Simulating the Ising Model

Max Heimann 580560, Lyding Brumm 604522, Burak Varol 623941

January 2024

1 Introduction

In this report, we present our code which simulates the Ising model on a D dimensional lattice. The main challenge of the problem was the exponential scaling of the number of configurations: There are two spin values (-1 and 1) associated to each lattice point. For N lattice points, we have 2^N total configurations, which gives $\approx 10^{30}$ for 10×10 lattice.

The observation that not every configuration is equally important leads us to just include the configurations which gives the biggest contribution. This is done by the Metropolis Algorithm. We began the Markov Chain with a random initial configuration and each iteration in the chain refers to a Metropolis Iteration. Since the algorithm is probabilistic in nature, we use number of different Markov Chains (replicas) to calculate the statistical error. In the following, a list of modules can be found

Module Name	Summary
grid.c/grid_test.c	The grid which represents the Ising Model,grid related functions and their test
metropolis.c/metro_test.c	The Metropolis Algorithm and test
statistics.c/statistics_test.c	Statistical analysis functions calculating the mean values and errors; their tests
imdf.c/imdf_test.c	Data format which extracts the data from C and saves it in an analysable way and its test
main.c /test.c	The main code and its test
delta_h_test	the test of the delta h function

Table 1: Summary of modules

2 Problem

We have to calculate:

$$\langle M(s) \rangle = \frac{\sum_s M(s) e^{\frac{H(s)}{k_b T}}}{\sum_s e^{\frac{H(s)}{k_b T}}} \tag{1}$$

with

$$\frac{H(s)}{k_b T} = -\beta \sum_n \sum_k s(n) s(n + e_k) + b \sum_n s(n) \tag{2}$$

where $\beta = \frac{J}{k_b T}, b = \frac{\mu B}{k_b T}$.
This is however very hard to calculate and would take an insane amount of time. With the Markov chain,

$$s_1 \to s_2 \to s_3 ... \to s_{N_{thermal}} \to ... \to s_{N_{config}} \tag{3}$$

The observable for one replica is calculated as:

$$\langle M \rangle = \frac{1}{N_{config} - N_{thermal}} \sum_{k=N_{thermal}+1}^{N_{config}} M(s(k)) \pm \Delta M(s(N_{config})) \tag{4}$$

The observable over all replicas can be calculated as (R = number of replicas):

$$\bar{\bar{M}} = \frac{1}{R} \sum_{r=1}^{R} \bar{M}^{(r)} \tag{5}$$

The error is

$$err(\bar{\bar{M}}) = \sqrt{\frac{1}{R(R-1)} \sum_{r=1}^{R} (\bar{M}^{(r)} - \bar{\bar{M}})^2)} \tag{6}$$

3 Experiments

We created 500 replicas and used the values $D = 2$, $N = 100$, $\beta = 0.43$, $b = 0.01$, $N_{config} = 1000$. For each replica, we use a different seed (from 1 to 501) and luxury number 2 to initialize the pseudo random number generator RANLUX in order to get statistically independent replicas. To find the $N_{thermal}$ value, we inspect the energy (1) and magnetization (2) graphs, look for the iteration at which the graph starts to oscillate and eyeball $N_{thermal}$ as $N_{thermal} = 200$.

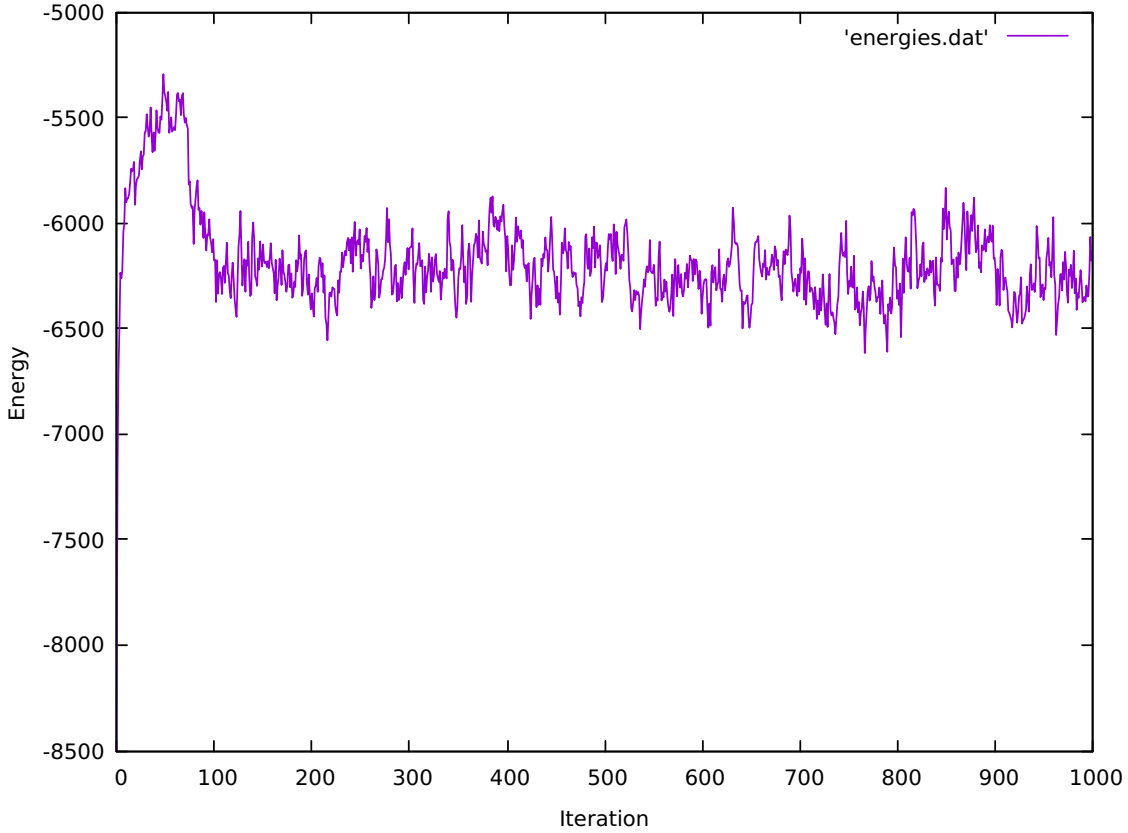

Figure 1: The Energy Thermalisation Graph

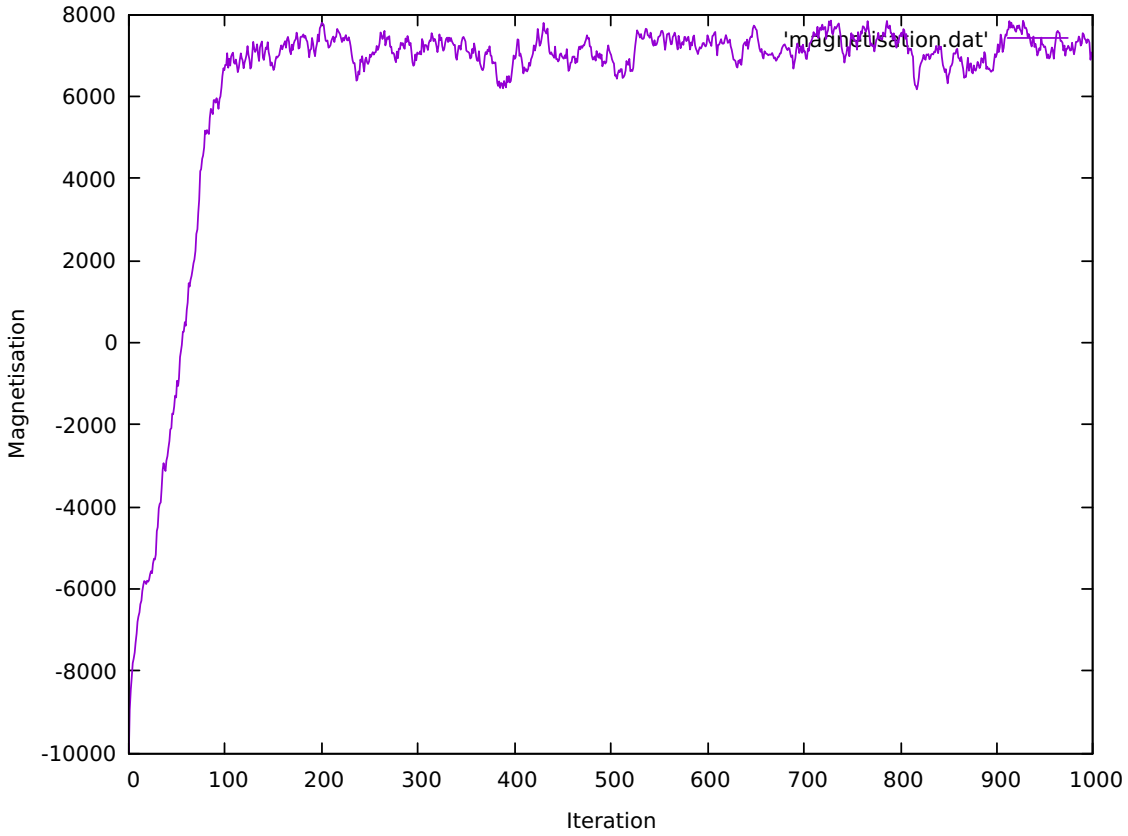

Figure 2: The Magnetization Thermalisation Graph

With these values in mind, we find the energy and magnetization as the following (using (5) and (6)):

$$E = (-6237.077049 \pm 0.628868)k_B T \tag{7}$$

$$M = 7184.701846 \pm 2.171860 \tag{8}$$

3.1 Large R and Large N_{config} limit

- If we send the number of replicas to infinity, the statistical error must go to zero. From the definition of standard deviation and the formula (6), we can write:

$$err = \frac{\sigma}{\sqrt{R-1}} \tag{9}$$

  For $R \to \infty$, we obviously expect the statistical error to vanish. This was experimentally verified see figure 5 and 6

- The Central Limit theorem says that, the sampling distribution of the mean will be always normal distributed with the $\sigma_{mean} = \frac{\sigma}{N}$ if the sampling size is large enough. We can apply the theorem to our case since the sampling size will be large, the samples are independent and identically distributed, and we have a finite variance. In our case, $N = N_{config}$ and the σ_{mean} is (roughly) the statistical error. We see that $\sigma_{mean} \to 0$ if N goes to ∞, meaning the statistical error will vanish for large enough N_{config}. This was experimentally verified see figure 3 and 4.

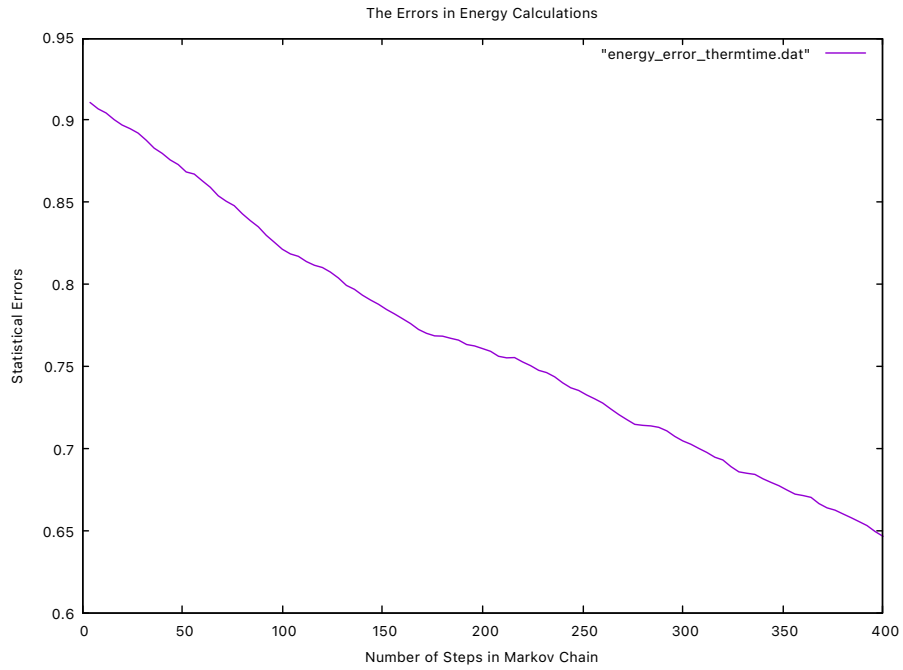

Figure 3: Energy Uncertainty as Different Lengths of Markov Chain (After Thermalisation)

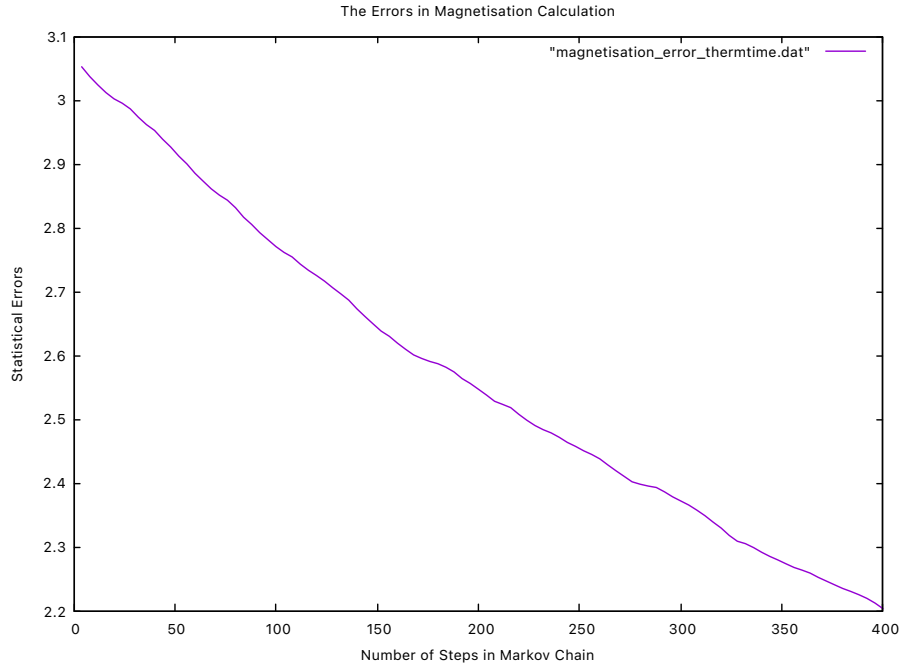

Figure 4: Magnetisation Uncertainty as Different Lengths of Markov Chain. (After Thermalisation)

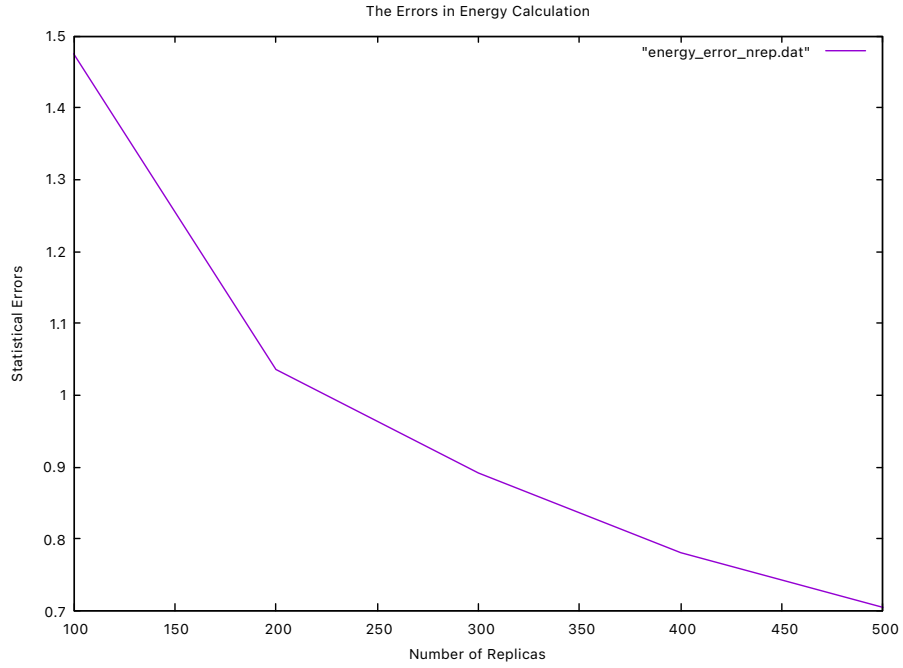

Figure 5: Energy Uncertainty as Different Number of Replicas

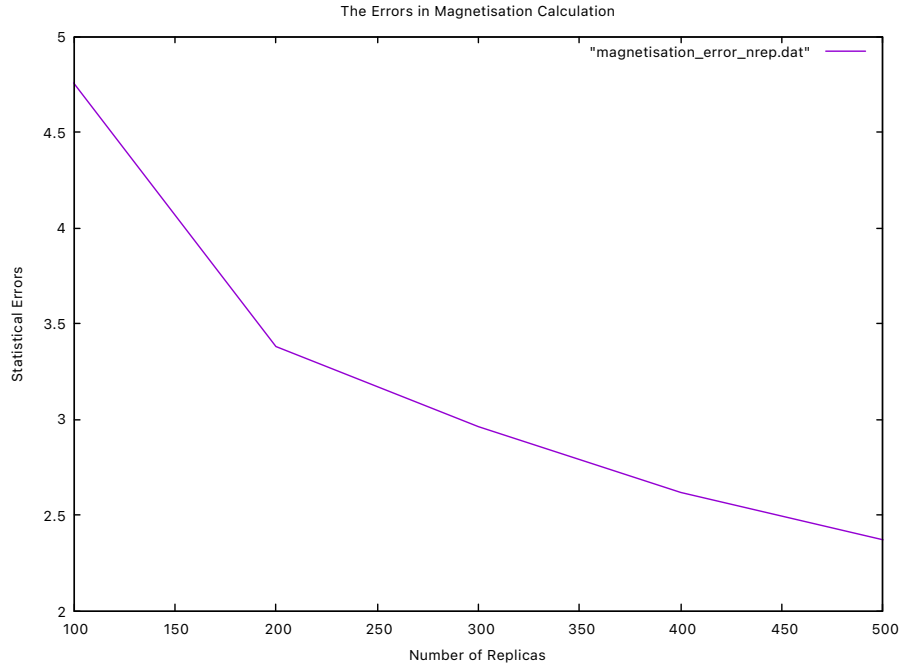

Figure 6: Magnetisation Uncertainty as Different Number of Replicas

We present our results for large R and large N_{config} in the following way: For the simulated data, we choose $N_{thermal}$ higher or consider less replica. By this method, we can demonstrate for increasing N and R the error decreases.

4 Probability distributions

To get an idea on how actual probability distribution looks like, we plotted the probability of the energy/magnetization is within one bin of a certain width. For the energy $\Delta E = 8k_BT$ and for the Magnetization $\Delta M = 30$. Plotting this against the normalized frequency we get the probability distribution. These are shown in 8 and 7.

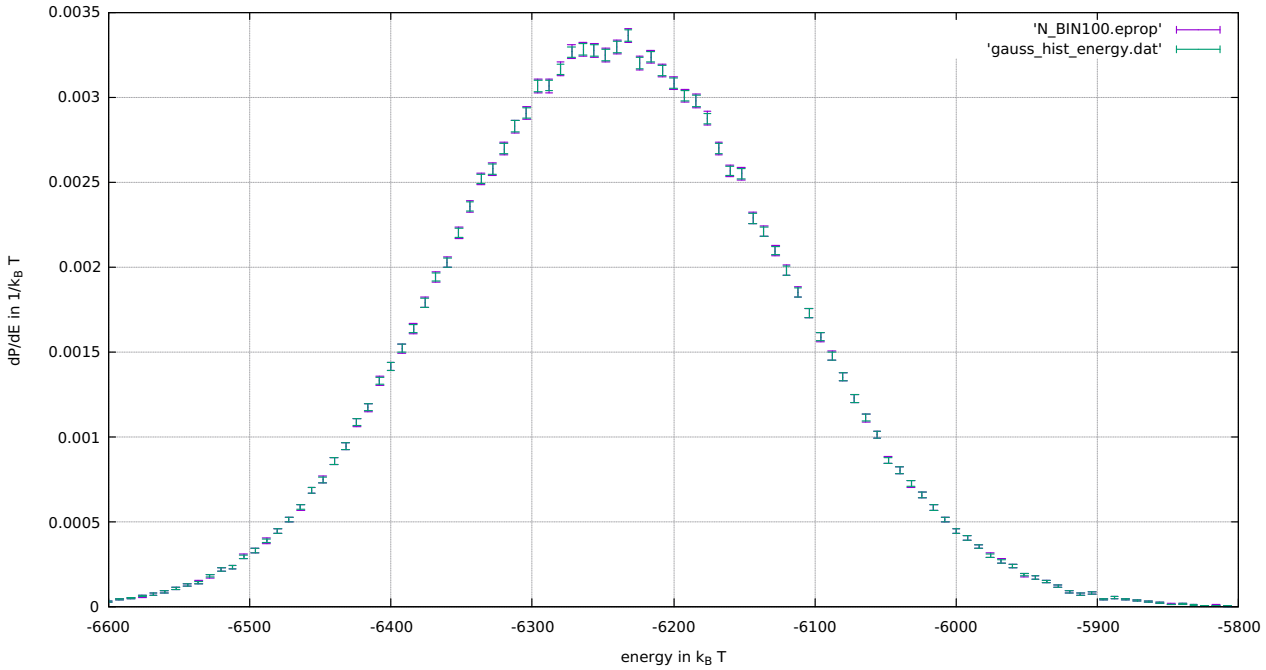

Figure 7: Probability distribution for the energy

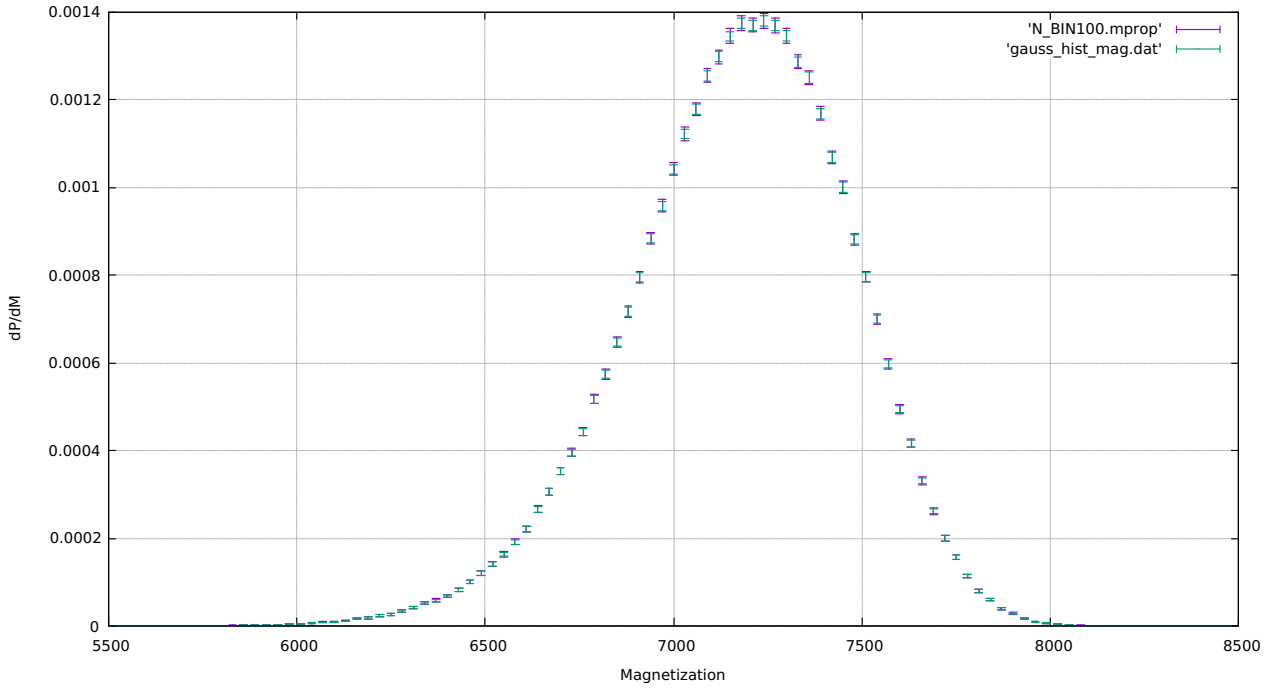

Figure 8: Probability distribution for the magnetization

4.1 Calculation of statistical uncertainty

In both cases the actual value for the PDF via

$$p_i = \frac{k_i}{N_{bins}} = P(E \in (X_i, X_i + \Delta X)) \tag{10}$$

$$\frac{\mathrm{d}P}{\mathrm{d}X}(X_i) \approx \frac{p_i}{\Delta X} \tag{11}$$

k here is the counts of energies within one bin. For the actual value we just average over all replicas. As is seen in 7 and 8 we calculated the error-bars in two different ways: first in the *"standard"* by just calculating the variance of p and divide it by the number of replicas and hence $err(X) = \frac{\sigma_X}{\sqrt{R}}$. these are the green bars in the plots. However this approach is flawed for a few reasons:

1. If we where to calculate the error for a bigger confidence interval than 1σ we would predict the existence of negative count-rates which is illogical
2. If for some i $k_i = 0$, then $err(p_i) = 0$. This is unreasonable. Just because we did not measure any points in this particular bin, does certainly not mean there is no finite possibility that the system cannot reach this value for the observable, however unlikely it might be

This is why we also model the error in a different way. For that we view each bin i as a Bernoulli Experiment: When we measure the energy for one configuration it has the probability p_i to be within this bin, where p_i is defined as before. Now for a Bernoulli experiment the confidence interval for a given confidence level $Z_{\alpha/2}$ ($= 1$ in our case) can be computed via various methods. We decided to use the Wilson-Score method which is easy to compute and does not have the issues mentioned in above. The Error interval is then given by (p_l, p_u) where

$$p_{u/l} = \frac{1}{1 + \frac{Z^2_{\alpha/2}}{n}} \left(p \; + \frac{Z^2_{\alpha/2}}{2n} \pm Z_{\alpha/2} \sqrt{\frac{p(1-p)}{n} + \frac{Z^2_{\alpha/2}}{4n^2}} \right) \tag{12}$$

Where in our case $n = (N_{config} - N_{thermal})R$.
However this method also does have flaws of its own as it assumes uncorrelated data which we do not have. This is why we will always underestimate the magnitude of the error.

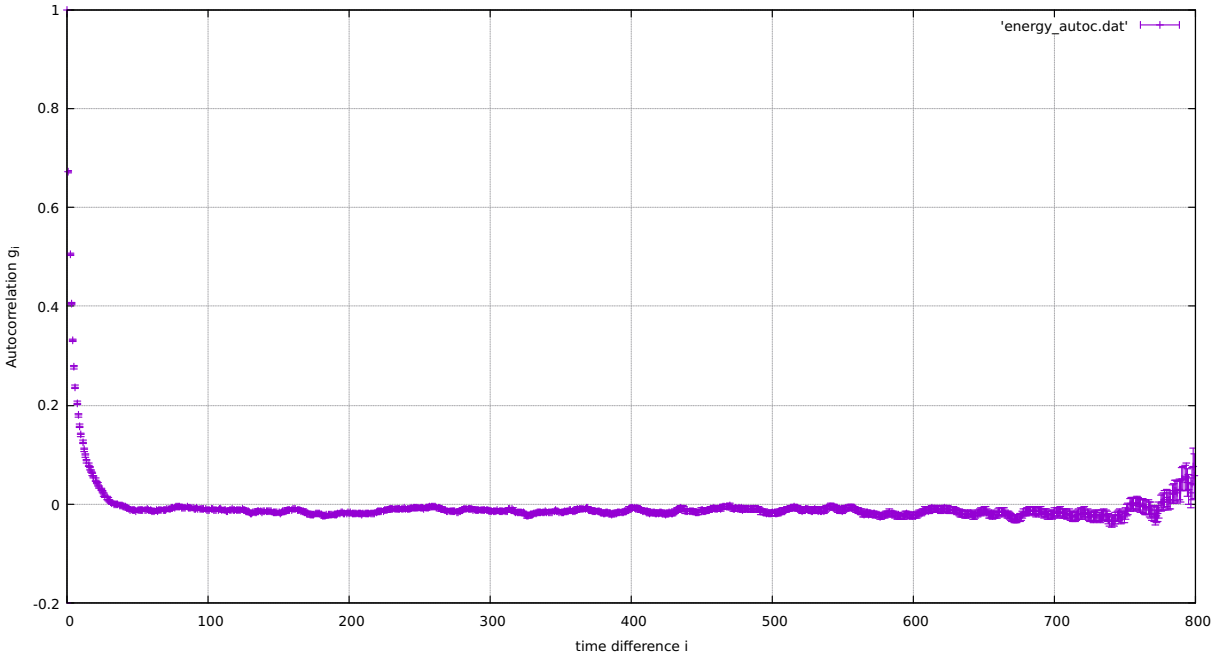

Figure 9: Autocorrelation of the energy

Now we would need to quantify by what amount we underestimate the error. For this we reduce the effective number of points to the number of independent points by dividing by the square root of the correlation time. So we get

$$n = \frac{(N_{config} - N_{thermal})R}{\sqrt{N_{corr}}} \tag{13}$$

The results are the purple plots in 7 & 8. We eyeballed the correlation time for the energy to be about 3 and the magnetization to be about 10. This gives good overlap with the errorbars calculated form the variances. Another approach would be to model the the actual count rate as Poisson distribution, which of course would yield the same flaws as before. What might be a more fitting model for the count rate is a superpoissonian distribution which may be reconstructed with the help of the replicas. This however would be a too elaborated method for the scope of this project.

5 Calculation of of $T(s|0)$

The goal of this task was to analytically calculate the transition probability $T(s|0)$ (s here is any configuration) for the set of parameters given in the script by two approaches: analytically and numerically. We represent each configuration as a number, where we read the configuration as a binary number and and the individual spins as bits. So $(\uparrow,\uparrow,\uparrow,\uparrow) = (0000)_b = 0$, $(\uparrow,\uparrow,\uparrow,\downarrow) = (0001)_b = 1$ and so on.[1] This is also very compatible with the way we implemented our configurations in the first place.

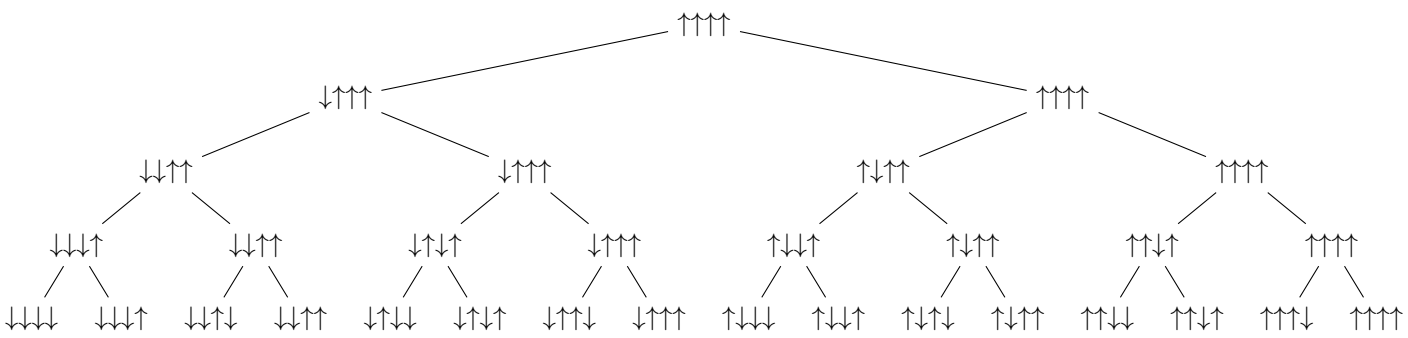

Figure 10: Decision Tree for Metropolis Algorithm

[1] This convention actually flips the sign of the b parameter, but does not change the outcome otherwise

5.1 Analytical approach

For the analytical approach we calculate the probability distribution by multiplying the chain of corresponding elementary updates.

$$T(s|0) = t_{v-1}(s|s^{v-2})...t_0(s^0|0) \tag{14}$$

Here s^i is the configuration where the spins up until the index i are flipped like in s. So for $s = (\downarrow, \downarrow, \uparrow, \downarrow)$, $s^2 = (\uparrow, \downarrow, \uparrow, \downarrow)$. $t(s^i|s^{i-1})$ is then given as in the script. Note that we start counting at 0.
To actually calculate $T(s|0)$ we initialize an array with length of the total number of possible configurations 2^v going from 0 to $2^v - 1$ as described above and fill it up with all 1. Then we iterate over all indices and calculate for each configuration $t(s^i|s^{i-1})$ and multiply the corresponding array value with it. To better understand the approach, we visualise the process in a table, where each row is an iteration step over the index and each column is the sequence of $t(s^i|s^{i-1})$ towards $T(s|0)$.

s	0	1	2	3	4	5	6	7
i = 0	$t_0(0\|0)$	$t_0(1\|0)$	$t_0(0\|0)$	$t_0(1\|0)$	$t_0(0\|0)$	$t_0(1\|0)$	$t_0(0\|0)$	$t_0(1\|0)$
i = 1	$t_1(0\|0)$	$t_1(1\|1)$	$t_1(2\|0)$	$t_1(3\|1)$	$t_1(0\|0)$	$t_1(1\|1)$	$t_1(2\|0)$	$t_1(3\|1)$
i = 2	$t_2(0\|0)$	$t_2(1\|1)$	$t_2(2\|2)$	$t_2(3\|3)$	$t_2(4\|0)$	$t_2(5\|1)$	$t_2(6\|2)$	$t_2(7\|3)$

Table 2: Example calculation for $v = 3$

Now taking the product over each column yields $T(s|0)$. Note that around half of the calculations we made in 2 are redundant and can be copied.

5.2 Numerical approach

For the numerical calculations we just use initialize a grid to be in the state 0, then run the metropolis algorithm one time, look what final configuration s and add +1 to counter variable k_s for this particular configuration. Then repeat the process a $N_{Iterations}$ times and calculate $p_s = \frac{k_s}{N_{Iterations}}$. As each of the outcomes is completely independent from each other one, we can again utilize the Wilson-Score method to calculate the errorbars as we did when calculating the errors for the pdf.

5.3 Comparison

We compared the analytical calculated result with the numerical approach. The result can be seen in 11:

s	$p_{analytical}$	$p_{numeric}$	$p_{numericlower}$	$p_{numericupper}$
0	6.2516e-01	6.2355e-01	6.2201e-01	6.2508e-01
1	3.2374e-03	3.4500e-03	3.2694e-03	3.6404e-03
2	1.5880e-02	1.5640e-02	1.5252e-02	1.6037e-02
3	2.9808e-03	3.0000e-03	2.8319e-03	3.1779e-03
4	1.5880e-02	1.5550e-02	1.5163e-02	1.5946e-02
5	0.0000e+00	0.0000e+00	0.0000e+00	9.9999e-06
6	1.4622e-02	1.4700e-02	1.4324e-02	1.5085e-02
7	0.0000e+00	0.0000e+00	0.0000e+00	9.9999e-06
8	7.7901e-02	7.8700e-02	7.7852e-02	7.9555e-02
9	1.4622e-02	1.4670e-02	1.4294e-02	1.5055e-02
10	1.9789e-03	1.9100e-03	1.7768e-03	2.0531e-03
11	1.3463e-02	1.2820e-02	1.2469e-02	1.3180e-02
12	7.1728e-02	7.3330e-02	7.2509e-02	7.4158e-02
13	2.2255e-03	1.9500e-03	1.8153e-03	2.0945e-03
14	6.6043e-02	6.4990e-02	6.4214e-02	6.5773e-02
15	7.4273e-02	7.5740e-02	7.4907e-02	7.6580e-02

Figure 11: Comparison between Analytical and Numeric Results, $N = 100000$

5.4 Covariance Matrix

While in the first part of this project we used the replica methode, i.e. multiple Monte-Carlo simulations to estimate the uncertainty of the observables we now implemented the covariance matrix methode which allows

us to get and estimation of the uncertainties from just one simulation. If we want to calculate the variance of $\bar{X_N}$, we can do so by calculating:

$$var(\bar{X_N}) = \frac{1}{(N - N_{th})^2} \sum_{m,n=N_{thermal}+1}^{N} \langle\langle \delta X(s_m) \delta X(s_n) \rangle\rangle \tag{15}$$

Here $\langle\langle \delta X(s_m) \delta X(s_n) \rangle\rangle$ are the elements of the covariance matrix. Then the error of $\bar{X_N}$ is given as

$$std(\bar{X}_N) = \sqrt{var(\bar{X}_N)} \tag{16}$$

We can get an estimator, if we replace:

$$\langle\langle \delta X(s_m) \delta X(s_n) \rangle\rangle = [X(s_m) - \bar{X}_N][X(s_n) - \bar{X}_N] \tag{17}$$

which will be very noisy. Covariance matrix is really close to zero if $|n - m|$ is large. Therefore we can define W, *integration or summation window* and the following:

$$\delta(w) = \frac{1}{(N - N_{th})^2} \sum_{m,n=N_{thermal}+1, |m-n|<W}^{N} [X(s_m) - \bar{X}_N][X(s_n) - \bar{X}_N] \tag{18}$$

If we choose a $W = \bar{W}$ value such that error caused by the neglected terms is small. We require:

$$|\frac{\delta(\bar{W} + 1) - \delta(\bar{W})}{\delta(\bar{W})}| < 5\% \tag{19}$$

That approach is supported further by the fact, that s changes less and less for large enough w as seen in figure **??** And the error is given as:

$$err(\bar{X_N}) = \sqrt{\delta(\bar{W})} \tag{20}$$

One can see that the error decreases with increasing number of configurations as one would expect(12

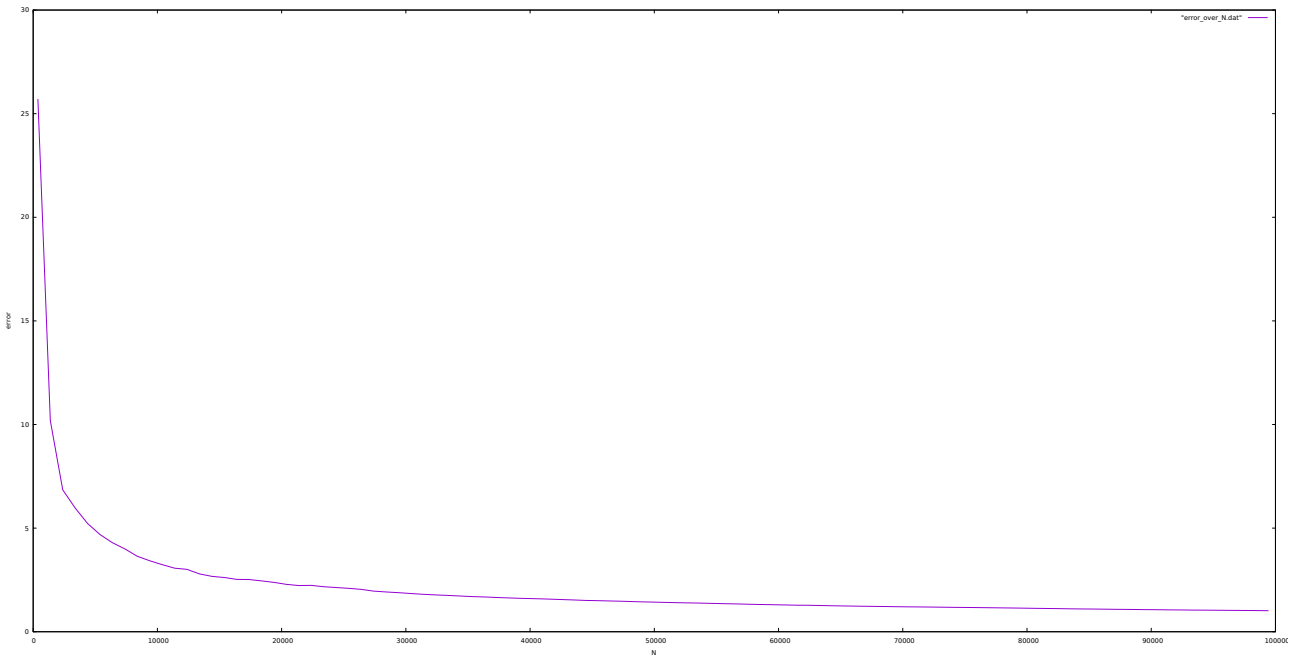

Figure 12: Error as a function of iterations

For $N = 100, D = 2, \beta = 0.43, b = 0.01, N_{conf} = 100000, \text{seed} = 12345$ we get an error of 1.017477 for the energy and 3.380548 for the magnetisation.

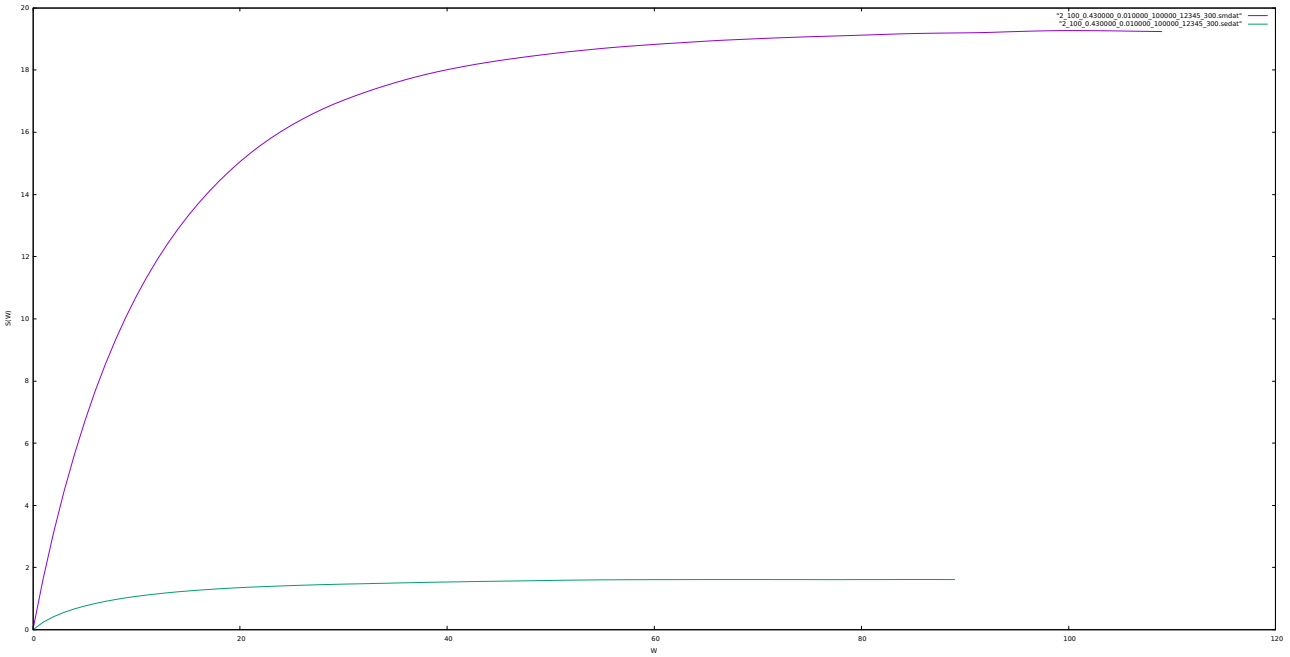

Figure 13: s as a function of w for both magnetisation and energy

6 Phasetransition

It is well known that the magnetisation of a ferromagnet depends heavily on its temperature. This is clearly reflected by our calculations of the magnetisation as seen in figure 14 Figure 15 shows the energy density of the ferromagnet for different values of b.

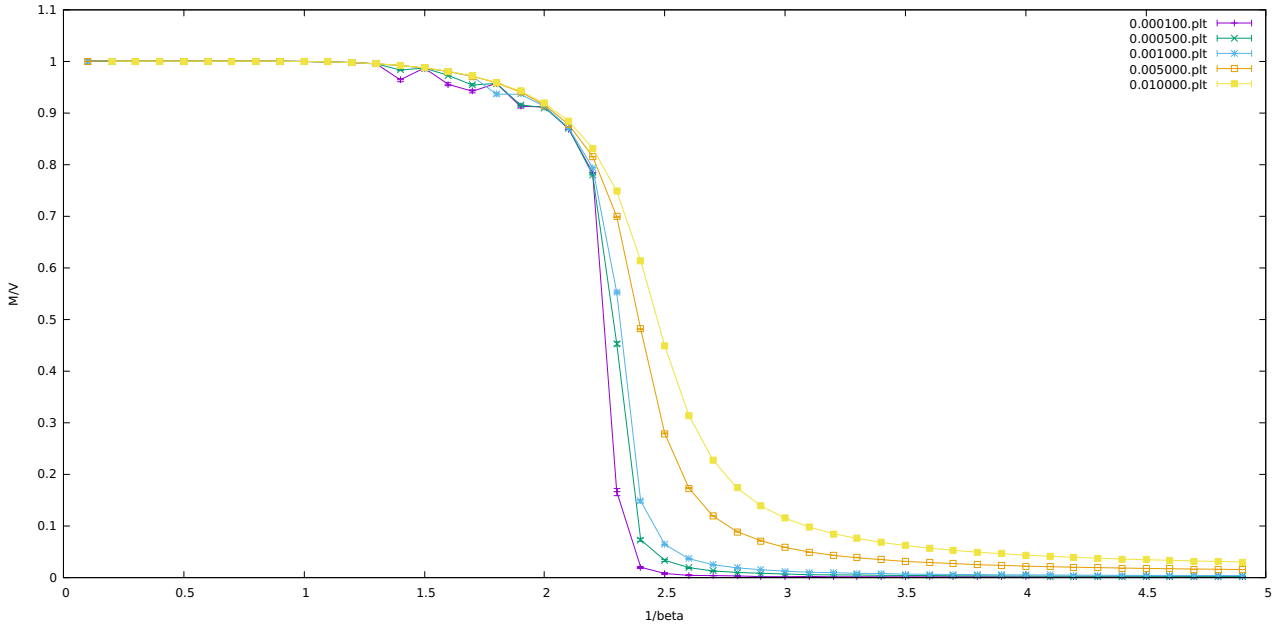

Figure 14: Magnetisation as a function of temperature for different values of b

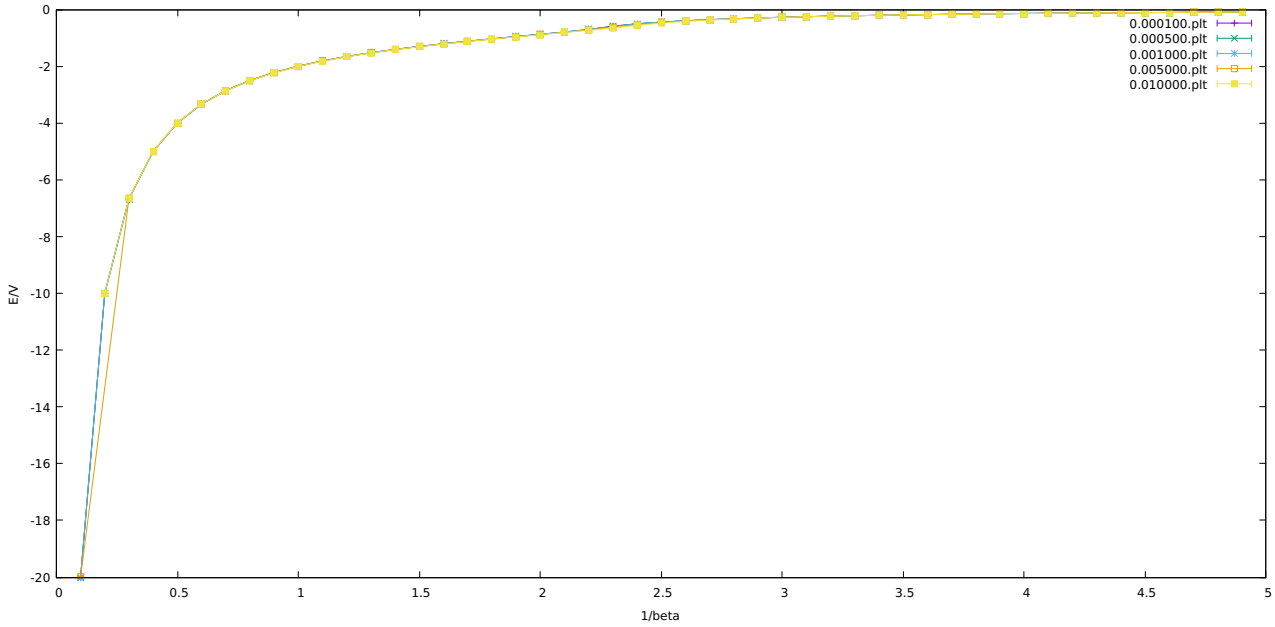

Figure 15: Energy density as a function of temperature for different values of b

7 Usage of the Code

The code is purely written in C and can be compiled with a Makefile. Be aware that the RANLUX library is present in the 'lib' subdirectory and has to be compiled for the architecture of your machine. The main part of the code can be build with 'make'. After a successful build the binary 'main' and 'analysation' will be present in the subdirectory 'app'. The executable 'main' implements the Metropolis algorithm and produces two files storing the energy and magnetisation of each iteration in a new-line seperated format, ready for gnuplot's plot function. Its parameters are:

- Dimension of the grid (integer)
- Number of Gridpoints per Dimension (integer)
- β (float)
- b (float)
- Number of iterations (integer)
- seed (integer)

It therefor calculates one replica. Running the main program therefor might look something like './app/main 2 100 0.43 0.01 1000 12345' The second executable takes two arguments, the path to a directory containing replica files and the thermalisation time. It calculates the expectation value of the energy and magnetisation as well as their error. Running the analysation program might look something like './app/analysation ./dataset2/ 300'.