Fys4150
Project 4

Karl Jacobsen

in collaboration with Peter Killingstad
`https://github.com/kaaja/fys4150`

November 19, 2017

Note to instructurs reagarding Github repository

If the above Github-link does not work, it is eighter because you have not yet accepted our invite to the repository, or you have not yet provided us with an e-mail addres available at Github so that we can invite you. The Github user you will be invited from is "kaaja". If the latter applies to you, please send us an e-mail with an e-mailadress available in Github or your Github username so that we can send you an invite. My e-mailadress: karljaco@gmail.com.

Abstract

Monte Carlo simulations and the Metroplois algorithm are applied to study phase transitions in a magnetic sysmtem using the two-dimensional Ising model. The Metroplis algorithm is motivated by a theoretical derivation linking our problem to Markov chains. The main theoretical aspects of the Ising model is shown. Expressions for the implemented Metroplis algorithm that is shown to reduce FLOPS drastically are derived. For the 2×2 lattice case we derive analytical results for energy, magnetization, specific heat capacity and suspecptibility, and these are reproduced well numerically. For a 20×20 lattice we study the equilibration time and the share of accepted moves for two different temperatures (one low and one close to the critical temperature) and two different initial spin configurations (one random and one fixed giving ground state). For a low temperature, a fixed initial spin configuration giving the ground state gives significantly lower equilibration times, while all other combinations of temperature and initial spin configuration gives about the same equailibration time (2^{18} MC cycles). The acceptance share is always lower for the low temparature, approaching zero towards equilibrium. The probability distriubution for energy and its variance does not vary much between type of initial spin configurations, while there are huge differences between temperatures. In the low temperature case the distribution is very left skewed, with the ground energy occupying 90 per cent. In the high temperature cases, the probability distribution becomes much wider. We derive an estimator for the critical temperature $T_C(L = \infty)$, and the numerical results for lattice sizes from 40×40 and 60×60 up to 100×100 lattices all deviates less than one percentage from the analytical result found by Onsager. Parallel computing with MPI is used. The computational time seems to be $\mathcal{O}(1/\#\text{ processors})$.

1 Introduction

Monte Carlo simulations of Markov chains are widely used in many areas of science, from physics to finance. Knowledge about these subjects is important. In this report the main workings of these methods will be shown by analytical expressions and through numerical implementation of these equations.

I order to calculate a Markov process, a so called stochastic matrix, containing transition probabilities between all possible states of the system, is necessary. In pricatice the stochastic matrix in the basic Markov chain equation is unknown. In order to calculate the Markov process, we need to approximate the stochastic matrix. The Metropolis algorithm, which according to Hjorth-Jensen [1] p. 3 is among the top ten algorithms in the world, is such an approximation of the the stochastic matrix. We will derive the Metropolis algorithm.

The above mentioned methods are applied to the physical problem of phase transitions in magnetic systems by using the two dimensional Ising-model. The Ising model is much used in the study of phase transitions, and is one of the most wideliy used models in statistical physics. Knowledge about the Ising-model is important if one wants to simulate complex phase transitions. We will set up and explain the main equations of the Ising model.

Monte Carlo methods are statistical methods, and as such the validity of the results depends greatly on having a significant number of observations. Knowledge about how the simulation results behave as a function of the number of simulations is necessary for judging the validity of the results. To investigate the dependence of the results on the number of simulations, we study the so-called equilibration time. The equilibration time is the time, which in our case is the number of Monte Carlo cycles, needed for different statistical quantities to stabilize around their so-called steady state levels. Equilibration times for the mean energy, the mean magnetization, and their variances are studied.

The results from our studies of equilibration times reveal what is often true for Monte Carlo simulations: one needs a lot of simulations to reach steady state. Monte Carlo methods quickly become computationally expensive. Hence methods to improving the efficiency of the implemented Monte Carlo algorithms can be of great benefit. In these regards we firstly use our knowledge about the Ising model when we implement the Metropolis algorithm. By using our knowledge about the Ising model, we make the implemented Metropolis algorithm considerable more efficient in terms of FLOPS compared to a more brute force implementation. FLOP counts of the difference between our implemented Metroplois method and a straight forward method are derived. Secondly, we apply MPI using parallell computing to speed up the computation times. We investigate the speed ups as function of number of processors, and the results show great gains with regard to computational speed when parallel computing is applied.

In order to investigate the quality of our numerical implementation, we make several tests. Firstly, we study the share of accepted moves in from the Metropolis algorithm as a function of temperature and type of initial spin configuration. We find that the acceptance share is almost zero for the low temperature when the initial spin configuration is fixed to give the lowest energy. This is as expected, since the equilibrium for low temperature is close to the ground state which we started from. For a random initial spin configuration, the acceptance share starts out higher, as the energy moves down to the ordered equilibrium, and then stabilizes areound zero. For a temperature close to the critical temperature, the temperature where the phase transition takes place, the share of accepted moves is higher compared to the low temperature case regardless of initial spin configuration type. Also this is as expected, since the equilibrium state is a state with higher energy, increasing the probability of accepting moves.

The effects of initial spin configuration and temperature on the equilibration times are studies. For the low temperature case we find the expected effect of initial spin configuration. For low temperatures, an ordered fixed initial spin configuration, where the spins gives the lowest energy, gives considerably lower equilibration times. Somewhat unexpected we find that for a temperature close to the critical temperature, the temperature where the phase transition takes place, there is little difference in equalibration time between the fixed initial spin configuration and a random initial spin configuration. Based on the Ising model we expected lower equilibration times for the random initial spin configuration, since the equilibrium state in this case would be unordered.

We investigate the probability distribution of energy states for the two different temperatures and the two different initial spin configurations. For the low temperature cases the distribution is verly left skewed, with 90 per cent of the occurances in the lowest energy state. This was expected, since the acceptance share is low when the lowest energy is reached in the low temperature case. In the high temperature cases the variation among energy states are significanlty higher compared to the low temperature cases. For high temperatures, the probability distribution becomes much wider, reflecting the Boltzman-distribution.

The final tests of our results is comparison agains analytical results. Firstly, we derive analytical expressions for the 2D Ising model in the case of only two spins and perioc boundary conditions. We run the same case with our program and reproduce the analytical results well. Secondly, we derive an estimator for the critical temperature that applies simulated critical temperatures from different lattice sizes. The estimator gives results that differ less than 1% from Onsager's analytical result when applying only a 40×40 and a 60×60 lattice. Including larger lattices improves the results further.

2 Theory

2.1 The Ising model

We study the relationship between quantities such as magnetism, energy and temperature in a material by applying the Ising model. The Ising model is an atomic lattice model, where one models the material by assuming that it consists of a lattice of atoms. Each atom has a spin, representing the magnetism, which can have two values, up or down. In our case we apply a 2 dimensional lattice.

The model gives the following equation for the energy

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

where $J > 0$ is a constant reflecting the strength of the interaction between neighbouring spins, $s_k = \pm 1$ is the spin at position k, β is an external magnetic field, N is the total number of spins, and $< kl >$ means neigbours so that we sum over only nearest neigbours. In our anayzis we simplify by assuming $\beta = 0$. Note that $\beta = 0$ does not mean that there is no presence of magnatization. A given magnetic moment can through interaction with neigbours indirectly influence non-neigbouring spins through a kind of chain event.

The magnetization is given by

$$M = \sum_{j=1}^{N} s_j. \tag{2}$$

The above formulae gives the energy and the magnetic moments for a given spin configuration. That is good. But what if we do not know the spin configuration? We are then interested in the most likely state. This is where statitistics comes in to play. Expectation values is of great interest. For a given quantity x, the expectation value, in the discrete case, is given as

$$< x >= \frac{1}{Z} \sum_{j} P_j x_j, \tag{3}$$

where j is the possible states and P_j/Z is the probability distribution function (PDF) representing the probability of being in the given state. Z is the partition function that ensures that the PDF is normalized.

We apply the Boltzmann PDF

$$P_i(\beta) = \frac{e^{-\beta E_i}}{Z}, \tag{4}$$

where

$$\beta = 1/k_B T, \tag{5}$$

and $Z = \sum_j p_j$, where $\sum_j$ represents all possible states, is the partition function, which ensures that the Boltzmann PDF is normalized.

Another statistical valriable that is central, is the variance, which for a given variable x is

$$\sigma^2 =< x^2 > - < x >^2 . \tag{6}$$

The variance σ_x^2 says something about the fluctuations in the variable x. The standard deviation, $\sigma_x = \sqrt{\sigma_x^2}$, is the average deviation of x from the mean of x.

So in order to calculate expectation values for given variables, we need the expressions for these variables at the given states and also the PDF. Applying the energy formula without external magnetization, we get the following energy for a given state i

$$E_i = -J \sum_{kl}^{N} s_k s_l \tag{7}$$

In the next section we will calculate exact solutions for a 4 spins case for the 2D Ising model, but we will apply some results from Hjorth-Jensen [1] directly, so that it might not be fully clear when reading this report what is really going on when calculating energy states. To make up for this, I follow Hjorth-Jensen's [1] 1D 2-spin example on p 422, so that it is clear how energies are calculated.

With 2 spins, we can draw two arrows next to each other that represent the spins. Arrow up is numerically represented by 1 and arrow down is represented by -1. Now we will apply the formula for energy in a given state (7) to this spin system of 2 spins.

Firstly, there will be $2^2 = 4$ different spin configurations, since there are two possible spin values and there are two spins in the lattice. Note that this does not mean that there are 4 different energy states E_i. Assume free ends, meaning that the spins at the ends are not multiplied with anything. Following Hjorth-Jensen [1] we get

$$E_1 = E_{\uparrow\uparrow} \overset{(7)}{=} -J \cdot 1 \cdot 1 = -J \tag{8a}$$

$$E_2 = E_{\uparrow\downarrow} \overset{(7)}{=} -J \cdot 1 \cdot (-1) = +J \tag{8b}$$

$$E_3 = E_{\downarrow\downarrow} \overset{(7)}{=} -J \cdot (-1) \cdot (-1) = -J \tag{8c}$$

$$E_4 = E_{\downarrow\uparrow} \overset{(7)}{=} -J \cdot (-1) \cdot 1 = +J \tag{8d}$$

From the above equations, (8), we see that there are four different spin configurations, while there are only two different energy states. We say that the two different energies both have a degeneracy of two. As we very soon will see, knowledge about the number of different energy and magentic states and their degeneracy will ease the calculations of the analytical expressions for statistical variables such as expectation values.

The expectation value of energy is given by

$$< E >= \frac{1}{Z}\sum_i E_i P_i \overset{(4)}{=} \frac{1}{Z}\sum_j e^{-\beta E_i}. \tag{9}$$

If we have precalculated the different energy states E_i and their degeneracy, the calculation of expectation values becomes easier. In our example above with 1D and 2 spins, the expectation value is calculated as follows

$$< E > \overset{(9)}{=} \frac{1}{Z}\sum_j e^{-\beta E_i} \tag{10a}$$

$$= \frac{1}{Z}\Bigg(2 \cdot (1e^{-\beta(-J)}) + 2 \cdot (1e^{-\beta(J)})\Bigg) \tag{10b}$$

So instead of calculating the expected energy applying all four spin configurations, which would give four terms, we above only need half the number of terms. This method is used in most calculations in the next section.

2.2 Phase transitions

A phase transition is when some characteristics of the material changes type. Examples of phase transitions are boiling of water where the material changes from liquid to gas, and, in our case, when a material changes from being magnetic to not being magnetic.

We are interested in the so-called critical temperature, T_C, the temperature for which the phase transition happens. There exists an analytical result for $T_C(L = \infty)$, where L is the Lattice size, Onsager [3]. This analytical result gives us an opportunity to test the validity of our numerical implementation. Hence we are interested in estimating $T_C(L = \infty)$ from our numerical results.

Following Hjorth-Jensen [4] p.3 the behavior of finite and infinite lattices relates according to

$$T_C(L) - T_C(L = \infty) = aL^{-1/\nu}, \tag{11a}$$

where a is a constant and ν is a parameter in the formula for the so-called correlation length

$$\xi(T) \sim |T_C - T|^{-\nu}\,. \tag{12}$$

The correlation length is the length for which the overall and bulk properties of the material starts to differ, Hjorth-Jensen [1] p. 429.

We will follow Hjorth-Jensen [4] p.3 and choose $\nu = 1$ so that (11) reduces to

$$T_C(L) - T_C(L = \infty) = aL^{-1}. \tag{13}$$

Now we need an estimate for the constant a. As a first step of making this estimate we get rid of $T_C(L = \infty)$ in (13), by solving (13) for two different lattice lengths and combining the results

$$T_C(L_i) - T_C(L = \infty) = aL_i^{-1}, \tag{14a}$$

$$T_C(L_j) - T_C(L = \infty) = aL_j^{-1}. \tag{14b}$$

Now (14a) - (14b) gives

$$\begin{aligned}
T_C(L_i) - T_C(L_j) &= aL_i^{-1} - aL_j^{-1} && (15a)\\
&= a(\frac{1}{L_i} - \frac{1}{L_j}) && (15b)\\
\rightarrow a &= \frac{T_C(L_i) - T_C(L_j)}{\frac{1}{L_i} - \frac{1}{L_j}}. && (15c)
\end{aligned}$$

We now insert a (15c) with $L = L_i$ into (13) and solve for $T_C(L = \infty)$

$$\begin{aligned}
T_C(L = \infty) &\overset{(13)}{=} T_C(L_i) - aL_i^{-1} && (16a)\\
&\overset{(15c)}{=} T_C(L_i) - \frac{T_C(L_i) - T_C(L_j)}{\frac{1}{L_i} - \frac{1}{L_j}} L_i^{-1} && (16b)\\
&= T_C(L_i) - \frac{T_C(L_i) - T_C(L_j)}{1 - \frac{L_i}{L_j}} && (16c)
\end{aligned}$$

(16c) is an estimator for $T_C(L = \infty)$. However, (16c) is only based on one L. Since a is supposed to be a constant that is independent of L, our estimator for $T_C(L = \infty)$ should improve by inclusion of all the L's we have available. We can include all the available L's by taking the average of all the possible $T_C(L = \infty)$'s using (16c)

$$T_C(L = \infty) = \frac{1}{\binom{N}{2}} \sum_i \sum_{i \neq j} \Big(T_C(L_i) - \frac{T_C(L_i) - T_C(L_j)}{1 - \frac{L_i}{L_j}} \Big), \tag{17a}$$

where N is the numner of available lattice sizes. (17a) will be applied in our numerical simulations, testing the model's ability predicting T_C.

2.3 Analytical results: 2D Ising model with 4 spins

A very useful method to check an implemented numerical algorithm, is to check it against analytical results. In our case we are so lucky that we can calculate analytical resutls for the 2D Ising model with 4 spins. These analytical expressions will be derived in this section.

For the case of periodic boundary conditions (PBCs), we will here derive the analytical expressions for the partition function Z, the expected value of energy $< E >$, the expected values of the absolute magnetic moment $< |M| >$, the specific heat capacity C_V and the susceptibility χ.

Before deriving the above mentioned expressions, a few words about PBCs. In our case we evaluate neigbouring spin values, $s_i s_{i+1}$. Say we have N spins. The question is what to do about the last neighboring spins, $s_N s_{N+1}$, as we do not have s_{N+1}. One approach, called the free end approach, is to just ignore this last value. Another alternative, called PBCs, is to assume $s_{N+1} = s_1$, where s_1 is the first spin. This last alternative is what we are implementing and deriving analytical expressions for below. According to Hjorth-Jensen [1] p. 423 - 424, the choise between the free end and periodic BCs, only affect the results for energy for small dimensions of the lattice. For large lattices, the results are unaffected.

We start by deriving the so-called partition function. In simple terms, this is a function that ensures normality of the probability distribution function PDF. For the Boltzmann PDF, that we use, we have

$$p_i = \frac{e^{-\beta E_i}}{Z}, \ i = 1, ..., N, \tag{18a}$$

where $Z = \sum_j p_j$, where $\sum_j$ represents all possible states, is the partition function, which ensures that the Boltzmann PDF is normalized.

Now we will apply the method described in the above section for calculation of statistical values when we know the different states and their degeneracy. We apply the results listed in table 13.4 on p. 424 in Hjorth-Jensen [1], where Hjorth-Jensen has calculated all possible states. For convinence of reading, we reprint the table here.

Number of spins up	Degeneracy	Energy	Magnetization
4	1	$-8J$	4
3	4	0	2
2	4	0	0
2	2	$8J$	0
1	4	0	-2
0	1	$-8J$	-4

Table 1: Energy and magnetization for 2D Ising model with 2 × 2 spins and period BCs. Source: Hjorth-Jensen [1] p. 424

Applying the results from the table 1, we get the following calculation for the partition function

$$\begin{aligned}
Z_N &= \sum_{i=1}^{\text{Number of states}} e^{-\beta E_i} && (19a)\\
&= 1 \cdot \left(e^{-\beta E_i}\right)_{4 \text{ up}} + 4 \cdot \left(e^{-\beta E_i}\right)_{3 \text{ up}} + 4 \cdot \left(e^{-\beta E_i}\right)_{2 \text{ up}} + 2 \cdot \left(e^{-\beta E_i}\right)_{2 \text{ up}} + 4 \cdot \left(e^{-\beta E_i}\right)_{1 \text{ up}} + 1 \cdot \left(e^{-\beta E_i}\right)_{0 \text{ up}} && (19b)\\
&= 1 \cdot e^{8\beta J} + 4 \cdot e^{0\beta J} + 4 \cdot e^{0\beta J} + 2 \cdot e^{-8\beta J} + 4 \cdot e^{0\beta J} + 1 \cdot e^{8\beta J} && (19c)\\
&= 12 + 2(e^{8\beta J} + e^{-8\beta J}) && (19d)\\
&= 12 + 4\cosh(8\beta J) && (19e)\\
&= 4\,(3 + 1\cosh(8\beta J)) && (19f)
\end{aligned}$$

Next we calculate the expted value of energy, $< E >$ by applying a trick from the lectures:

$$\begin{aligned}
-\frac{\partial \ln Z}{\partial \beta} &= -\frac{1}{Z}\frac{\partial Z}{\partial \beta} && (20a)\\
&\overset{(19a)}{=} \frac{\sum_{i=1}^{N} E_i e^{-\beta E_i}}{Z} && (20b)\\
&\overset{(9)}{=} < E > && (20c)
\end{aligned}$$

By applying (20) on our partition function (19d) we obtain

$$
\begin{aligned}
< E > &= -\frac{\partial \ln Z}{\partial \beta} && \text{(21a)}\\
&= -\frac{\partial \ln \Big(12 + 2(e^{8\beta J} + e^{-8\beta J})\Big)}{\partial \beta} && \text{(21b)}\\
&= -\frac{1}{Z} 8J \cdot 2(e^{8\beta J} - e^{-8\beta J}) && \text{(21c)}\\
&= -\frac{1}{Z} 16J(e^{8\beta J} - e^{-8\beta J}) && \text{(21d)}\\
&= -\frac{32Jsinh(8\beta J)}{Z}. && \text{(21e)}
\end{aligned}
$$

The expected value of the absolute magnetic moment, $< |M| >$, we find utlizing the results in table 1

$$
\begin{aligned}
< |M| > &= \sum_{i=1}^{2^4} p_i |M_i|\\
&= \frac{1}{Z}\Big(1 \cdot \big(e^{-\beta E_i}|M_i|\big)_{4 \text{ up}} + 4 \cdot \big(e^{-\beta E_i}|M_i|\big)_{3 \text{ up}} + 4 \cdot \big(e^{-\beta E_i}|M_i|\big)_{2 \text{ up}} + 2 \cdot \big(e^{-\beta E_i}|M_i|\big)_{2 \text{ up}}\\
&+ 4 \cdot \big(e^{-\beta E_i}|M_i|\big)_{1 \text{ up}} + 1 \cdot \big(e^{-\beta E_i}|M_i|\big)_{0 \text{ up}}\Big)\\
&= \frac{1}{Z}\Big(1 \cdot 4e^{-\beta(-8J)} + 4 \cdot 2e^{-\beta(-0J)} + 4 \cdot 0e^{(}) + 2 \cdot 0e^{(}) + 4 \cdot 2e^{-\beta(-0J)} + 1 \cdot 4e^{-\beta(-8J)}\Big)\\
&= \frac{1}{Z}\Big(4e^{8\beta J} + 8 + 8 + 4e^{8\beta J}\Big)\\
&= \frac{8}{Z}(e^{8\beta J} + 2).
\end{aligned} \tag{22a}
$$

The specific heat capacity, C_V, is given by

$$
C_V = \frac{1}{K_B T}\Big(< E^2 > - < E >^2 \Big). \tag{23a}
$$

We calculate the two terms in (23) separately. Utlizing the results intable 1 we get

$$
\begin{aligned}
< E^2 > &= \frac{1}{Z}\Big(1 \cdot \big(e^{-\beta E_i}E_i^2\big)_{4 \text{ up}} + 4 \cdot \big(e^{-\beta E_i}E_i^2\big)_{3 \text{ up}} + 4 \cdot \big(e^{-\beta E_i}E_i^2\big)_{2 \text{ up}} + 2 \cdot \big(e^{-\beta E_i}E_i^2\big)_{2 \text{ up}}\\
&+ 4 \cdot \big(e^{-\beta E_i}E_i^2\big)_{1 \text{ up}} + 1 \cdot \big(e^{-\beta E_i}E_i^2\big)_{0 \text{ up}}\Big)\\
&= \frac{1}{Z}\Big(1 \cdot (-8J)^2 e^{-\beta(-8J)} + 4 \cdot (0)^2 e^{-\beta(-0J)} + 4 \cdot (0)^2 e^{(}) + 2 \cdot (8J)^2 e^{-\beta(8J)} + 4 \cdot (0)^2 e^{(}) + 1 \cdot (-8J)^2 e^{-\beta(-8J)}\Big)\\
&= \frac{1}{Z}\Big(64J^2 e^{8\beta J} + 128J^2 e^{-8\beta J} + 64K^2 e^{8\beta J}\Big)\\
&= \frac{256\cosh(8\beta J)}{Z}.
\end{aligned} \tag{24a}
$$

For $< E >^2$ we have

$$
\begin{aligned}
< E >^2 &\overset{(21e)}{=} \Big(-\frac{32Jsinh(8\beta J)}{Z}\Big)^2 && \text{(25a)}\\
&= \frac{32^2 J^2 \sinh^2(8\beta J)}{Z}. && \text{(25b)}\\
& && \text{(25c)}
\end{aligned}
$$

Insertion of $< E^2 >$ (24) and $< E >^2$ (25b) into C_V (23) gives

$$C_V = \frac{1}{K_BT}\Big(< E^2 > - < E >^2 \Big) \tag{26a}$$

$$= \frac{1}{K_BT}\Big(\frac{256\cosh(8\beta J)}{Z} - \frac{32^2 J^2 \sinh^2(8\beta J)}{Z}\Big) \tag{26b}$$

$$= \frac{1}{K_BTZ}\Big(256\cosh(8\beta J) - 32^2 J^2 \sinh^2(8\beta J)\Big). \tag{26c}$$

Now for the susceptibility, χ, which becomes

$$\chi = \frac{1}{K_BT}\Big(< M^2 > - < M >^2 \Big) \tag{27a}$$

$$= \frac{1}{K_BT}\Big(\sum_{i=1}^{2^4} p_i M_i^2 - (\frac{8}{Z}(e^{8\beta J}+2)^2\Big) \tag{27b}$$

$$= \frac{1}{K_BT}\Bigg(\frac{1}{Z}\Big(1e^{-\beta(-8J)}4^2 + 4e^0 2^2 + 0 + 0 + 4e^0(-2)^2 + 1e^{-\beta(-8J)}(-4)^2\Big) - \frac{64}{Z^2}(e^{8\beta J}+2)^2\Bigg) \tag{27c}$$

$$= \frac{1}{K_BT}\Bigg(\frac{1}{Z}\Big(16e^{-\beta(-8J)} + 16 + 16 + 16e^{-\beta(-8J)}\Big) - \frac{64}{Z^2}(e^{-8\beta J}+2)^2\Bigg) \tag{27d}$$

$$= \frac{32}{K_BTZ}\Bigg(e^{\beta 8J} + 1 - \frac{2}{Z}(e^{8\beta J}+2)^2\Bigg). \tag{27e}$$

3 Numerical algorithm

The magnetism and energy for the lattice changes with the spin configurations. We are interested in the most likely state, the so-called steady state of the material. Since we do not know the order of the spins in the lattice, we want to calculate statistical values such as expectation values and variances. In principe we could calculate the expectation vaslues and variances by the same method used in the case above for the 2×2-lattice, by calculating the PDF directly. However, a direct calculation of the PDF would in cases of larger lattices become computationally expensive, since the partition function would include $2^{\text{Number of spins}}$ terms.

One alternative to computing the partition function directly, is to introduce randomness in the lattice configuration together with physics (Boltzmann PDF) and try to simulate the steady state by applying the Monte Carlo (MC) method. This is the approach that will be followed here.

For each temperature we apply the MC method to try find the steady state. The MC algorithm goes as follows

```
for cycle in Monte Carlo Cycles
   - Run through the lattice and change spins according to a rule (Metropolis algo)

   - When lattice is swept through: update energy, magnetism
end Monte Carlo loop

Compute average energy, magnetisme, variances, etc.
```

When one runs the MC algorithm above, one hopes that the the expectation values of the physical qunatitites stabilizes when the number of MC cycles has reached a point that does not involve too many MC cycles. Running more cycles past this point should then not affect the averages, and we have then reached a steady state.

In the MC algorithm above, there is one black box, and that is the point about ”change spins according to a rule (Metroplos algo)”. The Metropolis algorithm will be discussed in the next subsection.

3.1 The Metropolis algorithm

A run through the lattice corresponds to one step of a Markov process. A Markov process typically follows a rule like

$$w_i = \sum_j W_{j\to i} w_j, \tag{28}$$

where w_i is the probaility of being in state i, w_j is the probability of being in state j, and $W_{j\to i}$ is the transition probability representing the probability of going from all different states j to state i. In our case, we have that the probability distribution is given by the Boltzmann distribution

$$w_i = e^{-\beta E_i}/Z. \tag{29}$$

The w's are part of a probability vector. One element in the probability vector contains the probability of being in that specific state at the given time. The length of the probability vectors is one, since all possible states is included.

In our case the dimension of the probability vector could equal the number of possible spin configurations, which would be $2^{\text{LatticeNumber}_x \cdot \text{LatticeNumber}_y}$, where "LatticeNumber$_x$" and "LatticeNumber$_y$" is the number of lattice points in the x and y direction respetively. The word "could" is used, since spin configuration is just one alternative variable. If one e.g. wants to analyze energy, one would most likely have another number of possible energy states, changing the dimension of the probability vector for energy.

$W_{j\to i}$ in (28) is a row in a stochastic matrix, meaning that the column elements of this matrix sums to one. This stochastic matrix contains the information about the transition probabilities between all different states. So the above equation (28) feeds a given probability distribution of states into the stochastic matrix, which then produces the new probability distribution of states. In our lattice case, say we have one spin configuration. Then we apply the stochastic matrix to this configuration, and we get a new configuration. So applying the matrix W is the same as one full sweep through the lattice in our case.

From linear algebra we know that Markow chains converges to a steady state solution, and that this solution is given by the eigenvector of the largest eigenvalue. The largest eigenvalue is one. In a steady state we would have

$$\mathbf{w}(\mathbf{t} = \infty) = W\mathbf{w}(\mathbf{t} = \infty), \tag{30a}$$

which is nothing but an eigenvalue problem with eigenvalue equal to one.

There is a problem with the above: we don't know the stochastic matrix! This is where the Metropolis algorithm comes into play. With the Metropolis algorithm the stochastic matrix is split into two matrices

$$W(j \to i) = A(j \to i)T(j \to i), \tag{31}$$

where $T(j \to i)$ is the probability of proposing a given state i from state j, and $A(j \to i)$ is the probability of accepting the proposed jump $T(j \to i)$. The T-matrix must have columns summing to one, since all states are included.

Assume we want to study the system over time using the above decomposition of the stochastic matrix. (28) now becomes

$$w_i(t+\epsilon) = \sum_j \left(\overbrace{w_j(t)T(j \to i)A(j \to i)}^{\text{Coming from } j} + \overbrace{w_i(t)T(i \to j)(1 - A(i \to j)}^{\text{Staying in } i} \right). \tag{32a}$$

To get an expression for the steady state, we manipulate (32)

$$\begin{aligned}
w_i(t+\epsilon) &\overset{(32)}{=} \sum_j \Big(w_j(t)T(j\to i)A(j\to i) + w_i(t)T(i\to j)(1-A(i\to j)\Big) && \text{(33a)}\\
&= \sum_j \Big(w_j(t)T(j\to i)A(j\to i) - w_i(t)T(i\to j)A(i\to j)\Big) + \sum_j w_i(t)T(i\to j) && \text{(33b)}\\
&= \sum_j \Big(w_j(t)T(j\to i)A(j\to i) - w_i(t)T(i\to j)A(i\to j)\Big) + w_i(t) && \text{(33c)}\\
w_i(t+\epsilon) - w_i(t) &= \sum_j \Big(w_j(t)T(j\to i)A(j\to i) - w_i(t)T(i\to j)A(i\to j)\Big) && \text{(33d)}\\
0 &\overset{\text{Steady state}}{=} \sum_j \Big(w_jT(j\to i)A(j\to i) - w_iT(i\to j)A(i\to j)\Big). && \text{(33e)}
\end{aligned}$$

The above steady state condition (33e) turns out to be a too mild condition. Applying only the above condition can result in oscillatory solutions, which is not a steady solution. The extra condition of detailed balance.

$$w_jT(j\to i)A(j\to i) = w_iT(i\to j)A(i\to j), \tag{34}$$

guarantees that the steady state is reached.

In the detailed balance equation (34) we only have the w's, and these are given by the Boltzmann distribution (29). Lets manipulate the detailed balance equation (34) slightly

$$\begin{aligned}
w_jT(j\to i)A(j\to i) &\overset{(34)}{=} w_iT(i\to j)A(i\to j) && \text{(35a)}\\
\frac{w_j}{w_i} &= \frac{T(i\to j)A(i\to j)}{T(j\to i)A(j\to i)}. && \text{(35b)}
\end{aligned}$$

Now assume an uniform probability distribution function (PDF) for T

$$T(i\to j) = \frac{1}{N} = T(j\to i). \tag{36a}$$

The interpretation of the uniform PDF assumption for T is that we assume that the probability for jumping between states is equal between all states.

Insertion of the uniform PDF (36) into the detailed balance condition (35b) gives

$$\begin{aligned}
\frac{w_j}{w_i} &= \frac{T(i\to j)A(i\to j)}{T(j\to i)A(j\to i)} && \text{(37a)}\\
\frac{w_j}{w_i} &\overset{(36)}{=} \frac{A(i\to j)}{A(j\to i)}. && \text{(37b)}
\end{aligned}$$

So now we are only left with determining the acceptence probabilitis A. Using the the Boltzmann distribution (29) in the left hand side of the last version of the detailed balance equation (37b) we get

$$\begin{aligned}
\frac{w_j}{w_i} &= \frac{e^{-\beta E_j}}{e^{-\beta E_i}} && \text{(38a)}\\
&= e^{-\beta\Delta E} && \text{(38b)}
\end{aligned}$$

Firstly, note that the partition function cancells out above. This is compuatationally very coest saving! Secondly, from (38b) we see that in our case, with Boltzmann PDF, a lower energy, $E_j < E_i \to \Delta E < 0$, gives $\frac{w_j}{w_i} > 0$, implying that lower energies are more likely. Combining this with (37b), we see that $E_j < E_i$, implies that $A(i \to j) > A(j \to i)$, so that the the probability of accepting moves to lower energy is larger than the probability of accepting moves to higher energy. This is reasonalble , since energy will be minimized in equilibrium, so that we would accept moves to lower energies more often than moves to higher probabilities. Based on this we make the first part of our acceptance rule, A

$$\text{If } \frac{w_j}{w_i} \geqslant 1 \to A(i \to j) = 1 \text{ accept.} \tag{39}$$

A tempting completion of the acceptance rule is adding the condition

$$\text{If } \frac{w_j}{w_i} < 1 \to A(i \to j) = 0 \text{ reject.} \tag{40}$$

The 2nd statement (40), however, would give bad results. One would only accept moves going to lower energies. This would violate the ergodicity condition, which states that every state of a system should be possible to reach from any given current state, if the system is simulated for a long enough time, see Hjorth-Jensen [1] p. 399 and p. 403. With (40) one could only reach all states in the single situation where one starts at the maximum energy. So we need an alternative to (40) that includes the possibility of going up in energy ($\frac{w_j}{w_i} < 1 \to A > 0$ is needed).

Now comes a trick. Set $A(j \to i) = 1$ in (37b), meaning that we accept all proposed moved from j to i

$$\frac{w_j}{w_i} \overset{(37b)}{=} \frac{A(i \to j)}{A(j \to i)} \tag{41a}$$

$$= \frac{A(i \to j)}{1} \tag{41b}$$

$$= A(i \to j). \tag{41c}$$

Now we have an expression for the acceptance probability in the cases where $\frac{w_j}{w_i} < 1$

$$\text{If } \frac{w_j}{w_i} < 1 \to A(i \to j) = \frac{w_j}{w_i}. \tag{42}$$

The 1st part of the acceptance rule (39) and the 2nd part of the acceptance rule (42) gives the following acceptence rule

$$A(i \to j) = \begin{cases} 1 \text{ (always accept)} & \text{if } \frac{w_j}{w_i} \geqslant \text{if } 1 \iff \frac{E_j}{E_i} \leqslant 1, \\ \frac{w_j}{w_i} = e^{-\beta(E_j - E_i)} & \text{otherwise.} \end{cases} \tag{43}$$

In the implementation we do not apply the acceptance rule exactly as stated in (43). We work with a binary acceptance probability, $A(i \to j) \in [0, 1]$, instead of the ratio $\frac{w_j}{w_i}$ in (43). We do this by picking a random number, $r \in [0, 1]$, and compare this number to the ratio $\frac{w_j}{w_i}$ to get

$$A(i \to j) = \begin{cases} 1 \text{ (always accept)} & \text{if } \frac{E_j}{E_i} \leqslant 1, \\ 1 & \text{if } \frac{E_j}{E_i} > 1 \text{ and } r < e^{-\beta(E_j - E_i)}, \\ 0 & \text{otherwise.} \end{cases} \tag{44}$$

The implemented acceptance rule (44) can be more compactly be written as

$$A(i \to j) = \begin{cases} 1 \text{ (always accept)} & \text{if } r < e^{-\beta(E_j - E_i)}, \\ 0 \text{ (always reject)} & \text{otherwise.} \end{cases} \tag{45}$$

(45) is what we implement in c++.

3.2 Energy and momentum calculations in Metropolis

There are several ways the Metropolis algorithm (45) can be implemented, and the different implementations have huge differences in numerical costs. Here I will describe two methods for implementing the Metropolis algorithm: a method where one runs through the full lattice flipping spins randomly before calculating the new energy E and feeding this E into Metropolis, and a method where one only flip one spin at the time, calculate E and call Metropolis for each spin flip. We start by deriving some FLOP numbers for the first method.

To calculate ΔE for the Metropolis algorithm (45), one must apply (7) to calculate the new energy:

$$E_i = -J\sum_{kl}^{N} s_k s_l \tag{46}$$

Applying periodic BCs (described in the section on analytical results) for a 2D lattice with 2 × 2 spins, (46) gives

$$\begin{aligned} E_i &\overset{(46)}{=} -J\sum_{kl}^{N} s_k s_l && \text{(47a)}\\ &= -J(s_1s_2 + s_2s_1 + s_3s_4 + s_4s_3 + s_1s_3 + s_3s_1 + s_2s_4 + s_4s_3) && \text{(47b)} \end{aligned}$$

(47b) gives $\Big(1(J\cdot\sum)+8(s_ks_l)+7(\sum)\Big)$ = 16 FLOPS. In the general case, (47b) gives $\approx 1+N_{Spins}^2 \approx N_{Spins}^2$ FLOPS. These FLOPS will be repeated $N_T \times N_{MCS}$ times (N stands for "number of"), so in total we get $\approx N_{Spins}^2 N_T \times N_{MCS}$ FLOPS when applying the method where one runs thorugh the full lattice before calculating E and applying Metropolis. If we can reduce the factor coming from the sum, N_{Spins}^2, this will quickly give large savings computationally. Note that the subtraction of the new energy, to get ΔE, would also give a FLOP. Since we have already approximated the FLOP count, and noting that the effect of this FLOP would be small compared to N_{spins} as N_{spins} grow, we leave it out of the count.

Now for the second method, which is the method that we apply. In this method we flip only one spin at the time, and then calculate ΔE for each flip we do. The key here is that only the neighbors of the flipped spin needs to be taken into account when calculating the new energy. The calculation of ΔE becomes

$$\begin{aligned} \Delta E &= E_j - E_i && \text{(48a)}\\ &\overset{(7)}{=} -J\sum_{kl} s_k^j s_l^j - (-J\sum_{<kl>} s_k^i s_l^i) && \text{(48b)}\\ &\overset{s_k^i = s_k^j}{=} J\sum_{<kl>} s_k(s_l^i - s_l^j) && \text{(48c)}\\ &\overset{s_j=-s_i}{=} 2Js_l^i \sum_{<k>} s_k && \text{(48d)} \end{aligned}$$

In (48d) the sum gives 4 FLOPS, since each spin has four neighbours, and the multiplications give 3 FLOPS, so in total we get 7 FLOPS. However, $2J$ in (48d) can be precalculated, so in practice we get 6 FLOPS. These 6 FLOPS will be repeated $N_T \times N_{MCS} \times N_{spins}$ times.

Compared to the method where one runs thorugh the whole lattice before calculating E, the difference is $\Big(N_{spins} - 6\Big) \cdot \Big(N_T \times N_{MCS} \times N_{spins}\Big)$ FLOPS. We see that as N_{spins} grow, the difference in FLOPS between the methods quickly becomes large. Note that the calculation for the FLOP difference is not exact, as we used an approximated expression for the FLOPS for method 1. This should not have much to say as N_{spins} get some size, because the left out term in the approximation would then be small compared to N_{spins}.

We conclude that flipping one spin at the time, and computing ΔE and calling Metropolis at each step when one runs through the lattice, is more computationally efficient compared to the method of running thorugh the whole lattice before calculating ΔE and calling Metropolis.

Still the method by flipping one spin at the time can be made more efficient. When applying the Metropolis algorithm above, (45), we see that we need to calculate $e^{-\beta E_j/E_i} = e^{-\beta \Delta E}$ many times. Say we have a 2D lattice with $N \times N$ spins. ΔE would then have to be calculated $MCS \times N \times N \times N_T$ times in total, where MCS is the number of MC cycles and N_T is the number of different temperatures. MCS is typically large, $MCS \geqslant 10^6$, so the calculations of $e^{-\beta \Delta E}$ quickly becomes expensive.

Following Hjorth-Jensen [1] p. 436, we will now show that there actually are only five possible ΔE's in the case where only one spin is flipped at the time. Also we will show how to take advantage of knowing the 5 different ΔE's when computing $e^{-\beta \Delta E}$.

Assume all spins in our 2D Ising model all turn up. Then switch one spin down. In the calculation of energy the spin flip will only affect the terms which inlcudes the spins that has been flipped. This means that we can ignore all other spins than the spin that is flipped and its neighbours! The following shows the case mentioned above in addition to the other possible ΔE's.

$$E \overset{(9)}{=} -4J \qquad \begin{array}{ccc} & \uparrow & \\ \uparrow & \uparrow & \uparrow \\ & \uparrow & \end{array} \qquad \Longrightarrow \qquad E \overset{(9)}{=} 4J \qquad \begin{array}{ccc} & \uparrow & \\ \uparrow & \downarrow & \uparrow \\ & \uparrow & \end{array} \qquad \Longrightarrow \qquad \Delta E = 8J$$

$$E \overset{(9)}{=} -2J \qquad \begin{array}{ccc} & \uparrow & \\ \downarrow & \uparrow & \uparrow \\ & \uparrow & \end{array} \qquad \Longrightarrow \qquad E \overset{(9)}{=} 2J \qquad \begin{array}{ccc} & \uparrow & \\ \downarrow & \downarrow & \uparrow \\ & \uparrow & \end{array} \qquad \Longrightarrow \qquad \Delta E = 4J$$

$$E \overset{(9)}{=} 0 \qquad \begin{array}{ccc} & \uparrow & \\ \downarrow & \uparrow & \uparrow \\ & \downarrow & \end{array} \qquad \Longrightarrow \qquad E \overset{(9)}{=} 0 \qquad \begin{array}{ccc} & \uparrow & \\ \downarrow & \downarrow & \uparrow \\ & \downarrow & \end{array} \qquad \Longrightarrow \qquad \Delta E = 0$$

$$E \overset{(9)}{=} 2J \qquad \begin{array}{ccc} & \downarrow & \\ \downarrow & \uparrow & \uparrow \\ & \downarrow & \end{array} \qquad \Longrightarrow \qquad E \overset{(9)}{=} -2J \qquad \begin{array}{ccc} & \downarrow & \\ \downarrow & \downarrow & \uparrow \\ & \downarrow & \end{array} \qquad \Longrightarrow \qquad \Delta E = -4J$$

$$E \overset{(9)}{=} 4J \qquad \begin{array}{ccc} & \downarrow & \\ \downarrow & \uparrow & \downarrow \\ & \downarrow & \end{array} \qquad \Longrightarrow \qquad E \overset{(9)}{=} -4J \qquad \begin{array}{ccc} & \downarrow & \\ \downarrow & \downarrow & \downarrow \\ & \downarrow & \end{array} \qquad \Longrightarrow \qquad \Delta E = -8J$$

The above, which is based on code from Hjorth-Jensen's webpage [2], means that we before we run the Monte Carlo cycles (so once for each temperature) can precalculate an array for $e^{-\beta \Delta E}$ consisting of only five elements (rememeber $\beta = 1/k_b T$). Actually, we only need to directly calculate two of the five values, $e^{-\beta 8J}$ and $e^{-\beta 4J}$, since the other values will be equal or larger than one, always giving acceptance probability of 1 according to (45). The need for evaluation of $e^{-\beta \Delta E}$ at every MC sampling is then avoided. ΔE is calculated applying (48d), and from this we can access the correct element of the pre-calculated array with the $e^{-\beta \Delta E}$'s. Not counting the few FLOPS it takes generating the 5 element long array with the $e^{-\beta \Delta E}$'s, we have saved $\left(N_{MCS} + N_{spins}\right)$ evaluations of $e^{-\beta \Delta E}$ by avoiding the calculation of $e^{-\beta \Delta E}$ inside the loops. As $N_{MCS} \geqslant 10^6$ this seemingly little trick soon becomes cost saving!

Also for the magnetic momentum there are savings with respect to FLOPS by choosing the method of flipping only one spin at the time and calculating ΔM directly. We get

$$\Delta M = M_j - M_i \tag{49a}$$

$$\overset{(2)}{=} \sum_{k=1}^{N} s_k^j - \sum_{k=1}^{N} s_k^i \tag{49b}$$

Method 1 (49b) would give $1 + 2 \cdot N$ FLOPS. This would be repeated $N_{Temperature} \times N_{MCS}$ times, giving $(1 + 2 \cdot N) \cdot (N_{Temperature} \times N_{MCS})$ FLOPS.

Lets see what (49b) gives for FLOPS, when we turn only one spin at the time. Assume the spin s_l is the spin that is turned. We get

$$\Delta M \overset{(49b)}{=} \sum_{k=1}^{N} s_k^j - \sum_{k=1}^{N} s_k^i \tag{50a}$$

$$= s_l^j - s_l^i \tag{50b}$$

$$\overset{s_l^j=-s_l^i}{=} -2s_l^i. \tag{50c}$$

(50c) involves 2 FLOPS. Compared to method 1, we get a difference of $\Big(2-(1+2\cdot N)\Big)\cdot(N_{Temperature}\times N_{MCS}) = (2N-1)\cdot(N_{Temperature}\times N_{MCS})$ FLOPS. As for the energy, there are considerable savings in FLOPS by choosing the method of one flip and calcuation of ΔM after each flip, compared to doing all the flipping prior to one calculation of ΔM.

3.3 FLOP count

Summing up the contents of this section including the FLOP analyzis, we get the following algorithm with FLOP counts for simulating the steady state of the Ising model

```
for temperature in temperatures                                             #Temperatures FLOPS
   Initialize variables E, M

   for cycles in MonteCarloCycles                                                #MC cylces FLOPS
      Sweep through the lattice at random locations                                   #Spins FLOPS
         Calculate delta E                                                                 6 FLOPS
         Feed delta E into Metropolis acceptance rule
            If accepted: flip spin, update statistical variables                           3 FLOPS
            Else: Do not flip spin, do not change statistical variables                    0 FLOPS


      Update averages (+ squares and |M|)                                               9 FLOPS
```

In the above algorithm we have not included the final calculation of averages, where one divides by the number of MC cycles. The reason for this is that we have implemented this step in another part of our program (in the write to file part.)

Looking at the above algorithm, we get the following FLOP count

$$\text{Total number of FLOPS} = N_{temperatures}\times N_{MCcycles}\times\Bigg(N_{spins}\times(6+p_{accept}\cdot 3)+9\Bigg) \tag{51a}$$

In (51a) p_{accept} equals the share of accepted moves. Using the fact that $0\leqslant p_{accept}\leqslant 1$, (51a) gives that the total number of FLOPS will be

$$N_{temperatures}\cdot N_{MCcycles}\cdot(N_{spins}\cdot 6+9)\leqslant \text{Total number of FLOPS} \overset{(51a)}{\leqslant} N_{temperatures}\cdot N_{MCcycles}\cdot(N_{spins}\cdot 9+9). \tag{52a}$$

From (52a) we see that the maximum difference in total FLOPS is $3\cdot N_{temperatures}\cdot N_{MCcycles}\cdot N_{spins}$, so the acceptance share has a considerable effect on the total number of FLOPS.

From (51a) we conclude that the algorithm is $\mathcal{O}(N_{temperatures}\cdot N_{MCcycles}\cdot N_{spins})$ in FLOPS.

4 Programming

4.1 Parallell computing

With the above FLOP count, (51a), and noting that $N_{MCcycles} \geqslant 10^6$ and that we will simulate lattices of size 100×100, we see that our problem will be very costly with regards to FLOPS (and hence computational time). In order to speed up the computations, we us parallell computing by applying the parallell computing library MPI.

By splitting up the total workload of the full program into subtasks and distributing these sub tasks among the CPU cores of the computer, MPI makes it possible to take advantage of the multiple cores by running the sub tasks on the different cores in parallell. Our results suggest a typical speed up follows $T_p = T_1/p$, where T_1 and T_p is time used for 1 and p cores respetively. Parallell computing is simple to implement in problems where there is a low degree of dependence between the different parts of the computational domain.

An example might clarify this. Say one wants to calculate an 1D integral of $y = f(x)$ using say the Trapeziodal rule of integration. The rule would require $(x_i, f(x_i))$ for all x_i-values. Each pair $(x_i, f(x_i))$ is independent on all the other pairs. The independence between $(x_i, f(x_i))$ pairs implies that we can integrate over the domain in any order we want, as long as we sum the contributions when all pairs are calculated. This means that we can divide the computatinal domain, the x-range, into parts, sending the parts to separate CPU cores, having each core computing the sum over the domain it is given, and finally adding together the results of the different cores.

In our Monte Carlo simulation there is independence over the MC cycle domain, because each MC cycle is independent on the previous MC cycle. This means that we can parallelize the MC computations. In MPI one has the flexibility to decide which parts of the program that should be parallellized. If one e.g. solves for different temperatures, one can choose to distribute the different temperatures among the cores. Then each core will solve different temperatures. Another alternative is to divide the Monte Carlo domain into subdomain, and let each core solve for a specific subdomain. The last alternative is the altnerative we apply.

Using MPI, we did not have to had many lines of code to our original unparallelized version of the program. The lines we added will be listed and commented uppon below. Where possible I have limited the commenting of the code to comments inside the code listings.

```
//  MPI initializations
MPI_Init (&argc, &argv);                 # Intialization of MPI
MPI_Comm_size (MPI_COMM_WORLD, &numprocs);    # how many processes participate in a given MPI computation.
MPI_Comm_rank (MPI_COMM_WORLD, &my_rank);   # rank of given process, equals number in [0,number of
    processes]
```

In the above code, the comments are based on Hjorth-Jensen [1] p. 134.

The next part of the code, listed below, splits our MC domain into parts, where each part is to be calculated on different processors in parallel.

```
/ MPI variables
int no_intervalls = mcs/numprocs;             # Division of full interval into subintervalls
int myloop_begin = my_rank*no_intervalls + 1;
int myloop_end = (my_rank+1)*no_intervalls;      # Larger than 1, since rank 0 is master
if ( (my_rank == numprocs-1) &&( myloop_end < mcs) ) myloop_end = mcs;
```

```
// Broadcast to all nodes common variables
MPI_Bcast (&n_spins, 1, MPI_INT, 0, MPI_COMM_WORLD); #Tranfer "n" data on one processor (our master
node) to be shared with all other processors
MPI_Bcast (&initial_temp, 1, MPI_DOUBLE, 0, MPI_COMM_WORLD);
MPI_Bcast (&final_temp, 1, MPI_DOUBLE, 0, MPI_COMM_WORLD);
MPI_Bcast (&temp_step, 1, MPI_DOUBLE, 0, MPI_COMM_WORLD);
```

The comment in the above code is taken from Hjorth-Jensen [1] p. 141, and the following comments builds on the same source. Basically the broadcast function makes sure that every process (if we are using 4 cores, we say we have 4 processes) has copies of the variable we have used in the command. The 2nd input to the broadcast-function, which is ”1” in all the lines above, is the number data sent, which in this case is ”1” since we are broadcasting scalars. The third input is the data type of the object that we wish to share. In our case this is a double or integer depending on the variable. The fourth input, which is ”0” in all our cases, specifies the process which has the original data. The ”0” means that it is the master process, being process 0, that contains the original data. The master node is the node where the results from the other nodes are added toghether to get the final result. The last argument ”MPI_COMM_WORLD” deals with somthing called communicators, which opens up for dividing the total number of tasks into socalled communicators where each comminucator contains processes that comunicates with each other, see Hjorth-Jensen [1] p 136 for more about this. ”MPI_COMM_WORLD” is the option that ensures that all MPI processes that are initiated are included in one common communicator, ”MPI_COMM_WORLD”.

Finally, the below code shows the important command ”MPI_Reduce”, which in our case basically tells the program to send the results from the processes to the master node when done.

```
for( int i =0; i < 6; i++){
MPI_Reduce(&average[i], &total_average[i], 1, MPI_DOUBLE, MPI_SUM, 0, MPI_COMM_WORLD);
}

  MPI_Finalize ();                # terminate the MPI computation and clean up
```

In the above ”$\&average[i]$” is the nodes where results are sent from, ”$\&total\_average[i]$” is the node where all the data is sent, ”1” is the number of variables, ”MPI_DOUBLE” says that the type of the variable is a double, and ”MPI_SUM” means that we sum the results from the different nodes. ”0” and ”MPI_COMM_WORLD” have the same functions as in ”MPI_Bcast” commented uppon above. These comments were based on Hjorth-Jensen [1] p. 141.

4.2 Classes

We solved all tasks by applying Python and C++. Everything is run from one python-program. In this program we set up all the scenarios, and create tables and figures from the computational results from C++.

We built the c++ program by starting with Hjorth-Jensen's program [5]. To make the program more easy to apply in the future, we re-wrote the program from Hjorth-Jensen, which is function based, to become a class based program.

There is one class ”isingmodel” and one main-file. The main-file mainly deals with the the issues related to the current problem in addtion to MPI-related commands. We read input in the main file, and loop over the temperatures. Except for this, the main file mainly contains calls to the class methods.

4.3 Random number generator

Since no random number generator (RNG) is truly random, it is very important to know the period of the random number generator when one applies MC calculations. The period is the number of values the RNG can produce before it starts repeating itself, hence when the RNG does not produce random numbers anymore.

The RNG in the lib-library has a period of 10^9. We are at the most running 10^6 MC cycles with a lattice of 100×100 size, which in total gives 10^{10} numbers to be drawn. Hence the random number generator in the lib-library has a too short period to be used in parts of our MC experiments.

In order to get a long enough period for our RNG generator, we apply the Mersenne Twister RNG, which according to to the slide ”Which RNG should I use?” in Hjorth-Jensen [6], has a period of 2^{19937}, which should suffice for our use.

5 Results

5.1 Scales

To make the equations easier implement, we apply $k_B = 1$ and $J = 1$ to get

$$\beta \overset{(5)}{=} \frac{1}{k_B T} \tag{53a}$$

$$= \frac{1}{T}, \tag{53b}$$

and

$$\Delta E \overset{(48d)}{=} 2Js_l^i \sum_{<k>} s_k \tag{54a}$$

$$= 2s_l^i \sum_{<k>} s_k. \tag{54b}$$

(53b) and (54b) gives the following expression in the Metropolis test

$$r \overset{(45)}{<} e^{-\beta \Delta E} \tag{55a}$$

$$r < e^{-\frac{2s_l^i \sum_{<k>} s_k}{T}}. \tag{55b}$$

The above implies that energy and temperature have the same units in our calculations.

5.2 2×2 lattice: Comparison with analytical results.

In this section we report our numerical results in the case of $T = 1$ for mean energy, mean absolute magnetization, the specific heat capacity and the suscpetibility for a 2×2 lattice with 4 spins. These results will be compared to our analytical results from the section "Anaytical results 2D Ising model with 4 spins".

The first table below shows our estimation results, while the second table shows the percentage difference between the numerical and analytical results.

$\log_{10} mcs$	$< E > /L^2$	$< \|M\| > /L^2$	C_V/L^2	χ/L^2
2	-2.000000	1.000000	0.000000	0.000000
3	-1.992000	0.997000	0.063744	0.009964
4	-1.995400	0.998450	0.036715	0.004690
5	-1.996440	0.998815	0.028429	0.003544
6	-1.995660	0.998564	0.034645	0.004265
7	-1.996033	0.998676	0.031673	0.003971
8	-1.995990	0.998661	0.032017	0.004020

Table 2: Numerical results 2×2 lattice.
The mean values quickly stabilizes, while the variables dealing with variances, C_V and χ, need more MC cycles to stabilize.

$\log_{10} mcs$	$< E > /L^2$	$< \lvert M \rvert > /L^2$	C_V/L^2	χ/L^2
2	0.201300	0.134106	-100.000000	-100.000000
3	-0.199505	-0.166296	98.688799	148.432987
4	-0.029163	-0.021102	14.441058	16.945765
5	0.022942	0.015447	-11.386410	-11.627691
6	-0.016137	-0.009636	7.986720	6.333443
7	0.002571	0.001519	-1.275680	-0.986165
8	0.000412	-0.000013	-0.203735	0.224759

Table 3: Percentage deviations from analytical results.
The deviation in the mean values is less than 1 % already for 10^3 MC cycles, while the variance related variables C_V/L^2 and χ/L^2, need respectively 10^8 and 10^7 MC cycles to have an error less than 1 %.

Firstly, the tables above show that the numerical results approaches the analytical results to a high degree of accuracy as the number of MC cycles increaes, where I define "high degree" to be aorund 1 % deviation between numerical and exact results.

Secondly we observe that the variables related to variance, C_V and χ, given by (23) and (27a) repsectively, need more MC cycles to converge to the analytical solution. We conclude that our numerical implementation seems correct.

5.3 20×20 lattice: Steady state.

In this section we study the number of MC cycles needed for convergence in the mean energy $< E >$ and mean absolute magnetic moment $< |M| >$ in the case of a 20×20 lattice. We are interested in the equilibrium state, since it is this state that lays behind the modelling in the Metropolis algorithm. It is in equilibrium that the Boltzmann-distribution should hold.

Further, we study the effect of temperature and the effect of type of starting spin configuration (random or fixed) on the the convergence properties of $< E >$ and $< |M| >$. The figures below show the results.

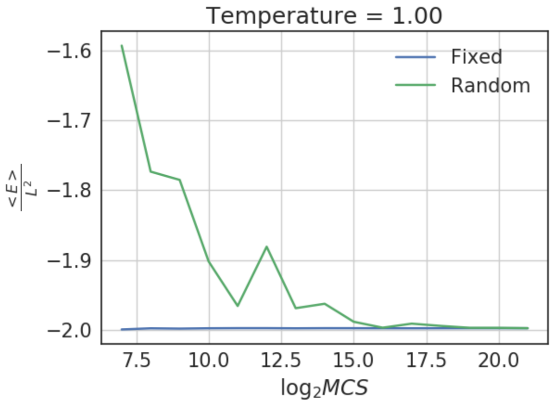

Figure 1: Expected Energy divided by L^2. $T = 1.0$.
The fixed configuration reaches equilibrium very fast, while the random configuration needs around 2^{16} MC cycles.

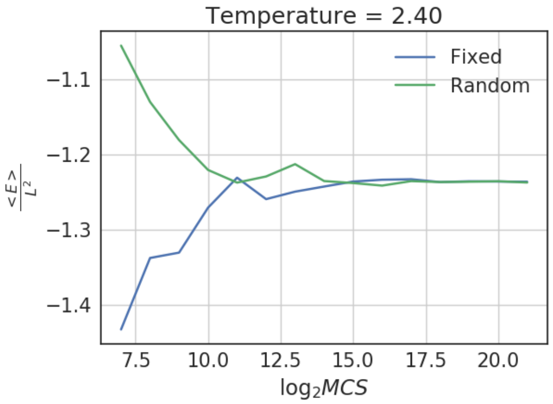

Figure 2: Expected Energy divided by L^2. $T = 2.4$.
For temperature closer to the cirtical temperature, more cycles are needed to reach equilibrium than for the low temperature case with fixed initial spin configuration. There seems to be little difference in equilibration time with respect to initial spin configuration.

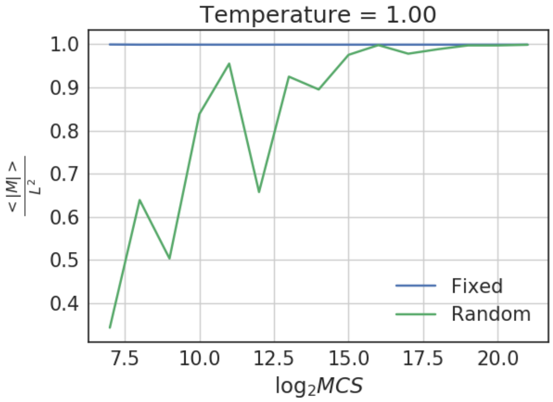

Figure 3: Expected absolute magnetic momentum divided by L^2. $T = 1.0$.
Equilibration times looks similar as for $< E >$ at $T = 1$. The random spin configuration starts out further from the equilibrium than the case for $< E >$.

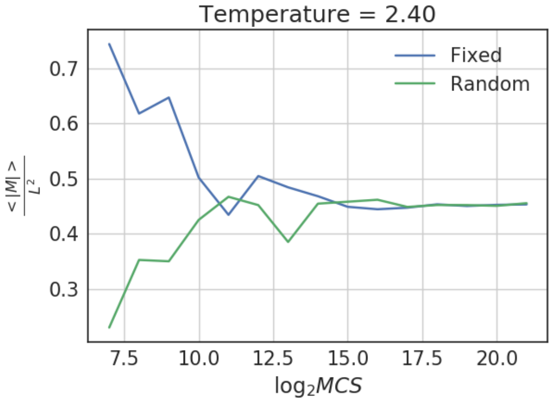

Figure 4: Expected absolute magnetic momentum divided by L^2. $T = 2.4$.
Equilibration times looks similar as for $< E >$ at $T = 2.4$.

Before we start interpreting the figures, a few words about temperature and equilibrium behavior. For low temperatures the spins will be in a more ordered configuration, pointing in the same direction, resulting in higher net magnetization and lower expected energy. For higher temperatures the spin ordering will be less fixed, reducing the net magnetization and increasing the expected energy. With higher temperatures more states become likely according to the Bolzmann-distribution, so that the variance in the estimated averages should be higher with higher temperature.

For the types of initial spin configurations, we start by discussing the low temperature case. In the fixed spin configuration all spins point in the same direction, yielding the lowest possible energy of the system, see (7). When the temperature is low, equilibrium would consist of an ordered spin configuration, explaining why the fixed initial spin configuration gives lowest equilibration time, since we are starting out close to equilibrium.

To see from our implemented Metropolis algorithm why the energy and magnetization stabilizes so fast for fixed initial spin configuration with all spins in the same direction in the low temperature case in our algorithm, we study the array used for acceptance probability, (45), in the Metropolis algorithm. The mentioned array, which has elements of form $e^{-\beta \Delta E}$, will for the low temperature have $\Delta E = 8$, see the section "Energy and momentum calculations in Metropolis". Hence, in the Metropolis test, a random number will be compared against $e^{-1 \cdot 8} \approx 0.0003$ (where we have used our scaling $\beta = 1/k_B T = 1/1 = 1$). We see that the random number in the Metrpopolis test is compared against a very low number, and the probability of getting a random number less than 0.0003 is per definition 0.0003. Hence the number of accepted moves will be very low in the case of $T = 1$, reflecting the principle of energy minization in equilibrium, and a fixed initial spin configuration equal to the ground state. The low acceptance rate also explains the fast convergence in absolute magnetization.

If a random initial spin configuration is chosen for the low temperature, we see that both energy and magnetization now needs typically 2^{18} MC cycles to fully converge. When I say "fully converge", I mean that the osciallitions seems to have died out. Others call the equilibration time the time where only the largest oscillations have died out, so in that case we could say that the equilibration time is around 2^{15} MC cycles for the low temperature with random initial spin configuration. Whichever requirement one chooses for defining the equilibration time, the increase in equilibration time going from fixed to random initial spin configuration in the low temperature case is drastic.

The reason why a random initial spin configuration takes longer time to equilibrate in the low temperature case, is that for low temperatures the equilibrium state is an ordered state resulting in low energy. With a random initial spin configuration, both the energy and magnetization start further away from equilibrium. Hence more time is needed to reach equilibrium. This shows how much utlization of knowledge about the equilibrium state can play for the computational times!

Now we move to the high temperature cases. The Boltzman PDF gives that a larger number of states is probable when the temperature increases. With temperatures closer to the critical temperature, equilibrium would consist of a disordered spin configuration. Hence a fixed initial spin configuration, giving the ground state, would start the simulations far from equilibrium, in the same way a random initial spin configuration started the simulations far from equilibrium in the low temperature case. On the other hand, a random initial spin configuration would probably start the simulations closer to the equilibrium state in the high temperature case.

What is a little suprising, is that the equailibration time for the high temperatures is about the same for the random initial spin configuration as for the fixed initial spin configuration with all spins in the same direction. Based on the discussion in the prevuous paragraph, we expcted the equilibration time to be lower in the random case compared to the fixed case, since the random case, most likely having a disordered configuration, would start of closer to equilibrium.

The next figure shows the percentage share of accepted moves in the four different scenarios.

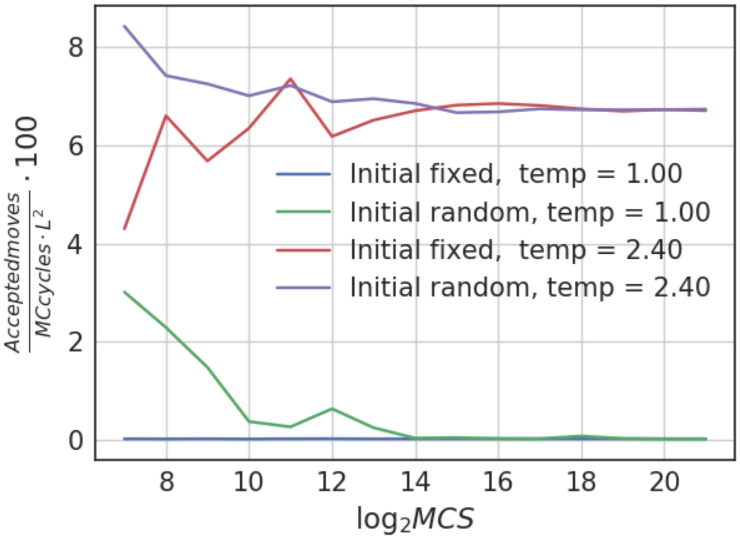

Figure 5: Percentage share of accepted moves in the Metropolis algorithm.
The share of accepted moves is considerably higher in the high temperature case than in the low temperature case. All shares stabilizes, and are stable aorund 2^{18} *MC cycles. For* $T = 1.0$ *the share is always close to 0 for fixed initial spin configuration. For the high temperature the random configuration mainly has the largest acceptance share.*

As expected, from the discussion above the figure, the acceptance move probability at the lowest temperature is close to zero in the case where the initial spin configuration is fixed to give minimum energy. We are already close to equilibrium, and the probability of accepating a move at a low temperature equilibrium state is, as discussed, very low.

The acceptance share for the low temperature starts out higher for the random case, since the probability of accepting a move in Metropolis is higher in disordered states. Mathematically, higher energies than the minimum gives $\Delta E < 8$, which gives us a higher number to compare the random number with in the Metropolis algorithm, increasing the acceptance probability. As the energy approaches equilibrium, the share of accepted moves shrinks toward zero.

When equilibrium is reached in the low temperature with a random initial spin configuration, the share of accepted moves remains close to zero when equilibrium is reached, since the probability of accepting a move to a higher energy state is very low.

For the high temperature cases, we see that the acceptance share is considerable higher compared to the low temperature cases. The reason for this is that at higher temperature, the energy will be larger in equilibrium. With higher energy in the Metropolis algorithm, the number that the random number is to be compared against will be larger, giving a higher acceptance probability, since $e^{-\Delta E/T}$ will be larger with higher temperature. This results in a higher accepted moves share in equilibrium. Physically the high temperature state is more agitated, and the Boltzmann PDF gives that more states are probable when the temperature increaes.

The fact that the acceptance share for random initial configuration is higher compared to the fixed initial spin configuration for high temperatures, is in line with our expectations. When starting out at the lowest energy, as in the fixed case, there will only be suggestions of moving up in energy in the Metropolis algorithm, and this is always less likely than moving down in energy.

In the above we have seen that there can be large differences between the initial estimates of statistical quantitites from the early MC cycles and the equilibrium values. We also noted that large differences between the initial MC cycles and the results from the equilibrium ranges contributed to larger variance in the estimated quentitites. This increase in variance can be omitted by ignoring the cycles before equilibrium is reached. A used rule is to ignore the first 10 per cent of the MC cycles when calculating the statistics.

5.4 20×20 lattice. Probability distribution.

The below tables show simulated values of mean energy, mean magnetization and their variances in our four cases for the 20×20 lattice when the steady states are reached.

As a control, our calculated measures of the mean energy and its variance, $< E >$ and $\frac{<E^2>-<E>^2}{L^2}$, are compared to direct calculations of the same quantities from the array with all the energies.

$\log_2 MCs$	T	μ_E/L^2	$< E > /L^2$	$(\frac{\mu_E/L^2}{<E>/L^2} - 1) \cdot 100$	σ_E^2/L^2	$\frac{<E^2>-<E>^2}{L^2}$	$(\frac{\sigma_E/L^2}{1/L^2(<E^2>-<E>^2)} - 1) \cdot 100$
20.0	1.0	-1.997172	-1.997172	-8.152091e-07	0.023281	0.023281	1.080300e-06
20.0	2.4	-1.236940	-1.236940	3.699787e-06	8.185407	8.185407	-4.893184e-07

Table 4: Statistics. Fixed initial config.
The mean energy is lower for the lowest temperature, and the variance is considerably higher for the highest temperature. How the mean and variance are computed does not matter.

$\log_2 MCs$	T	μ_E/L^2	$< E > /L^2$	$(\frac{\mu_E/L^2}{<E>/L^2} - 1) \cdot 100$	σ_E^2/L^2	$\frac{<E^2>-<E>^2}{L^2}$	$(\frac{\sigma_E/L^2}{1/L^2(<E^2>-<E>^2)} - 1) \cdot 100$
20.0	1.0	-1.997043	-1.997043	4.680687e-07	0.039074	0.039074	-3.837571e-07
20.0	2.4	-1.236217	-1.236217	2.899155e-06	8.114528	8.114527	3.297618e-07

Table 5: Statistics. Random initial config.
The results mimics the results in the fixed initial spin configurations in the table above.

As expected the mean energies grow with temperature. That also the variance grows with temperature is not a big suprise, given our discussion about the share of accepted moves in the previous section. Higher temperatures increases the number of accepted moves, and this gives more fluctuations in energy. Physically higher temperature gives higher energy and makes more states possible, since the system is more "agitated".

For the low temperature we observe that the variance is slightly higher in the random initial spin configuration case than in the fixed configuration case. This reflects the difference in energies between random and fixed initial spin configurations that we observed for the initial MC simulations: figure 1 shows that there in the beginning of the MC simulations are more variation in the energy estimates in the random case compared to the fixed case. Even though we are considering steady states, the numbers from the full period is part of the estimated means and variances for the steady state, so that differences in earlier parts play a role and shows up in the variances. We note that the noise from the earlier MC cycles could have been avoided by excluding the first MC cycles from the calculation of the statistical variables.

The below figure shows the probability distributions for the four different cases.

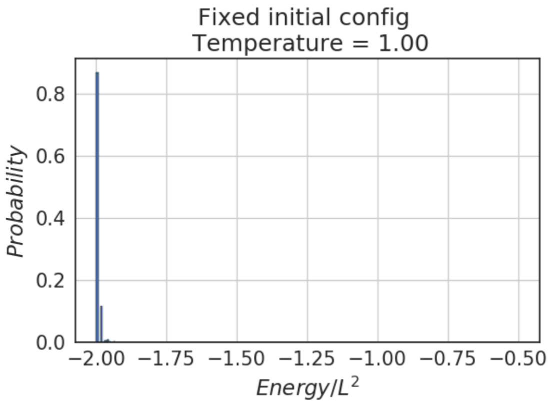

Figure 6: Probability distribution. Fixed intital spin configuration. $T = 1$.
The probability distribution is very skewed to the left and very narrow, centered around the expected value.

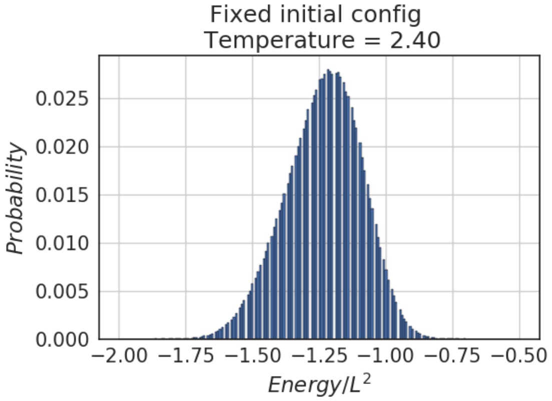

Figure 7: Probability distribution. Fixed intital spin configuration.$T = 2.4$.
With higher temperature several more states become likely, and we get a fatter distribution centered around the expectation value.

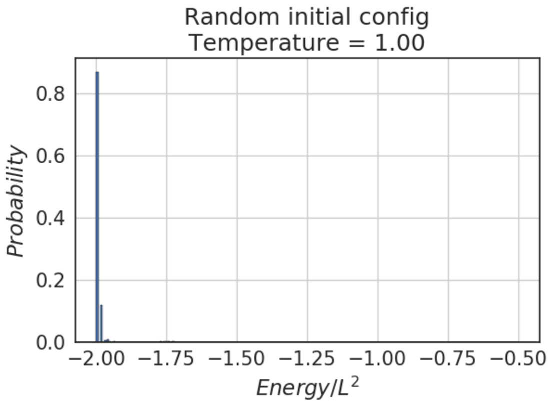

Figure 8: Probability distribution. Random intital $T = 1$.
The distribution looks very much the same as for the fixed case.

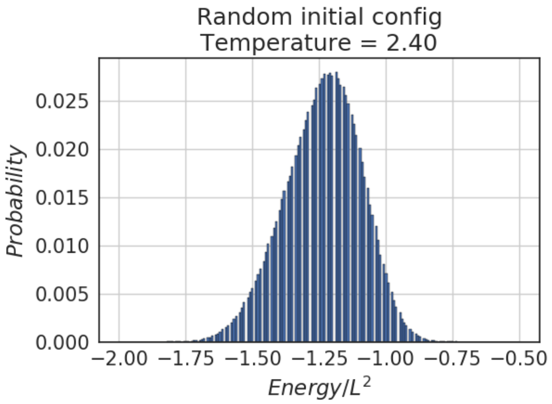

Figure 9: Probability distribution. Random intital $T = 2.4$.
The distribution looks very similar to the fixed case with same temperature.

Based on figures 1, 3 and 5, we expected the lowest energy to be the dominant energy in the histogram for the lowest temperature, and that is what we get. We see from figures 6 and 8 that the variation among enery states is very low, and that the occurences are centered around the expected values.

Looking at the figures above for the low temperature, figures 6 and 8, it might seem surprising that the histograms look as similar as they do for fixed and random inititial condition. The reason the histogram for the random initial spin configuration does noe contain more bins, as one might expect based on the above mentioned figures, is that all the higher energies in the first MC cycles are obtained very few times. Hence the bins for these higher energies are very low. From the two tables above we also note that the variances are very similar for the two types of initial spin configuration, suppoerting the similarity of the histograms.

For high temperatures, several more energy states are visited. This is as expected based on the earlier results for e.g. the acceptance share, since the acceptance probability was larger for higher temperatures. The probability distribution becomes much wider compared to the low temperature case, and is centered around the expected value.

As for the low temperature case, the similarity between the type of initial spin configuration, seen from figures 7 and 9, is supporteded by previous figures, see figures 2 and 4. Figures 2 and 4 show that the behaviour towards equilibrium looks of similar kind with respect to variations and that the equaulibrium values are the same. Further, the tables 4 and 5 show that both the means and the variances are very similar for the initial spin configuration type, supporting the equality of distribution shown in figures 7 and 9.

5.5 Timing of parallelization

When we are going to estimate the critical temperature, the number of simulations will be large. In this case parallel computing will save us a lot of CPU time. The following shows timing results for 1, 2, 3 and 4 processors.

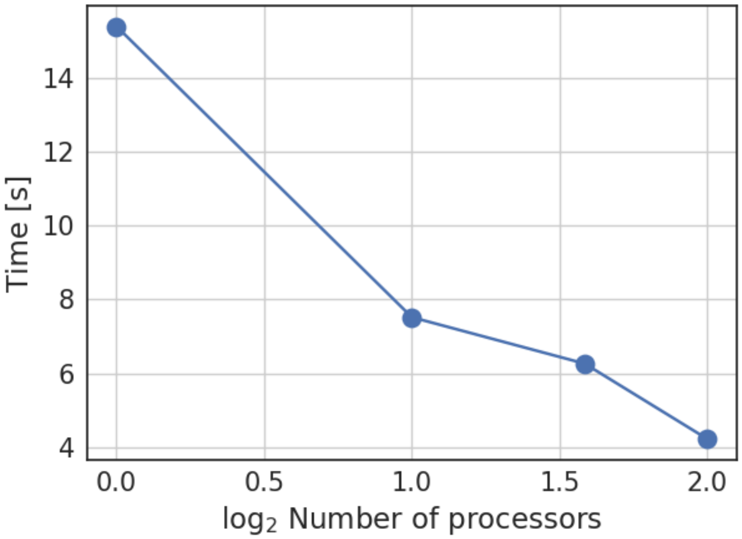

Figure 10: Timings of the main algorithm. $T = 1$, $mcs = 2^{20}$, lattice $= 20 \times 20$. Number of processors $= 1, 2, 3, 4$. *The time seems to be $\mathcal{O}(\#$ processors$)$.*

From the above figure we see that the computational time goes like $\mathcal{O}(1/\#$ processors$)$. By using 4 processors, like we do, we reduce the time of the main algorithm to a fourth of the time when not applying parallell computing running on only 1 processor.

5.6 Critial temperature

In this section we will study behavior around the critical temperature, and applying the estimator we derived for the crititcal temperature, test how well our numerical results matches Onsager's analytical result for the critical temperature.

When we start the simulations for a new temperature, we apply the equilibrium spin configuration from the previous temperature as initial spin configuration. Since the temperature changes are small, the change in spin configurations between the different temperatures should be small. Hence, starting out with the equilibrium spin configuration from the the previous temperature as initial spin configuration for a new temperature, should reduce the equilibration time.

The below figures show the results for mean energy, mean absolute magnetization, specific heat capacity and susceptibility for different lattice lengths as functions of temperature.

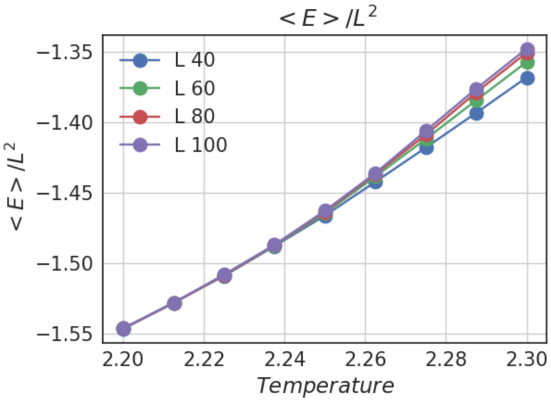

Figure 11: Expected value energy.
All lattices starts out at the same energy level, and starts to diverge close to the anaytical critial temperature, larger lattice sizes having more energy. The energy seems to converge with lattice size.

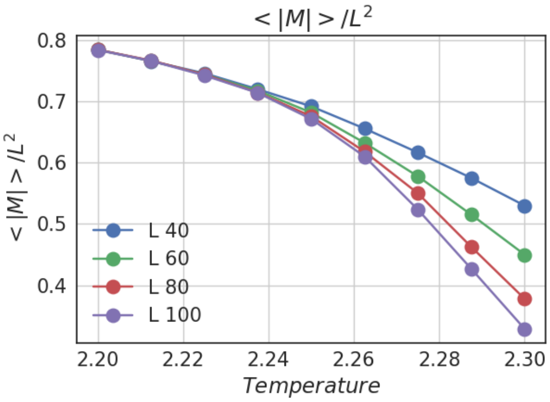

Figure 12: Expected value magnetic moment.
As for the energy, the absolute magnetic moment starts out being the same for all lattice sizes before the moments starts to diverge when the temperature approaches the analytical critical temperature. The differences in moments are larger for the lowest lattice sizes.

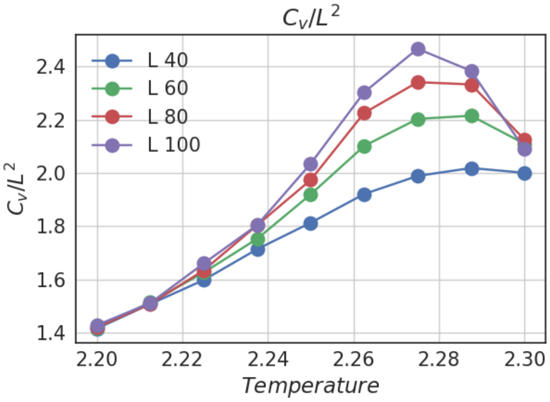

Figure 13: Specific heat capacity.
The heat capacity starts to diverge between the lattice sizes at lower temperatures than the energy and moment. The temperature where maximum C_V is reached converges with increased lattice size. The peak becomes sharper with lattice size.

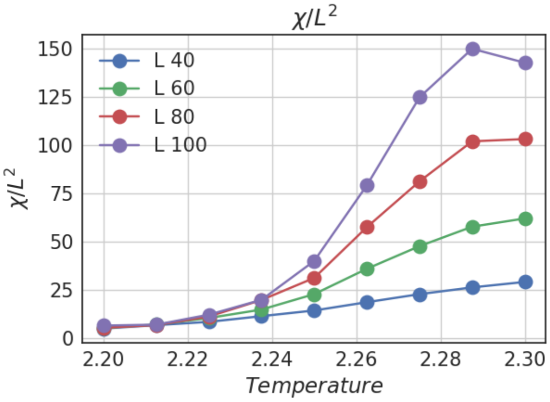

Figure 14: Susceptibility.
Divergence between the lattice sizes starts around the same temperautre as for C_V. Maximum χ does not stabilize with lattice size. The temperature where maximum χ is reached seems to decrease with lattice size. The peak in χ becomes sharper with lattice size.

The figure with energy above shows that there is a gradual increase in energy when the the critical temperature, $T_C = 2.269$ is approached. Also, the energy around the critical temperature seems to converge with lattice size .

The magnetization experiences a drop when the the critical temperature is approached, and the drop grows with lattice size. The difference between the lattice sizes seems to become smaller with lattice size. The figures suggest that the absolute magnetization will approach zero for larger temperatures than displayed. This suggests that there is a phase transition aorund the critical temperature. We are going from states where many spins points in the same direction, creating magnetic moments, to unordered states where the direction of the spins are random, cancelling each other out, and hence reducing the magnetization.

Throughout this report we are using the absolute value of magnetization, and not magnetization itself. The reason for this is that we want to avoid oscillations in magnetization close to the critical temperature. Close to the critical temperature the net spin might by slightly positive or slightly negative, reflecting the unordered state at this temperature. A small change in spins, when we are close to T_C, would result in flucations between positive and negative magnetization, see Hjorth-Jensen [1] p. 444. We remove this oscillating behavior by displaying the absolute magnetization.

From theory we know that the specific heat capacity is supposed to diverge for $T_C(L = \infty) = 2.269$. From the figure we see that the peak in C_V increases with lattice size when the critical temperature is approached, indicating that a phase transition is taking place.

The below table shows the results for estimation of $T_C(L = \infty)$, when we have applied the estimator (17a) derived in the section "Phase transitions".

Spin combos	$T_c^{Estimate}(L = \infty)$	$(\frac{T_c^{Estimate}(L=\infty)}{T_{c,exact}} - 1) \cdot 100$
$[40, 60]$	2.2875	0.80710401536
$[40, 60, 80]$	2.27916666667	0.439865020769
$[40, 60, 80, 100]$	2.27708333333	0.348055272121

Table 6: Estimated T_C.
All estimates are below 1 % in difference from the anaytical value. The estimations seems to converge with lattice size.

The above table shows that we get a good approximation of $T_C(L = \infty)$ applying our results. This supports the quality of our numerical implementation.

6 Conclusions

In this report we have applied Monte Carlo simulations and the Metroplois algorithm to study phase transitions using the two-dimensional Ising model. The calculations are done using classes in c++, while tables and plots are generated in Python.

Our implementation of the Metroplis algorithm is motivated by a theoretical derivation linking our problem to Markov chains. The main theoretical aspects of the Ising model are also shown.

Expressions that increases the eficiency of the implemented Metropolis algorithm considerably are derived. We do a FLOP count that show these expressions are very cost saving computationally.

For the 2×2 lattice with 4 spins we derive analytical results for energy, magnetization, specific heat capacity and suspecptibility. We simulate the same case for $T = 1$ and compare against the corresponding analytical results. We find that we need few MC cycles to approach energy and magnetization below 1% in difference from the theoretical results, while we need some more cycles to get the specific heat capacity and the susceptibily to the same level of precision.

For a 20×20 lattice we study the equilibration time for two different temperatures (one low and one close to the critical temperature) and two different initial spin configurations (one random and one fixed giving ground state). We find that the fixed initial spin configuration gives a very low equilibration time for the low temperature case, while all other combinations of temperature and initial spin configuration give equilibration times around 2^{18} MC cycles.

The share of accepted moves is shown to be lowest in the low temperatures cases, always being close to zero for the fixed initial spin configuration and approaching zero at equilibrium for the random initial spin configuration. For the high temperature, the share is considerable higher, around 7 per cent, and stabilizes when equilibrium is reached.

There is not large differences in the probability distriubution for energy and its variance between the different initial spin configuration types (random and fixed). The probability distributions in the low temperature and the high temperature case, on the other hand, differens significantly. The lowest energy state occuies 90 per cent of the time in the lower temperature cases, so that the distribution is very skewed to the left. In the high temperature cases, the probability distribution becomes much wider. These differences are reflected in the variance estimates, which are considerably higher in the high temperature cases. We do a small test of the calculated expected values and variances for energy and magnetization from our c++ program by calculating these once again in python based on the data of all saved energies. The python results and c++ results are very similar.

We derive an estimator for the critical temperature $T_C(L = \infty)$ which is implemented. The results for lattice sizes from 40 × 40 and 60 × 60 up to 100 × 100 lattices all deviates less than one percentage from the analytical result found by Onsager.

Parallel computing with MPI is used. The most significant commands are commented uppon and explained. The computational time seems to be $\mathcal{O}(1/\#$ processors$)$, which makes a hige difference in the simulations of the critical temperature especially.

For the future, we note that noise in the calculations of the statistical variables could have been reduced by ignoring the first MC cycles in the calculations of the statistical averages.

7 Feedback

7.1 Project 1

This project has been extremely educational. We learned about about c++, especially pointers and dynamic memory allocoation. Also which for us was a well forgotten subject, we learned about dangerous of numerical round-off errors.

We feel the size of the project is large, much larger than typical assignments in other courses. However, the quality and quantity of the teaching without a doubt made the workload managable. The detailed lectures, combined with the fast and good respones on Piazza helped a lot!

We think the project could have gone even smoother, if we on the 2nd lab-session had learned basic branching in Github. We used a considerable amount of time finding out of this.

All in all, two thumbs up!

7.2 Project 2

- catch: We ended up using a lot of time making this work properly. Still we have some problems with catch and Qt. We think we might had benefited from a demonstration at the lab.
- We were not able to understand the revised Sturm-Bisection algorithm from Barth et al.'s [**?**] paper on the revised Sturm-Bisection.
- Apart from the small details above, we are very happy about this project. How would have thought linear algebra could be fun?!

7.3 Project 3

- Classes: Very useful!
- Extra fun working with a system that one "knows" a little bit about. Also easier to judge the quality of the results, and find potential errors.

7.4 Project 4

- Plain fun. Nothing more to add.

8 Bibliography

[1] Hjorth-Jensen, M.(2015) Computational physics. Lectures fall 2015. https://github.com/CompPhysics/ComputationalPhysics/tree/master/doc/Lectures

[2] https://github.com/CompPhysics/ComputationalPhysics/blob/master/doc/pub/statphys/ipynb/statphys.ipynb

[3] Onsager, L.(1944) "Crystal statistics. I. A two-dimensional model with an order-disorder transition". *Phys. Rev.*, Series II, 65: 117–149.

[4] https://github.com/CompPhysics/ComputationalPhysics/blob/master/doc/Projects/2017/Project4/pdf/Project4.pdf

[5] https://github.com/CompPhysics/ComputationalPhysics/blob/master/doc/Programs/LecturePrograms/programs/StatPhys/cpp/ising_2dim.cpp

[6] http://compphysics.github.io/ComputationalPhysics/doc/pub/mcint/html/mcint.html