On phase transitions in magnetic systems

A classical approach

Sakarias Frette
Anders Vestengen
William Hirst

November 22, 2020

1 Introduction

In this paper we aim to study a phase transition in a magnetic lattice through numerical computation of the classical case, without quantum mechanics. To do this we are using the widely popular Ising model, because when set up for our use-case it exhibits a sudden phase-transition at a critical temperature. This is ideal as we are looking to simulate the sudden change of magnetization that happens when a magnetic structure is subjected to rising temperatures, causing the structural magnetic lattice to become unstable, and the unison magnetic field to be disrupted. Specifically we are looking to numerically replicate the work of Lars Onsager in 1944[2], when he discovered the critical temperature of a two dimensional lattice, using the Ising model.

2 Theory

2.1 Monte Carlo simulation and metropolis.

A Monte Carlo method is used to simulate a system when a variable is unknown (for example) due to a random interference, making a calculation on the system impossible or very hard. The method is to use many random values to replace the unknown value, and use the average of all the calculations as an estimate. Given enough samples (random numbers) the estimate improves. The algorithm can be explained as follows; given a function f and an unknown variable X, we can give an estimate of f(x) by as

$$f(x) \approx \frac{1}{N}\sum_{i}^{N} f(x_i)$$

, where x_i is a random number and N is the number of samples.

The Metropolis algorithm is a type of Monte Carlo simulation. The algorithm is used when the probability distribution, P for a state of a system, S (which dependent on many elements), is unknown. The idea is to find the states that would be generated if P was known (S_P) by using a Markov process. The process is to randomly chose an element in S and decide to change the state of that element dependent on a rate of acceptance, the previous state(hence Markov process) and an element of randomness. Given that certain requirements of A (like being proportional to P) and P are satisfied, if enough changes are made to the original state S, the state should converge towards S_P.

Therefore, even though we cannot know the probability distribution for a each element in a magnet, we can use the metropolis algorithm to simulate and find a realistic state (or at least a good approximation of one) for the magnet as a hole given enough cycles.

2.2 The Ising Model

Is a mathematical model for studying ferromagnetic properties in statistical mathematics. We will be skipping ahead to the two dimensional Ising model in this introduction. Consider a lattice of magnetically charges 'spins' each linked to each other in a two dimensional structure, like the figure below.

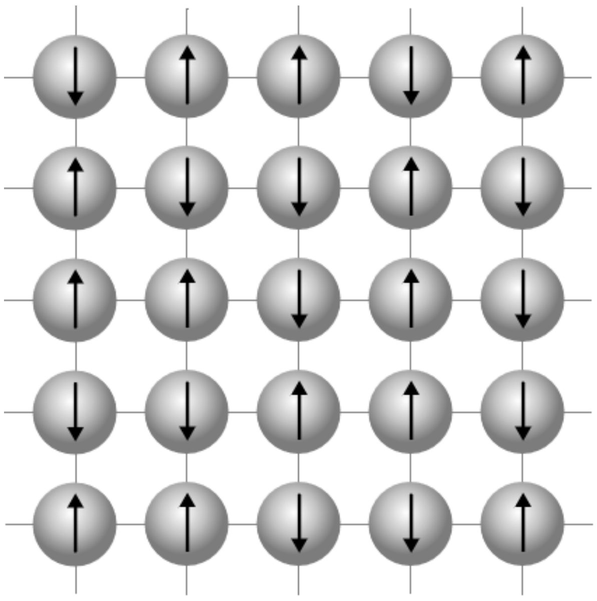


Figure 1: Two dimensional Ising model Source

Each spin can either point up or down, indicating its magnetic polarization. From this we can calculate the energy of the field using the Hamiltonian, by iterating over the lattice like so.

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

[1] The "kl" indicates we only sum over nearest neighbours to avoid duplicating energy. We then sum for number of spins (N). This also highlights a few things which we will be discussing later on when commenting on the code, such as the fact that a two dimensional model only has a limited number of energies, which can be precalculated. That's also were we're going to discuss the boundary conditions of our model.

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

[1]see Here for a more thorough discussion of the energy configurations. p.422

For Magnetization we simply iterate over the total number of spins. We don't have to take duplicates into account like we did for energy, because magnetization is an absolute value, and opposites will cancel each others contributions.

In this model, a phase transition happens when the temperature approaches the critical temperature T_c. In the Ising model, the critical temperature for a lattice of given size L^2 follows the following relation

$$T_c^{(L)} - T_c^{(L=\infty)} \propto aL^{-1/\nu} \tag{3}$$

2.3 Random numbers in numerical calculations

A random generated number (RNG) is very rarely truly random number, but are instead perfectly deterministic. Most random number generators are based on some algorithm to simulate randomness and repeat after a period of iterations. An example of such an algorithm could be to use the digits in an irrational numbers to simulate a "random" choice between 0 and 1. For each digit larger than 5 the generator would pick 1 and for each digit smaller than 5 the generator would generate 0. To most people this would seem random, but if you knew the algorithm and the irrational number, you could easily predict the result.

2.4 Statistical physics

In the canonical ensemble[1] we can calculate the expectation value for energy, $\langle E \rangle$, but we need a probability distribution in order to do so. This is given by the Gaussian distribution as

$$P_i(\beta) = \frac{1}{Z}e^{-\beta E_i} \tag{4}$$

where $\beta = \frac{1}{k_B T}$ is the inverse temperature in units of joules, E_i is a given energy for a microstate and Z is the partition function, given as

$$Z = \sum_{i=1}^{M} e^{-\beta E_i} \tag{5}$$

From the thermodynamic identities we have Heimholtz free energy given as

$$F = -k_B T \ln Z = < E > -T \left(k_B \ln Z + k_B T \left(\frac{\partial \ln Z}{\partial T} \right)_{N,V} \right) \tag{6}$$

This gives us the expectation value $< E >$ as

$$< E > = \sum_{i=1}^{M} E_i P_i(\beta) = \frac{1}{Z} \sum_{i=1}^{M} E_i e^{-\beta E_i} \tag{7}$$

This allowes us to calculate the variance of the energy as

$$\sigma_E^2 =< E^2 > - < E >^2= \frac{1}{Z}\sum_{i=1}^{M} E_i^2 e^{-\beta E_i} - \left(\frac{1}{Z}\sum_{i=1}^{M} E_i e^{-\beta E_i}\right)^2 \tag{8}$$

By the same token we find that the expectation value for the magnetization is given by

$$< M >= \frac{1}{Z}\sum_{i=1}^{M} M_i e^{-\beta E_i} \tag{9}$$

and the variance of magnetization as

$$\sigma_M^2 =< M^2 > - < M >^2= \frac{1}{Z}\sum_{i=1}^{M} M_i^2 e^{-\beta E_i} - \left(\frac{1}{Z}\sum_{i=1}^{M} M_i e^{-\beta E_i}\right)^2 \tag{10}$$

This gives us the following quantities heat capacity C_v and susceptibility χ. They are given as

$$C_v = \frac{\sigma_E^2}{k_B T^2} \tag{11}$$

and

$$\chi = \frac{\sigma_M^2}{k_B T^2} \tag{12}$$

3 Implementation

The programs used in this research article can be found on this Github address.

3.1 The code

The code is structured to have one large implementation, to which there are several smaller configurations able to run for specific use-cases. We find this kind of development advantageous, because it let's us develop faster towards a general goal. We also acknowledge this leads to some rewrites when finalizing, but feel the early speed of implementation makes up for it. For a more in-depth look at the code see the READ-ME on our GitHub page.

3.2 The algorithm

The two main algorithms for this research is the Metropolis algorithm and the Monte Carlo integration algorithm as shown below.

Listing 1: Metropolis algorithm

```
void solver::Metropolis(){
// loop over all spins
    for(int spin =0; spin < m_tot_spins; spin++){
      int ix = (int) (ran1()*(double)m_spins);
      int iy = (int) (ran1()*(double)m_spins);
      int deltaE = 2*m_smatrix(iy, ix)*
      (m_smatrix(iy, periodic(ix,m_spins,-1)) +
      m_smatrix(periodic(iy,m_spins,-1), ix) +
      m_smatrix(iy, periodic(ix,m_spins,1)) +
      m_smatrix(periodic(iy,m_spins,1), ix));
      if ( ran1() < m_w(deltaE+8) ) {
        m_smatrix(iy, ix) *= -1; // flip one spin and
        accept new spin config
        // update energy and magnetization
        m_M += (double) 2*m_smatrix(iy, ix);
        m_E += (double) deltaE;
        m_counter++;
      }
    }// End of the Metropolis function.
}
```

Listing 2: Monte Carlo V1

```
void solver::MonteCarloV1(){
    m_counter =0;
    // Monte Carlo cycles
    for (int cycles = 1; cycles <= m_mcs; cycles++){
        //Run Metropolis algorithm
        Metropolis();

        // update expectation values
        m_E_vals[cycles] = m_E;
        m_average(0) += m_E; m_average(1) += m_E*m_E;
        m_average(2) += m_M*m_M; m_average(3) += fabs(m_M);
        m_cycles = cycles;
        output();
    }
}// end function MonteCarloV1
```

3.3 Unit tests

In order to verify that our code works properly, we run unit tests. We chose three tests to ensure the validity of our results. First we check if the spins are randomly

ordered, meaning that the pseudorandom algorithm actually randomized the spins in the matrix. Because the distribution of spin up and spin down evens out more and more as we increase the amount of spins. Therefor we check if the average sum for a given lattice of 10 by 10 is larger than the average sum for a given lattice of 20 by 20. We are to expect the average sum to go zero as L^2, the amound of spins in a lattice, goes to infinity.
The second test checked if the energy variance increased with an increased temperature. From equation 8 we have that β decreases with an increase in temperature, which in turn increases the overall expectation value. This in turn increases the variance. The physical interpretation of this is quite simple. By increasing the temperature, we are adding energy to the system, which in turn increases the amount of microstates, and so we would then expect that the probability for a given energy would get more wide, as the standard deviation, the square root of the variance, gets larger.
The third and final test compares the numerical and analytical values for the expectation value of the energy and magnetization up to a certain threshold. The analytical expressions are given by equation 7 and 9, and we put a threshold of one thousandth of the analytical value.

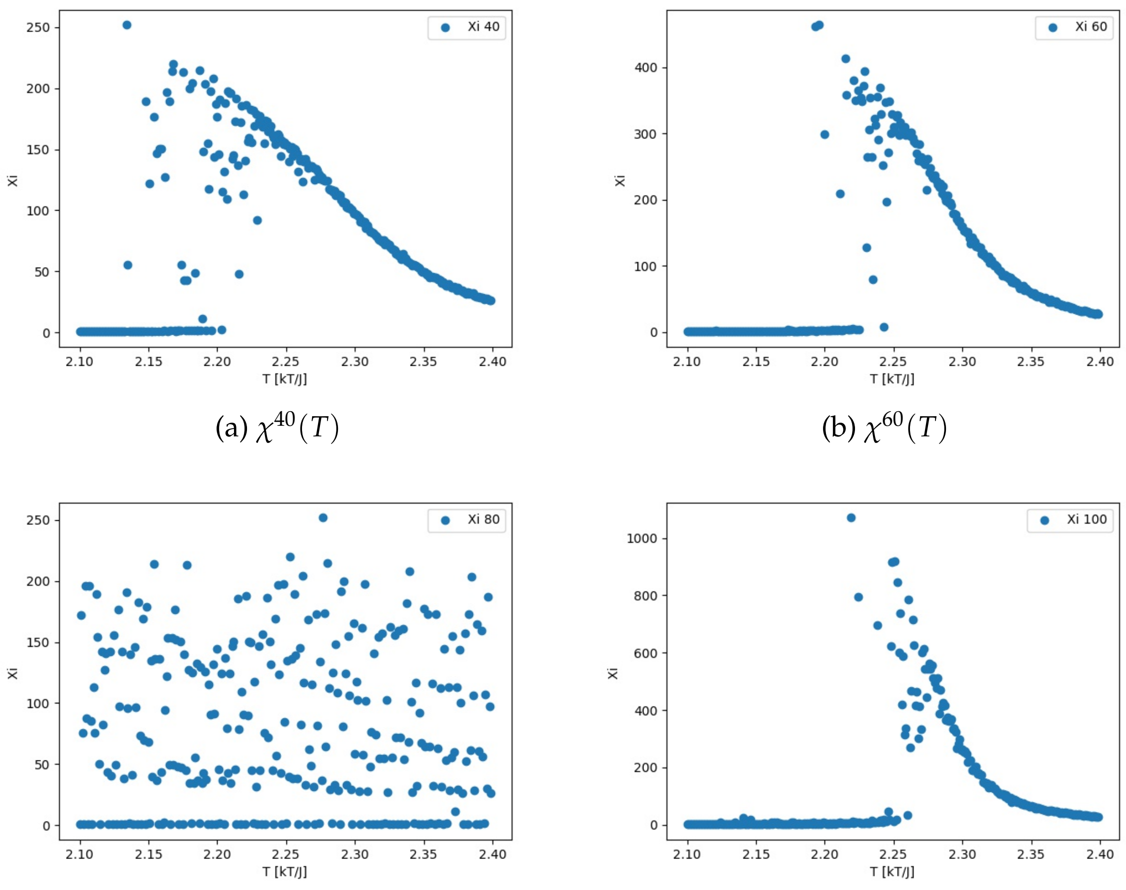
4 Results

	$\langle E\rangle$	$\langle \vert M\vert \rangle$	C_v	χ
Analytical values	-7.98	3.99	0.13	0.016

Table 1: The table shows the analytical values for the mean energy and magnetization, the heat capacity and the susceptibility. The values are calculated for T =1 and L=2.

Monte Carlo cycles	$\langle E\rangle$	$\langle \vert M\vert \rangle$	C_v	χ
10^4	-7.98	3.99	0.14	0.014
10^5	-7.98	3.99	0.12	0.013
10^6	-7.98	3.99	0.12	0.014
10^7	-7.98	3.99	0.13	0.016

Table 2: The table shows the numerical values for the mean energy and magnetization, the heat capacity and the susceptibility. The values are calculated for T =1 and L=2 and number of Monte Carlo cylces ranging from $10^4 \rightarrow 10^7$.



(a) $\chi^{40}(T)$ (b) $\chi^{60}(T)$

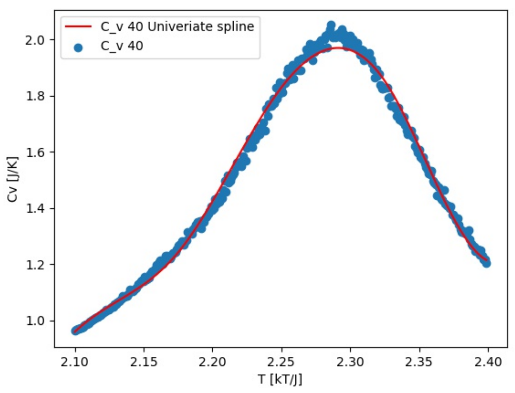
(c) $\chi^{80}(T)$ (d) $\chi^{100}(T)$

Figure 2: χ as a function of T in energy units, showing a move towards the critical temperature as L increases.

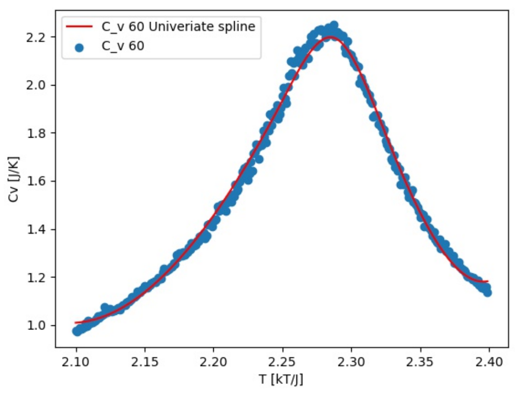
5 Discussion and analysis

5.1 Critical Temperature

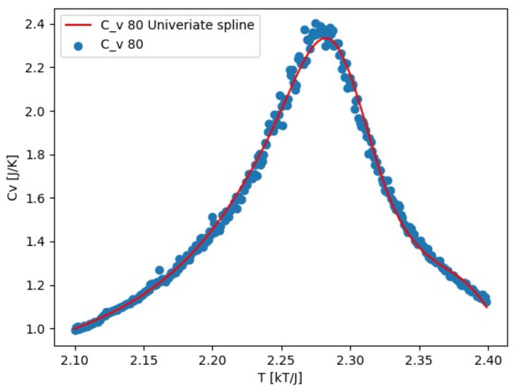
From our results in figure 3 and figure 2 we got a mean value for the critical temperature equal to 2.265, which is 0.004 off from what Lars Onsager found. The plots showing the susceptibility show the phase transition at the critical temperature as a drastic increase followed by a rough reciprocal inverse function form. This is shown also in figure 3 and figure 4 where the peak heat capacity marks the phase transition at the critical temperature.
Using equation 3, we found the mean value for a, and used it to find the mean value for the critical temperature at $L = \infty$. Now, we chose to use 10^6 Monte Carlo cycles with a Temperature $T \in [2.1, 2.4]$ with 300 steps. Had we increased the number of Monte Carlo cycles and temperature steps, we could assume better results. Having



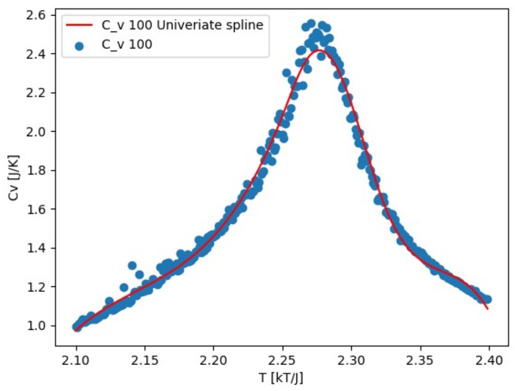
(a) $C_v^{40}(T)$



(b) $C_v^{60}(T)$



(c) $C_v^{80}(T)$



(d) $C_v^{100}(T)$

Figure 3: C_v as a function of T in energy units with univeriate spline interpolation

said that, we ran the calculations on a desktop with a Intel i7 4770k processor, and they took approximately 16,5 hours, so we can only speculate to the degree of which the data would be more precise. Another point to note is that we used Scipy's Univariate spline interpolation, which interpolated the curves with a given smoothness. In order to fit the data more accurately we tuned this parameter, but one can surely tune it even more than we did. This method has a residual function, given in the table below. We see here that the spline method gets less and less accurate with the

L	**Residual**
40	1.29
60	5.04
80	9.27
100	12.97

increase in spins, which is expected. From picture 3 we see that the data spreads more

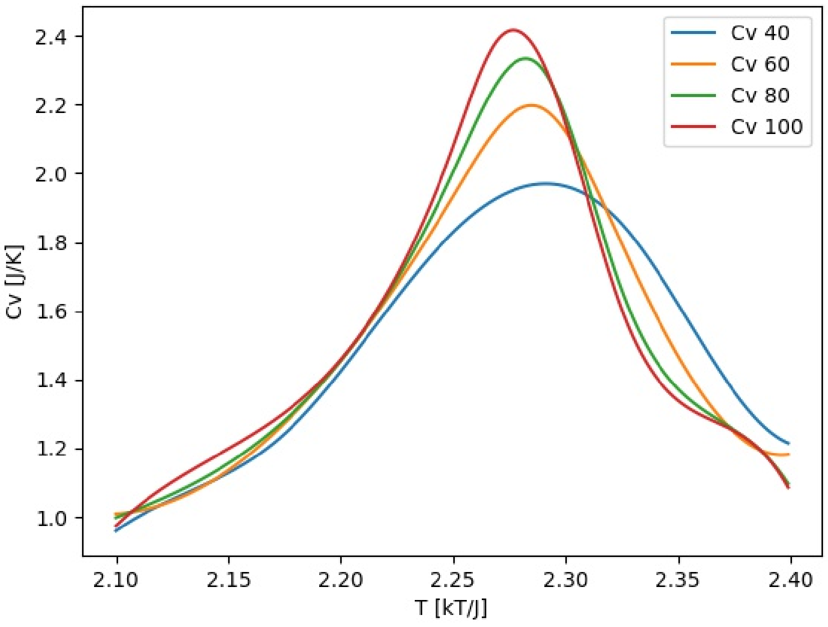
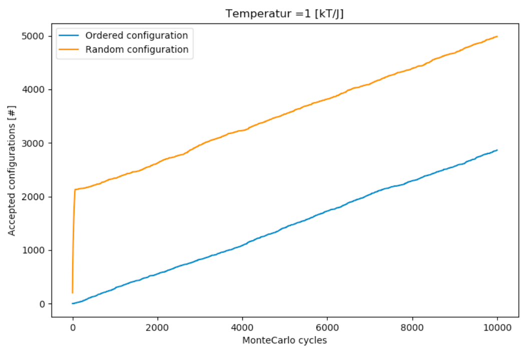
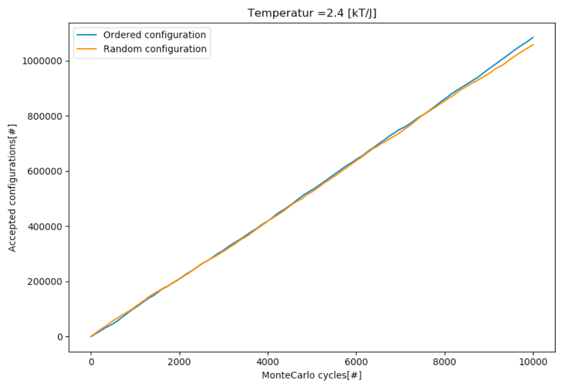


Figure 4: Univariate spline interpolation for the four latices as a function of T in energy units.



(a) The figure shows the number of accepted configurations as a function of Monte Carlo cyles. Both of the curves were made with T=1 and L=20. The blue curve shows ordered orientation(all up) as starting configuration and orange shows random.



(b) The figure shows the number of accepted configurations as a function of Monte Carlo cyles. Both of the curves were made with T=2.4 and L=20. The blue curve shows ordered orientation(all up) as starting configuration and orange shows random.

Figure 5: Energy and magnetization

out when we increase L. As mentioned earlier we could minimize this by increasing

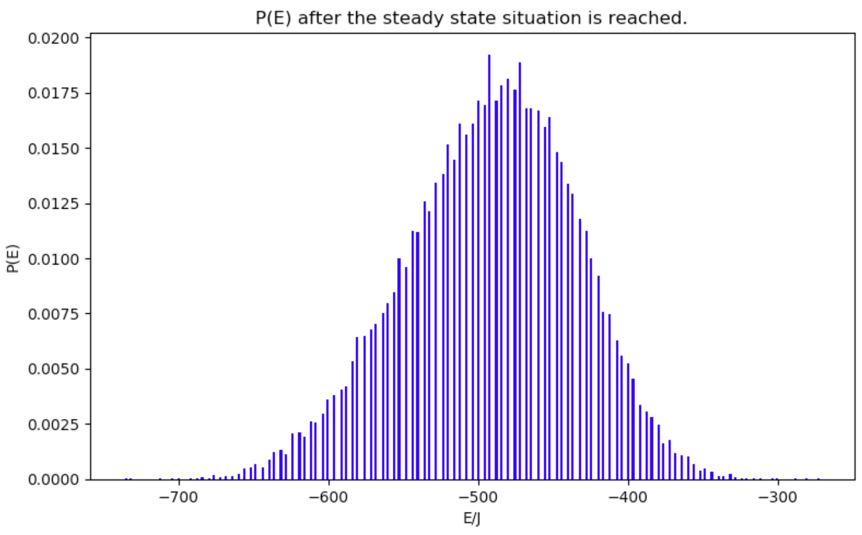


Figure 6: The figure shows a histogram of the Energy change per with 300 bins between the lowest and the largest E value.

the number of cycles, narrowing the temperature domain and increasing the amount of steps.

5.2 Unit tests

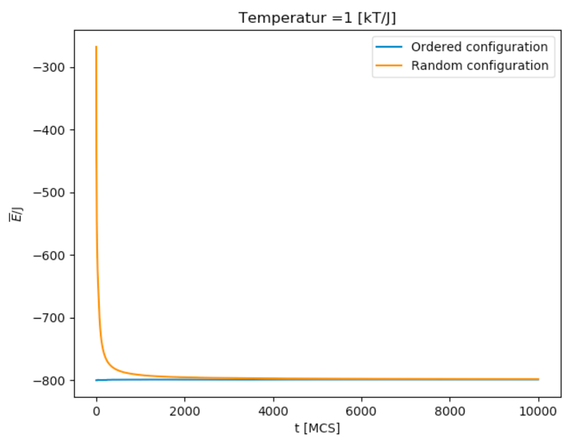
6 Conclusion

7 References

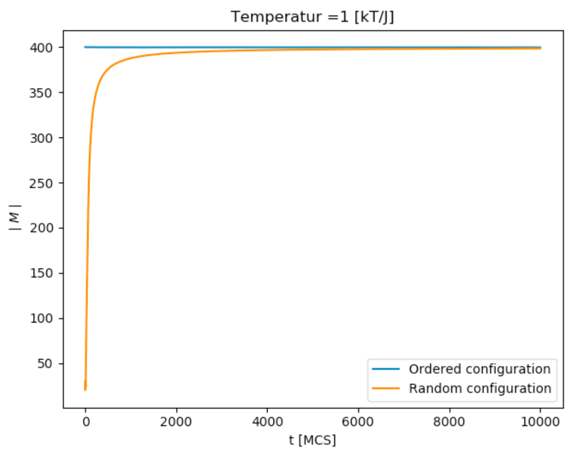
References

[1] Morten Hjort Jensen. *Statistical physics*. URL: https://github.com/CompPhysics/ComputationalPhysics/blob/master/doc/HandWrittenNotes/NotesSeptember3.pdf. (accessed: 22.11.2020).

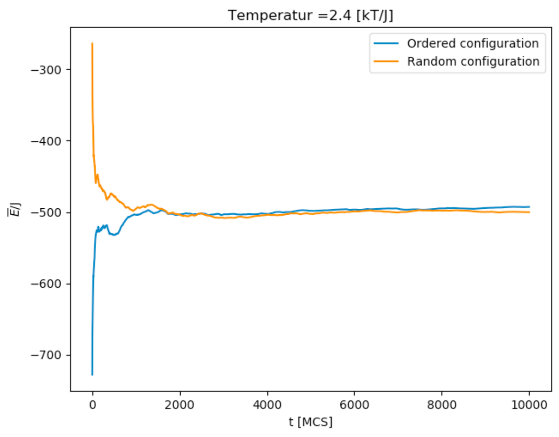
[2] Lars Onsager. "Crystal statistics. I. A two-dimensional model with an order-disorder transition". In: *Physical Review* Series II.65 (3–4) (1944), pp. 117–149. DOI: https://ui.adsabs.harvard.edu/abs/1944PhRv...65..117O/abstract.



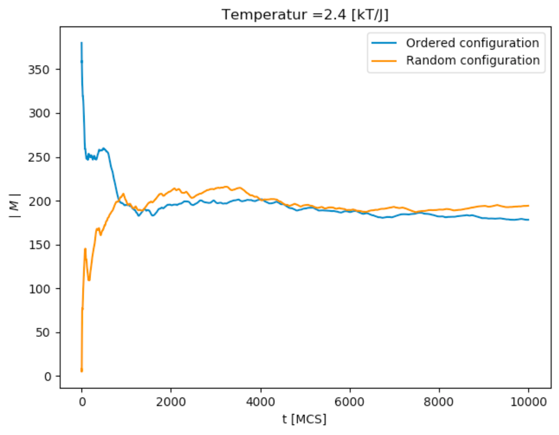
(a) The figure shows the mean Energy as a function of time/Monte Carlo cycles. The curves were generated with L=20 and kT/J=1.The blue curve shows ordered orientation(all up) as starting configuration and orange shows random.



(b) The figure shows the absolute value of the mean magnetization as a function of time/Monte Carlo cycles. The curves were generated with L=20 and kT/J=1. The blue curve shows ordered orientation(all up) as starting configuration and orange shows random.



(c) The figure shows the mean Energy as a function of time/Monte Carlo cycles. The curves were generated with L=20 and kT/J=2.4.The blue curve shows ordered orientation(all up) as starting configuration and orange shows random.



(d) The figure shows the absolute value of the mean magnetization as a function of time/Monte Carlo cycles. The curves were generated with L=20 and kT/J=1. The blue curve shows ordered orientation(all up) as starting configuration and orange shows random.

Figure 7: Energy and magnetization