FYS4150 - Project 4
Crafting and Investigating the Ising Model

Daniel Herman

November 19, 2017

Abstract

The Ising model is a commonly utilized model of statistical systems. In this project, we create our own Ising model using the programming language c++. The code is parallelized using OpenMPI, and python programs were written to extract and plot data output by the c++ program. The Monte Carlo Markov Chain is used to cycle through our model, and accepted changes to the model are determined using the Metropolis algorithm. Both of these methods utilize random number generators picking from a uniform distribution. This approach has proved very successful for run times of at least 10^6 Monte Carlo cycles, showing convergence to the analytical solutions predicted by statistical mechanics.

1 Introduction

The Ising Model is a common way to analyze phase transitions for physical phenomena. In this project, we look at the two-dimensional Ising Model. For this specific case, we're looking at the phase transition between a magnetized state, and a state with no magnetization. Each lattice site is binary, in that it can take one of two possible states, namely spin up, or spin down. In the initial state, all of the lattice site spins are either aligned, giving a fully magnetized system, or randomly distributed. As the temperature of the system changes, the spin orientation of each lattice site has a potential to change as determined by the Boltzmann distribution described in section 2. The total energy of the lattice at any given moment is given by the simple relation

$$E = -J \sum_{<kl>}^{N} s_k s_l \tag{1}$$

where $s_k = \pm 1$. N represents the total number of spins and J is a coupling constant that expresses the strength of the interaction between neightboring spins. For ease, we scale our calculations by setting $J = 1$. The lower index of the sum $< kl >$ states that we only sum over the nearest neighbors.
We investigate the two-dimensional case of the Ising model specifically as it has analytical expressions for the expectation values of the quantities which we study through our simulation. We can utilize these analytical expressions to compare with our results in section 4.

2 Methods

In order to understand the results of our Ising model simulation, we have to investigate the analytical expressions of the Ising model as well as Monte Carlo cycles and the Metropolis algorithm.

2.1 The Ising Model

We use analytic expressions of the Ising model to help understand the behavior of the model, and verify the results of our code. We find analytic expressions for the energy expectation value $< E >$, the mean absolute value of the magnetic moment (mean magnetizaion) $< |M| >$, the specific heat C_V and the susceptibility χ as functions of the temperature T. In order to find these expressions, we first must clarify some information about the Ising model.

2.2 Probability Distribution Function (PDF).

The energy of the system is described in equation 1. The probability of finding an energy E_i at a given temperature T is given by the Boltzmann distribution

$$P_i(T) \propto e^{-E_i/k_B T} \tag{2}$$

For simplification, we redefine the quantity $\dfrac{1}{k_B T} = \beta$, and scale the distribution by setting $k_B = 1$. In order to find the actual probability value for each energy, we introduce a normalization constant

$$P_i(\beta) = \frac{e^{-E_i \beta}}{Z} \tag{3}$$

where $Z = \sum_{i=1}^{M} e^{-E_i \beta}$ is the partition function of the distribution. M is the number of possible spin configurations.

2.3 Analytical Expectation Values.

Now we can calculate the expectaion value for the energy in the typical fashion

$$< E >= \sum_{i=1}^{M} E_i P_i(\beta) \tag{4}$$

We utilize the PDF again to determine the mean magnetization as well, using the definition of the magnetization of a state $M_i = \sum_{i=1}^{d} s_i$ where $s_i = \pm 1$ is the value of spin i.

$$< |M| >= \sum_{i=1}^{M} M_i P_i(\beta) \tag{5}$$

Similarly, we calculate the values $< E^2 >$ and $< M^2 >$ so that we can determine the variance σ^2 of the energy and magnetization values.

$$\begin{aligned} < E^2 >&= \sum_{i=1}^{M} E_i^2 P_i(\beta) \\ < M^2 >&= \sum_{i=1}^{M} M_i^2 P_i(\beta) \\ \sigma_E^2 &= < E^2 > - < E >^2 \\ \sigma_M^2 &= < M^2 > - < M >^2 \end{aligned}$$

2.4 Analytical Specific Heat and Susceptibility.

Using some statistical mechanics tricks, we can drive the specific heat C_V and magnetic susceptibility χ expressions given below

$$C_V = \frac{\sigma_E^2}{k_B T^2} \tag{6}$$

$$\chi = \frac{\sigma_M^2}{k_B T} \tag{7}$$

2.5 Computational Energy Expectation Value.

However, as the number of spins increases, calculating the partition function becomes more and more taxing. So in our case, instead of attempting to calculate the partition function, we average the energies found after each Monte Carlo (MC) cycle by dividing by the number of MC cycles N as such.

$$< E >\simeq \frac{1}{N} \sum_{i=1}^{N} E_i, < M >\simeq \frac{1}{N} \sum_{i=1}^{N} M_i \tag{8}$$

$$< E^2 >\simeq \frac{1}{N} \sum_{i=1}^{N} E_i^2, < M^2 >\simeq \frac{1}{N} \sum_{i=1}^{N} M_i^2 \tag{9}$$

2.6 Phase Transitions.

When the system approaches a phase transition, the physical quantities expressed above evolve as a function of the temperature difference between the systems temperature T, and the critical temperature T_C. These relations are found in [1]. Important quantities change as follows

$$< M(T) > \sim (T - T_C)^{\beta} \tag{10}$$
$$C_V(T) \sim |T_C - T|^{\alpha} \tag{11}$$
$$\chi(T) \sim |T_C - T|^{\gamma} \tag{12}$$

Where the experimental values are $\beta = 1/8$, $\alpha = 0$, and $\gamma = 7/4$.

For our model the functionality of the above relations is dependent upon the correlation length, which charaterizes the strength correlation between different part of the lattice. When the temperature approaches T_C, the spins become more correlated with one another and the correlation length should span the entire lattice. The behavior of ζ near T_C is

$$\zeta(T) \sim |T_C - T|^{-\nu} \tag{13}$$

As a result of this change in correlation length, we can relate the behavior of finite lattices to the results from an infinitely large lattice. We see that the critical temperature scales according to the size of the lattice L by

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

Substituting this back into 10, where we set $T = T_C$,

$$< M(T) > \sim (T - T_C)^{\beta} \rightarrow L^{-\beta/\nu}$$
$$C_V(T) \sim |T_C - T|^{-\alpha} \rightarrow L^{\alpha/\nu}$$
$$\chi(T) \sim |T_C - T|^{-\gamma} \rightarrow L^{\gamma/\nu}$$

2.7 Monte Carlo Cycles

The Monte Carlo method is a very popular way to solve complex probabilistic problems not only in physics but in other fields such as economics. It is often used when the changes of the system simulated are not deterministic in nature. Monte Carlo methods use randomly selected variables from a probability distribution to make changes in the system. This is particularly useful in the Ising model simulation as the behavior over time is statistical in nature.

2.8 Metropolis Algorithm

The Metropolis algorithm was crafted to estimate distribution functions (PDF1) by making random samples based off of a different distribution (PDF2) which is proportional to the distribution function we want to find. For each iteration of the algorithm, the random sample is not always accepted. The move is accepted or denied based off of PDF1. If the random move selected moves towards a position with higher probability then it is accepted. This will quickly cause the distribution of random samples to look like PDF1.

2.9 Code Parallelization

The code we run is not very memory intensive, however it makes many many calculations per second, especially as the lattice size increases. Because our analysis is statistical in nature, and our code is random in nature, we want to look at many iterations of our code for different lattice sizes. We want to run each lattice with 10^6 MC cycles many times to be able to study the behavior of our model. In order to save time with our calculations, we will use many processors simultaneously. We utilize OpenMPI, which is an open source Message Passing Interface to parallelize our code. Our code is simple to parallelize, because the processors do not need to communicate with each other while the code is running. Each processor can run the code independently. The output files are sent to the master compter for analysis.

3 Implementation

Code is written in c++ to create the Ising model and can be found at https://github.com/hermda02/Project4. The Monte Carlo method and the Metropolis algorithm are applied in the code. I will describe the behavior of these methods for our specific case below.

3.1 Monte Carlo Cycles

For our model to behave in a realistic matter, we cannot expect the flipping of spins to be deterministic. We implement the Monte Carlo Markov Chain to simulate a random flipping of spins. In order to do that, we create a loop over all the spins in the lattice, picking one randomly using a randomizer picking from a normal distribution. Since we have a two dimensional lattice, first we pick a random row, then a random column. We determine whether or not the spin is flipped based off of the Metropolis algorithm. For each Monte Carlo cycle, we randomly pick N spins, one by one, where N is the total number of spins in the lattice. This means that the same spin could potentially be flipped multiple times, and some spins may never be selected during the course of a Monte Carlo cycle.

3.2 Metropolis Algorithm

Once a spin has been selected, the current energy of that spin E is calculated. If the spin were to be flipped, the change in energy of the system would be equal to $2 * E$ (i.e. if the spin has an energy $E = 4$, when it is flipped, the energy becomes $E = -4$, so the change in energy $|\Delta E| = 8$). This value ΔE is then used to calculate the probability of the spin changing, according to the Boltzmann distribution 3. The use of the Boltzmann distribution is handy, as the maximum value P_i can take is when $\Delta E = 0$, where $P_i = 1$. A random number, x is then picked from a normal distribution ($x \in [0, 1]$), and if $x \leq P_i$, then the spin is flipped. This guarantees that as the model evolves, the lattice moves towards the most probabilistic state.

3.3 Parallelization and Optimization

We utilize OpenMPI in our c++ code. At the Institute for Theoretical Astrophysics (ITA), we have access to many super computer clusters. We use our access to the beehive cluster to run our code quickly and efficiently. This is optimal for producing large data sets necessary for in depth analysis of our model. More information on ITA's computing resources can be found at: http://www.mn.uio.no/astro/english/services/it/help/basic-services/compute-resources.html.

4 Results

To understand the behavior of our Ising model, we investigate lattices of different sizes L and at different temperatures T. All lattices that we investigate are square (L x L). To verify that our code behaves as we would like, we compare the results from the $L = 2, T = 1$ case to the analytical expressions for this case. According to our analytical expressions, if the code is working as expected we would see $< E >= -2$ (minimized energy), and we would expect the system to remain in and ordered magnetic state $< |M| >= 1$. According to our results, our code is behaving as expected, with an ordered final state $< |M| >= 1$ and minimized energy $< E >= -2$. The number of Monte Carlo cycles it takes for our code to reach a steady state is very small ($< 10^3$) in this case, which is expected as there are only 4 potential spin flips per MC cycle. The analytical expressions becomes very difficult to calculate as the size of the lattice increases, so future analysis will focus more on comparing trends rather than specific values.

4.1 Number of MC Cycles until Convergence

To better understand the number of MC cycles it takes for the lattice to reach a steady state, we look at a larger case. We set $L = 20$, and look at the $T = 1$ and $T = 2.4$ case. The evolution of $< E >$ as a function of MC cycles is shown below.

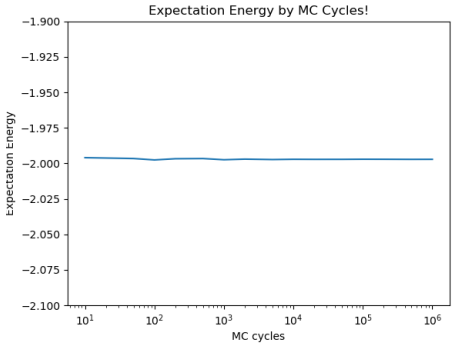

Figure 1: $< E >$ as a function of MC cycles for $T = 1.0$, with an ordered initial state.

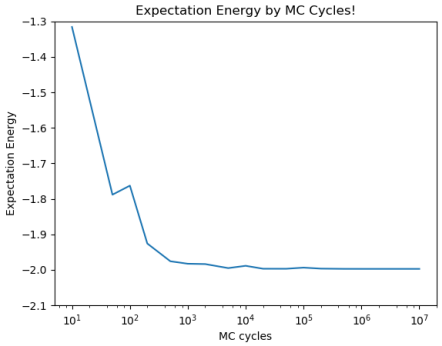

Figure 2: $< E >$ as a function of MC cycles for $T = 1.0$, with an random initial state.

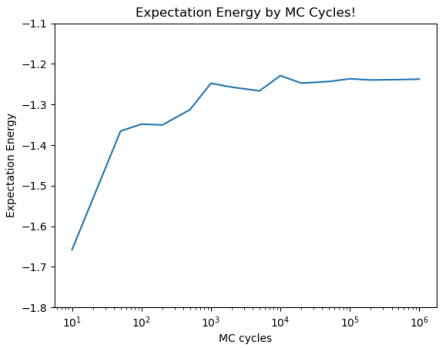

Figure 3: $< E >$ as a function of MC cycles for $T = 2.4$, with an ordered initial state.

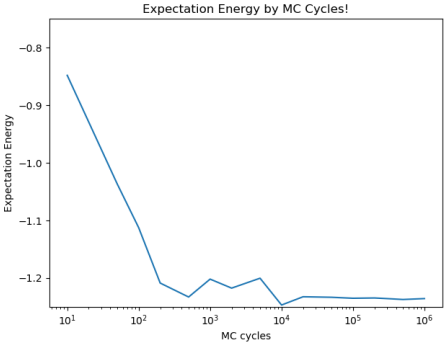

Figure 4: $< E >$ as a function of MC cycles for $T = 2.4$, with an random initial state.

As seen in figures 1, 2, 3 and 4, it takes between 10^5 and 10^6 MC cycles to reach an equilibrium state. Knowing this, we will use 10^6 MC cycles for the remainder of our calculations, to ensure that our simulations reach a steady state. The figures above only show the energy, yet the same converging behavior is observed for the other physical quantities.

For each MC cycle, there are N^2 opportunities for a spin to flip (where $L = N$.) The spin is only flipped if the new configuration of the lattice is accepted according to the Boltzmann distribution. Below is a plot of the log-log plot of the number of accepted configurations as a function of MC cycles. The number of accepted states grows linearly with MC cycles. Comparing figures 5 and 6 shows us that there are more accepted configurations for higher T.

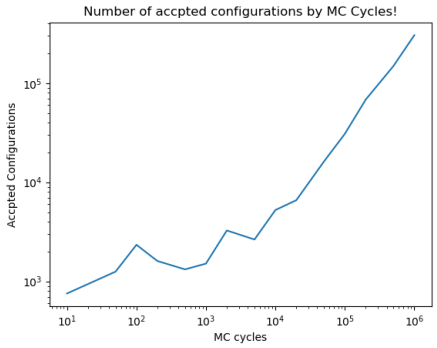

Figure 5: Number of accepted states as a function of MC cycles for $T = 1.0$, with a random initial state.

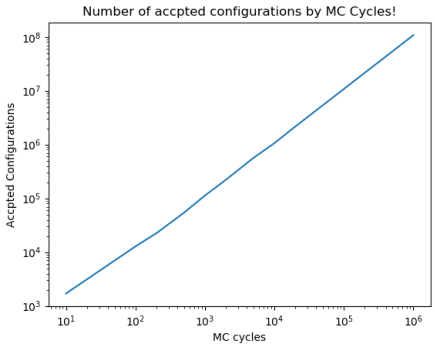

Figure 6: Number of accepted states as a function of MC cycles for $T = 2.4$, with a random initial state.

4.2 Lattice Behavior

Now that we know how much time it takes before the lattice converges to values, we can study the lattice after convergence to truly see how our model behaves for different sizes and temperatures.

4.2.1 Energy Probability

Once the system has reached a steady state, there is still a possibility that spins flip, and the total energy of the system "floats" around the steady state value. We store the values of the energy after the system becomes stable, making a histogram of the potential values of the energy of the system. The energy of the system is stored after each MC cycle, for 10^5 MC cycles after the lattice has reached an equilibrium state (10^6).

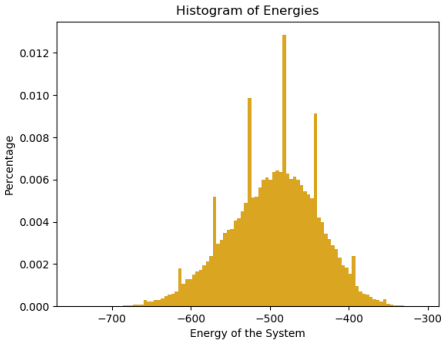

Figure 7: Energy distribution of 10^5 MC cycles after reaching equilibrium, for $T = 2.4$.

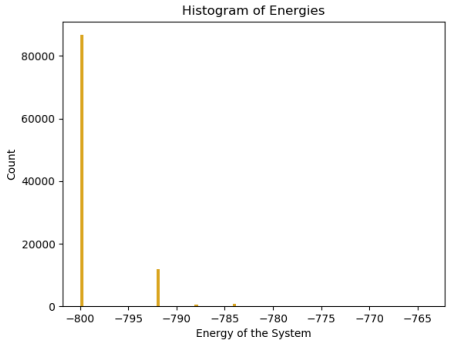

Figure 8: Energy distribution of 10^5 MC cycles after reaching equilibrium, for $T = 1.0$.

The probability distribution at $T = 2.4$, figure 7 follows the Boltzmann distribution well, however we notice large spikes at the stable, most likely states at the given temperature. Even the most likely state has a probability on the order of 1%.

When $T = 1.0$ the system is sitting in an ordered state, with minimized energy as seen in figure 8. As a result, the probability of the lattice remaining in this minimized state, $E = -800$, is dominant (86.7%), and the next most common state, $E = -792$ (11.9%). Every other populated state ($E = -788, -784, -780, -776$) all have probabilities of under 1%.

4.2.2 Phase Transitions

Comparing the behavior of our model at $T = 1.0$ and $T = 2.4$, we can see that there is some sort of transition in this temperature range. We investigate further, and find that there is a distinct change within the temperature range of $T \in [2.1, 2.4]$. We run our simulation for this range of temperatures, with temperature steps of 0.02. Different sized lattices are used as well.

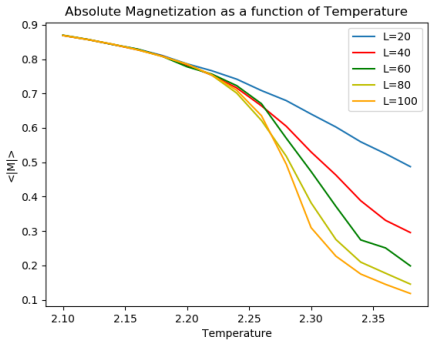

Figure 9: $< |M| >$ as a function of temperature.

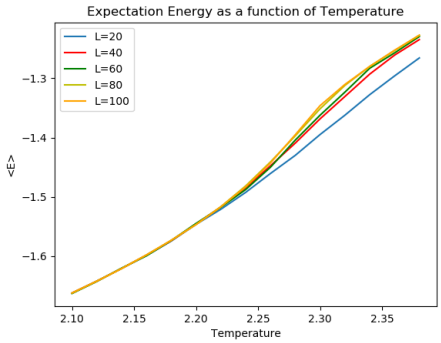

Figure 10: $< E >$ as a function of temperature.

The trends of the expectation values of the energy and absolute value of magnetization look more functional as the lattice size increases, following the relations as show in 2.6. Something is causing a slight change in system energy, but a large change in magnetization in the range $T \in [2.25 - 2.30]$.

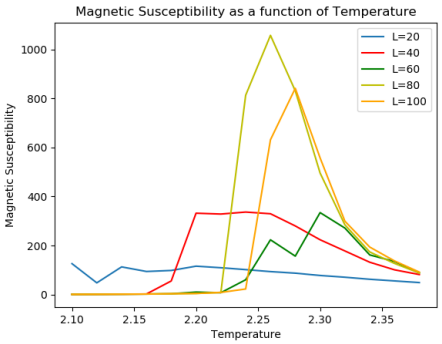

Figure 11: χ as a function of temperature.

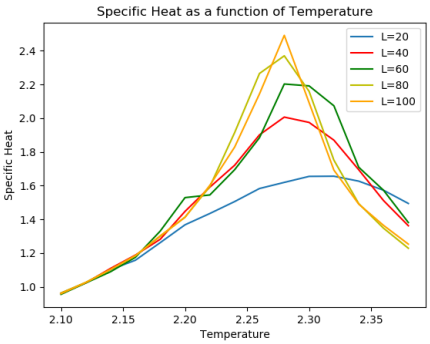

Figure 12: C_V as a function of temperature.

When we plot the behavior of the magnetic susceptibility χ (fig. 11) and the specific heat C_V (fig. 12), we see that there is a drastic change in the range $T \in [2.25 - 2.30]$. We also see that the stability and resolution of these changes become more defined as the lattice size increases.

To find a more where this critical temperature T_C is located, we need to make some approximations using the relations used in in 2.6. First, we want to solve for the correlation length proportionality constant a. We are utilizing equations (12) and (13) and magnetic susceptibilty values from our data at apparent peaks, assuming the peaks are near the critical temperature. We use the given values for $\gamma = 7/4$ and $\nu = 1$.

$$\chi_i(T) = 1057.17 \sim aL^{\gamma/\nu} = a80^{7/4}$$
$$\chi_f(T) = 841.134 \sim aL^{\gamma/\nu} = a100^{7/4}$$
$$a \simeq \frac{1057.17 - 841.134}{80^{7/4} - 100^{7/4}} = -0.2113$$

Using equation 13, and the accepted value given ($T = 2.269$)

$$\begin{aligned} T_C(L) - T_C(L = \infty) &= aL^{1/\nu} \\ T_C(100) - 2.269 &= (-0.2113) * 100^{-1} \\ T_C(100) &= 2.269 - 2.113 * 10^{-3} = 2.2668 \end{aligned}$$

For a 100x100 lattice, we find a critical temperature with a relative error of 0.001 from the accepted value.

5 Discussion

The code for this project was simple to create, and utilizes helpful c++ libraries such as armadillo and random. It runs quickly for small lattice sizes, but as the size of our simulation grows, the computation time grows accordingly. OpenMPI cut down significantly on the time, especially with access to the supercomputers at ITA. On standard laptops, the code for the larger simulations used in this project take on the order of 10s of hours, but with the super computers at ITA, the computations take on the order of 10s of minutes.

The Monte Carlo Markov Chain and the Metropolis algorithm have proven very successful in solving this problem. Both of these methods involve random moves, and the Metropolis algorithm utilizes calculated probabilities inherent to the problem. As a result, for "small" number of MC cycles, the results appear random with slow convergence. However, after roughly $5 * 10^5$ cycles, the convergence of the method becomes apparent. Most importantly, as the model converges, we approach the analytical solutions to the model, proving that this method is effective.

References

[1] Morten Hjorth-Jensen. Project 4. Helpful material and instructions for Project 4 in FYS4150.