Project 4 - FYS4150

Shako Farhad

November 16, 2016

Abstract

In this project I am investigating particles with spin up, 1, or spin down, -1. The two dimensional system of particles can be understood as a thin magnet sheet with magnetization and energy. Each particle will be affected by the four particles surrounding it, above, below, to the left and to the right. This is the Ising model, and I have written a program implementing this Ising model with the metropolis algorithm. With this I am exploring energy levels, magnetization levels, heat specific, susceptibility and phase transitions as functions of temperature and size of the lattice.

Introduction

When trying to do scientific computing on quantum mechanical phenomena, we always end up with programs that take a huge amount of time to finish, unless we simplify the problem. The Metropolis algorithm gives us a way of reducing the complexity of the problem by using a combination of Marko chains with Monte Carlo computing.

I wrote a program in c++ implementing the Ising model and the Metropolis algorithm, and this program was also parallellized get a speedup in computing. As we will discuss later, the parallellization did not work as intended, but I still got many results that give insight into the system. See the discussion below.

I tested the program using a $L = 2$ (2x2-system) where it was possible to calculate the analytical values by hand. These values are our baseline, and they show that the program works as intended.

Theoretical models and methods

The Ising model is that each particle will be affected by the four particles surrounding it, above, below, to the left and to the right. We are using the 2-dimensional version of the Ising model which will simulate a thin, magnet sheet. If all the particles have spin up or spin down, then the magnetic sheet attracts and repulses respectively. And from this model there are formulas for calculating the energy, $< E >$, the magnetic moment, $< |M| >$, the heat specific, C_v, and the susceptibility χ analytically.
To find the energies and magnetic moments of the different configurations, we use the formulas $E_i = -J \sum_{<kl>} s_k s_l$, and $M_i = \sum_i s_i$ respectively.

I make a table with the different possible states, their energies, magnetic moments, and number of micro-states.
I can use this to find a expression for the Z-function, $Z = \sum_{i=0}^{N} e^{-\beta E_i}$:

Nr. of up-spin	E	M	Nr. of microstates
4	-8J	4	1
3	0	2	4
2	0	0	4
2	8J	0	2
1	0	-2	4
0	-8J	-4	1

Table 1: The first row shows the different configurations. The second and third row shows the different energy- and magnetization levels for the different configurations respectively. The fourth row shows how many instances of the different configurations there are.

$$Z = \sum_{i=1}^{16} e^{-\beta E_i} = 2e^{\beta J 8} + 2e^{-\beta 8 J} + 12 e^{-\beta \cdot 0}$$
$$= 2(e^{8J\beta} + e^{-8J\beta}) + 12$$

I use this to find the expectation values $< E >$, $< |M| >$, C_v and χ:

$$< E >= \frac{1}{Z}\frac{\partial Z}{\partial \beta} = \frac{1}{Z}\sum_{i=1}^{16} E_i e^{-\beta E_i}$$
$$< E >= \frac{1}{Z} 2(e^{-8J\beta})$$
$$< |M| >= \sum_{i=1}^{16} \frac{|M_i| e^{-\beta E_i}}{Z}$$

Heat-specific:

$$\sigma_E^2 =< E^2 > - < E >^2$$
$$C_v = \frac{1}{kT^2} \cdot \sigma_E^2$$

Susceptibility:

$$\chi = (< M^2 > - < M >^2) \cdot \beta$$

Calculating analytic values for $L = 2$

The heat specific, C_v:

$$C_v = \frac{1}{kT^2} \cdot \sigma_E^2$$
$$\sigma_E^2 =< E^2 > - < E >^2$$
$$C_v = \frac{1}{kT^2}\frac{(8J)^2}{Z}(2e^{8J\beta} + e^{-8J\beta}) - \frac{1}{kT^2}(\frac{8J}{Z})^2(e^{-8J\beta} - 2e^{8J\beta})^2$$
$$= \frac{(8J)^2}{kT^2 Z}(2e^{8J\beta} + e^{-8J\beta} - \frac{1}{Z}(e^{-8J\beta} - 2e^{8J\beta})^2)$$

The mean magnetization:

$$< |M| >= \sum_{i=1}^{16} \frac{|M_i| e^{-\beta E_i}}{Z} = \frac{1}{Z}(|4|e^{8J\beta} + |2|e^0 + |-2|e^0 + |-4|e^{8J\beta}) = \frac{1}{Z}(8e^{8J\beta} + 4) = \frac{4}{Z}(2e^{8J\beta} + 1)$$

The susceptibility:

$$\chi = \frac{1}{kT}(< M^2 > - < |M| >^2) = \frac{1}{Z}\sum_{i=1}^{16} M_i^2 e^{-\beta E_i} - (\frac{1}{Z}\sum_{i=1}^{16} |M|_i e^{-\beta E_i})^2$$

$$= \frac{1}{kTZ}((4^2 e^{8J\beta} + 2^2 e^0 + (-2)^2 e^0 + (-4)^2 e^{8J\beta}) - \frac{1}{Z^2}(|4|e^{8J\beta} + |2|e^0 + |-2|e^0 + |-4|e^{8J\beta})^2)$$

$$= \frac{8}{kTZ}(4e^{8J\beta} + 1 - \frac{2}{Z}(2e^{8J\beta} + 1)^2)$$

Then we inserted $k = 1.0$, $J = 1$ and $T = 1.0$ into the equations above, and get a value for each expression by using a short python-script.

$$\text{Energy} = -1.99598219825$$

$$\text{Magnetization} = 0.998158549645$$

$$\text{Specific Heat} = 0.128318560222$$

$$\text{Susceptibility} = 0.0280697957033$$

Program results

After running the program we got these values:

```
Printing out relevant values for the 2x2 system case.
Run values have been set to the following:
1000000 Monte Carlo cycles, temperature = 1 and ordered system.

The energy is -1.99596
The mean absolute magnetisation is 0.998652
The specific heat is 11.9838
The susceptiblity is 3.99293
```

The calculated energy and mean magnetization is very close to the analytical values, but unfortunately the specific heat and the susceptibility is far off. This could be due to the formulas for the specific heat and susceptibility being wrong, or the calculation of these values within the big program might be wrong. I will never the less use the program to investigate the quantum mechanical system.

Results

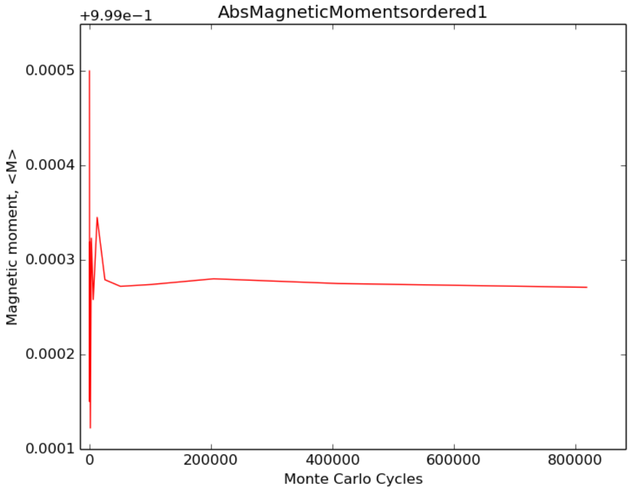


Figure 1: Temperature is set to 1 and the system is ordered. On the x-axis it shows Monte Carlo cycles and on the y-axis it shows the mean of the absolute value of the magnetic moment, $< |M| >$. The steady state is reached at about 100000 Monte Carlo cycles, and the y-values vary between 0.0001 and 0.0005.

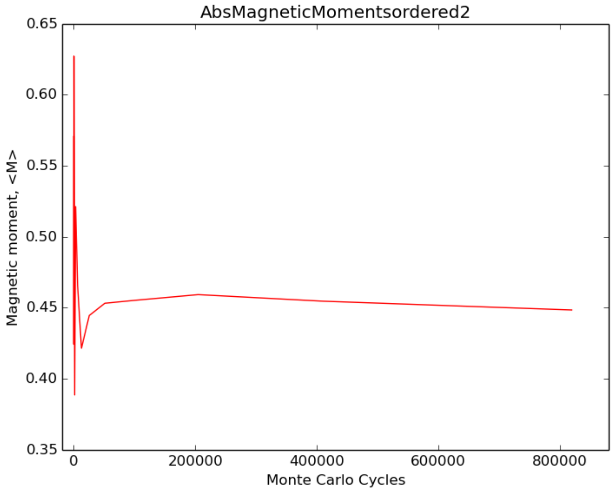


Figure 2: Temperature is set to 2.4 and the system is ordered. On the x-axis it shows Monte Carlo cycles and on the y-axis it shows the mean of the absolute value of the magnetic moment, $< |M| >$. The steady state is reached at about 80000 Monte Carlo cycles, and the y-values vary between 0.39 and 0.63.

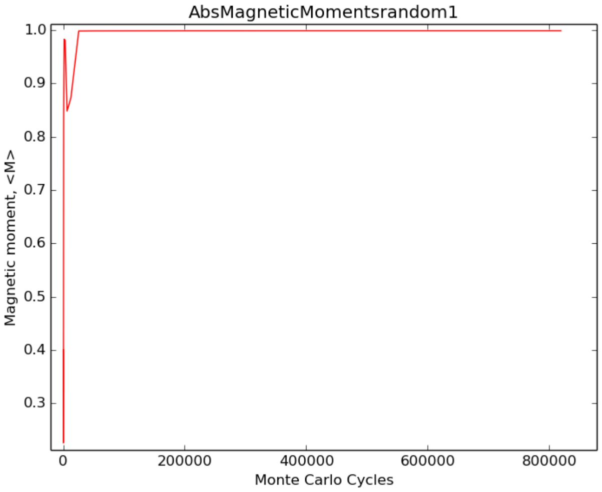


Figure 3: Temperature is set to 1 and the system is random. On the x-axis it shows Monte Carlo cycles and on the y-axis it shows the mean of the absolute value of the magnetic moment, $< |M| >$. The steady state is reached at about 70000 Monte Carlo cycles, and the y-values vary between 0.26 and 1.0.

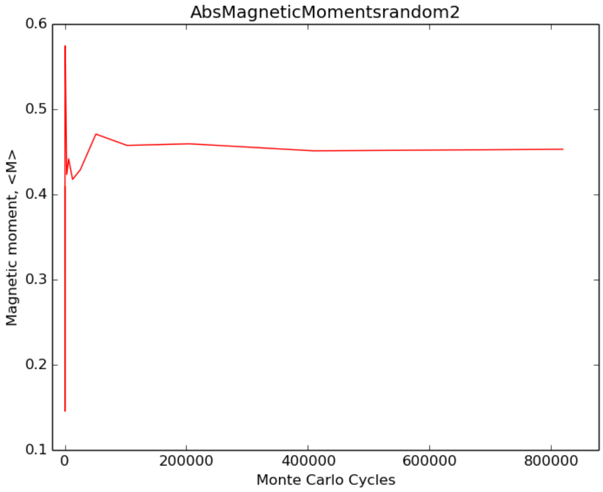


Figure 4: Temperature is set to 2.4 and the system is random. On the x-axis it shows Monte Carlo cycles and on the y-axis it shows the mean of the absolute value of the magnetic moment, $< |M| >$. The steady state is reached at about 120000 Monte Carlo cycles, and the y-values vary between 0.42 and 0.47.

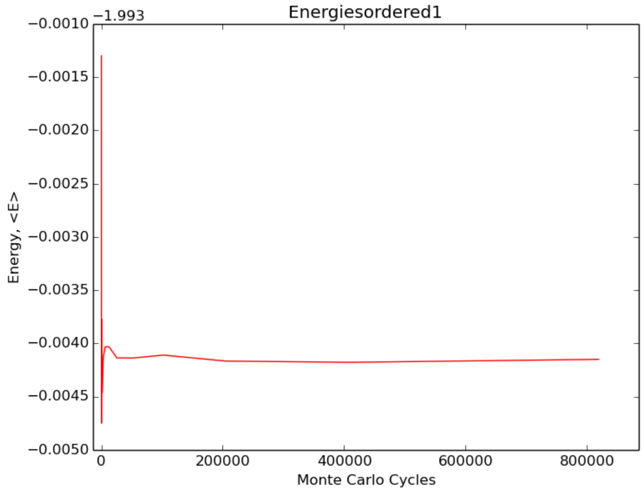


Figure 5: Temperature is set to 1 and the system is ordered. On the x-axis it shows Monte Carlo cycles and on the y-axis it shows the mean energy, $< E >$. The steady state is reached at about 150000 Monte Carlo cycles, and the y-values vary between -0.0043 and -0.0013.

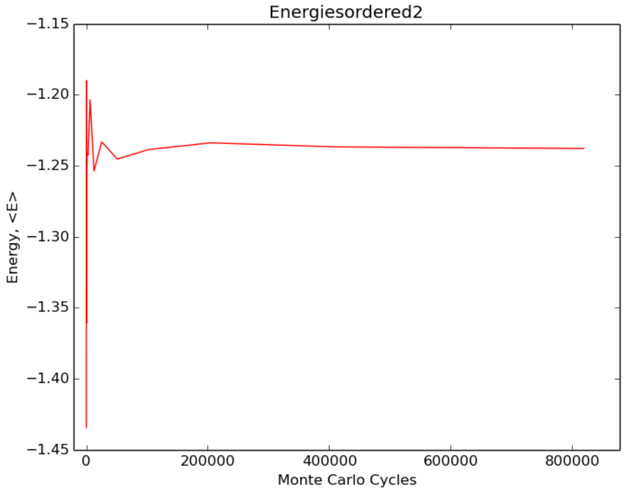


Figure 6: Temperature is set to 2.4 and the system is ordered. On the x-axis it shows Monte Carlo cycles and on the y-axis it shows the mean energy, $< E >$. The steady state is reached at about 100000 Monte Carlo cycles, and the y-values vary between -1.43 and -1.18.

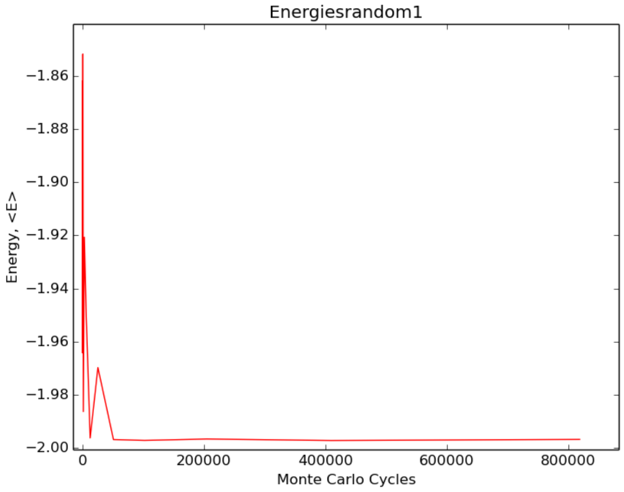


Figure 7: Temperature is set to 1 and the system is random. On the x-axis it shows Monte Carlo cycles and on the y-axis it shows the mean energy, $< E >$. The steady state is reached at about 55000 Monte Carlo cycles, and the y-values vary between -1.99 and -1.85.

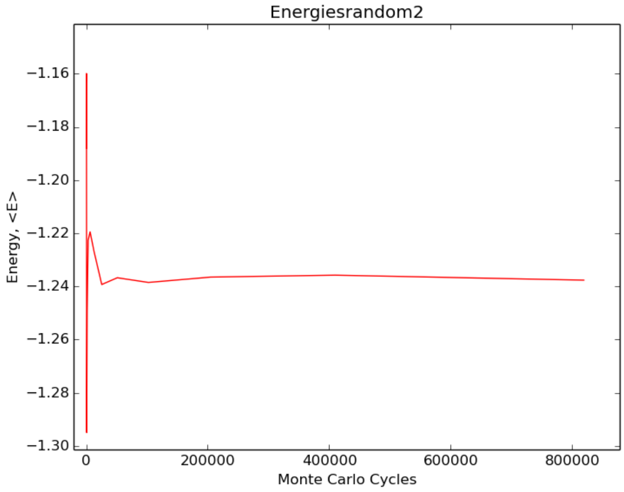
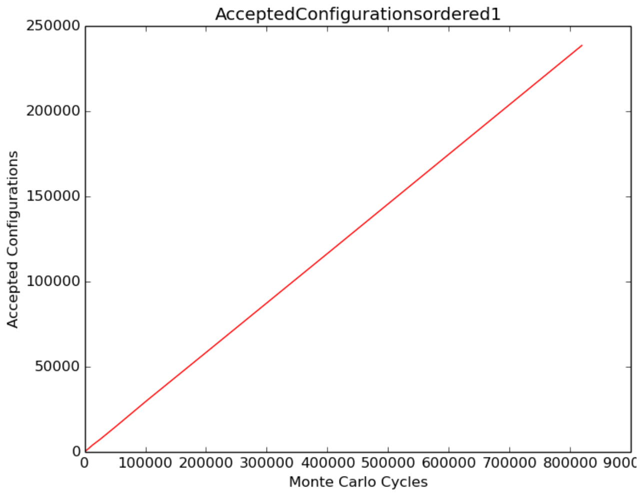


Figure 8: Temperature is set to 2.4 and the system is random. On the x-axis it shows Monte Carlo cycles and on the y-axis it shows the mean energy, $< E >$. The steady state is reached at about 80000 Monte Carlo cycles, and the y-values vary between -1.29 and -1.16.



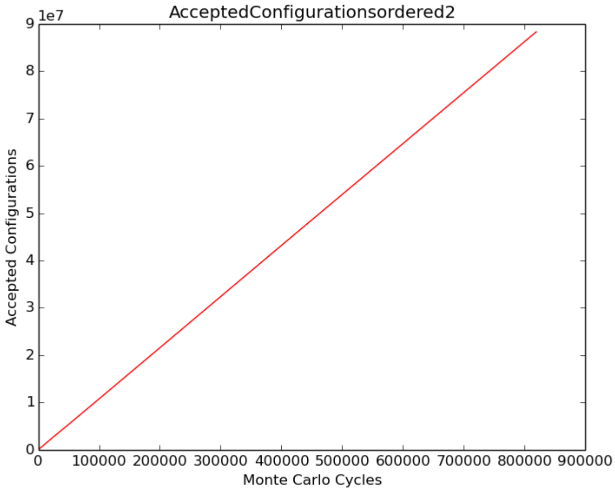
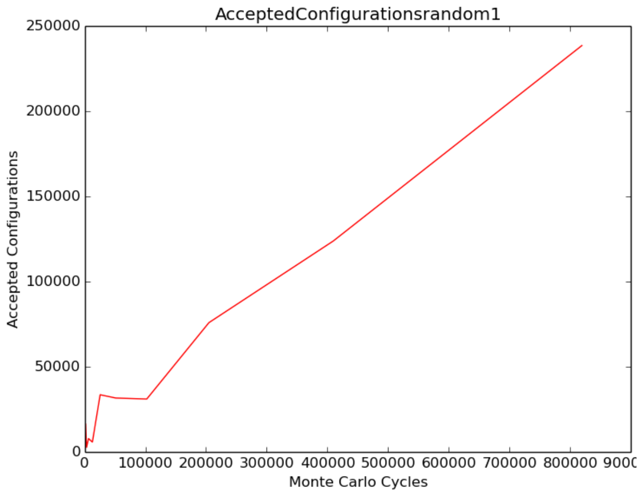


Figure 9: Temperature is set to 1 and the system is ordered. On the x-axis it shows Monte Carlo cycles and on the y-axis it shows the accepted configurations. The function is completely linear and the y-values vary between 0 and 240000.

Figure 10: Temperature is set to 2.4 and the system is ordered. On the x-axis it shows Monte Carlo cycles and on the y-axis it shows the accepted configurations. The function is completely linear and the y-values vary between 0 and 90000000.



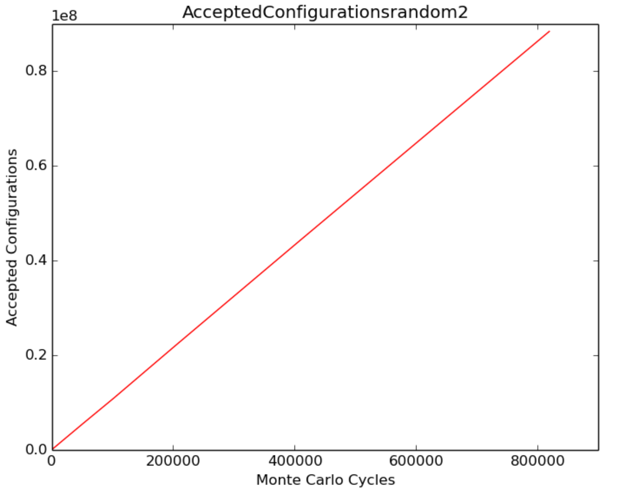


Figure 11: Temperature is set to 1 and the system is random. On the x-axis it shows Monte Carlo cycles and on the y-axis it shows the accepted configurations. The function is linear after about 400000 Monte Carlo cycles, and the y-values vary between 0 and 240000.

Figure 12: Temperature is set to 2.4 and the system is random. On the x-axis it shows Monte Carlo cycles and on the y-axis it shows the accepted configurations. The function is completely linear and the y-values vary between 0 and 85000000.

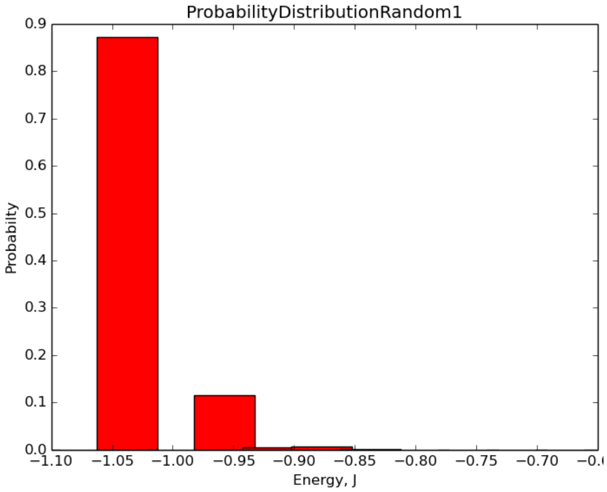


Figure 13: Temperature is set to 1 and the system is random. On the x-axis it shows energy and on the y-axis it shows the probability of a certain configuration assuming a certain energy.

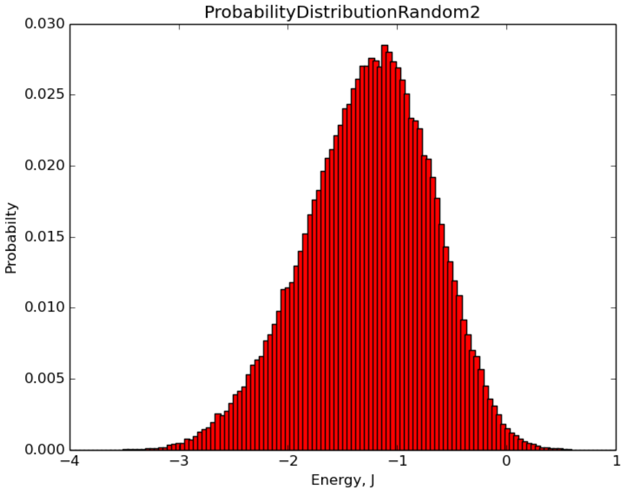


Figure 14: Temperature is set to 2.4 and the system is random. On the x-axis it shows energy and on the y-axis it shows the probability of a certain configuration assuming a certain energy.

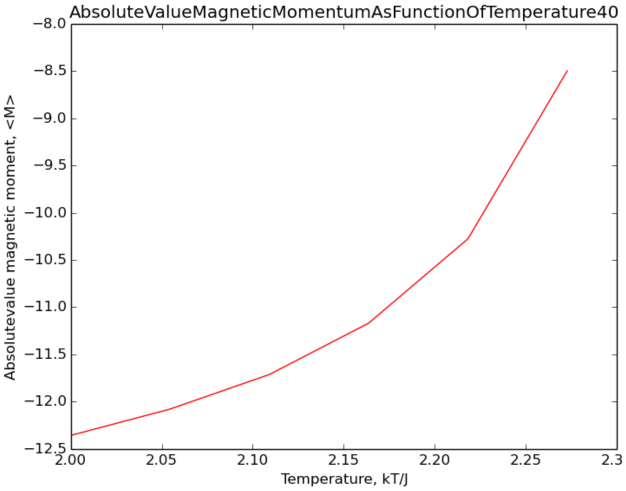


Figure 15: The system is a 40x40 random lattice. On the x-axis it shows the temperature and on the y-axis it shows the mean of the absolute value of the magnetic moment, $< |M| >$. There is rapid change in $< |M| >$ the closer to 2.3 in temperature we get.

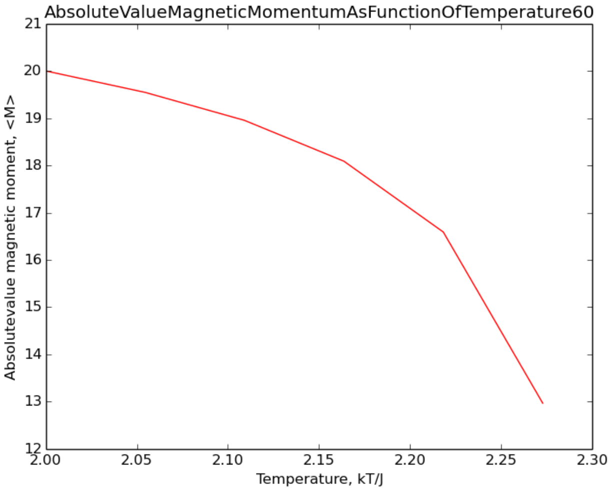
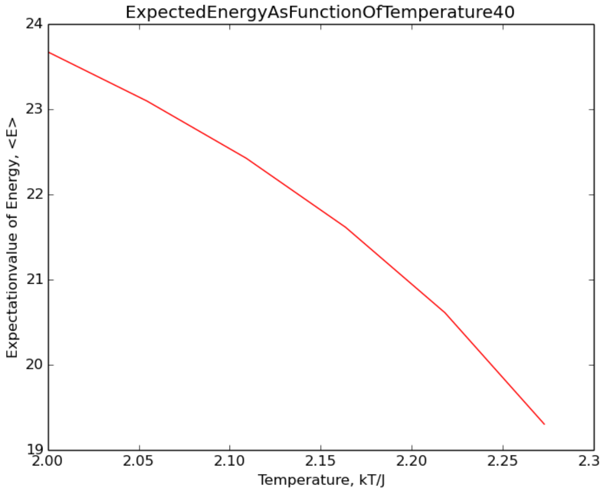


Figure 16: The system is a 60x60 random lattice. On the x-axis it shows the temperature and on the y-axis it shows the mean of the absolute value of the magnetic moment, $< |M| >$. There is rapid change in $< |M| >$ the closer to 2.3 in temperature we get.



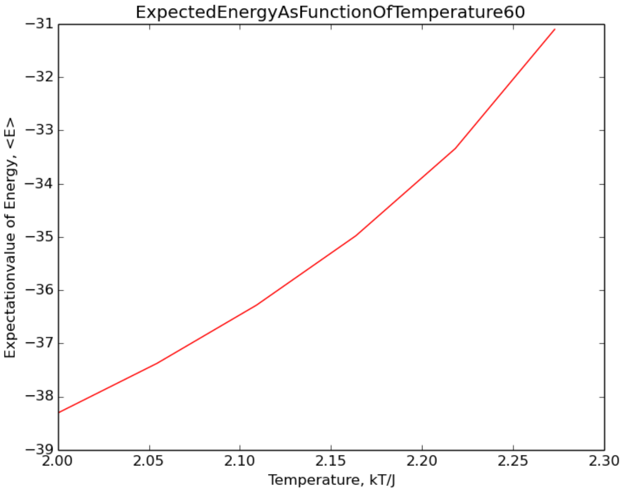
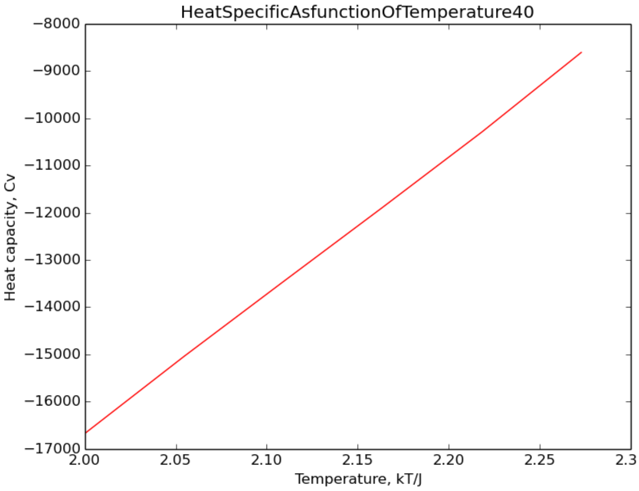


Figure 17: The system is a 40x40 random lattice. On the x-axis it shows the temperature and on the y-axis it shows the mean energy, $< E >$. The graph is almost linear, but $< E >$ is decreasing faster the closer to 2.3 in temperature we get.

Figure 18: The system is a 60x60 random lattice. On the x-axis it shows the temperature and on the y-axis it shows the mean energy, $< E >$. The graph is almost linear, but $< E >$ is increasing faster the closer to 2.3 in temperature we get.



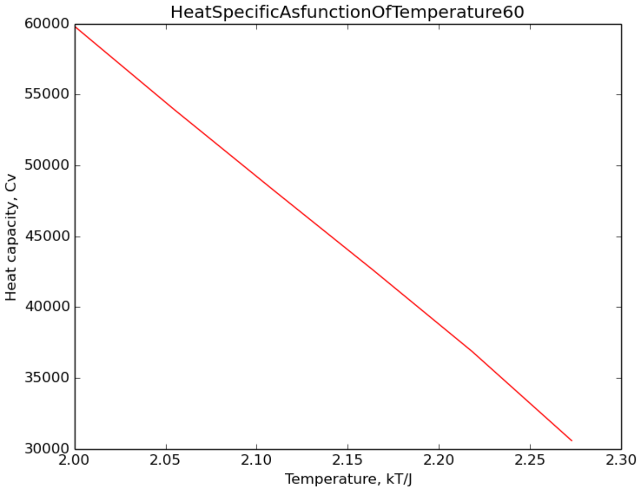


Figure 19: The system is a 40x40 random lattice. On the x-axis it shows the temperature and on the y-axis it shows the heat specific, C_v. The graph is linear.

Figure 20: The system is a 60x60 random lattice. On the x-axis it shows the temperature and on the y-axis it shows the heat specific, C_v. The graph is linear.

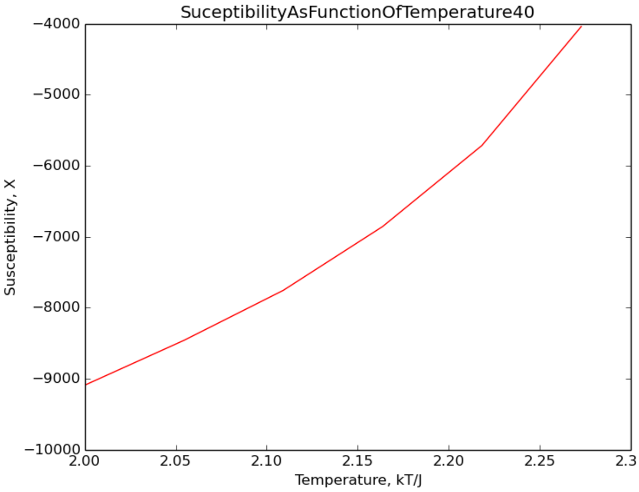


Figure 21: The system is a 40x40 random lattice. On the x-axis it shows the temperature and on the y-axis it shows the susceptibility, χ. The graph increasing exponentially with temperature.

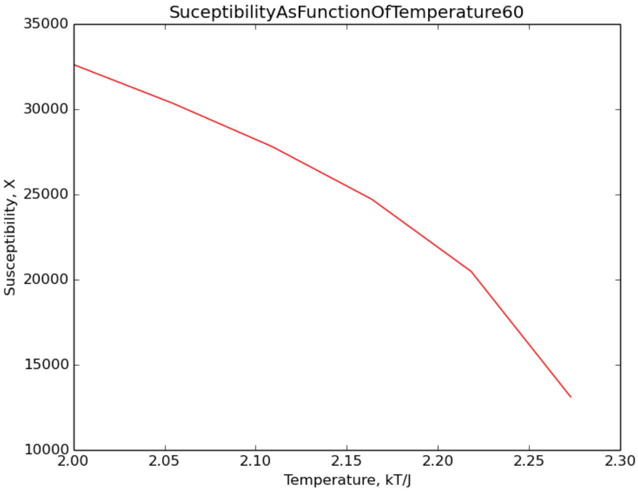


Figure 22: The system is a 60x60 random lattice. On the x-axis it shows the temperature and on the y-axis it shows the susceptibility, χ. The graph decreasing exponentially with temperature.

Discussion

Steady state

In almost all of the figures (figure 1 to figure 12), we have ran up to 1000000 Monte Carlo cycles. Just by reading the graphs, I can see that they all get to steady state between 10000 and 100000 Monte Carlo cycles. $\frac{100000}{1000000} = 0.1$, so it is safe to assume that the system has reached steady state after 10% of the maximum amount of Monte Carlo cycles.

When the temperature is low there is less change in the system. But from the graphs I can see that if the system is started in an ordered configuration with low temperature, and later is started in an ordered configuration with higher temperature, I need more Monte Carlo cycles to reach steady state for the higher temperature. If the system is started in a random configuration, then it requires the same amount of Monte Carlo cycles to reach steady state no matter the temperature.

The accepted configurations as functions of Monte Carlo cycles are all linear for ordered and random systems. The linearity is broken only in the case where the temperature is low and the system is random, but then quickly becomes linear after around 250000 Monte Carlo cycles. Another thing to note is that for low temperatures, about 230000 configurations are accepted after 800000 Monte Carlo Cycles. For higher temperatures about 85000000 configurations were accepted after about 800000 Monte Carlo cycles. So increasing the temperature from 1 to 2.4 gave us an increase of about 37000% in accepted configurations. And this is true for ordered and random systems.

Energy probability distribution

From figure 13 and 14, we can see that for low temperatures, the energy does not change much at all, but for higher temperatures the probability distribution of the energy looks like a normal distribution. After running the program for 1000000 Monte Carlo cycles, we got this output:

```
The computed variance in energy for 1 in temperature is 4.24358.
The computed variance in energy for 2.4 in temperature is 30.1368.
```

As we noticed earlier, the calculated value of the variance is much higher than it should be. This might be because the variance should have been divided by the size of the system ($L \cdot L$). This could be the reason why the variance is so much higher than what we can see is the case in figure 13 and 14.

But we expected the energy probability distribution to be close to a normal distribution because the test for accepting and rejecting configurations in our Metropolis algorithm is a normal distribution.

Parallel programming

Because MPI did not function correctly on the laboratory computers, I had to split the workload in 6 and run 6 instances of my program with different input arguments.

I ran the parallellized program and left it running for about 20 hours. And the program has only given us values for half of the T values that I had set (every other T value; $2.0, 2.05454, 2.10909$ etc., for $\Delta T = 0.0272727$. And after 20 hours of only writing every other T, the program only wrote $L = 40$ and $L = 60$ and it is still working on $L = 100$. I know this because 6 of the CPUs are working 100%.

I believe that this massive slowdown is due to some problem with writing the values to the .txt file, because when I run the program with no parallellization and not writing to file, I got these time values for the two runs I conducted:

```
RUN 1 OUTPUT:
Program finished the task in 1.42264e+10 microseconds.
```

```
RUN 2 OUTPUT:
Program finished the task in 2.18352e+10 microseconds.
```

When I convert these values into hours, we get that the first and second run took about 3.95177778 hours and 6.06533333 hours respectively. So with $\Delta T = 0.0272727$ and $L = 40, 60, 100, 140$, it took between 4 and 6 hours to finish with only one thread.

We also ran the parallellized program without writing to file, but even after 24 hours, it did not finish. We suspect there is a problem in how the for loop is constructed when it is parallellized, but then again, when we ran the same program with writing to file, it did finish, but only writing half of the values it should have. So the program will finish after a long time, and for small L it does its job smoothly, but something happens for large L and when we use the parallellization.

Critical temperature

The data from the figures (figure 15 to 22) is actually correct, but the end-temperature is too low. If I had ran the program for longer, the trend would be a top that decreases in height for larger L.

We are about to see a phase transition in the figures (figure 15 to 22), but because of not enough T values around 2.3 and the lattice size only going up to 60, we can not see it. This is regrettable, but unfortunately the parallellization did not work as intended.

Conclusion

The analytical and numerical values correlate quite well. There are some instances of error, but those are most likely related to parallellization not working correctly, and the absence of the normalization of the different values from dividing by the lattice size. Other than that this project has been a great way of getting insight in to the quantum mechanical world and learning more about parallell programming.

It was especially interesting to look at how the test in the Metropolis algorithm gave us a nearly normal distributied probability distribution. Finding the steady state by looking at the different figures, and actually improving the our numerical approximations was also great. This all opened me to a deeper understanding of scientific reporting and Monte Carlo methods.

Appendix

The code and other result-material can be found at:

`https://github.com/ShakoFarhad/Project-4`