FYS4150 - Project 4

Eirik Ramsli Hauge

16. November 2016

Abstract

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:

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

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

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:

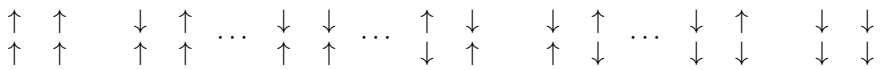

Table 1: Possible spin configurations (degeneracy): All spins up (1), one spin down (4), two spins down side by side (4), two spins down diagonally (2), three spins down (4), all spins down (1). Should I remove degeneracy?

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

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	

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)

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

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

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 $i = 0 \rightarrow i = L$ **do**
  for $j = 0 \rightarrow j = L$ **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. REF!

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

Do this when you know what to do

Experimental

The programs used in this project can be found in the GitHub repository REF 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 REF HER ikke nedenfor 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 textREF! 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 fairly similar to the one in task b), but we have added a counter for each accepted configuration. WRONG! SAME FOR BOTH! `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 Kreve dette? 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. Eller: but only when all the cores have ran through all Monte Carlo Cycles with its temperature.

Results

This is what we got when we did the stuff we knew

Discussion

Why did we end up with what we did when we did the stuff we knew? Who knows?

Conclusion

I'm afraid I seem to have strayed somewhat from my original brief. But in a nutshell: sex is more fun than logic. One cannot prove this, but it 'is' in the same sense that Mount Everest 'is', or that Alma Cogan 'isn't'.

Goodnight.