FYS4150 - Project 4

Eirik Ramsli Hauge

16. November 2016

Abstract

The aim of this project is to study the Ising model using the Metropolis algorithm and Monte Carlo cycles where one cycle is one runthrough through the Metropolis algorithm. We will mostly change the variables spin lattice size L^2 and temperature to aquire our results. First we will compare our analytical expression to a numerical for a 2x2 case. Then we will expand to a 20x20 case. We will also look at the number of accepted spin flips after all cycles for differen temperatures and find the probaility of finding an energy as a function of two temperatures and look at variance. Lastly we will use parallelization and look at many temperatures with increasing L and find the critical temperature.

We found that there was a good agrement between our analytical and numerical expression. The number of accepted states increased rapidly with T, the probability plots made sense, the variance did not and our critical temperatures came closer and closer to the limit found by Onsager with increasing L.

Introduction

Phase transitions is ofthe thought of as a transition between liquid and vaport or solids. This however is not always the case. We can also have magnetic transitions and a popular model to study this is the Ising model. This model studies the phase transition between a magnetic phase and a phase with zero magnetization. To simplify our problem, we will look at a two dimensional system without an external magnetic field. We will mostly look at the values for $\langle E \rangle$, $\langle M \rangle$ C_v and χ as a function of T.

First of all we will study the 2x2 case analytically, develop a code based on the Ising model and the metropolis algorithm and compare the numerical values with the analytical ones. The metropolis algorithm will always try to diminish the Energy of the system so that we reach the stabil state.
When we know our code is working we will increase to a 20x20 lattice and find how many Monte Carlo cycles are needed before the different eigenvalues reach an equilibrium for different temperatures. We will also look at how many times we flip a spin to across all Monte Carlo cycles. Er det dette vi gjør?

Afterwards we will look at the probability $P(E)$ of finding an energy after equilibrium is reached by counting each energy we find.

Lastly we will use parallelization to calculate larger with L-values for many values of T. To do this we will use MPI. The results will be analyzed and used to find the critical temperature T_C in the thermodynamical limit $L \to \infty$.

There are several good reason for why we are doing this experiment as a lesson in Computational Physics. By using the Ising model and the metropolis algorithm over many cycles we get a good insight into Monte Carlo calculations and by using MPI we learn about paralellization and the benefits of being able to use every core at once. These tricks teaches us that it's allowed to think before you code, which is always a good thing.

Theory

The Ising model's total energy for a L x L spin lattice in state i can in it's simplest form be expressed as follows:

$$E_i = -J \sum_{\langle kl \rangle}^{N} s_k s_l - \mathcal{B} \sum_{k}^{N} s_k \tag{1}$$

Where J is a coupling constant expressing the strength of the interaction between neighboring spins. The $\langle kl \rangle$ expresses that we only wish to look at neighboring spins, $s = \pm 1$, N is the number of spin states and $\mathcal{B}$ is an external magnetic field which will be set to 0 in this experiment.

To calculate the wanted expectation values $\langle E \rangle$ and $\langle |M| \rangle$ we have the expressions:

$$\langle E \rangle = \sum_{i} E_i P_i(\beta) \tag{2}$$

and

$$\langle |M| \rangle = \sum_{i} |M_i| P_i(\beta) \tag{3}$$

E_i and M_i are respectively the energy (equation (1)) and the magnetization in the state i. The magnetization is given as:

$$M_i = \sum_{k}^{N} s_k \tag{4}$$

$P_i(\beta)$ is the probability distribution for the state i and is given as:

$$P_i(\beta) = \frac{e^{-\beta E_i}}{Z} \tag{5}$$

where $\beta = 1/(k_B T)$ where k_B is the Boltzmann constant and T is the temperature. and Z is the partition function given as:

$$Z = \sum_{i=1}^{M} e^{-\beta E_i} \tag{6}$$

When all the above equations are known, it is easy to find the heat capacity C_v and susceptibility χ:

$$C_v = \frac{\sigma_E^2}{k_B T^2} = \frac{\langle E^2 \rangle - \langle E \rangle^2}{k_B T^2} \tag{7}$$

$$\chi = \frac{\sigma_M^2}{k_B T} = \frac{\langle M^2 \rangle - \langle |M| \rangle^2}{k_B T} \tag{8}$$

where for any discrete variable A σ_A^2 is called the variance of A and

$$\langle A^2 \rangle = \sum_i A_i^2 P_i(\beta) \tag{9}$$

all other variables are as above.

The 2x2 case

If we have a lattice with 2x2 spins, there are 16 different spin configurations possible:

Table 1: Possible spin configurations (degeneracy): All spins up (1), one spin down (4), two spins down side by side (4), two spins down diagonally (2), three spins down (4), all spins down (1).

↑	↑	↓	↑	...	↓	↓	...	↑	↓	↓	↑	...	↓	↑	↓	↓
↑	↑	↑	↑		↑	↑		↓	↑	↑	↓		↓	↓	↓	↓

The properties of the different spin states illustrated in table 1 are better expressed in table 2

Table 2: Energy, magnetization and degeneracy as a function of number of spins up for a 2 x 2 lattice. Energy and magnetization calculated by equation (1) and (4)

Number of spins up	Degeneracy	Energy	M	
4	1	$-8J$	4	
3	4	0	2	
2	4	0	0	(side by side)
2	2	$8J$	0	(diagonal)
1	4	0	-2	
0	1	$-8J$	-4	

By using equation (6) we find the distributin function:

$$\begin{aligned} Z &= e^{\beta 8J} + 2e^{-\beta 8J} + e^{\beta 8J} + 12 \cdot e^0 \\ &= 2e^{\beta 8J} + 2e^{-\beta 8J} + 12 \end{aligned}$$

And then by using equations (2), (3), (7) and (8) we can easily find respectively $\langle E \rangle$, $\langle |M| \rangle$, C_v and χ

$$\begin{aligned}
\langle E \rangle &= \frac{1}{Z}\sum_{i}^{16} = E_i e^{-\beta E_i} \\
&= \frac{1}{Z}\big[-8Je^{\beta 8J} + 2\cdot 8Je^{-\beta 8J} - 8Je^{\beta 8J}\big] = -\frac{1}{Z}\big[16Je^{\beta 8J} - 16Je^{-\beta 8J}\big] \\
&= -\frac{32J}{Z}\sinh(\beta 8J) \\
\langle |M| \rangle &= \frac{1}{Z}\sum_{i}^{16} = |M_i| e^{-\beta E_i} \\
&= \frac{1}{Z}\big[4e^{\beta 8J} + 4\cdot 2e^{0} + 4\cdot 2e^{0} + 4e^{\beta 8J}\big] = \frac{1}{Z}\big[8e^{\beta 8J} + 16\big] \\
\langle E^2 \rangle &= \frac{1}{Z}\sum_{i}^{16} = E_i^2 e^{-\beta E_i} = \frac{128J^2}{Z}\big[e^{\beta 8J} + e^{-\beta 8J}\big] = \frac{256J^2}{Z}\cosh(\beta 8J) \\
\langle M^2 \rangle &= \frac{1}{Z}\sum_{i}^{16} = M_i^2 e^{-\beta E_i} = \frac{32}{Z}(e^{\beta 8J} + 1) \\
C_v &= \frac{1}{k_B T^2}\Big[\frac{256J^2}{Z}\cosh(\beta 8J) - (-\frac{32J}{Z}\sinh(\beta 8J))^2\Big] \\
\chi &= \frac{1}{k_B T}\Big[\frac{32}{Z}(e^{\beta 8J} + 1) - (\frac{1}{Z}\big[8e^{\beta 8J} + 16\big])^2\Big]
\end{aligned}$$

Why is the 2x2 case relevant? There are two good reason:
1. It is good to have an analytical example to compare with.
2. It gives an easy understandig periodic boundary conditions. Periodic boundary conditions meens that the spin lattice can be looked upon as a 3D torus folded out where if we walk over the edge of the 2D lattice, we end up at the other side. This is an easy consept to grasp if you have played video games like Chrono Trigger where the map functions in the same way. In our 2x2 example this means that the top most left corner will "see" the bottom left spin both above and belove itself. And likewise it will see the top most right spin both to its right and to its left.

The Metropolis Algorithm

Through this project we wish to start with a spin lattice where all spin are known. Then we wish to achieve the stable state, e.i. the state with the lowest energy. To do this we will use both Monte Carlo and the Metropolis algorithm. The Metropolis Algorithm bases itself upon a pretty simple principle and applied to the Ising model it goes like this:

Algorithm 1 Metropolis for the Ising Model

Initialize a state spin lattice with energy E_0 and magnetization M_0
for $i = 0 \to i = L$ **do**
  for $j = 0 \to j = L$ **do** ▷ we loop over L^2 giving all spins a chance to be picked.
    Choose random spin $s_{l,k}$
    Find the difference in energy (ΔE) if we had flipped that spin.
    if $r \leq e^{-\beta \Delta E}$ **then** ▷ where r is a random number between 0 and 1.
      Flip spin
      M += $2 \cdot s_{l,k}$
      E += ΔE
    end if
  end for
end for
Update Eigenvalues

By first initializing a spin lattice and then running the Metropolis algorithm many times using the previous spin lattice as the initial state we do a Monte Carlo simulation where each run of the Metropolis is one cycle.
For a more in-depth study of the Metropolis Algorithm in general you may read in the Lecture Notes and especially note figure 12.7 as this gives a good overview. [**?**]

Stabilization and probability

The reason we wish to run the Metropolis algorithm many times is to reach the steady state. While the steady state is hard to guess, we will reach it bit by bit by reducing the energy. As one can see from algorithm 1 we only accept a flip spin if $r \leq e^{-\beta \Delta E}$. As $\beta > 0$ there will be a bigger chance for the negative ΔE values to cause a spin flip and thus cause a lower overall energy.

Once this steady state is reached we can look at the probability of finding a given energy. This is done by simply counting how often a given energy appears and then dividing by the total number of energies.

Estimate the Critical Value

Onsager found that for $L \to \infty$ the critical temperature was $T_C \simeq 2.269$. In our case we have the function:

$$T_C(L) - T_C(L = \infty) = aL^{-1/v} = a/L \tag{10}$$

we are given that v = 1. a is a constant.
Numerically we do this by finding the peak of C_V and χ as a function of T since this is the phase transition between non-zero magnetization and then zero magnetication. This is caused by L. The finding of the peak is the estimation.

Experimental

The programs used in this project can be found in the GitHub repository [**?**] in the `/src/` folder. When running the program, it takes five command line arguments: `task`, spin lattice size `L`, `Monte Carlo cycles`, `T` which is either `temperature` or `Number of temperatures` depending on task and

`spin direction`. The latter is optional and can be set to either `up`, `rand` or `down` which respectively intializes the spin lattice with all spins up, all spins randomly either up or down or all spins down. By default it is set to `up`.
All files containing data will be stored in GitHub under `/benchmarks/` with a separate subfolder for each task. The program has four different run modes, which will be discussed in more detail below. Many of the run modes are fairly similar, but with some small changes. To make the program more versatile `L` and `T` are optional for all `task` even though the text [**?**] ask us to run for specific values in each task.

Parallelization

This program uses `MPI` for programming. When running the program one should write:
`-n (Number of cores to use) Filename task L (Monte Carlo cycles) T (spin direction)`
where everything within a paranthesis describes one argument.
Although every task can be run with MPI, it is only task e) that is meant to. We therefore recommend setting number of cores to use to one when running all the other tasks.

Task b)

Setting `task` to `b` runs the part of the program which is meant to solve task b. Here the text would have us set L = 2, but we have left this optional. For this task `T` is `temperature` and should be set to 1.0. All the data is written to:
`eigenvalues_MC<Monte Carlo cycles>_dim<L>_dir<spin direction>_T<temperature>.txt`
where `<a>` means the value of a.

Task c)

Setting `task` to `c` runs the part of the program which is meant to solve task c. `L` should be set to 20. The Metropolis algorithm for this part of the program is the same as above. `T` should be set to either 1.0 or 2.4. All the data is written to:
`eigenvalues_MC<Monte Carlo cycles>_dim<L>_dir<spin direction>_T<temperature>.txt`

Task d)

Setting `task` to `d` runs the part of the program which is meant to solve task d. `L` must be set to 20. The reason why `L` is fixed here is that we have som threshold values for when the system is in a steady state, as this is affected by the size of the spin lattice. Failing to set `L` to 20 will make the program stop. The Metropolis algorithm for this part of the program is the same as for `c`, but when we update the eigenvalues we also store the energy if the number of Monte Carlo cycles are above a threshold. Then the program counts how many times every energy is achieved. `T` is once again temperature and should be set to either 1.0 or 2.4 and `Monte Caro cycles` should be set to 1000000 for the threshold values to be right. The eigenvalues for each Monte Carlo cycles are written to
`eigenvalues_MC<Monte Carlo cycles>_dim<L>_dir<spin direction>_T<temperature>.txt`
And all the energies and their counters are written to:
`EnergyValues_T<temperature>_dim<L>_dir<spin direction>.txt`

Task e)

Setting `task` to `e` runs the part of the program which is meant to solve task e. `L` is optional. `T` sets the number of temperatures in an array from 2.0 to 2.3. The difference between two values in this array, the dt, must be less than or equal to 0.05 or else the program will stop.
This part of the program is also specially designed to use `MPI` for paralellizing and by giving each core a specific temperature to work with we can solve for all the temperatures more efficiently. Since we are only interested in the last eigenvalues we do not write to file for each cycle, but only when all the Monte Carlo cycles have been run through for every core with its temperature.

Results

All runs were performed with 1 000 000 Monte Carlo cycles.

L = 2

Running the program for `task` b with L = 2, 1 core and T = 1.0 gave us the results in table 3:

Table 3: Values returned from the program as a function of Monte Carlo Cycles. The first row of variables is from our analytical calculations.

Monte Carlo cycles	$< E >$	$< \|M\| >$	σ_E^2	σ_M^2
Analytical	-1.99598	0.99866	0.03208	0.00401
100	-1.86000	0.94000	1.04160	0.20560
1000	-1.98400	0.99350	0.12698	0.02283
10000	-1.99440	0.99820	0.04467	0.00519
100000	-1.99556	0.99848	0.03544	0.00464
1000000	-1.99572	0.99857	0.03418	0.00428

L = 20

Setting L = 20, T = 1.0 or 2.4 and running `task` c for 1 core gave us the following results:

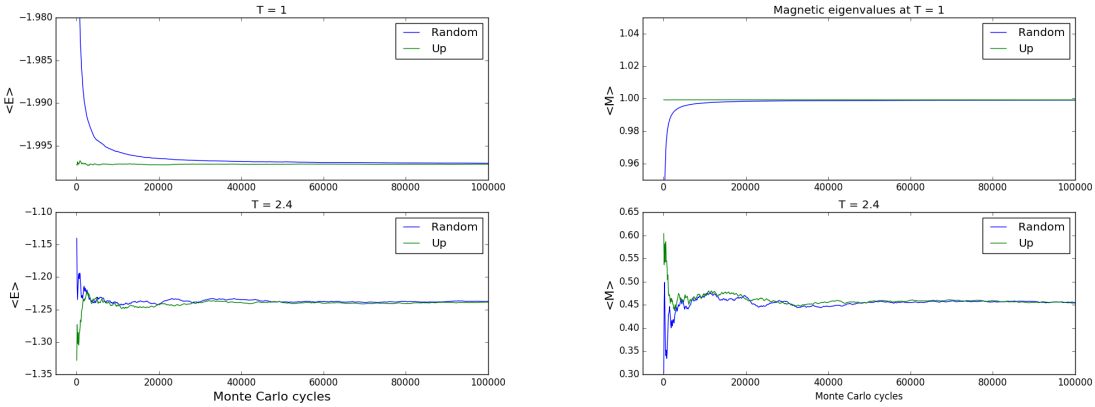

Figure 1: Energy and Magnetic eigenvalues as a function of Monte Carlo Cycles for different temperatures. One of the graphs represents the case with all spins fixed upwards. The other representes the case with all spins randomly directed.

We can note from figures 1 that there seems to be an equilibrium around 80 000 Monte Carlo cycles for T = 1.0 and around 60 000 for T = 2.4.
By looking at the total number of accepted states at the end of all Monte Carlo Cycles as a function of different temperatures we found the correlation shown in figure 2:

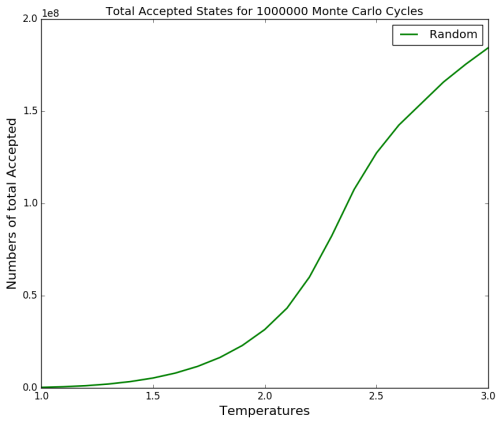

Figure 2: Number of accepted states after all Monte Carlo cycles as a function of temperature. With L = 2

By counting each energy for T = 1.0 and T = 2.4 we found the probabilities shown in figure 3

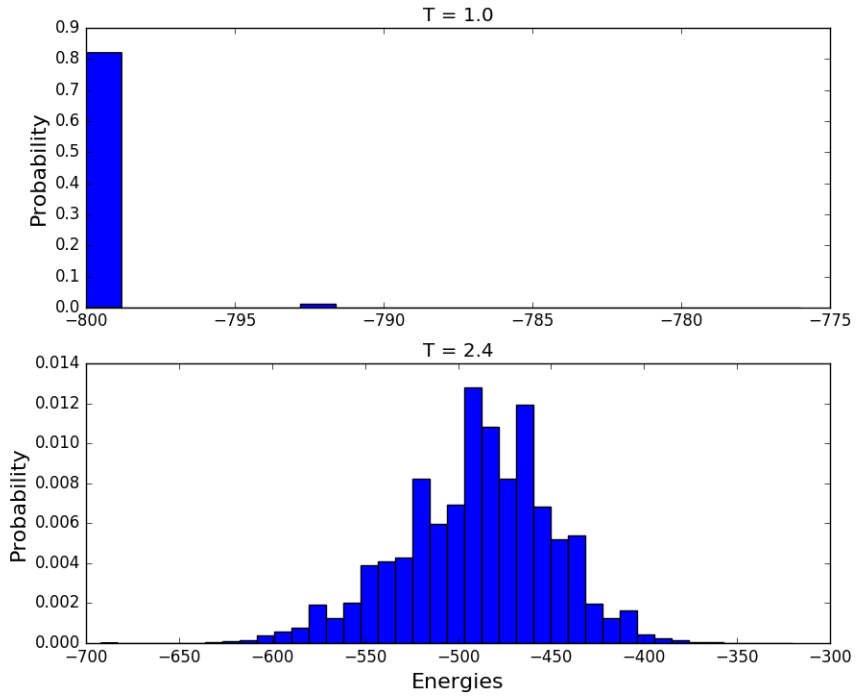

Figure 3: Probabilities for different energy values for T = 1.0 and T = 2.4. All spins were sett to be initialized randomly.

By taking the variance of our energies with `np.var()` and comparing to the last value of $sigma_E^2$ gave us the following presented in table 4

Table 4: Variance given directly from the program compared to taking the variance of our energies. As we can see, they are quite different

Temperature	σ_E^2	`var(E)`$/L^2$
1.0	0.03731	0.0025
2.4	8.125	4.044

L $\geq$ 20

By using MPI we were able to fairly quickly run the program for different temperatures as we had access to a computer with 64 cores. By letting each core have one or maximum two values we produced the results presented in figure 4. Here we have that all simulations runs in the interval $T \in [2.0, 2.3]$ with a time step $dt = 0.01034$ for $L \in [40, 100]$ and $dt = 0.00612$ for L = 140.

Table 5: Critical temperature for C_v and χ as a function of L.

L	C_v	χ
40	2.28	2.3
60	2.2793	2.3
80	2.2793	2.2897
100	2.2793	2.2897
140	2.2755	2.2755

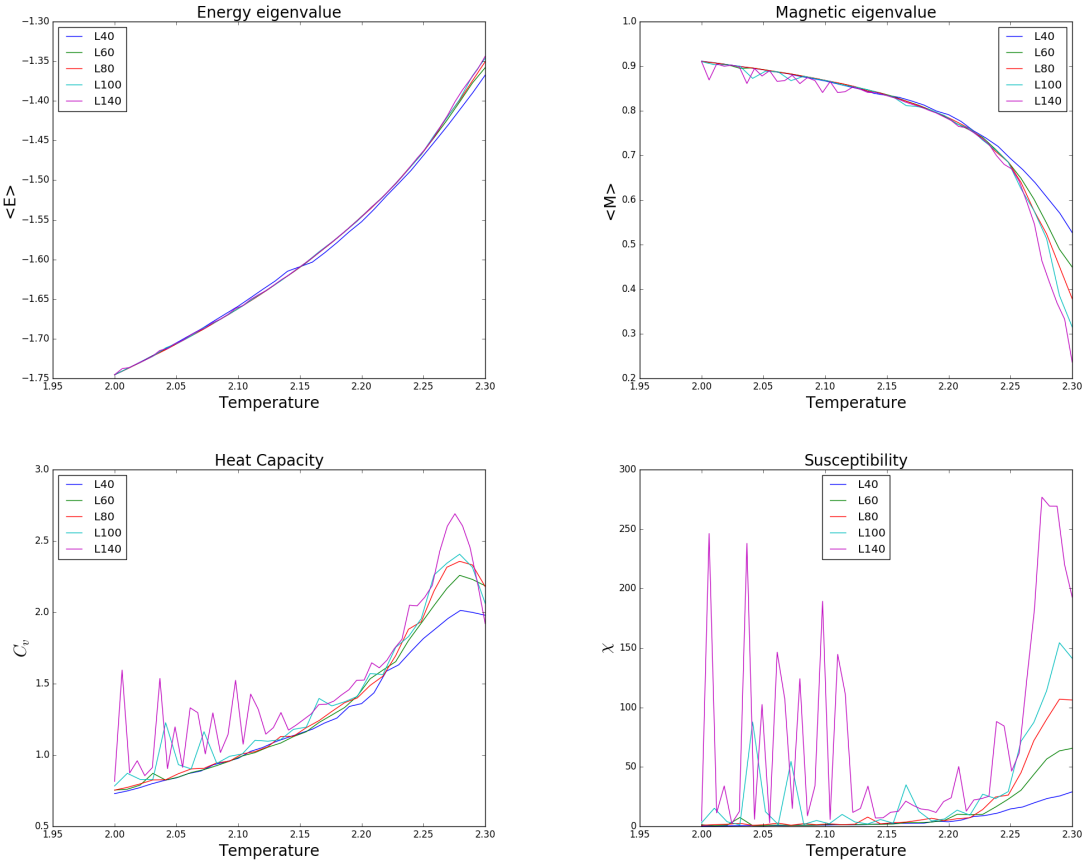

Figure 4: Energy eigenvalues, magnetic eigenvalues, Heat capcity and Susceptibility as a function of different temperatures in the interal $T \in [2.0, 2, 3]$ for different L-values. $dt = 0.1034$ for $L \in [40, 100]$ and $dt = 0.00612$ for L = 140. As we can see, there is a good deal unstability at L = 140.

The phase shift will happen at the maximum value of C_v and χ. In table 5 the different max temperatures for C_v and χ for each L is written down:

Discussion

L = 2

Looking at table 3 we can see that the values comes closer and closer to the analytical ones and then starts fluctuating around it with increasing Monte Carlo cycles. This is as we should expect given and I would deam this run stable at about 100000 Monte carlo Cycles. Comparing with our grafs in figure 1 this seems to be a reasonable assumption.

L = 20

From figure 1 we can see that the values stabilizes at about 80 000 for T = 1.0 and 60 000 at T = 2.4. These values makes sense. When we start of the Monte Carlo cycles all spins are either randomly divided or all are pointing upwards. And as we see there is a big difference in the start before they close in on each other. The reason why we have such a large threshold for T = 1.0 is maybe not clear. After all the y-axis on these plots are a lot shorter than for T = 2.4. It is simply because this was the point where the overall stability seemed to be from the graph. One could argue that since we don't have such a detailed graph of T = 2.4 we should have a lower threshold for T = 1.0 than we do for T = 2.4, but as we have 1000000 Monte Carlo cycles there is little harm done, and the graph is more stable at 80000 than at say, 30000.

Figure 2 shows us that the number of accepted states increases rapidly with temperature. My theory is that since we only accpet $r \leq e^{-\beta \Delta E}$ in the metropolis and β decreaces with larger T, this will make $e^{-\beta \Delta E}$ come closer to 1 for positive ΔE. Therefore we will go faster towards equilibrium. Simply summarized: When we increase T, β decreases and we have more accepted states. What is interesting to note is that we have the most rapid increase about around where the critical temperature found by Onsager is.

The plots in fiugre 3 were a pain. I don't know where the problem is and after about 15 hours on this task alone I have given up and I am content with my results as they are. If I had done this project again I would have asked for help about this more than I did this time and wait until I was 100% sure I had the correct answer. As we can see from the upper most plot, there are most energies around the start energy. This makes sense as we can see from figure 1 the cases of T = 1.0 are quickly becomming stable and there is little fluctiuation.
For T = 2.4 the energies are more spread out. Once again, this makes sense. First of all, there will be more accpeted states and as we see from figure 1 the values fluctuate more here than compared to the T = 1.0 case.
The variance values in table 4 are completely off and I think it has something to do with how we aquire our values. If I had done the task again, I would especially have focused on how I write to file and find out why the variance is weird while the plots make sense.
Another problem with the plots are that they are not correctly normalized. This is because I use the `.hist` function from `matplotlib.pyplot` and the normalize function there is not working for me.

L $\geq$ 20

This was the most fun part of the program as we did huge calculations in a relatively short time period. As we can see from figure 4 the values are as eexpecting increasing with L and temperature with a peak in temperature at the critical temperature. From table 5 we can see that the critical

temperatures comes closer and closer to the limit found by Onsager and this shows us that we have good data sets.
Why the L = 140 curve is acting out is a mystery to me, but I have two thoughts: 1. The algorithm becomes unstable for so large values of L.
2. Since we have more temperatures, there may be the same fluctuation in all graphs, but it is only here it shows.
I think the first point is the most likely.
Had I done the experiment again, I would have done this for even higher L and with the same temperatures for all L.

All timing was done by my friend who hasn't given me the data, but the last was done in right under two hours.

Conclusion

This has been a fun project when one overlooks task d. We have found that there was a good correlation between the analytical and the numerical values. The algorithm becomes stable at around 100000 both for L = 2 and L = 20. Accepted states increases as temperature increases because $\beta = 1/T$. Our histograms in figure 3 are badly normalized, but make sense. The variance values found numerically with numpy are terribly wrong.

References