UNIVERSITY OF PISA, PHYSICS DEPARTMENT

Simulation of 2-dimensional Ising model

Marco Cocciaretto, Miriam Patricolo

October 6, 2022

ABSTRACT

We ran a simulation of a 2-dimensional Ising model; our scope was to study the behaviour of the system near its critcal temperature T_c and to derive its critical exponents. Since our computers are not infinitely powerful, we simulated the system for finite lattice sizes, but we were able to study the phase transitions through Finite Size Scaling, which enabled us to understand how the system would behave if it were of infinite size.

1 ISING MODEL

The Ising model represents the interaction between classical spins fixed on a grid. Its hamiltonian is:

$$H = -J \sum_{\langle i,j \rangle} S_i S_j - h \sum_i S_i \equiv H[\{S_i\}] \tag{1.1}$$

Where $S_i = \pm 1 \; \forall i$ and $\langle i, j \rangle$ indicates that the sum must be computed $\forall i, j$ that satisfy the proximity conditions, which in our case means that the sites i and j must be adjacent in our grid, either vertically or horizonatally. The collection $\{S_i\}$ represents a state of the system, so that we can write the probability distribution of the states:

$$\rho(\{S_i\}) = \frac{1}{Z} \exp\left(-\beta H[\{S_i\}]\right) \tag{1.2}$$

$$Z = \sum_{\{S_i\}} \exp\left(-\beta H[\{S_i\}]\right) \tag{1.3}$$

Where $\beta = 1/(k_B T)$ and T is the temperature of the system. In our simulation, we fixed the following quantities:

$$\begin{aligned} k_B &= 1 \\ J &= 1 \\ h &= 0 \end{aligned} \tag{1.4}$$

1.1 Continuous phase transition

With our choice (1.4), in the thermodynamical limit, we expect to find a continuous phase transition at an inverse temperature:

$$\beta_c = \frac{1}{T_c} \simeq 0.4406868 \tag{1.5}$$

In a neighbourhood of β_c, we expect a critical behaviour of the system described by the variable $t \equiv (T - T_c)/T_c \propto \beta - \beta_c$ and the power laws:

$$\xi \sim |t|^{-\nu} \tag{1.6}$$

$$\langle M \rangle \sim |t|^{\beta} \qquad t < 0 \tag{1.7}$$

$$\chi \equiv \frac{\partial \langle M \rangle}{\partial h} = \frac{V}{T}\left(\langle M^2 \rangle - \langle M \rangle^2\right) \sim |t|^{-\gamma} \tag{1.8}$$

$$C \equiv \frac{\partial \langle \epsilon \rangle}{\partial T} = \frac{V}{T^2}\left(\langle \epsilon^2 \rangle - \langle \epsilon \rangle^2\right) \sim |t|^{-\alpha} \tag{1.9}$$

Where ξ is the correlation length, V is the volume of the system, M is the magnetization density, ϵ is the energy density, χ is the magnetic susceptibility and C is the thermal capacity. The theoretical values of the critical exponents in a 2-dimensionsonal Ising model are the following:

$$\alpha = 0 \quad \beta = 1/8 \quad \gamma = 7/4 \quad \nu = 1 \tag{1.10}$$

2 Metropolis

In order to sample our system, we devised a Markov chain that we implemented using an *ad hoc* version of the Metropolis algorithm, which can be summarized along these lines:

1. Let us call the current state of the system S
2. We select a site of the lattice and we compute a *trial* spin value for that site, i.e. we invert the sign of the spin, so that if we substituted the current value of the spin in that site with the trial value, we would end up in a new state that we call T
3. Using (1.2) we can compute the probabilities for both states, called p_S and p_T. If we select the i-th site, the ratio of the probabilities is:

$$\frac{p_T}{p_S} = \exp\left(-2\beta\left(\sum_{\langle i,j \rangle} S_j\right) S_i\right) \tag{2.1}$$

   Note that the sum above only has 4 terms given our proximity conditions, since i is fixed in the sum
4. Now the Metropolis test is run: the state T becomes the actual state of the system with probability p_T/p_S (if $p_T/p_S > 1$ then the state T always becomes the new state of the system)
5. We repeat the steps above for all the sites of the system

3 Bootstrap

Markov chains produce data which are not uncorrelated (each state has a well defined probability of being sampled given the previous state), rendering error estimation more bothersome than usual. To account for this, we used the bootstrap block resampling technique. Let us consider an array of data points of size `measures` and let us assume that this array represents the

states of the lattice during the simulation[1], then, if we want to compute the error associated to a given observable, the bootstrap algorithm runs as follows:

- Selects a resampling size `bin_size`
- Extracts a random number `k` in the range of `measures`
- Resamples all the `bin_size` consecutive elements of the array starting from the `k`-th element and stores them into another array[2]
- Repeats the last two steps `measures/bin_size` times, storing the data in the same array as the previous step, so that the resampled array would have approximately the same size (exactly the same size if `bin_size` is a divisor of `measures`) of the initial array
- Calculates the desired observable from the resampled array, saving it into another array
- Repeats the steps above for an arbitrary times, which we called `resamplings`, so that the array of observable would end up having `resamplings` elements (we fixed `resamplings` = 100)
- Computes the standard deviation of the array of observables and stores its value into another array
- Repeats the steps above for increasing values of `bin_size`. Specifically, the algorithm executes the line `bin_size` = `bin_size` $*$ 2 at the end of the last step
- Finally, returns the maximum value of the array of standard deviations, which we infer to be the correct estimate of the error of that given observable

The central values of the observables are calculated directly from the starting arrays.

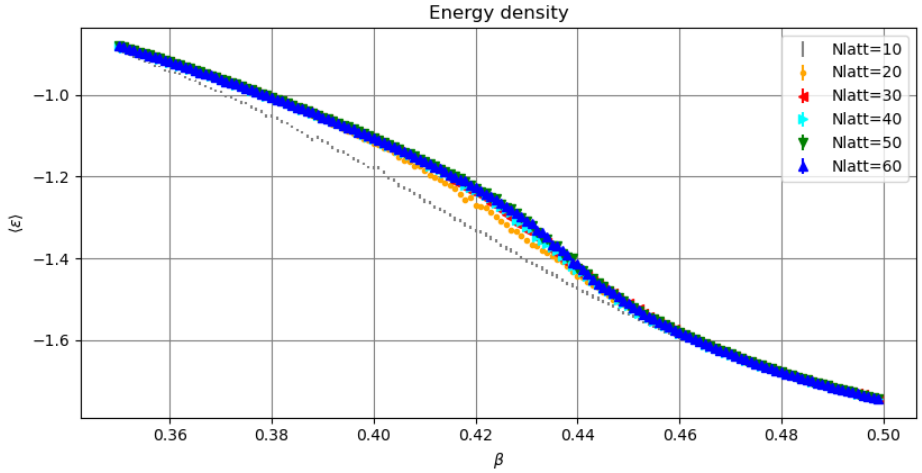
4 Simulation

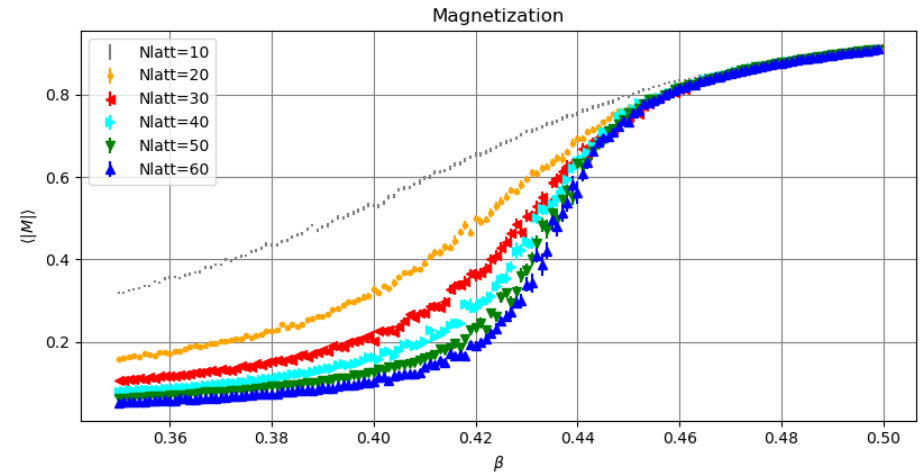
Basically, our simulation does the following:

1. Reads all the variables needed to run the simulation from `.txt` files which had been created for this purpose.
2. Suppose the sides of the lattice are long L, then the algorithm creates a square lattice matrix of size `Nlatt`, so that $L = a$`Nlatt`, where a is the distance between two adjacent spins, initializing it in 3 possible states, which are selected based on the value of the variable `init_flag`:
   - `init_flag` = 0 $\Rightarrow$ cold state, all the spins in the lattice are initialized having the value 1.0
   - `init_flag` = 1 $\Rightarrow$ hot state, all the spins are selected at random
   - `init_flag` $\neq 0, 1 \Rightarrow$ old state, the program reads a `.txt` file where we stored the final state of the lattice of the previous iteration of the simulation
3. Computes the proximity conditions and the boundary conditions (the latter define a toroidal structure)
4. Starts the Markov chain using the Metropolis algorithm described in section 2; each data point is taken once every time the Metropolis algorithm has been called `decorrel_len` times, for a total number of `measures` data points. These data points correspond to the values of the magnetization density and of the energy density of the lattice and are stored into `.txt` files.
5. Saves the final lattice configuration into a `.txt` file

[1] The physical quantities that we want to compute can be derived from just 2 arrays, which store the magnetization density and the energy density of the states explored by the Markov chain, so it is not necessary to save each spin configuration during the simulation and the bootstrap algorithm can be implemented using these arrays

[2] If `k` + `bin_size` > `measures` − 1, then the algorithm resamples the last `bin_size` elements of the array





6. Runs the bootstrap algorithm as described in section 3 and computes all the physical quantities that will be discussed in the following sections with their associated errors

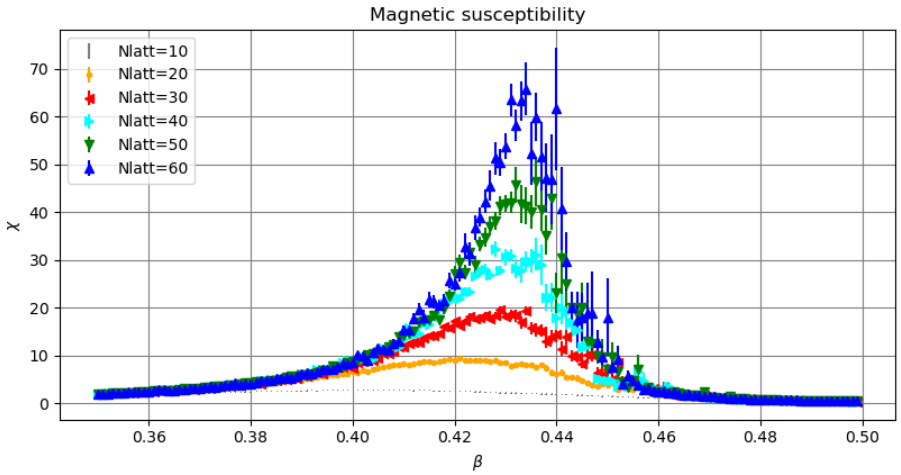
The complete source code can be found in section 7. We were interested in measuring the energy density, the magnetization density, the magnetic susceptibility, the heat capacity and the Binder cumulant for the magnetization on the lattice. We ran the simulation for different temperatures and different values of `Nlatt`, obtaining the results illustrated in the following sections.

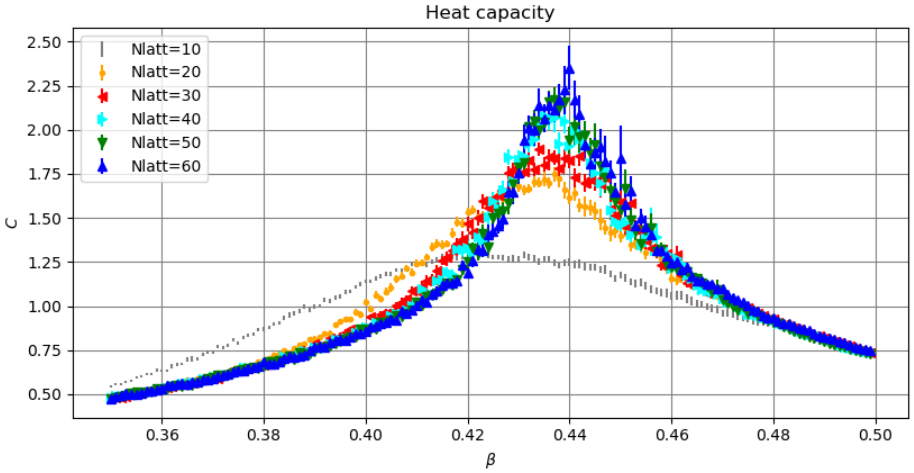
5 Observables as functions of lattice size and temperature

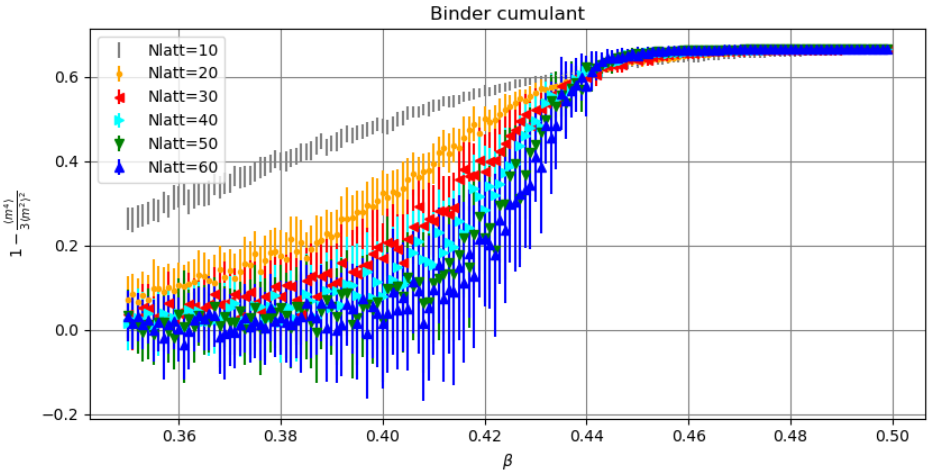
We report the plots for the observables in relation to the inverse temperature and lattice size. From the plots, especially from the susceptibility plot, we can clearly see the limitations of our simulation: the finite size of the system means that we do not observe the continuous phase transition that we would expect in the thermodynamical limit, in fact, in our simulation, the correlation length ξ cannot diverge:

$$\xi \leq L \tag{5.1}$$

Which means that the power laws in (1.6) can not be satisfied.


Magnetic susceptibility
Nlatt=10
Nlatt=20
Nlatt=30
Nlatt=40
Nlatt=50
Nlatt=60
χ
β


Heat capacity
Nlatt=10
Nlatt=20
Nlatt=30
Nlatt=40
Nlatt=50
Nlatt=60
C
β


Binder cumulant
Nlatt=10
Nlatt=20
Nlatt=30
Nlatt=40
Nlatt=50
Nlatt=60
β

6 Finite Size Scaling

In order to account for the limitations summarized in (5.1), we relied on Finite Size Scaling (FSS). Let us focus our attention on the susceptibility χ, knowing that what follows can be easily transposed onto the other quantities defined in (1.6). In the thermodynamical limit we have:

$$\chi \sim |\xi|^{\gamma/\nu} \tag{6.1}$$

We know that the quantities that we have measured during our simulation, which we will call $\chi(\beta, \texttt{Nlatt})$ to make their dependencies explicit, can not diverge but from (5.1) we can infer that in a neighbourhood of the peak:

$$\chi(\beta, L) = \chi(\xi, L) = L^{\gamma/\nu}\hat{\varphi}(\xi, L) \tag{6.2}$$

Where $\xi(\beta)$ is the expected correlation length in the thermodynamical limit at temperature $1/\beta$. Furthermore, during a phase transition, $\xi \to \infty$, so we expect the system to "lose memory" of its own microscopic structure, i.e. the value of a, meaning that we can assume that the only relevant length during the transition is the ratio between the two remaining lengths L and ξ, so that:

$$\chi(\beta, L) = L^{\gamma/\nu}\tilde{\varphi}(L/\xi) = L^{\gamma/\nu}\tilde{\phi}\left(tL^{1/\nu}\right) = L^{\gamma/\nu}\varphi\left(\left(\beta - \beta_c\right)L^{1/\nu}\right) \tag{6.3}$$

$$L^{-\gamma/\nu}\chi(\beta, L) = \varphi\left(\left(\beta - \beta_c\right)L^{1/\nu}\right) \tag{6.4}$$

7 Codes

All the codes used in the simulation, although a little messy, are available on GitHub