# TFY4235: Exam

Karl Kristian Ladegård Lockert

May 8, 2020

## Abstract

Exam solutions for TFY4235: Computational physics spring 2020. The code for the tasks are attached, and also available in an open GitHub repository[1].

## 1 Assignment 2

The final report for Assignment 2: Biased Brownian motion are attached together with code for the project. The core functionality of the program is written in Python, with heavy utilisation of compiled packages, as well as functionality for compiling python code. Most plots and development have been done in Jupyter Notebooks, which can be found in the Notebooks folder. If you do not have a Notebook viewer available, the very same files can be found and viewed in my GitHub repository. These files are, however, much more experimental and less documented, as they are created for very specific tasks or testing. The repository also contains additional plots, but these should be looked upon with great caution; if they are not included in the report there is no guarantee that the represent actual results rather than tests for different functions.

[1] https://github.com/kklocker/NumFys

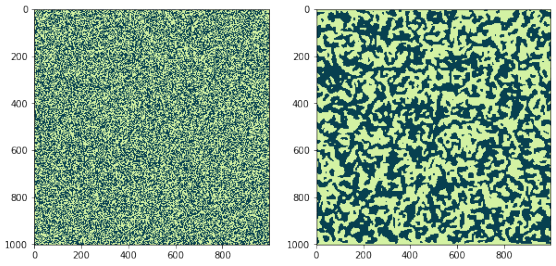


Figure 1: Initial and Low-$T$ states for a $1000 \times 1000$-lattice.

## 2 Ising model

The entirety of this project is written in Python 3.7 with testing in Jupyter Notebooks[2]. In the attached code you will find that most (if not all) functions has an "@njit"-decorator in front of its definition. This is a trick for drastically speeding up Python, which is an interpretive language, and thereby slow by default. "jit" is short hand for "Just-In-Time-compilation". The trick is that the functions will be compiled the first time calling them, and provides C-like speeds after compilation.

From earlier computations of the Ising model, I know how the low-T states tends to be for ferromagnetic coupling, and the implementation reproduces states that look familiar, shown in fig. 1. This is not important, but it is helpful to view the states as a guide for knowing if the implementation is on the right track. Letting $J \to -J$ leads to a checker board pattern

[2] See previously stated help for viewing these files!

for low $T$ (see fig. 16), which is expected in the antiferromagnetic case. Note that I have implemented the boundary conditions following the indices presented in the exam paper, meaning that the directions $x, y$ in my implementation are reversed from the description in ref. [1]. Also note that i will refer to the "original algorithm" multiple times, meaning the algorithm where the ensemble average is taken over $H_{++}$, and not $H'$ as the method they used in the paper.

### 2.1 Original Algorithm

The first three days of the home exam have been spent debugging erroneous results produced by the original Mon-Jasnow algorithm. The main challenge is computing the intermediate quantity

$$A = \left\langle e^{-\frac{H_{+-}-H_{++}}{T}} \right\rangle_{++} \tag{1}$$

to obtain

$$\tau = -\frac{T}{N} \ln A. \tag{2}$$

This is done by generating states in the $(++)$-ensemble and computing the energy difference in these states. Monte Carlo- simulations have the advantage that the Boltzmann weight for the statistical average is already incorporated in the generation of states, provided enough states are generated. We can thus compute a "normal" average of the sampled states. Discussing with teaching assistants have both been helpful, but also a bit frustrating due to miscommunications on both parts. There has been good clarification in what to expect from the results, but the interpretation of the ensemble average showed to be trickier to communicate. Including the mixed messages of the mysterious factor 2, which at the start was a relief giving better results for $\tau$, but was later changed. After discussions with several other students, it seems as many have interpreted the Hamiltonian differently. My interpretation of the $H_{++}$ and $H_{+-}$-systems is that the fixed spins are never summed over as localised spins, and only contribute as nearest neighbours to the spins adjacent on the interior side of the bulk. There has been a lot of errors and head scratching as to why my result were initially wrong, but in the end, I finally found yet another bug which when solved gave more expected results. In figs. 2 and 3 the resulting interfacial tension is presented for the original implementation. The shape is reminiscent of that in ref. [1], and $T_C$ appears to be correct. There is a clear

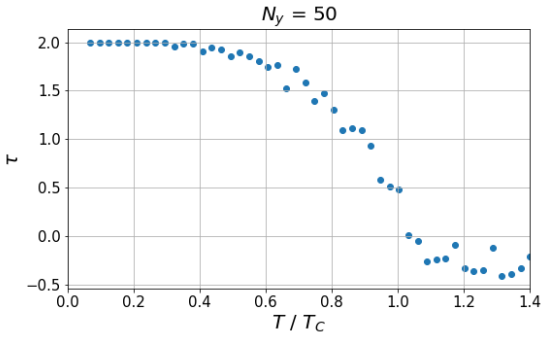


Figure 2: The original Mon-Jasnow algorithm. For $N_y = 50$. The simulation was done with 10000 sweeps of $N^2$ spin-flip tries.

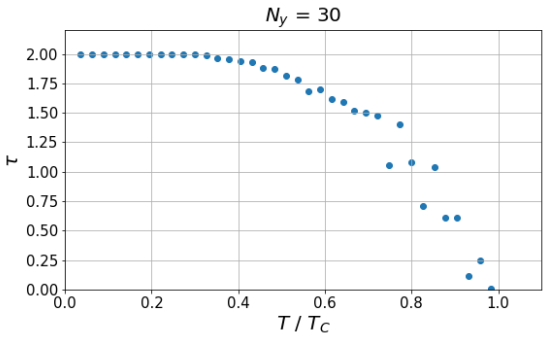


Figure 3: The original Mon-Jasnow algorithm. For $N_y = 30$. The simulation was done with 30000 sweeps of $N^2$ spin-flip tries.

transition at $T/T_C = 1$. I started implementing the Klein Bottle and torus system before getting decent result for the $H_{++}$-system, and immediately got good initial results, as we shall see later. How is this implemented? The basic idea is to follow the description in ref. [1] for the boundary conditions. The metropolis algorithm for one Monte-Carlo "sweep" follows the pattern described in Algorithm 1.

```
Initial state {σ}, T;
for iterator = 1 : N^2 do
    Pick random indices i, j;
    Compute W = e^{-βΔH} by
     flipping spin at i, j;
    Draw random (uniform)
     r ∈ [0, 1];
    if W ≥ r then
        Accept flip;
    else
        Reject flip;
    end
end
```

**Algorithm 1:** Basic idea of metropolis algorithm.

Although this algorithm describes the main idea behind the implementation, there are some special cases that have to be handled. For example, for small temperatures, the exponential might be too large for the computer to be able to interpret the result as something different from infinity. These states must be discarded in the average, because they will shift the results drastically. Although successful implementations of the Hamiltonian for the different systems are present in the code, they are never computed explicitly. This is because the relevant quantities in the algorithm, say $\Delta H$ can for the different Hamiltonian's be expressed compactly as $\mathcal{O}(N)$ functions, which is favourable to the double $\mathcal{O}(N^2)$ summation for the complete Hamiltonian. For example, the quantity $H_{+-} - H_{++}$ in eq. (1) is just two times the sum of the next to last row in $H_{++}$, as stated just after Equation (2) in ref. [1].

The implementation of the original algorithm concerning the interfacial tension was based upon the ensemble average for the $H_{++}$-system. The expectation value follows

$$\langle A \rangle_{++} = \frac{1}{Z_{++}} \sum_{\{\sigma\}_{++}} A e^{-\beta H_{++}}, \qquad (3)$$

where $\{\sigma\}_{++}$ is a sloppy notation for the states generated in the $(++)$-ensemble, with a probability following the Boltzmann distribution. Since the metropolis Monte-Carlo (MC) algorithm generates these states, we can get expectation values by computing the normal mean of a sufficiently large sample size. In the implementation of the sampling for $\tau$, I have defined multiple quantities to tune the sampling. A "sweep" is $N^2$ attempts of flipping a spin, as described in Algorithm 1. "Skips" is the number of sweeps between each sample. Starting in an initial configuration with all spins pointing "up", a sufficient number of skips are helpful for more accurately describing the equilibrium state for a given temperature $T$. A "run" is the number of times the lattice is to be reset to its initial configuration. So for each run, we sample every "skip" sweep consisting of $N^2$ attempts to flip random spins.

### 2.2 Extended Mon-Jasnow

The extended algorithm is very similar in form as the original: We generate states in some ensemble, this time the ensemble where $H = H_t$ is the Ising Hamiltonian with periodic boundary conditions in both directions. The interfacial line tension is still described by eq. (2), but the Hamiltonians and average of eq. (1) are differnt. The implementation of the torus - and Klein Bottle boundary conditions is added as an extension to previously implemented functions. Again, the Hamiltonian for the different

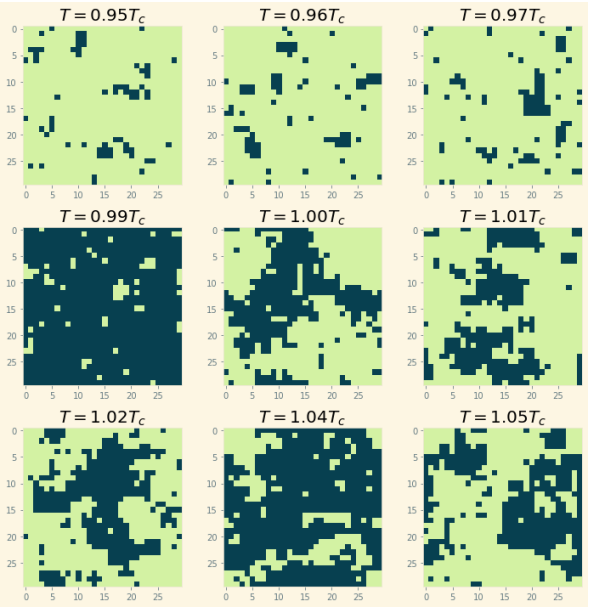


Figure 4: Uncorrelated states of a $30 \times 30$ system governed by the torus Hamiltonian after some hundred sweeps, near $T_C$.

systems are implemented, but never used in the sampling process. They are, however, used to visualise the states of the system, ensuring correct implementation. In fig. 4 different uncorrelated states are shown for temperatures close to $T_C$. The states indicate something happening at $T = T_C$, which is promising preliminary results. Even if the Hamiltonian is correct, doing a double summation at each iteration to compute the energy of the system is not wanted. Since we in this case is weighting the average over states in the torus-ensemble, $\langle \cdot \rangle_t$, we have to find the energy associated with flipping a spin using $H_t$. This is a simple sum over four lattice sites, since there are no difficult boundary conditions apart from a modulo system size at each edge[3]. The slightly more challenging quantity is $H_k - H_t$. By testing different approaches, I found that the energy

[3]See the function "spin_flip_energy" in "lattice_utils.py"

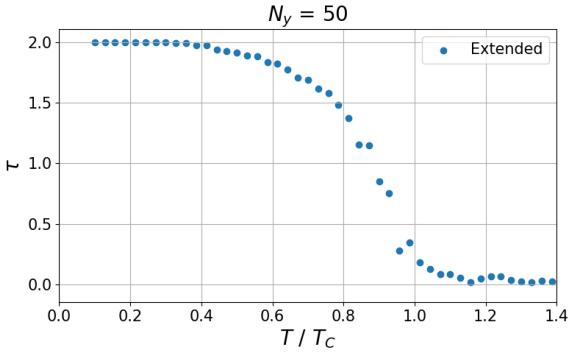


Figure 5: The extended Mon Jasnow algorithm for 10000 sweeps on a $50 \times 50$ lattice, sampling every third sweep after 50 initial sweeps.

difference could indeed be written as a $\mathcal{O}(N)$ summation, and found it was given by

$$H_k - H_t = \sum_{i=0}^{N-1} (\sigma_{0,N-1-i} + \sigma_{0,i})(\sigma_{N-1,i}). \quad (4)$$

This is implemented in the function "energy_diff". Cutting of an order of magnitude is important, since we need to compute the energy difference for every sample, ideally a very large number. Thus all the tools for efficient sampling are implemented, and the results look promising. We can more clearly see that there is a phase-transition at $T = T_C$. By increasing the lattice size, we get consistent results, shown in fig. 5 where $\tau$ is plotted for the same parameters as fig. 2, making a direct comparison possible. We can show that the phase transition happens at a steeper rate by doing runs for different $N$, and is shown in fig. 6 where evenly solutions for evenly spaced $N$ from 16 to 192 are plotted against temperatures close to $T_C$.

### 2.3 Low-$T$ scaling

We now consider the scaling of $\tau$ in the limit where

$$t \equiv \frac{T_C - T}{T_C} \to 0^+ \quad (5)$$

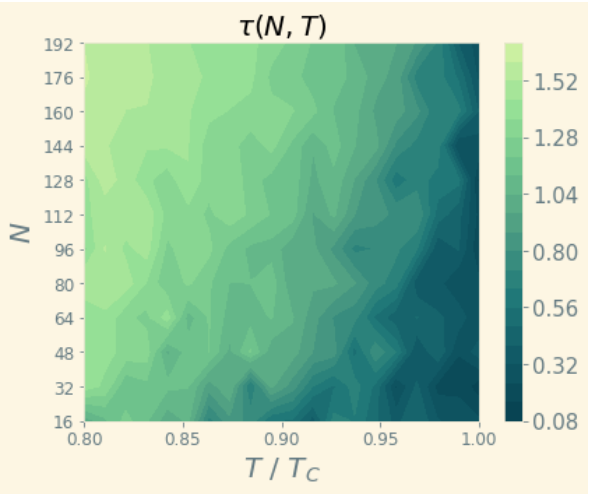


Figure 6: Contourplot of the interfacial tension in the extended Mon Jasnow algorithm. The phase transition happens more rapidly for higher N. These results were based on one run for each temperature consisting of 20000 sweeps, for 20 equally spaced temperatures in the range $T \in [0.8T_C, T_C]$.

Since I am computing independent solutions for varying $N$, i figured that the computation might be done in parallel, saving time. This was done using a framework called Dask [2]. This made able to perform 1 million sweeps for all lattices in the range $N \in [10, 32]$ in reasonable time. For this and the next sections, most of the programming has been too specific to justify implementing own functions for different tasks, and I have resorted to the use of Jupyter Notebooks for tweaking in plots and such. As of now, a sufficient amount of simulations for different parameters have been computed, and the core functionality of the program has done its job. This run has been the most successful one, and is shown in fig. 7. Based on the scaling ansatz $\Sigma(z) \sim z^{-\mu}$ as $z \to 0^+$, we expect

$$\lim_{t \to 0^+} \tau = \tau_0 t^{\mu} (N^{\frac{1}{\nu}} t)^{-\mu} = \tau_0 N^{\frac{-\mu}{\nu}} \qquad (6)$$

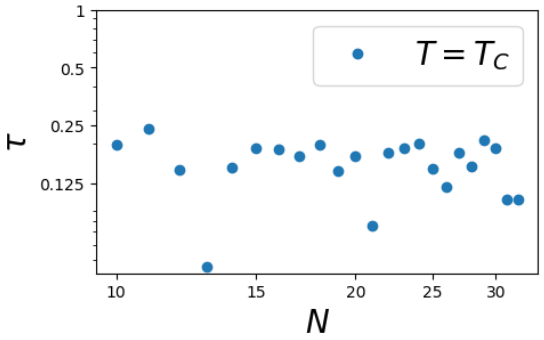


Figure 7: $\tau$ against $N$ at critical temperature. Average of every third sweep of a total of 1000000 sweeps.

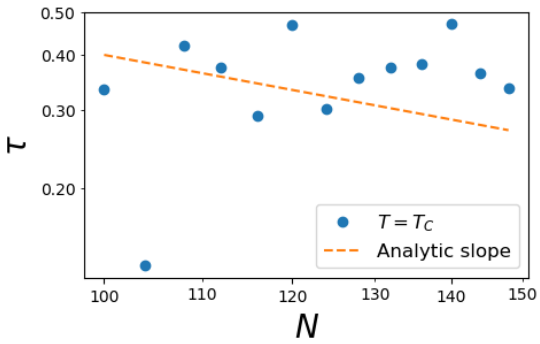


Figure 8: Scaling of $\tau$ against $N$ for larger lattices. 8000 sweeps per lattice. **NB**: The vertical placement of the orange dashed line is arbitrary, and just for visual aid comparing slopes.

Inserting $\mu = \nu = 1$ for the 2D Ising model, we should get $\tau \sim 1/N$ for temperatures close to Tc. This should appear as a straight line with a slope of -1 in a log-log-plot. In fig. 8 the same results are shown together with a line depicting the analytic slope of $\tau$ against $N$, but for far fewer sweeps, resulting in larger uncertainty. In fig. 9, essentially the same results are shown to be consistent for large ranges of $N$. These results are not in agreement with the expected slope of $-1$. There might be many reasons for this, one likely reason being in an erroneous implementation. Another possible explanation of

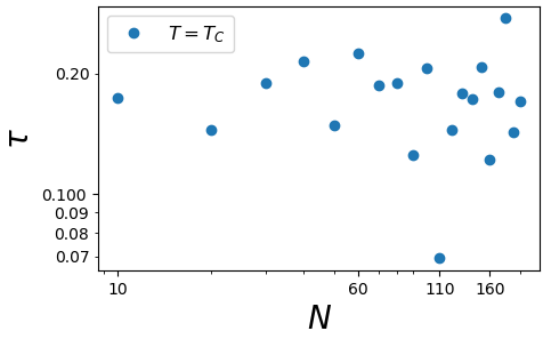


Figure 9: Scaling of $\tau$ for a large range in $N$. Every 10th $N$ from 10 to (including) 200 is computed with 100000 sweeps $N$.

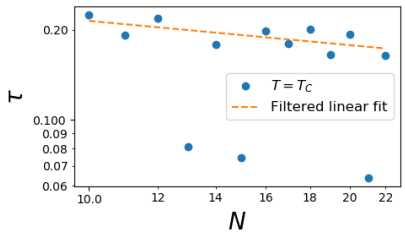


Figure 10: $\tau$ at $T = T_C$ for small $N_y$, with 1000000 sweeps. In addition a filtered linear fit with a slope of -0.9161, where the apparent outliers have been filtered out from the fitting.

the poor results are that the system is incredibly sensitive at $T = T_C$, and things near this limit is hard to compute exactly. A finale (call it desperate) attempt was made for smaller lattices in the range $N = 10$ to $N = 32$ with 1 million sweeps. This is shown in fig. 10. As we can see, there are a couple of outliers in the computation, breaking a seeming trend. If one were to remove these outliers, one would get (by linear fitting of the logarithms) a slope of $-0.9161(4)$. **NB: messing with the data in this way is a very bad way of doing things.**

A run of 1 million sweeps and 20 temperature points near $T_C$ for lattices ranging from 10 to 20 was computed to further investigate

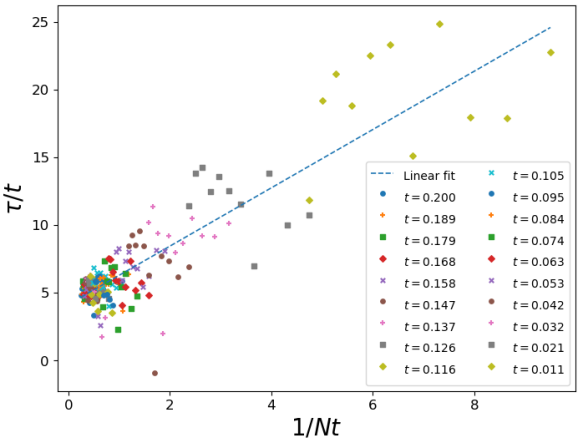


Figure 11: Scatter plot for multiple values of $t$ near $T = T_C$, together with a linear fit of all scattered plots.

the scaling of $\tau$ as a function of $N$ and $T$. By plotting $\tau/t$ as a function of $1/(Nt)$, and computing a linear fit to all pairs of points, an estimate of $\tau_0$ can be made. By inserting 0 in the interpolating function, $\tau_0$ is found to be

$$\tau_0 = 4.1322155(1). \tag{7}$$

This is shown in fig. 11. Note that the $t = 0$ cases have been removed due to infinities occurring in the numerical case[4] By shifting the x-axis of fig. 11 and scaling the axis to a log-log-plot, we get a result more reminiscent of that in ref. [1], shown in fig. 12.

Finally, we make a few comparisons between the original and extended algorithm. In fig. 13 $\tau$ is plotted with equal parameters. Notice how the original method seems to break down for $T > T_C$. Although this is stated in ref. [1], we can add the following physical interpretation: Above $T_C$, thermal fluctuations will begin to dominate the spin-flip process. Since two of the edges are fixed with spin up (down on the (+-)-lattice), the interface will to a much lesser

[4] In the analytic case these infinities would be cancelled by the scaling of $\tau$.

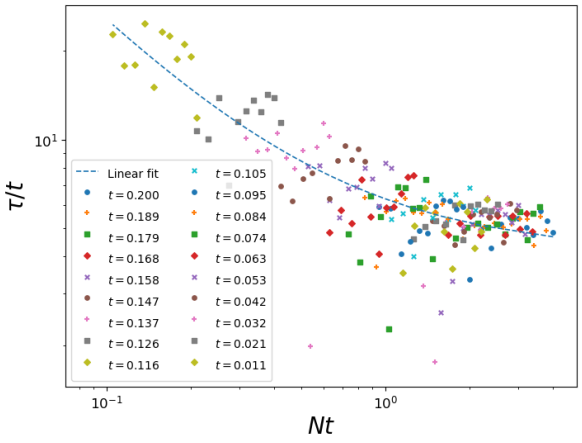


Figure 12: Log-log plot of fig. 11 with inverted x-axis.

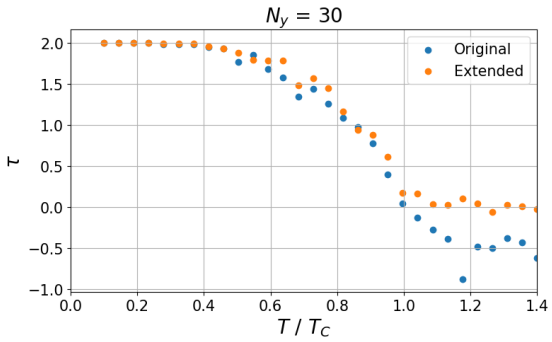


Figure 13: Comparison using 10000 sweeps, sampling every third sweep for $N_y = 30$.

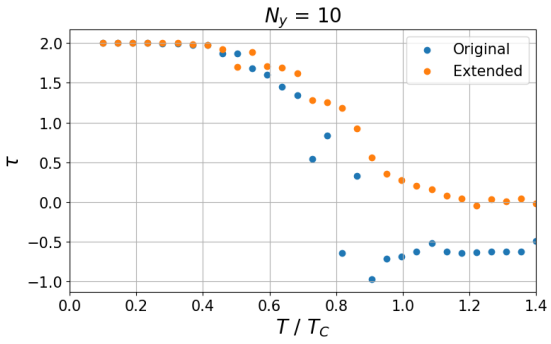


Figure 14: Comparison using 10000 sweeps, sampling every third sweep for $N_y = 10$.

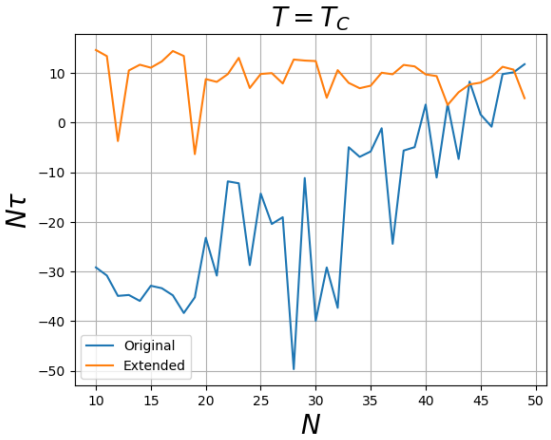


Figure 15: Comparison of the scaling of $\tau$ at $T = T_C$. The sampling consisted of every third sweep of a total of 100000 sweeps for each $N$.

degree act as an infinite system. Having constant spin up on the edges will effectively be the same as having infinitely strong localised magnetic fields, which is far from the analytic results. These will intuitively affect the systems for smaller values of $N$, while the periodic boundary conditions for the torus will compensate this to a much higher degree. We thus suspect the extended algorithm to be more stable for lower values of $N$. This does indeed seem to be the case, as shown in fig. 14 for $N = 10$. Here we see that the extended algorithm tends to zero for $T > T_C$, while the original algorithm is much more unstable, and does not even go to zero at once after passing $T_C$. This is further shown in fig. 15 by comparing $N\tau$ to $N$ for the two methods with otherwise equal parameters. As previously discussed, the extended algorithm is much more stable at $T_C$ than the original algorithm. This might be attributed to the interfaces introduced. In the original algorithm, the spins at the interface are not allowed to flip, while this restriction is absent in the extended scheme, more accurately describing the importance of thermal fluctuations in

an infinite lattice, even for small $N$.

We have in this task shown that both the original and extended Mon Jasnow algorithms predict phase transitions at $T = T_C$, and that the extended algorithm is much more stable for smaller lattice sizes, and is thus much more suitable for large computations since the algorithm scales with the lattice size. In fact, a (very) crude estimate of the complexity of the algorithm would be something like

$$N_{\text{sweeps}} \left( \mathcal{O}(N^2) + \frac{1}{N_{\text{skips}}} \mathcal{O}(N) \right) \sim \mathcal{O}(N^2) \tag{8}$$

in the asymptotic limit, so getting better results for smaller lattice sizes is very favourable. The $\mathcal{O}(N^2)$-part is from each sweep, and the $\mathcal{O}(N)$ is from each computed energy difference. This solution has been written only in Python, with heavy utilisation of clever manipulations on a normally "slow" programming language which can provide C-like speeds for function calls, yet maintain the advantages of rapid development. The exam have given new insight in the Mon Jasnow algorithm as an implementation of the Monte Carlo metropolis algorithm, both its limitations and the importance of correct sampling for statistical averages. It has been quite challenging, but i hope that this "report" does justice to the many failures encountered and debugged during the examination period, as well as the many successes. Lastly I have (for fun) included how the system evolves for the antiferromagnetic case in fig. 16.

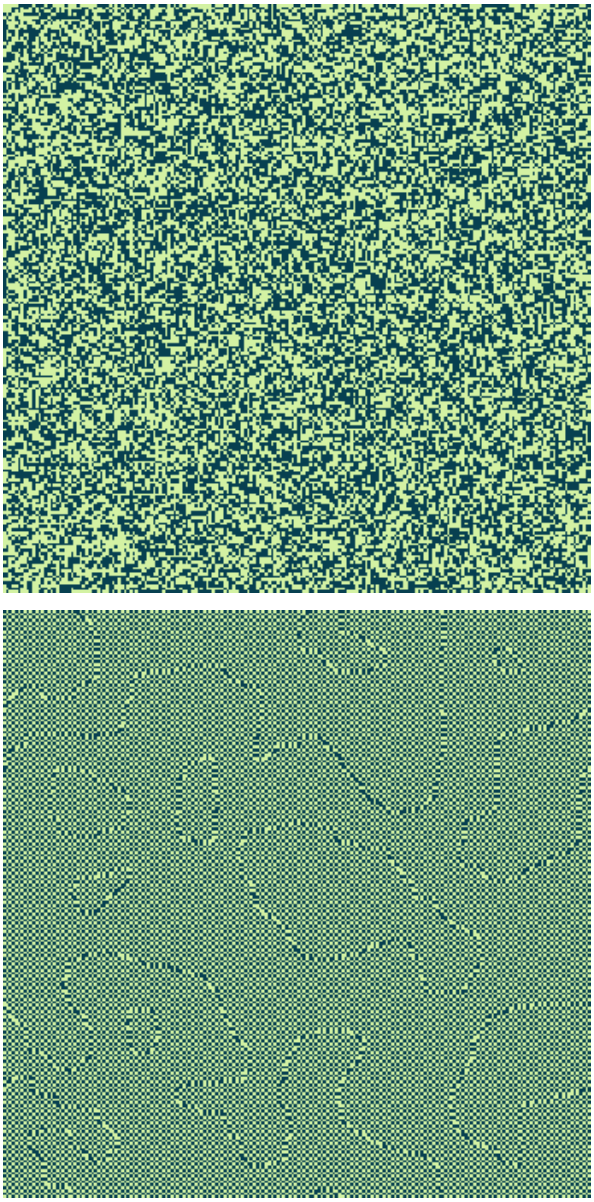

Figure 16: Initial state and low T -state after 100 sweeps in the antiferromagnetic case $J = -1$ for a $200 \times 200$ lattice.

## References


[1] K. K. Mon and David Jasnow. Direct calculation of interfacial tension for lattice models by the monte carlo method. *Phys. Rev. A*, 30:670–673, Jul 1984.

[2] Dask Development Team. *Dask: Library for dynamic task scheduling*, 2016.