get reasonable results.

The above graph of the susceptibility gives the following maxima for the heat capacity, ideally

corresponding to the critical temperatures:

L	T_C
40	2.271 79
60	2.287 18
100	2.276 92
140	2.271 79

Approximation to the thermodynamic limit:

$$T_C(L = \infty) = 2.275\,14$$

This was calculated by findcritical.py.

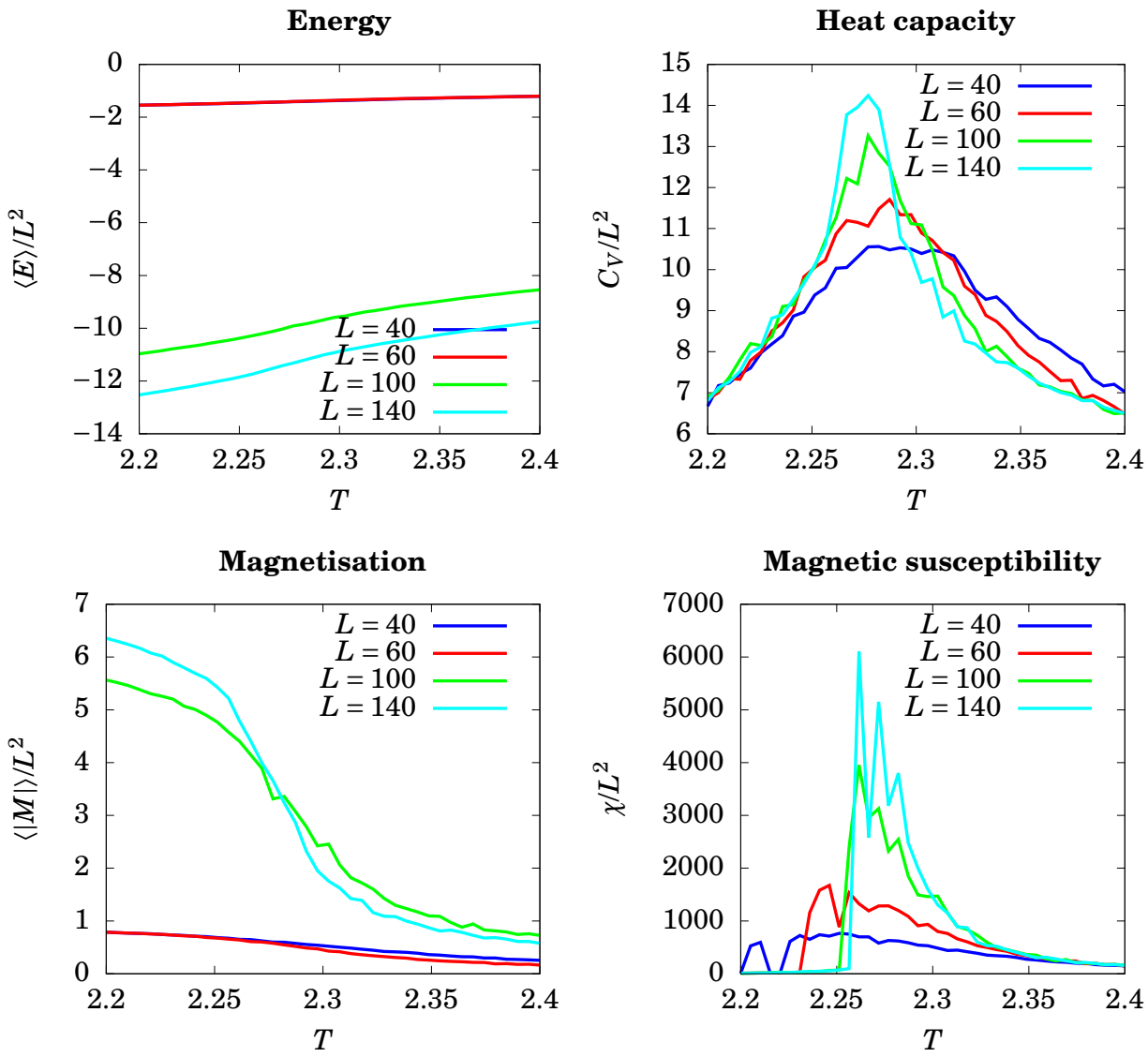


Figure 7: Simulation for 40 different temperatures with 500 000 Monte Carlo cycles (after skipping 20 000). Run by condormedium.cmd on the HTCondor system and generated by phases.cpp. The runtime was approximately 4 hours.

With five times as many Monte Carlo cycles, the data look a bit better and have a lot less noise, while still not super.

The above graph of the susceptibility gives the following maxima for the heat capacity, ideally

corresponding to the critical temperatures:

L	T_C
40	2.28205
60	2.28718
100	2.27692
140	2.27692

Approximation to the thermodynamic limit:

$$T_C(L=\infty) = 2.27558$$

This was calculated by findcritical.py. This is very close to Lars Onsager's analytical result of $T_C(L=\infty) = 2.269$. With 40 temperatures in the interval from 2.2 to 2.4, this is nearing the maximum precision possible.

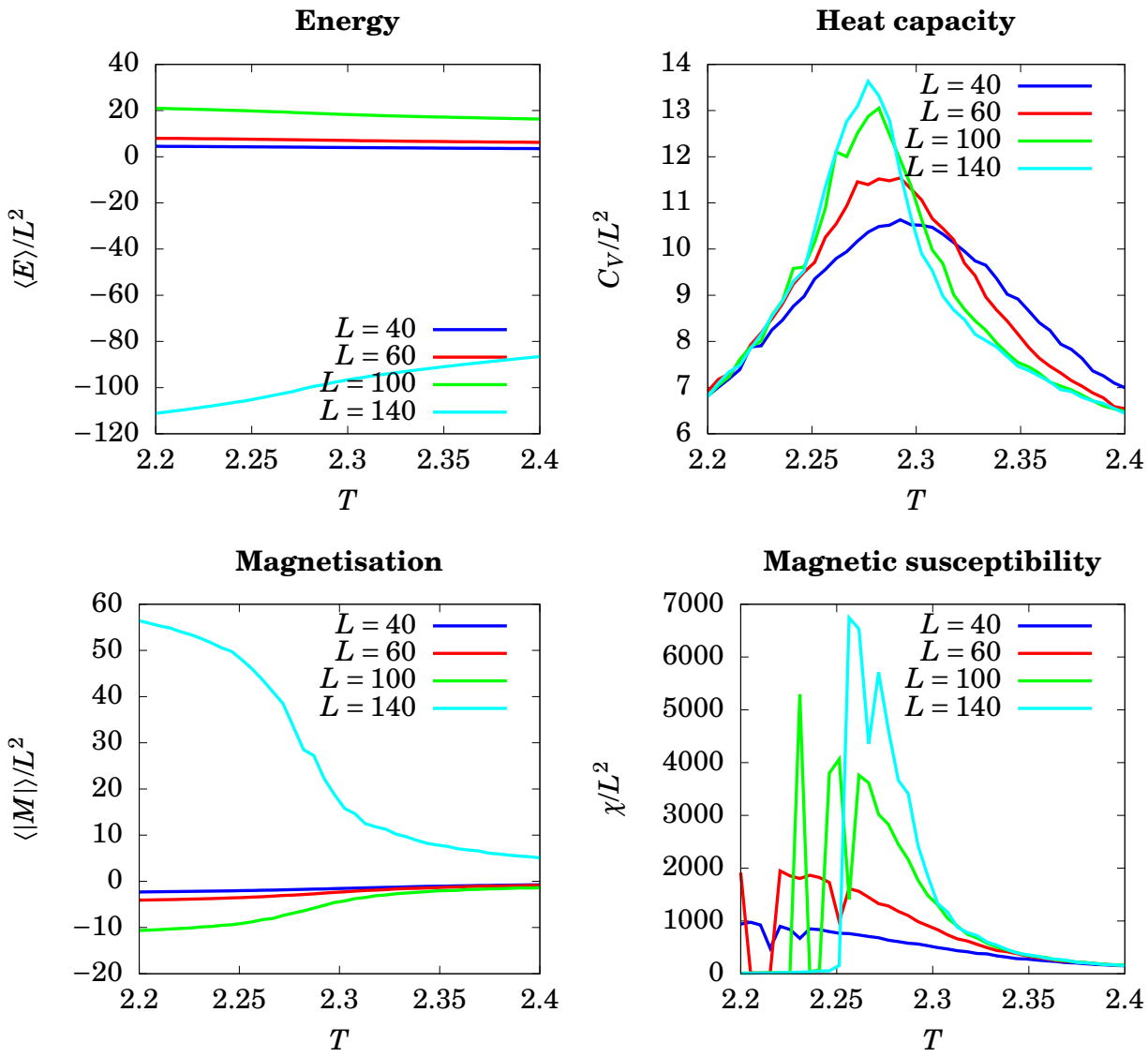


Figure 8: Simulation for 40 different temperatures with $2 \cdot 10^6$ Monte Carlo cycles (after skipping 20 000). Run on a UiO server (with 64 cores) and generated by phases.cpp. The runtime was approximately 15.5 hours.

With this many Monte Carlo cycles and a ginormous runtime, I was hoping for the graph of the magnetic susceptibility to look considerably better. The heat capacity, on the other hand, has improved, and this is the quantity used to calculate the critical temperature.

L	T_C
40	2.29231
60	2.29231
100	2.28205
140	2.27692

Approximation to the thermodynamic limit:

$$T_C(L = \infty) = 2.2724$$

This was calculated by findcritical.py. While the graphs don't look particularly pretty, the approximation to the thermodynamic limit is even closer to Lars Onsager's analytical result of $T_C(L = \infty) = 2.269$, and the distance is approximately equal to the best achievable accuracy with this choice of temperatures.

Unfortunately, the values for the expectations values for the energy and magnetisation in figure 7 and figure 8 are quite clearly wrong, which probably stems from these runs not being done with the latest versions of the programs due to the runtime (the simulation generating the data in figure 8 finished just two hours before the deadline). The critical temperatures are, however, not affected by this. I believe the error was caused by how the "skip the first steps" ability was implemented, as the implementation was changed without certain lines being updated, leading to a division by the wrong amount of Monte Carlo cycles. I may eventually redo these runs and upload the updated report on GitHub.

6 Conclusion

In this project, the magnetic properties of a ferromagnetic material has been studied through the Ising model. We started off with the comparison of numerical and analytical results for a 2×2 lattice, where it was shown that the results from the numerical simulations are indeed satisfactory. Then, the behaviour of the system over the course of the simulation was studied for two different temperatures, including the path towards a steady state. Finally, the various thermodynamic quantities were studied as functions of temperature using data from heavy simulations, and a critical temperature was found - both for the finite systems studied numerically and for the thermodynamic limit of an infite lattice. The results agree with the analytical result $T_C = 2.269$ found by Lars Onsager.

The main idea for further work is to do heavier simulations of the phase transitions, in order to get a smoother function of temperature with less noise, and thus a more precise value for the critical temperature in the thermodynamic limit. I did unforunately not have time to do this. This probably requires the change from `<thread>` to `OpenMPI` as a mean for parallelisation, as `OpenMPI` is a message parsing interface, enabling the programs to be run on multiple machines. On HTCondor, this would allow the program to run on many desktop machines at once.

References

[1] James Burridge and Steven Kenney. “Birdsong dialect patterns explained using magnetic domains”. In: *Physical Review E* 93.6 (2016), p. 062402.

[2] Barry A Cipra. “The best of the 20th century: editors name top 10 algorithms”. In: *SIAM news* 33.4 (2000), pp. 1–2.

[3] Morten Hjorth-Jensen. *Computational Physics*. Lecture notes. 2015. URL: https://github.com/CompPhysics/ComputationalPhysics/blob/master/doc/Lectures/lectures2015.pdf.