Computational Physics Exercise 4: The 1D and 2D Ising Model

Fynn Janssen 411556[1] and Tamilarasan Ketheeswaran 411069[2]

April 2023

Contents

[1] fynn.janssen@rwth-aachen.de
[2] tamilarasan.ketheswaran@rwth-aachen.de

1 Introduction

This exercise discusses the Ising model of a 1D and 2D spin-lattice, which is realized with the Metropolis Monte Carlo algorithm. We study the internal energy U, specific heat C, and magnetization M as a function of the temperature T for different lattice and sampling sizes. At first, the 1D case is discussed and after that, we extend the model to a 2D lattice which, however, share many similarities. Throughout this exercise, we use natural units $k_B = 1$.[1] Further, the code is written in the programming language `python` and the packages `numpy`[2] as well as `matplotlib` are used. To achieve reproducible results we use the seed `rand.seed(1069)`.

2 1D Spin-Lattice

2.1 Simulation Model and Method

At first, the 1D Ising model is introduced and the Metropolis Monte Carlo algorithm is presented. The investigated Ising model describes a 1D lattice in which each spin only interacts with its nearest neighbors. Additionally, in the 1D case, we examine free boundary conditions. Therefore, the energy of one specific configuration is given by

$$E = -\sum_{n=1}^{N-1} S_n S_{n+1}. \tag{1}$$

According to statistical mechanics, each configuration of energy E has a certain probability

$$P(E) = \frac{e^{-\beta E}}{\sum\limits_{\{S_1,\dots,S_N\}} e^{-\beta E}}, \tag{2}$$

to be realized, which contains a sum over all possible states. It would be inefficient to compute this sum, therefore, the Metropolis Monte Carlo algorithm is used. At first, we generate a random spin-lattice of size N, realized as an array containing ± 1. -1 corresponds to spin-down and $+1$ to spin-up. To create this array we use

```
rand.choice([-1,1], size=N).
```

In the next step, we pick a random spin S_i[3], of the lattice and determine the change in energy ΔE if this spin is flipped via

$$\begin{aligned} \Delta E &= 2S_i\left(S_{i-1} + S_{i+1}\right) \\ S_0 &= 0 \\ S_{N+1} &= 0, \end{aligned} \tag{3}$$

where $S_0 = 0$ and $S_{N+1} = 0$ account for the free boundaries. Statistical mechanics predicts that the probability of the change in energy induced by a spin flip is given by

$$q := \frac{P(E+\Delta E)}{P(E)} = e^{-\beta \Delta E}, \qquad \beta = 1/T. \tag{4}$$

[1] Note that neither the energy nor the temperature has units because only the ratio is of interest.

[2] `numpy` is abbreviated by `np` and we often make use of `numpy.random` which is abbreviated as `rand`.

[3] using `int(rand.random()*N)`

This can be realized by generating a random number r, which is uniformly distributed between zero and one, as a measure for the probability of this spin-flip. Hence, if $q > r$, the spin-flip is accepted and we obtain a new spin configuration. Otherwise, if $q \leq r$, we continue with the initial configuration. For an efficient simulation, the spin-lattice is updated via `S[i] *= -1`, if the condition for a spin-flip is met.[4] Note that this `i` is not the same as in eq.(3), since eq.(3) uses the mathematical convention to count from one instead of zero.

This model can now be used to make predictions about the internal energy per spin U/N and the specific heat per spin C/N. Statistical mechanics predicts that the internal energy can be calculated via

$$U = \frac{\sum\limits_{\{S_1,...,S_N\}} E e^{-\beta E}}{\sum\limits_{\{S_1,...,S_N\}} e^{-\beta E}}.^{5} \tag{5}$$

As already mentioned, we use the MMC[6] algorithm to avoid computing the sums over all possible spin configurations. Therefore, for any function $f(S_1, ..., S_N)$, the average F can be calculated in the following way

$$F = \frac{\sum f(S_1, ..., S_N) e^{-\beta E}}{\sum e^{-\beta E}} = \frac{1}{\#\Omega} \sum_{\{S_1,...,S_N\} \in \Omega} f(S_1, ..., S_N), \tag{6}$$

where Ω is the ensemble of all simulated configurations in thermal equilibrium and $\#\Omega$ is the number of configurations.

Thus, the internal energy per spin and the specific heat per spin yield

$$U/N = \frac{1}{N} \frac{\sum E e^{-\beta E}}{\sum e^{-\beta E}} = \frac{1}{N} \frac{1}{\#\Omega} \sum_{\{S_1,...,S_N\} \in \Omega} E \tag{7}$$

and

$$\begin{aligned} C/N =& \frac{\beta^2}{N} \left(\frac{\sum E^2 e^{-\beta E}}{\sum e^{-\beta E}} - U^2 \right) \\ =& \frac{\beta^2}{N} \left[\left(\frac{1}{\#\Omega} \sum_{\{S_1,...,S_N\} \in \Omega} E^2 \right) - U^2 \right], \end{aligned} \tag{8}$$

while the theoretical predictions read

$$U/N = -\frac{N-1}{N} \tanh(\beta) \tag{9}$$

$$C/N = \frac{N-1}{N} \left(\frac{\beta}{\cosh(\beta)} \right)^2. \tag{10}$$

[4] This is a tradeoff between clarity and performance of the code. We could also implement a function, that does the spin-flip, returns the new lattice, and call this function in every step. This would be more clear but the runtime of the code would increase.

[5] In the following equations we use "$\sum$" instead of "$\sum\limits_{\{S_1,...,S_N\}}$" for clarity. If the sum is not over the spin configurations $\{S_1, ..., S_N\}$ we explicitly state the summation.

[6] Metropolis Monte Carlo

After this introduction on how we calculate the physical quantities, we present the method to implement it. The goal is to obtain U/N and C/N as a function of the temperature $T \in \{0.2, ..., 4\}$ in 20 steps. To this end, we start with a randomly generated spin-lattice[7] at $T = 4$ and do $N_{wait} = 10^6$ steps in a loop to obtain a spin-lattice in thermal equilibrium. After that, we have a nested loop. The outer one loops over the temperature that decreases from $T = 4$ in steps of $\Delta T = 0.2$. Here, we initialize the energy $E(S)$ according to eq.(1) via

```
E(S) = -np.sum(S[1:]*S[:-1]).
```

Furthermore, we define the two variables `U_temp = 0` and `C_temp = 0`, which are updated in every step of the inner loop. The inner loop iterates $N_{run} = N \cdot N_{samples}$ times, while in each step we do 4 things. At the beginning of every iteration, we determine the energy of the current system with `E += dE`, where `dE` is zero at the first iteration and after that determined by

$$\texttt{dE} = \begin{cases} \Delta E & \text{if } q > r \\ 0 & \text{if } q \leq r, \end{cases} \tag{11}$$

according to eq.(3). The second and third calculation determines the average energy E and energy squared E^2 via eq.(6), where $\#\Omega = N_{run}$. This is done by calculating `U_temp += E/`N_{run} and `C_temp += E**2/`N_{run}, which are temporary variables that are initially zero for each new temperature. Finally, we do the MMC as already discussed, such that the spin-lattice is updated and the process repeats.

Once the inner loop is finished, we obtain the internal energy per spin and the specific heat per spin

$$U/N = \frac{\texttt{U}_\texttt{temp}}{N} \tag{12}$$

$$C/N = \frac{\texttt{C}_\texttt{temp} - \texttt{U}^2_\texttt{temp}}{N \cdot T^2}. \tag{13}$$

Both results are stored in an array that gets filled in each step of the outer loop. Thus, in this way we obtain one value for U/N and C/N for each temperature T.

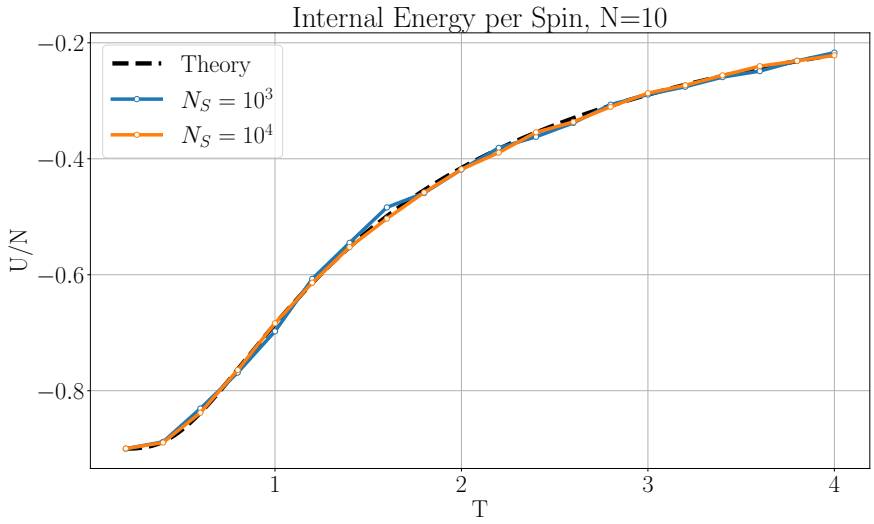
[7] This means that the orientations of the spins are randomly distributed

2.2 Results

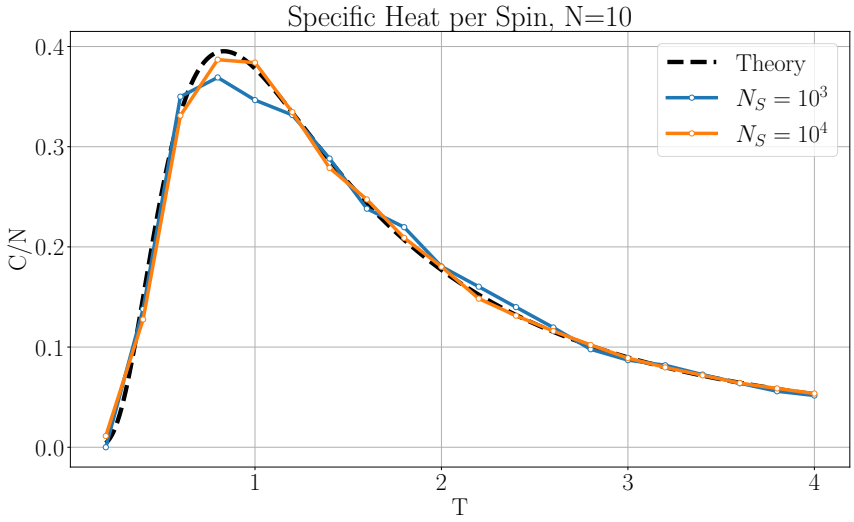
In this subsection, we present the results of the discussed model for different spin-lattice sizes $N = 10, 100, 1000$ and different sampling sizes $N_S = 1000, 10000$.[8] The results are organized according to the size of the system. Every plot also contains the theoretical prediction according to equations (9) and (10), represented by the black dashed line.

2.2.1 N=10 Spins

The results for a lattice size of $N = 10$ spins are presented in figures (1a) and (1b).



(a)



(b)

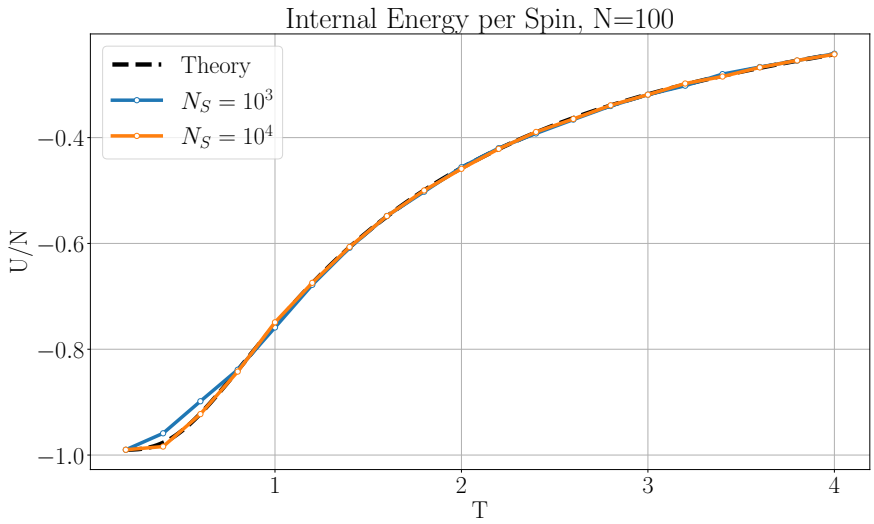
Figure 1: (a) internal energy per spin U/N, and (b) specific heat per spin C/N generated with $N = 10$ spins, $N_{wait} = 10^5$, $N_{run} = 10 \cdot N_{samples}$, and $N_{samples} = 1000, 10000$. The theoretical prediction is given by the black dashed curve.

[8] In this section, we use N_S instead of $N_{samples}$, for more clarity in the plot legends.

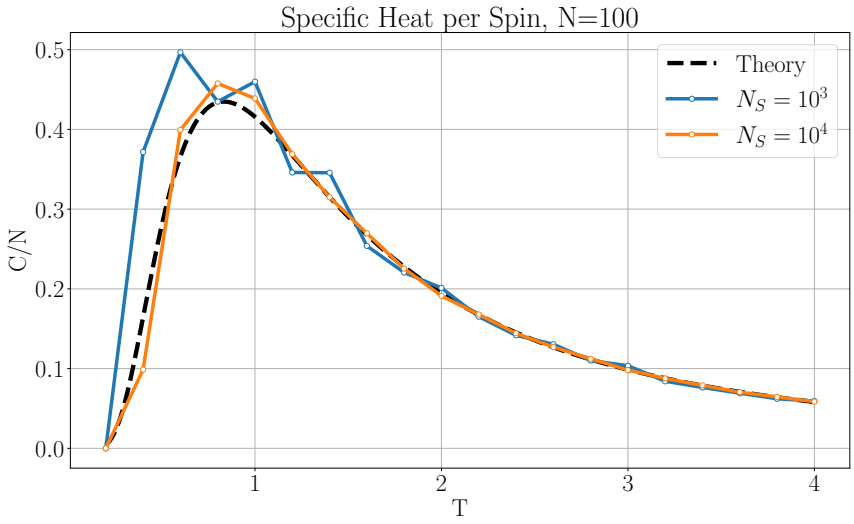
We can see that for the internal energy the data matches the theoretical prediction well. Notably, the simulation with $N_S = 1000$ yields slightly more fluctuating results. Similar characteristics can be seen in the results for the specific heat. Overall, both data curves follow the theory, with the most deviations at the peak maximum specific heat. Further, the result for $N_S = 1000$, also, deviates slightly more in comparison to $N_S = 10000$.

2.2.2 N=100 Spins

Secondly, we present the result for $N = 100$ spins, presented in figures (2a) and (2b).



(a)



(b)

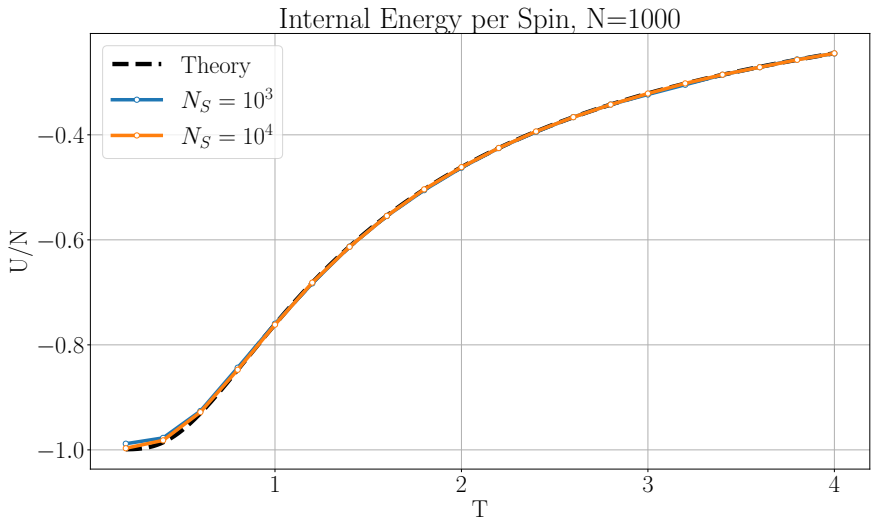
Figure 2: (a) internal energy per spin U/N, and (b) specific heat per spin C/N generated with $N = 100$ spins, $N_{wait} = 10^5$, $N_{run} = 100 \cdot N_{samples}$, and $N_{samples} = 1000, 10000$. The theoretical prediction is given by the black dashed curve.

The results are very similar to the ones obtained for $N = 10$, but more fluctuating.

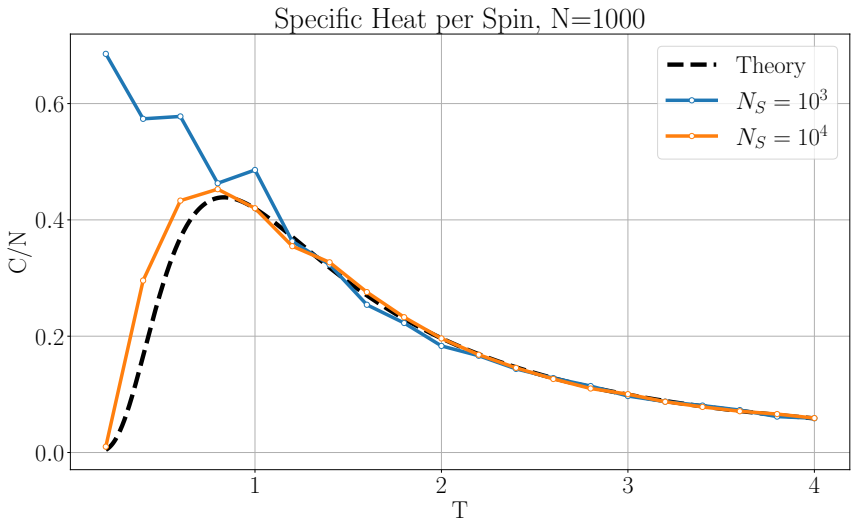
Again, we can see that for both, the internal energy and the specific heat, the results with a larger sample size are in better agreement with the theory.

2.2.3 N=1000 Spins

Finally, we present the results for $N = 1000$ spins in figures (3a) and (3b).



(a)



(b)

Figure 3: (a) internal energy per spin U/N, and (b) specific heat per spin C/N generated with $N = 1000$ spins, $N_{wait} = 10^5$, $N_{run} = 1000 \cdot N_{samples}$, and $N_{samples} = 1000, 10000$. The theoretical prediction is given by the black dashed curve.

Similar to both cases before, the results for $N_S = 10000$ fit the theoretical predictions quite well. Further, the internal energy for $N_S = 1000$ is also in agreement with the theory, whereas the result for the specific heat strongly deviates from the theory for temperatures at the peak of the specific heat and below.

2.3 Discussion

First of all, we have to note that we cannot make any predictions about whether the simulated results truly are in agreement with the theoretical predictions since we do not have any uncertainties on the data. However, we are able to compare the data and results qualitatively.

The results for the internal energy per spin seem to represent the theoretical prediction well, with the most derivations for low temperatures below $T = 1$. Here, we would expect to find greater uncertainties, since these derivations can be seen for all lattice sizes. Further, we noticed that the results for $N = 10$ are slightly more fluctuating in comparison to the results with $N = 100, 1000$. This is a consequence of the small lattice size. For $N = 10$ the fluctuations in the selection of the spins are greater, especially for small sample sizes. This can be easiest understood with an example. Let us assume that one spin is significantly less flipped than the other ones due to the randomness of the algorithm. In this case, 10% of the system might not be in thermal equilibrium. If we extend this example to larger lattice sizes but still consider that one spin is significantly less flipped, we find that only a smaller percentage of the system is not in thermal equilibrium. Hence, there are fewer fluctuations.

At last, we discuss the results for the specific heat. As already mentioned, the results seem to be in very good agreement for temperatures greater than the peak in the theoretical prediction. At the peak and for lower temperatures we find more fluctuating results, especially for $N_S = 1000$. Fig. 3b represents this behavior the best. We expect that the uncertainties in this region are significantly higher than for larger temperatures since all three results show similar behavior. For $N_S = 10000$, we still have more fluctuations in the region around the peak, but significantly less. A possible explanation for this behavior is the following. At low temperatures, the spins of the system tend to align in one direction and domains of the same spin form. Between domains of spin up and spin down, there is an energy barrier, which can take more iterations to overcome and the system takes longer to reach thermal equilibrium. This means, that the system gets trapped in a state in which the energy is at a local minimum but not at the global one. Furthermore, this is not only an explanation for the larger fluctuations at lower temperatures, but also for the less deviating results at larger temperatures. In this regime, the system does not form large domains but rather has more spin fluctuations. Thus, the system does not get trapped in a local minimum.

Furthermore, in the results, we can see that there is no phase transition. In our system, a phase transition would describe the transition between a non-magnetic (all spins are random) to a magnetic state (all spins are aligned). This is an effect of the formation of domains and the large energy barrier between different domains. Thus, even at low temperatures, there is no spontaneous phase transition. This can be described by the Peierls-argument, which concludes that spontaneous phase transitions are not possible in one dimension since the probability of the formation of different domains is greater than the probability that all spins align.

3 2D Spin-Lattice

3.1 Simulation Model and Method

For the 2D Ising model, the procedure is similar to the of 1D. While for the 1D model, the energy for a given spin configuration is given by the interaction of the spin with its two neighboring spins, in 2D one has to account for the neighboring spins along both axis. However, we neglect the interactions of two spins diagonally to each other. Hence the energy for the system is given by

$$E = -\sum_{i=1}^{N-1}\sum_{j=1}^{N} S_{i,j}S_{i+1,j} - \sum_{i=1}^{N}\sum_{j=1}^{N-1} S_{i,j}S_{i,j+1} \tag{14}$$

$$\underbrace{-\sum_{i=1}^{N} S_{i,1}S_{i,N} - \sum_{j=1}^{N} S_{1,j}S_{N,j}}_{\text{only for periodic boundaries}}\,. \tag{15}$$

The initialization of a 2D lattice can be done similarly to the 1D. Using

$$S = \texttt{rand.choice([-1, 1], size=(N,N))}$$

we obtain an array consisting of -1 and $+1$ of shape (N, N). This equals a square lattice with a total number of N^2 spins. Next, we want to know the energy difference ΔE, if one of the N^2 spins is flipped. We could calculate the total energy before and after the spin-flip and infer the energy difference from both energies, but as this would take too long we use the same approach as in eq.(3). The difference in energy if spin $S_{i,j}$ flipped is given by

$$\Delta E = 2S_{i,j} \cdot (S_{i-1,j} + S_{i+1,j} + S_{i,j-1} + S_{i,j+1}). \tag{16}$$

The choice which spin is flipped is done using

```
i,j = (rand.random(size=2)*N).astype(int).
```

This generates two random integers from 0 to $N-1$, specifying the index of the spin $S_{i,j}$ that is flipped.[9] Having generated the numbers defining the spin that is flipped, we have to take into account the different boundary conditions. For free boundary conditions, we set

$$S_{i,j} = 0, \qquad \text{if any } i \text{ or } j = \{0, N+1\}. \tag{17}$$

This can be implemented into python using `if`-statements. For the periodic boundaries we set

$$\begin{aligned} S_{N+1,j} = S_{1,j}, \quad S_{i,N+1} = S_{i,1}, \qquad \text{and} \\ S_{0,j} = S_{N,j}, \quad S_{i,0} = S_{i,N} \end{aligned} \tag{18}$$

for all i, j. In `numpy` indexing an array with -1 yields the last element, which solves the latter condition inherently. For the first condtion of eq.(18) we need an `if`-statement that if an element is indexed at $N+1$[10], the first element of the array is

[9] note that the `i,j` that are generated are not the same integers as the i, j, as python starts indexing arrays from 0. As the numbers are randomly generated it is of no avail to use different indices for the mathematical and code description.

[10] corresponds to index `N` in the `python` array

given.

Again, from statistical mechanics, we know that the probability for a spin-flip is given by eq.(4). With the same criterion as before we flip a spin if $q > r$, with r being a randomly generated number with $r \in [0, 1)$. In case this condition is met, the spin corresponding to the indices `i,j` of the lattice is flipped using `S[i,j]*=-1`. Now that we have defined the basic construct needed to implement a spin-flip, we discuss the quantities we are interested in. The goal is to obtain the internal energy U/N^2, specific heat C/N^2 and magnetization M/N^2 per spin as a function of the temperature $T \in \{0.2, ..., 4\}$ in 20 steps. While the calculation of U and C was discussed in the section before, for 2D the magnetization M is defined as

$$M = \frac{\sum\limits_{\{S_{1,1},...,S_{N,N}\}} \sum_{i,j=1}^{N} S_{i,j} e^{-\beta E}}{\sum\limits_{\{S_{1,1},...,S_{N,N}\}} e^{-\beta E}}. \tag{19}$$

Using eq.(6) and from the latter equation we can infer the formula using the MMC for the magnetization per spin M/N^2 to be

$$M/N^2 = \frac{1}{N^2} \frac{1}{\#\Omega} \sum_{\{S_{1,1},...,S_{N,N}\} \in \Omega} \sum_{i,j=1}^{N} S_{i,j}. \tag{20}$$

Here $\#\Omega = N^2 \cdot N_{samples}$, because the number of spins is given by N^2. From the theory we expect the following behavior for the magnetization

$$M/N^2 = \begin{cases} (1 - \sinh^{-4} 2\beta)^{1/8} & \text{if } T < T_C = \frac{2}{\ln(1+\sqrt{2})} \\ 0 & \text{if } T > T_C \end{cases}$$

To obtain the quantities of interest, analogous to the 1D lattice, we start at $T = 4$ and do $N_{wait} = 10^5$ spin-lattice updates in a loop to obtain a lattice in thermal equilibrium. After that, we have a nested loop. The outer one loops over the temperature that decreases from $T = 4$ in steps of $\Delta T = 0.2$. Here, we initialize the energy $E(S)$ and boundary conditions according to eqs (15) and (18), and the initial magnetization M_{temp} eq.(20) via

```
E(S) = -np.sum(S[1:,]*S[:-1,])-np.sum(S[:,1:]*S[:,:-1])
       -(np.sum(S[0,]*S[N-1,])+np.sum(S[:,0]*S[:,N-1]))
```

only for periodic boundaries

For the initial magnetization per spin, we sum over the initial spin states using

$$\texttt{M}_{\texttt{temp}} = \texttt{np.sum(S)/N**2}. \tag{21}$$

Further, we define the five variables

$$\texttt{U}_{\texttt{temp}} = 0,\ \texttt{C}_{\texttt{temp}} = 0,\ \texttt{dE} = 0,\ \texttt{M} = 0 \text{ and } \texttt{dM} = 0,$$

which are updated in every step of the inner loop. The inner loop iterates $N_{run} = N^2 \cdot N_{samples}$ times, while in each step we do the following.

At the beginning of each iteration, we determine the energy of the current system with `E += dE`, where `dE` is zero at the first iteration. Then, similar to the 1D case, we

calculate $\texttt{U}_{\texttt{temp}}\texttt{+= E/N}_{\texttt{run}}$ and $\texttt{C}_{\texttt{temp}}\texttt{+= E**2/N}_{\texttt{run}}$ to determine the average of E and E^2 according to eq.(6). Next, we do the MMC step, that we discussed in this section, in which the spin-lattice is updated. The energy difference `dE` in each iteration is calculated via,

$$\texttt{dE} = \begin{cases} \Delta E & \text{if } q > r \\ 0 & \text{if } q \leq r \end{cases} \tag{22}$$

and finally, the change in the magnetization per spin is determined via

$$\texttt{dM} = \begin{cases} 2 * \texttt{S[i,j]}/\texttt{N}^2 & \text{if } q > r, \\ 0 & \text{if } q \leq r. \end{cases} \tag{23}$$

In these formulae, ΔE is calculated according to eq.(16) and `S[i,j]` in eq.(23) describes the flipped spin.

At last, we calculate

$$\begin{aligned} \texttt{M}_{\texttt{temp}} \texttt{ = M}_{\texttt{temp}} \texttt{ + dM} \\ \texttt{M += M}_{\texttt{temp}}, \end{aligned} \tag{24}$$

which sums the magnetization of each configuration and yields the average magnetization according to eq.(6).

After all that, in the outer loop again, we calculate the quantities according to eq.(7), eq.(8), eq.(20) for a given temperature using

$$\begin{aligned} U/N^2 &= \frac{\texttt{U}_{\texttt{temp}}}{N^2} \\ C/N^2 &= \frac{\texttt{C}_{\texttt{temp}} - \texttt{U}^2_{\texttt{temp}}}{N^2 \cdot T^2} \\ M/N^2 &= \frac{|\texttt{M}|}{N_{run}} \end{aligned} \tag{25}$$

Note that `M` refers to a variable in the code that is updated during each iteration of the inner loop and M is the average magnetization that is calculated after the inner loop. After iterating the loops over all the temperatures, we are left with arrays containing the values of interest at each temperature. Note that this whole code runs twice, once for free boundary conditions and once for periodic ones

3.2 Results

In this subsection, we present the results of the discussed model for different spin-lattice sizes $N = 10, 50, 100$ and different sampling sizes $N_S = 1000, 10000$.[11] The results are organized according to the size of the system. The plot for the magnetization also contains the theoretical prediction according to eq.(20), represented by the black dashed line. For the specific heat plots, a black vertical bar is drawn at the critical temperature T_C.

3.2.1 N=10 Spins

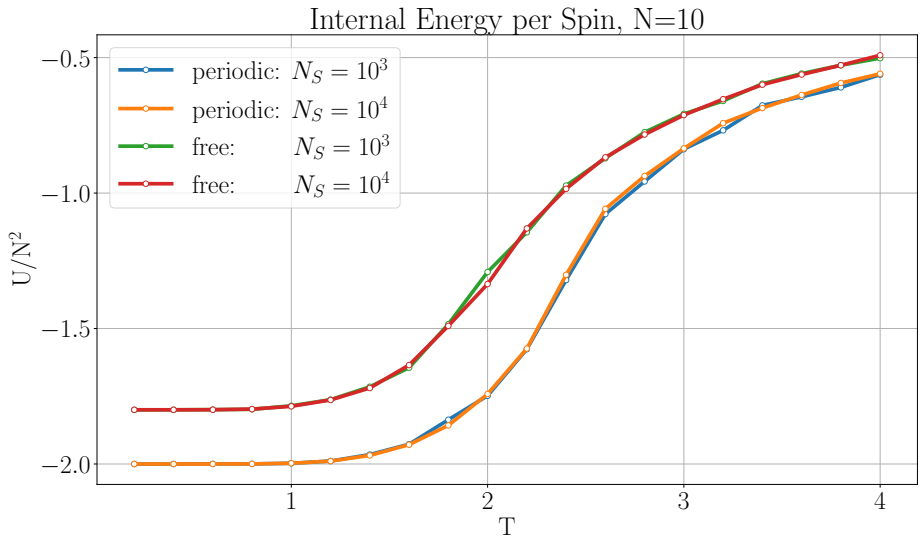


Figure 4: Internal energy per spin U/N^2. $N_{samples} = 10^3$ and 10^4 and number of spins $N^2 = 10^2$. The green and red curve are for free boundary conditions, while the orange and blue one are for periodic ones.

[11] In this section, we use N_S instead of $N_{samples}$, for more clarity in the plot legends.

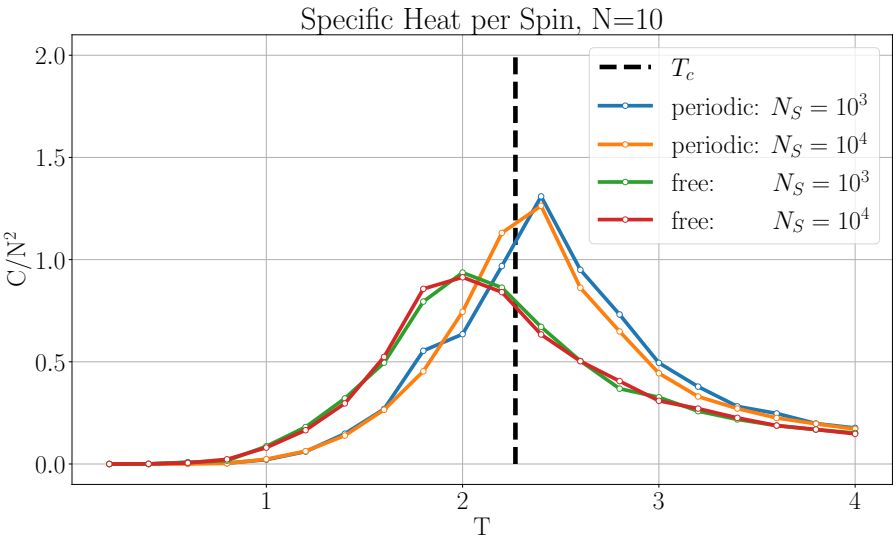


Figure 5: Specific heat per spin C/N^2. $N_{samples} = 10^3$ and 10^4 and number of spins $N^2 = 10^2$. The green and red curves are for free boundary conditions, while the orange and blue ones are for periodic ones. Vertical black bar depicts critical temperature T_C

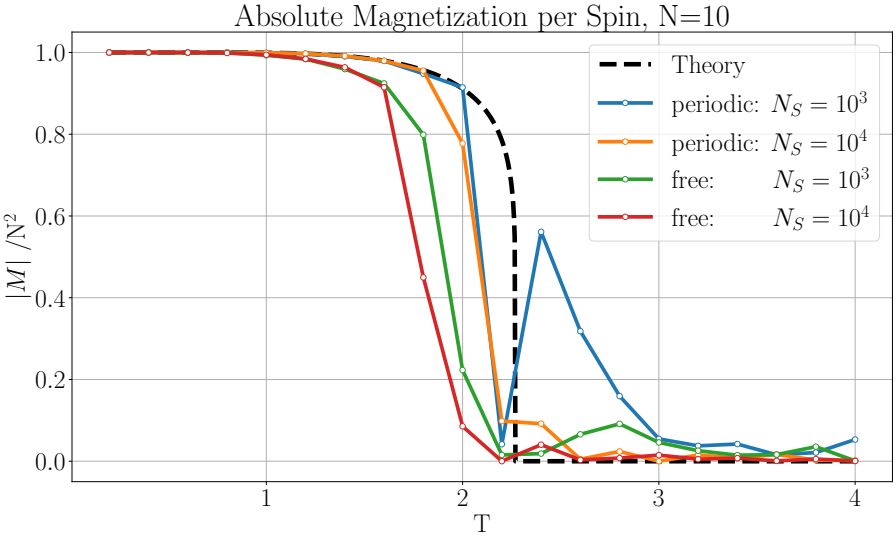


Figure 6: Absolute magnetization per spin M/N^2. $N_{samples} = 10^3$ and 10^4 and number of spins $N^2 = 10^2$. The green and red curves are for free boundary conditions, while the orange and blue ones are for periodic ones. Black dashed line depicts theory

3.2.2 N=50 Spins

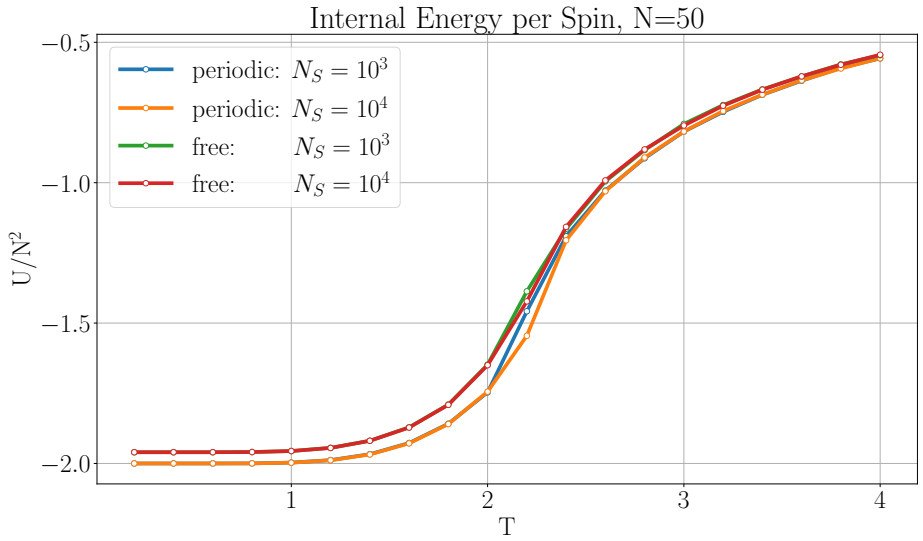


Figure 7: Internal energy per spin U/N^2. $N_{samples} = 10^3$ and 10^4 and number of spins $N^2 = 50^2$. The green and red curves are for free boundary conditions, while the orange and blue ones are for periodic ones.

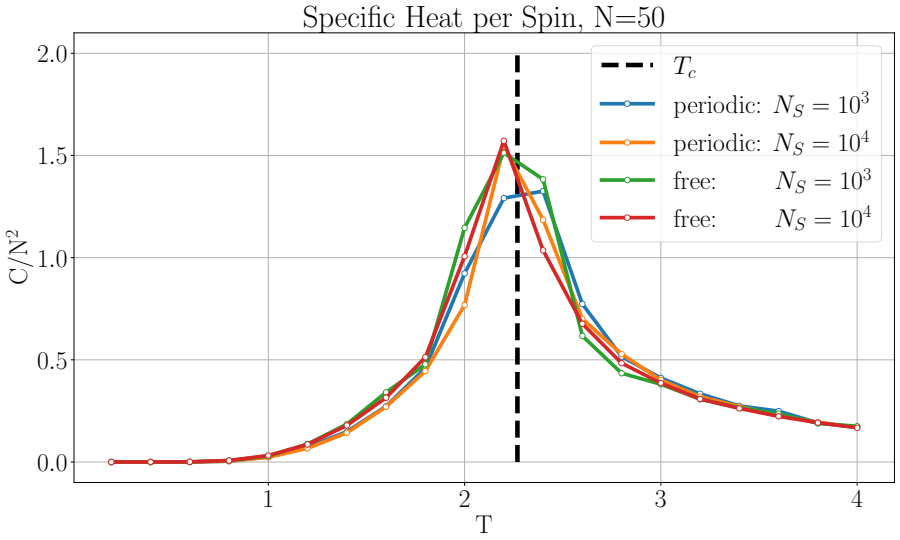


Figure 8: Specific heat per spin C/N^2. $N_{samples} = 10^3$ and 10^4 and number of spins $N^2 = 50^2$. The green and red curves are for free boundary conditions, while the orange and blue ones are for periodic ones. Vertical black bar depicts critical temperature T_C

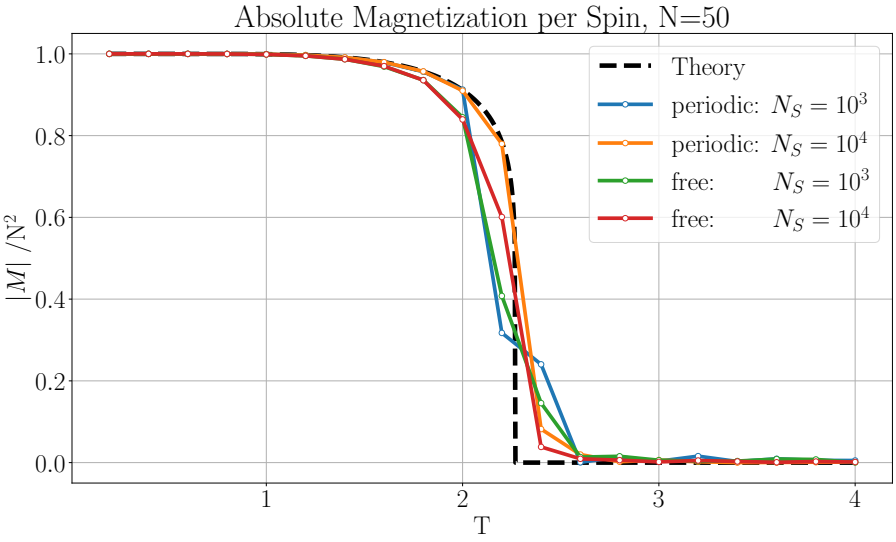


Figure 9: Absolute magnetization per spin M/N^2. $N_{samples} = 10^3$ and 10^4 and number of spins $N^2 = 50^2$. The green and red curves are for free boundary conditions, while the orange and blue ones are for periodic ones. Black dashed line depicts theory

3.2.3 N=100 Spins

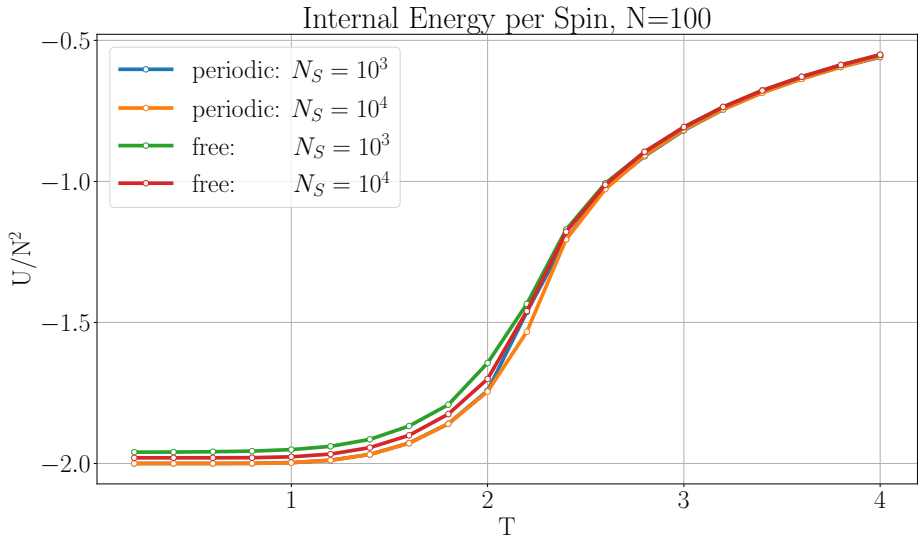


Figure 10: Internal energy per spin U/N^2. $N_{samples} = 10^3$ and 10^4 and number of spins $N^2 = 100^2$. The green and red curves are for free boundary conditions, while the orange and blue ones are for periodic ones.

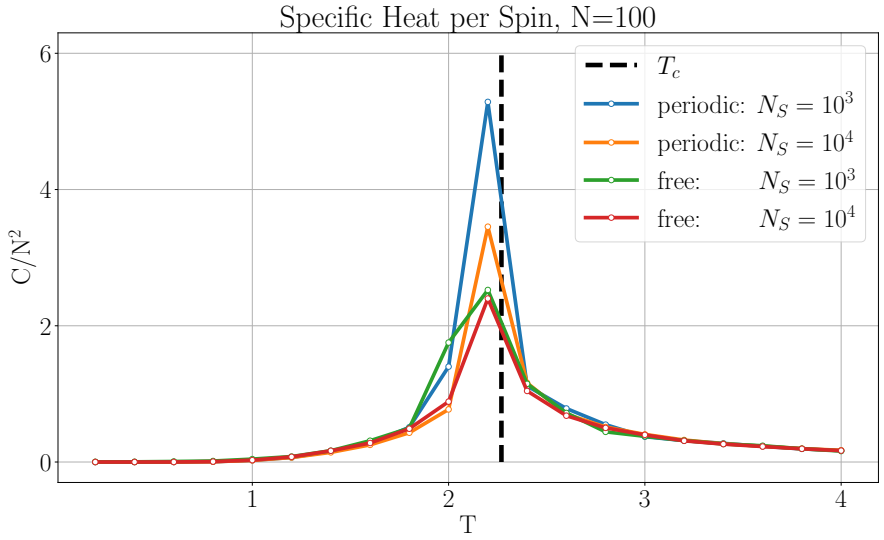


Figure 11: Specific heat per spin C/N^2. $N_{samples} = 10^3$ and 10^4 and number of spins $N^2 = 100^2$. The green and red curves are for free boundary conditions, while the orange and blue ones are for periodic ones. Vertical black bar depicts critical temperature T_C

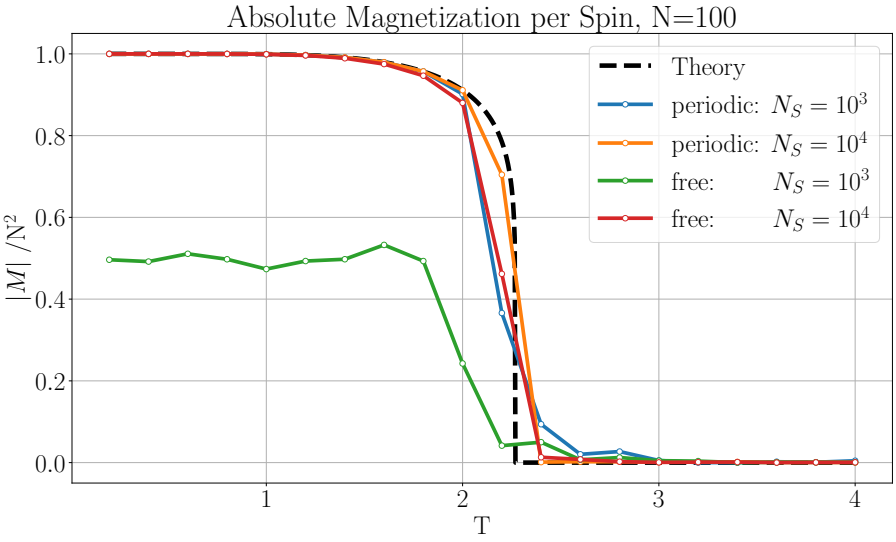


Figure 12: Absolute magnetization per spin M/N^2. $N_{samples} = 10^3$ and 10^4 and number of spins $N^2 = 100^2$. The green and red curves are for free boundary conditions, while the orange and blue ones are for periodic ones. Black dashed line depicts theory

3.3 Discussion

We start discussing the internal energies of the different systems. In fig.(4), we can see that the results for periodic and free boundary conditions follow a similar curve with a slight offset. Comparing it to the system of $N = 50$ and $N = 100$ in fig.(7) and fig.(10), we can see that the graphs with the periodic boundary conditions are the same except for slight fluctuations. This can be interpreted in a way that assuming periodic boundary conditions the finite lattice imitates an infinite one. Of course, there are not as many free spins that can be arranged in domains, but qualitatively this behavior can be understood in that way. For free boundary conditions, we can see an offset that gets smaller for increasing N. This is due to the fact that for smaller systems the boundaries occupy a larger fraction of the system. For $N \to \infty$ periodic and free boundary conditions are equal. Hence, the deviation of the internal energy for free to periodic boundary conditions decreases with increasing N, which is observable in our data. To understand why the internal energy decreases for small energies at all one has to understand that at high temperatures, far above the critical temperature, the system is in a disordered phase with randomly oriented spins. In this regime, the internal energy is relatively high. As the temperature decreases, the spins begin to align, and the internal energy decreases gradually. The aligned spins, however, form clusters that decrease the total energy. Near the critical temperature, these clusters become more prevalent, resulting in a sharp drop in the internal energy.

Next, we want to understand the curves of the specific heat given in fig.(5), fig.(8) and fig.(11). As expected the curves share a similar behaviour. For $N = 50$ and $N = 100$ both the curves with periodic and free boundary conditions are similar. For $N = 50$ the peaks near the critical temperature are of equal height independent of the boundary conditions chosen. For the $N = 100$ data, the periodic boundary conditions have higher peaks than the free ones. For $N = 10$, however, the peaks for the curve with periodic boundary conditions are shifted a little bit to the right from the critical temperature, while the curve with free boundary condition is shifted to the left from T_C. In all cases, the peak is in the vicinity of the critical temperature T_C. It is difficult to quantize the deviation from the critical temperature, as the sampling rate we have chosen for the temperature with steps of $\Delta T = 0.2$ is simply too coarse. However, with the chosen discretization of the temperatures, the critical temperature lies within the expected range. The deviations for $N = 10$ are likely due to the small size of the lattice and the spins on the edge being loosely bound to the system. In all systems, we can observe a specific heat going to 0 at small temperatures, a peak near T_C, and a subsequent decrease. To understand this behavior, we have to understand that the lattice experiences a phase transition close to the critical temperature, which is characterized by the occurrence of long-range correlations and fluctuations in the system. These critical fluctuations depend on the collective behavior of many spins. The specific heat reaches a peak as a result of these fluctuations. We can take into account the behavior of the spins and their energy fluctuations close to the phase transition to comprehend why the specific heat peaks at the critical temperature. Below the critical temperature, the spins are organized and aligned in domains. The energy of the system is significantly affected by these correlated spin clusters. The energy of the system varies according to changes in the size and arrangement of these clusters. The energy fluctuations are most pronounced at the critical temperature and depend on a variety of cluster sizes and configurations. The correlation length decreases and the system is less susceptible to large fluctuations as the temperature rises above the critical temperature. As a result, the specific heat away from the

critical temperature decreases.

For the absolute magnetization in fig.(6), fig.(9) and fig.(12), we see that the results fit the expectation given by the black dashed line. For $N = 10$, we can see the largest deviation from the curve. This was to be expected for free boundaries, as for small lattices the spins at the edges or boundaries of the system experience different interactions compared to those in the bulk. These spins have fewer neighboring spins, which can lead to deviations in the average magnetization compared to an infinite system. That is why for periodic boundary conditions it fits better than for free. The outliers in the curve of the periodic boundary conditions for $N_S = 10^3$ (in fig.(6)) are due to random deviations/ fluctuation, as for a sample size of $N_S = 10^4$ we do not observe such. For a higher number of spins, the results match the expectation really well, except for the curve of $N_S = 10^3$ in fig.(12). Again for a higher number of N_S, we do not observe this behavior. Hence we can conclude that this is due to numeric errors as well. To understand the curve we obtained, one has to keep in mind the fact that for high temperatures, well above the critical temperature, the system is of high disorder. This leads to the fact that the spins are randomly oriented in this regime, leading to a nonexistent average magnetization. The spins begin to align as the temperature drops, and the average magnetization gradually rises. The beginning of some level of order in the system is indicated by this increase. The magnetization rapidly changes as the temperature approaches the critical temperature. Large clusters of aligned spins emerge as a result of the spins going through a phase transition close to the critical temperature. As a result, the average magnetization per spin increases and approaches 1 for decreasing temperature.

Appendix A Code for the 1D Ising Model

A.1 Generate Data

```
import numpy as np

rand = np.random
rand.seed(1069)     #define random seed


# Calculate the energy of a given spin configuration S
def energy(S):
    return -np.sum(S[1:]*S[:-1])

# Calculate the energy difference for a given spin configuration S
# and a spin flip at index j
def energy_diff(S, N, j):
    # Make distinctions for the boundary spins
    if j==0:
        return 2*S[0]*S[1]
    elif j==(N-1):
        return 2*S[j]*S[j-1]
    else:
        return 2*S[j]*(S[j-1] + S[j+1])

# Do spin flip according to MMC Ising model
# Only used in the loop over N_wait since this is rather fast anyway
def spin_flip_new(S, N, T):
    j = int(rand.random()*N)
    r = rand.random()

    dE = energy_diff(S, N, j)
    q  = np.exp(-dE/T)

    if q > r:
        S[j] *= -1

    return S

# Main function that calculates the internal energy per spin and
# specific heat per spin
def calc_U_C(N_samples, N):
    K       = 20
    N_wait  = int(1e6)
    N_run   = int(N_samples*N)

    # define temperature, internal energy and spec. heat array
    T   = np.linspace(0.2, 4, K)
    UN  = np.zeros(K)
    CN  = np.zeros(K)

    # initial spin lattice
    S = rand.choice([-1, 1], size=N)

    # reach thermal eq. in N_wait steps
    for j in range(N_wait):
        S = spin_flip_new(S, N, T[K-1])

    # Nested Loop over each temperature (outer loop)
    # and over N_run spin-lattice updates (inner loop)
    for k in range(K):
        # define energy and temporary variables
```

```
        E       = energy(S)
        dE      = 0
        U_temp  = 0
        C_temp  = 0
        for i in range(N_run):
            # calc energy, mean energy and mean of energy squared
            E           += dE
            U_temp      += E/N_run
            C_temp      += E**2/N_run

            # Do the spin update according to the MMC Ising model
            j       = int(rand.random()*N)
            r       = rand.random()
            dE      = energy_diff(S, N, j)
            q       = np.exp(-dE/T[K-1-k])
            if q > r:
                S[j] *= -1
            else:
                dE = 0

        # Fill the arrays for U and C for each temp.
        UN[K-1-k] = U_temp/N
        CN[K-1-k] = (C_temp - U_temp**2)/T[K-1-k]**2/N
    return UN, CN




# Generate the data and save everything in a folder
def gen_data():
    N_arr           = np.array([10, 100, 1000])
    N_samples_arr   = np.array([1000, 10000])

    def text_rep(a, b, c):
        text = "1D_Data/1D_x_qN_zNS.npy"
        text = text.replace("x", str(a))
        text = text.replace("q", str(b))
        text = text.replace("z", str(c))
        return text


    for N in N_arr:
        for N_s in N_samples_arr:
            UN, CN = calc_U_C(N_s, N)
            np.save(text_rep("U", N, N_s), UN)
            np.save(text_rep("C", N, N_s), CN)

# generate the data
gen_data()
```

A.2 Generate Plots

```
import matplotlib.pyplot as plt
import numpy as np

# For fancy plots :)
plt.rcParams["text.usetex"] = True
plt.rcParams["font.family"] = "times new roman"
plt.rcParams["font.size"]   = "28"

def setup(title, xlabel, ylabel):
```

```
    plt.figure(figsize=(16, 9))
    plt.title(title)
    plt.xlabel(xlabel)
    plt.ylabel(ylabel)
    plt.xticks()
    plt.yticks()
    plt.locator_params(nbins=6)
    plt.grid()


# Theoretical expectations for internal energy and specific heat
def UN_theo(N, T):
    return -(N-1)/N * np.tanh(1/T)

def CN_theo(N, T):
    return (N-1)/N *(1/(T*np.cosh(1/T)))**2


# load the datasets
def dataset(N):
    def load_data(a, b, c):
        text = "1D_Data/1D_x_qN_zNS.npy"
        text = text.replace("x", str(a))
        text = text.replace("q", str(b))
        text = text.replace("z", str(c))
        return np.load(text)

    def create(name):
        arr = []
        Ns_arr  = [1000, 10000]
        for Ns in Ns_arr:
            arr.append(load_data(name, N, Ns))
        return np.array(arr).reshape(-1, 20)

    U = create("U")
    C = create("C")
    return [U, C]


# Plot the data
def plot_data(N, data, title, x_axis, y_axis, heat, name):
    lw_data = 4
    lw_theo = 5

    text = "1D_Plots/1D_ZN_X.pdf"
    text = text.replace("X", str(N))
    text = text.replace("Z", name)

    T       = np.linspace(0.2, 4, 20)
    T_theo  = np.linspace(0.2, 4, 1000)

    Ns_arr = ["10^3", "10^4"]

    setup(title, x_axis, y_axis)
    if heat:
        plt.plot(T_theo,  CN_theo(N, T_theo), color="black", ls="--",
                 linewidth=lw_theo, label="Theory")
    else:
        plt.plot(T_theo,  UN_theo(N, T_theo), color="black", ls="--",
                 linewidth=lw_theo, label="Theory")

    for i in range(2):
```

```
        plt.plot(T, data[i], linewidth=lw_data, marker="o",
                markerfacecolor='white',
                label=r" $N_S=Y$".replace("Y", Ns_arr[i]))
    plt.legend()
    plt.savefig(text)
    plt.close()


# Define the parameters that are used for the different plots
N_arr       = [10, 100, 1000]
title_arr   = ["internal energy per Spin, N=X",
               "Specific Heat per Spin, N=X"]
y_arr       = [r"U/N", r"C/N"]
heat        = [False, True]
name        = ["U", "C"]

# Loop over the different systems and create all resulting plots
for N in N_arr:
    data = dataset(N)
    for i in range(2):
        plot_data(N, data[i], title_arr[i].replace("X", str(N)), "T",
        y_arr[i], heat[i], name[i])
```

Appendix B Code for the 2D Ising Model

B.1 Generate Data

```
import numpy as np

rand = np.random
rand.seed(1069)                    # define random seed


# calculate energy for a given spin configuration S
def energy(S, boundary):
    N = len(S[0])
    S1 = S[1:,]                 # exclude the first row
    S2 = S[:-1,]                # exclude the last row

    S3 = S[:,1:]                # exclude the first column
    S4 = S[:,:-1]               # exclude the last column

    # distinguish between free and periodic boundaries
    per_bound_contr = 0
    if boundary == "per":
        S5 = S[0,]
        S6 = S[N-1,]

        S7 = S[:,0]
        S8 = S[:,N-1]

        per_bound_contr = np.sum(S5*S6) + np.sum(S7*S8)

    # Now we can express the sums via
    E = - np.sum(S1*S2) - np.sum(S3*S4) - per_bound_contr
    return E

# Calculate the energy difference for a given spin configuration S
# and a spin flip at indices i, j
def energy_diff(S, N, i, j, boundary):
```

```
    # distinguish between free and periodic boundaries
    if not boundary == "per":
        first_term  = S[i-1,j] if i > 0 else 0
        second_term = S[i+1,j] if i < N-1 else 0
        third_term  = S[i,j-1] if j > 0 else 0
        fourth_term = S[i,j+1] if j < N-1 else 0
    else:
        first_term  = S[i-1,j]
        second_term = S[i+1,j] if i < N-1 else S[0,j]
        third_term  = S[i,j-1]
        fourth_term = S[i,j+1] if j < N-1 else S[i,0]

    dE = 2*S[i,j]*(first_term + second_term + third_term
    + fourth_term)
    return dE

# Do spin flip according to MMC Ising model
# Only used in the loop over N_wait since this is rather fast anyway
def spin_flip(S, N, T, boundary):
    i, j = (rand.random(size=2)*N).astype(int)
    r = rand.random()

    dE = energy_diff(S, N, i, j, boundary)
    q  = np.exp(-dE/T)
    if q > r:
        S[i, j] *= -1

    return S


# Main function that calculates the internal energy per spin, specific
# heat per spin and abs. Magnetization per spin
def calc_U_C(N_samples, N, boundary):
    K       = 20
    N_wait  = int(1e5)              # N_samples
    N_run   = int(N_samples*N**2)       # int(N_samples*N)

    # define temperature, internal energy, spec. heat array
    # and magnetization
    T   = np.linspace(0.2, 4, K)
    UN  = np.zeros(K)
    CN  = np.zeros(K)
    MN  = np.zeros(K)

    # initial spin lattice
    S = rand.choice([-1, 1], size=(N,N))

    # reach thermal eq. in N_wait steps
    for j in range(N_wait):
        S = spin_flip(S, N, T[K-1], boundary)


    # Nested Loop over each temperature (outer loop)
    # and over N_run spin-lattice updates (inner loop)
    for k in range(K):
        # define energy, magnetization and temporary variables
        E       = energy(S, boundary)
        M_temp  = np.sum(S)/N**2

        dE      = 0
        dM      = 0
        M       = 0
```

```
        U_temp  = 0
        C_temp  = 0
        for i_ in range(N_run):
            # calc. energy, mean energy and mean of energy squared
            E           += dE
            U_temp      += E/N_run
            C_temp      += E**2/N_run

            # Do the spin update according to the MMC Ising model
            i, j    = (rand.random(size=2)*N).astype(int)
            r       = rand.random()
            dE      = energy_diff(S, N, i, j, boundary)
            q       = np.exp(-dE/T[K-1-k])

            # update spin lattice, the magnetization difference dM
            # and energy diff. dE
            if q > r:
                S[i, j] *= -1
                dM      = 2*S[i, j]/N**2
            else:
                dE = 0
                dM = 0

            # calculate mean magnetization
            M_temp      = M_temp + dM
            M           += M_temp

        # Fill the arrays for U and C and |M| for each temp.
        UN[K-1-k] = U_temp/N**2
        CN[K-1-k] = (C_temp - U_temp**2)/T[K-1-k]**2/N**2
        MN[K-1-k] = np.abs(M)/N_run
    return UN, CN, MN


# Generate the data and save everything in a folder
def gen_data(boundary):
    N_arr           = np.array([10, 50, 100])
    N_samples_arr   = np.array([1000, 10000])

    def text_rep(a, b, c, bond):
        text = "Computational/2D_Data/2D_x_qN_zNS_K.npy"
        text = text.replace("x", str(a))
        text = text.replace("q", str(b))
        text = text.replace("z", str(c))
        text = text.replace("K", str(bond))
        return text


    for N in N_arr:
        for N_s in N_samples_arr:
            UN2, CN2, MN2 = calc_U_C(N_s, N, boundary)
            np.save(text_rep("U", N, N_s, boundary), UN2)
            np.save(text_rep("C", N, N_s, boundary), CN2)
            np.save(text_rep("M", N, N_s, boundary), MN2)


# generate data ...
gen_data("per") #... for periodic boundaries
gen_data("free") #... for free boundaries
```

B.2 Generate Plots

```
import matplotlib.pyplot as plt
import numpy as np

# For fancy plots :)
plt.rcParams["text.usetex"] = True
plt.rcParams["font.family"] = "times new roman"
plt.rcParams["font.size"]   = "18"

def setup(title, xlabel, ylabel):
    plt.figure(figsize=(16, 9))
    plt.title(title)
    plt.xlabel(xlabel)
    plt.ylabel(ylabel)
    plt.xticks()
    plt.yticks()
    plt.locator_params(nbins=6)
    plt.grid()

# Theoretical expectations for magnetization
def MN_theo(T):
    T_c = 2/(np.log(1+np.sqrt(2)))
    return np.heaviside(T_c-T , 0)*np.abs((1-1/(np.sinh(2/T))**4))
    **(1/8)

# load the datasets
def dataset(N):
    def load_data(a, b, c, boundary):
        text = "2D_Data/2D_x_qN_zNS_K.npy"
        text = text.replace("x", str(a))
        text = text.replace("q", str(b))
        text = text.replace("z", str(c))
        text = text.replace("K", str(boundary))
        return np.load(text)


    def create(name):
        arr = []
        Ns_arr  = [1000, 10000, 1000, 10000]
        bounds  = ["per", "per", "free", "free"]
        for i in range(4):
            arr.append(load_data(name, N, Ns_arr[i], bounds[i]))
        return np.array(arr).reshape(-1, 20)

    U = create("U")
    C = create("C")
    M = create("M")
    return [U, C, M]


# Plot the data
def plot_data(data, title, x_axis, y_axis, magn, heat, Cmax, name):
    lw_data = 4
    lw_theo = 5

    text = "2D_Plots/2D_Z_X.pdf"
    text = text.replace("X", str(N))
    text = text.replace("Z", name)

    T       = np.linspace(0.2, 4, 20)
    T_theo  = np.linspace(0.2, 4, 1000)
```

```
    T_c      = 2/(np.log(1+np.sqrt(2)))

    Ns_arr = ["10^3", "10^4", "10^3", "10^4"]
    bounds = ["periodic,", "periodic,", "free,       ", "free,       "
    ]


    setup(title, x_axis, y_axis)
    if magn:
        plt.plot(T_theo,  MN_theo(T_theo), color="black", ls="--",
                linewidth=lw_theo, label="Theory")
    if heat:
        plt.vlines(T_c, 0, Cmax, color="black", ls="--",
                linewidth=lw_theo, label="$T_c$")
    for i in range(4):
        plt.plot(T, data[i], linewidth=lw_data, marker="o",
                markerfacecolor='white', label=r"X".replace("X",
                bounds[i]) + r" $N_S=Y$".replace("Y", Ns_arr[i]))
    plt.legend()
    plt.savefig(text)


# Define the parameters that are used for the different plots
N_arr       = [10, 50, 100]
title_arr   = ["internal energy per Spin, N=X", "Specific Heat per
    Spin,
                N=X", "Absolute Magnetization per Spin, N=X"]
y_arr       = ["U/N$^2$", r"C/N$^2$", r"|M|/N$^2$"]
magn        = [False, False, True]
heat        = [False, True, False]
Cmax        = [1.3, 2.5, 2.6]
name        = ["U", "C", "M"]

# Loop over the different systems and create all resulting plots
k = 0
for N in N_arr:
    data = dataset(N)
    for i in range(3):
        plot_data(data[i], title_arr[i].replace("X", str(N)),
                "T", y_arr[i], magn[i], heat[i], Cmax[k], name[i])
    k += 1
```