UNIVERSIDAD DE SEVILLA
DOBLE GRADO EN FÍSICA Y MATEMÁTICAS[1]
TRABAJO DE FIN DE GRADO

INSTITUT FÜR THEORETISCHE PHYSIK
WWU MÜNSTER

Width of Interfaces in the Two-Dimensional Ising Model

Author
Manuel CAÑIZARES GUERRERO[2]

Supervisor
Prof. Dr. Gernot MÜNSTER[3]

July, 2018

[1]Facultad de Física, Facultad de Matemáticas.
[2]m_cani02@uni-muenster.de
[3]munsteg@uni-muenster.de

Dedicated to everyone that made this work possible:
of course, to my supervisor G. Münster, for all his help,
to my family and friends, for the unconditional support,
to Dani, who drew Fig. 3,
and to Carlos, who designed the title page.
But also to all the researchers in whom I base this work,
and to all the free software and packages developers,
anonymous heroes without a cape that make our lives easier.

"I must not fear. Fear is the mind-killer. Fear is the little-death that brings total obliteration."
-F. Herbert.

Abstract

In this thesis, interfaces are studied in the context of the two-dimensional Ising model. In particular, one looks for the dependence of the width of these interfaces on the size of the lattice. Theoretical predictions are gotten from the convolution approximation between Capillary Wave and Mean Field Theory. A brief introduction to Monte Carlo simulations is included. Finally, the Metropolis algorithm is implemented and used to check the accuracy of the theoretical predictions, and to find a cutoff for the scale of the validity of Capillary Wave Theory.
Keywords: Ising model, interface, mean field theory, capillary wave theory, Monte Carlo method, Metropolis algorithm.

Contents

1 Introduction

Since the first half of the XX century, Monte Carlo methods have earned an important role in computation algorithms, when they were being developed simultaneously with modern computers. Like many other scientific advances, they both were intended to be used for war. In particular, Monte Carlo algorithms were devised to simulate neutron movement, with the finality of developing the hydrogen bomb.

However, like most of these war-born advances, it serves many purposes outside of armed conflict. Monte Carlo methods use randomness to simulate the behaviour of complex deterministic systems, and they are applied in all kind of sciences. They are very frequently used in statistical physics, due to the probabilistic nature of this discipline.

One of the main, probably most widely used examples in both statistical physics and Monte Carlo methods applications is the Ising model. It was also born in the first half of the XX century. It is used to model the behaviour of a magnetic material. If it is so widely used, it is probably because it is simple to understand, and yet widely applicable. It has been analytically solved for two dimensions, which also makes it suitable to test the reliability of simulation algorithms.

In this thesis, the Metropolis algorithm, one of the first of the Monte Carlo methods, devised by Nicholas Metropolis during the Manhattan Project, is used to study interfaces in the context of the two-dimensional Ising model. The profiles of these interfaces can be calculated approximately by Quantum Field Theory, in particular by a convolution between the mean field theory and the capillary wave theory. Mean field theory provides with an intrinsic profile for the interface, describing the interface at a microscopic level, while capillary wave theory describes the macroscopic behaviour, inserting a dependence of the interface width on the size of the system. Then, the classic Metropolis simulation for the Ising model is implemented with the right boundary conditions to check said approximation and estimate its range of validity.

2 The Ising Model

The Ising Model is a mathematical model in statistical mechanics for ferromagnetism. The model takes the name from Ernst Ising, who solved it in his thesis in 1924 for a one-dimensional lattice, by proposal of his professor Wilhelm Lenz. It is probably the most studied model in Statistical Physics. The two-dimensional model was solved by Onsager in 1944, and has been an important subject of study for statistical physicists during the XX century. However, a three-dimensional lattice raises a huge level of complexity, and an exact solution has not yet been found. Despite its dimensional restriction, the 2D Ising model is used for the study of the ferromagnetic phase transition. Furthermore, it has very many physical properties that can be exactly computed. All this makes it also a good model for the application of Monte Carlo methods.

The basic idea behind the model is that the magnetism of a bulk material is made up of the dipole moments of many atomic spins in the material. It postulates a lattice of sites, with a set of adjacency relations -in general, a graph-, in which spin variables are placed, which can only take the values ± 1. The spins only interact with their neighbours, and with the external magnetic field. If we denote these variables as s_i, the Hamiltonian of the system is given by [1] [5]:

$$\mathcal{H} = -J \sum_{<i,j>} s_i s_j \; + \; B \sum_i s_i \tag{2.1}$$

Here,

- $< i, j >$ denotes that the lattice sites i and j are adjacent.
- J is a the interaction parameter. If $J > 0$ the interaction is ferromagnetic, if $J < 0$ it is antiferromagnetic and, if $J = 0$, the spins are noninteracting. We can take $J = 1$ in our model for a ferromagnetic material.
- B corresponds to the extern magnetic field, which we will particularly take as null.

Usually, one chooses an easy geometry, i.e a D-dimensional finite array -in our case, $D = 2$- in which the adjacency relation is straightforward. On the edges of the lattice, different boundary conditions can be set, which will be discussed later.

The partition function for this model is given by

$$Z = \sum_{\{s_i\}} e^{-\beta \mathcal{H}} = \sum_{s_1=\pm 1} \sum_{s_2=\pm 1} \dots \sum_{s_N=\pm 1} \exp\left[\beta \sum_{<i,j>} s_i s_j\right] \tag{2.2}$$

where β is the *inverse temperature*, defined as $\beta = \frac{1}{k_B T}$, being k_B the *Boltzmann constant* and T the temperature. We can also take the units of $k_B = 1$. Under the framework of the *canonical ensemble*, we can calculate the probability of a state μ, corresponding to certain set of values of the spins s_i. This is given by the Boltzmann distribution:

$$P_\mu = \frac{1}{Z} e^{-\beta \mathcal{H}_\mu} \tag{2.3}$$

and thus, the expectation of an estimator f can be calculated in the usual manner:

$$\langle f \rangle = \sum_\mu P_\mu f_\mu = \frac{1}{Z} \sum_\mu e^{-\beta \mathcal{H}_\mu} f_\mu \tag{2.4}$$

For example, an interesting observable is the magnetization per spin:

$$m_\mu = \frac{1}{N} \sum_{i=1}^{N} s_i^\mu \tag{2.5}$$

where N is the number of lattice points.

$$\langle m \rangle = \frac{1}{NZ} \sum_\mu \left(e^{-\beta \mathcal{H}_\mu} \sum_{i=1}^{N} s_i^\mu \right) \tag{2.6}$$

In the two-dimensional Ising model, one finds a critical point at the Curie temperature $T_c = \left[\frac{1}{2} \log(1 + \sqrt{2}) \right]^{-1} \approx 2.269$ [5]. Above it, the material is found in the disordered phase, in which the random distribution of the spins sums up to an almost null magnetization, $< m > \approx 0$. Below it lies the ferromagnetic phase, in which the correlation between spins forces them to point in the same direction, giving raise to a net magnetization, $|< m >| > 0$. Around the Curie temperature, a phase transition occurs. In Fig.1, a representation of both theoretical and Monte Carlo simulated results for the phase transition is shown. The simulated results have been obtained with our own algorithm and periodic boundary conditions -explained in section 4- on a 200x200 square lattice. For an infinite lattice, there would be an spontaneous symmetry breaking, while in finite-size lattices as in the simulation, the phase transition is smoother.

The expression for the *correlation length*, an estimate of the distance at which one spin is correlated with another, is given below for low temperatures ($T < T_C$). When approaching β_C, the correlation length grows, giving raise to "bubbles", and ultimately diverging in the phase transition. [6]

$$\xi = \frac{1}{2} \left(2\beta + \log \tanh \beta \right)^{-1} \quad (\beta > \beta_C) \tag{2.7}$$

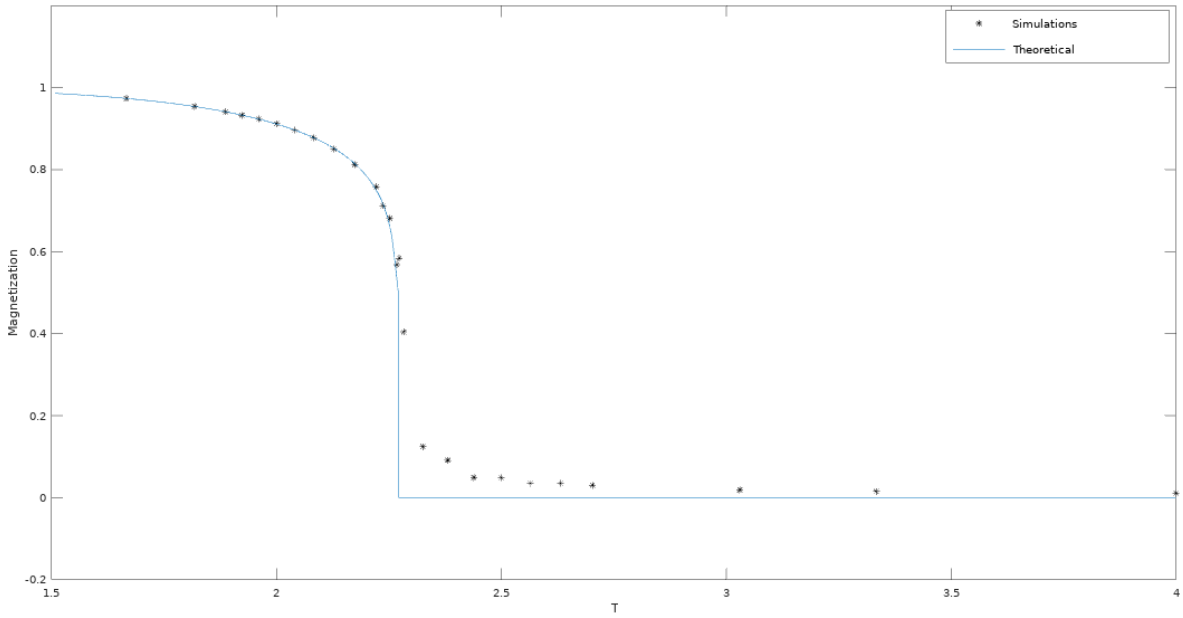


Figure 1: The phase transition in the two-dimensional Ising model is shown by representing the dependence of magnetization per spin on temperature. A comparison between theoretical prediction and Monte Carlo simulation on a 200x200 square lattice can be seen. The results show a nice agreement, and only differ near the critical point, where the symmetry breaking occurs.

3 Interfaces.

The subject of this study, however, is not a property of the whole material, but the behaviour of interfaces inside it. These interfaces are hypersurfaces that can be found, for example, separating magnetic domains in the ferromagnetic phase. These domains are regions in which the magnetization is homogeneous, but one can differ from one another in having opposite magnetizations. Ferromagnetic materials are usually separated in these domains when cooled down below the Curie Temperature from the disordered phase. In three-dimensional materials, interfaces are two-dimensional surfaces. However, from now on we will consider for simplicity a two-dimensional model, in which the interfaces appear as curves.

3.1 Mean Field Theory

These interfaces and magnetic domains were first approached theoretically by Pierre Curie and Pierre Weiss at the beginning of the XX century [4]. The technique used was the so called *mean field theory*. Curie and Weiss made an approximation in which the interaction of a spin was accounted as a mean interaction with the whole rest of the lattice, instead of an interaction only with its neighbours.

In our case, Landau theory can be used to describe the profile of the interfaces. This theory is embedded in the framework of QFT, out of the scope of our study. However, the basic supposition of this theory is that the free energy of the system is analytic and obeys the symmetry of the Hamiltonian. We may represent the profile of the interface as a continuous profile $\phi(x)$ representing the difference between concentrations of the two coexisting phases. In the framework of the Landau theory, this profile plays the role of a local order parameter. The free energy density is a functional to be minimized w.r.t. the profile function, and in this case is given by [2]:

$$\mathcal{F} = \frac{1}{2}(\phi'(y))^2 + V(\phi(y)) \tag{3.1}$$

with the double-well potential

$$V(\phi) = \frac{g}{4!}\phi^4 - \frac{m_0^2}{4}\phi^2 + \frac{3}{8}\frac{m_0^4}{g} \tag{3.2}$$

By minimizing $\mathcal{F}$ with boundary conditions appropriate for an interface perpendicular to the y-axis, one finds the mean field profile [2]:

$$\phi(y) = \tanh\left(\frac{1}{2\xi_0}(y-h)\right) \tag{3.3}$$

where $\xi_0 = \frac{1}{m_0}$ is the mean field correlation length which, as stated in section 2, depends on the temperature and diverges in the critical point, and h is the position of the center of the interface, which is arbitrary due to translational invariance.

Corrections due to order parameter fluctuations can be calculated with the *local potential approximation*, in which only corrections to the local potential $V(\phi)$ are taken into account. In

the lowest order local potential approximation, an analogous interface profile is obtained, this time including the physical correlation length ξ:

$$\phi(y) = \tanh\left(\frac{1}{2\xi}(y-h)\right) \tag{3.4}$$

This approximation does not fully account for long wavelength fluctuations. Thus, profile (3.4) does not contain the effects of the *capillary waves*, discussed below.

3.2 Capillary Wave Theory

The breaking of the translational symmetry by the appearance of an interface, like any other spontaneous symmetry breaking, is represented in the context of QFT as a Goldstone boson. The boson associated with this broken symmetry are long-wavelength excitations of the interface position, which have vanishing energy cost in the infinite wavelength limit. These capillary waves strongly influence interfacial properties like the interface width, but are neglected by the mean field theory, which supposes a flat interface. The free energy cost of capillary waves is basically due to the increase in interface length against the reduced interface tension σ. Being $h(x)$ the local interface position, the capillary wave free energy is given by [2]

$$F = \frac{\sigma}{\beta}\int dx\left[\sqrt{1+(\nabla h)^2}\right] \approx \frac{\sigma}{2\beta}\int dx\,(\nabla h)^2 \tag{3.5}$$

where $|\nabla h| \ll 1$, i.e. long wavelength and small amplitude, has been assumed. In the capillary wave model, one assumes that the local interface position is distributed as a Gaussian distribution

$$P(h) = \langle\delta(h(x)-h)\rangle = \frac{1}{\sqrt{2\pi}s}\int dx\,\delta(h(x)-h)\,e^{-\frac{h(x)^2}{2s^2}} = \frac{1}{\sqrt{2\pi}s}e^{-\frac{h^2}{2s^2}} \tag{3.6}$$

with variance

$$s^2 = \langle h^2\rangle = \frac{1}{2\pi\sigma}\int_{q_{min}}^{q_{max}} dq\,\frac{1}{q^2} \tag{3.7}$$

In [9], Gelfand and Fisher provide with an approximation via integration. However, we consider a system of finite extent L, so a much more precise approach changes the integral by sums, in which the allowed momenta are [3]

$$q_n = \frac{2\pi}{L}n, \quad \text{with} \quad n \in \mathbf{Z}, n \neq 0 \tag{3.8}$$

and the integrals are turned into sums of the shape:

$$\sum_{q_n}\frac{\Delta q}{2\pi} = \frac{1}{L}\sum_{q_n}. \quad \text{where} \quad \Delta q = \frac{2\pi}{L} \tag{3.9}$$

Then, the variance is given by

$$s^2 = \langle h^2 \rangle = \frac{1}{\sigma L} \sum_{q_n} \frac{1}{q^2} = \frac{1}{\sigma L}\, 2 \sum_{q_n > 0} \frac{1}{q^2} \tag{3.10}$$

The sum goes from $q_{\min} = 2\pi/L$, which sets an upper bound for the wavelength of the capillary wave, to q_{max}, which is for now arbitrary, and sets a cutoff for the scale of below which the theory is no longer valid. This q_{max} is expected to be inversely proportional to the correlation length ξ.

$$s^2 = \frac{2}{\sigma L} \sum_{q=q_{\min}}^{q_{\max}} \frac{1}{q^2} = \frac{2}{\sigma L} \sum_{q=q_{\min}}^{\infty} \frac{1}{q^2} - \frac{2}{\sigma L} \sum_{q=q_{\max}+\Delta q}^{\infty} \frac{1}{q^2}. \tag{3.11}$$

The first term is

$$s_a^2 = \frac{2}{\sigma L} \sum_{q=q_{\min}}^{\infty} \frac{1}{q^2} = \frac{2}{\sigma L} \left(\frac{L}{2\pi}\right)^2 \sum_{n=1}^{\infty} \frac{1}{n^2}. \tag{3.12}$$

With

$$\sum_{n=1}^{\infty} \frac{1}{n^2} = \frac{\pi^2}{6} \tag{3.13}$$

we obtain

$$s_a^2 = \frac{1}{12\sigma} L. \tag{3.14}$$

The second term is approximated by

$$s_b^2 = \frac{2}{\sigma L} \sum_{q=q_{\max}+\Delta q}^{\infty} \frac{1}{q^2} = \frac{2}{\sigma} \int_{q_{\max}}^{\infty} \frac{dq}{2\pi} \frac{1}{q^2} = \frac{1}{\pi \sigma q_{\max}} \tag{3.15}$$

Combining the terms and using $1/\sigma = 2\xi$, we obtain [3]

The interface tension σ can be seen a measure of the free energy per unit length of the interface, and was analytically calculated by Onsager for the two-dimensional Ising Model, giving the result $\sigma = \frac{1}{2\xi}$ [7].

In this framework, the instantaneous interface profile is a sharp step function between two phases. However, in the thermal average the capillary wave fluctuations produce a continuous density profile with a finite width whose square is proportional to the variance s^2, and thus to the system size L. Consequently, the interface width depends on the length scale and diverges in the thermodynamic limit.

$$\rho(y) = \int dh\, \mathrm{sgn}(y - h)\, P(h) = \mathrm{erf}\left(\frac{y}{\sqrt{2}s}\right) \tag{3.16}$$

3.3 Convolution Approximation and Interface Width

We can argue that the intrinsic profile describes the interface on a microscopic scale of the order of the correlation length, while capillary wave theory describes the macroscopic interface fluctuations of wavelengths much larger than this scale. [2] If we neglect the coupling between these structures, we can view the whole interface as a capillary wave theory membrane $h(x)$ to which an intrinsic profile $\phi(y)$ is "attached". If we perform the thermal average over the capillary waves, the final profile is mathematically given by a convolution

$$\mathfrak{m}(y) = \int dh\, \phi(y-h)\, P(h) \tag{3.17}$$

Now, we can define the interface width in various ways. For numerical purposes, a comfortable definition of the squared width is the second central moment of the distribution with probability distribution equal to the spatial derivative of the interface profile $\mathfrak{m}$:

$$w^2 = \frac{\int dy\, y^2\, \mathfrak{m}'(y)}{\int dy\, \mathfrak{m}'(y)} - \left[\frac{\int dy\, y\, \mathfrak{m}'(y)}{\int dy\, \mathfrak{m}'(y)}\right]^2 \tag{3.18}$$

Due to the linearity of this definition and of the convolution approximation, w^2 is the sum of the intrinsic squared width w_{int}^2 of the intrinsic profile ϕ and the capillary wave squared width w_{cw}^2 of the profile predicted capillary wave theory ρ:

$$w^2 = w_{int}^2 + w_{cw}^2 = \frac{\pi^2}{3}\xi^2 + \frac{1}{12\sigma}L - \frac{1}{\pi\sigma q_{\max}} \tag{3.19}$$

Inserting the known value for σ:

$$w^2 = \frac{\pi^2}{3}\xi^2 + \frac{L}{6}\xi - \frac{2}{\pi q_{\max}}\xi \tag{3.20}$$

4 Monte Carlo Methods

4.1 History and Definition

One could date the idea behind Monte Carlo methods back to the XVIII century, when Georges-Louis Leclerc proposed the "Buffon's needle problem" [8]. It consisted on calculating the probability of a needle touching one of the parallel, equidistant cracks on the floor, when dropped randomly. This probability is in fact inversely proportional to π, and thus one could estimate the value of π by throwing a sufficiently large number of needles and counting the proportion that touched the cracks.

However, the development and systematic use of these methods corresponds to the last century. [5] The main purpose of Monte Carlo methods was originally to estimate the values of the integral of poor-behaved functions, for which analytic integration was not a viable option. They became important in the 1930's, when they were used by Enrico Fermi to study neutron diffusion, and by the team of the Manhattan Project in Los Alamos to develop the Hydrogen bomb. Their popularization during this time comes in hand with the technological advances in computers. The name Monte Carlo method was coined by Nicholas Metropolis, who worked in the project, as a reference to the Monte Carlo casino in Monaco.

The Monte Carlo methods consist on solving a deterministic problem by using randomness. One generates a sample of random inputs for the problem, and performs the calculations on these inputs to approximate the results, expecting this result to asymptotically approach the deterministic solution when the sample size tends to infinity. They are widely used in mathematics, financial predictions and simulation of physical models, amongst other fields.

4.2 Markov Processes

The basic mathematical concept on which Monte Carlo methods rely is on *Markov Chains* or *Markov Processes*. A discrete-time Markov Process (DMP) is a random process in which the probabilities of the different possible states at the following step $i+1$ only depend on the current state i, and not in the past of the process.

The sampling in Monte Carlo methods is made using these processes. We only need to consider DMPs, since the sampling is made in discrete computing steps, and not in a continuous time. One establishes an initial state for the simulation and a set of transition probabilities $P_i(\mu \to \nu)$ from one state μ to another state ν. This way, with the use of random numbers, the evolution of the simulation is a set of states governed by the transition probabilities. Of course, this probability has to fulfill the axioms of a probability function, namely:

- $P_i(\mu \to \nu) \geq 0 \qquad \forall \nu \in S$
- $\sum_{\nu \in S} P_i(\mu \to \nu) = 1$ where S is the set of all possible states for the process.
- $P_i(\mu \to \cup_{j=1}^{\infty} \nu_j) = \sum_{j=1}^{\infty} P_i(\mu \to \nu_j)$ being $\{\nu_j\}_{j \geq 1}$ a countable set of disjoint states

4.3 Ergodicity and Detailed Balance

In our particular case of statistical mechanics, the goal of Monte Carlo methods is to to generate a succession of states with a probability that tends to that of the Boltzmann distribution. That way, we will be able to approximate observables by averaging over the different states generated.

For this purpose, we first should compel the simulation to fulfill the property of *ergodicity*. This property states that every state has a non-zero probability to be reached in a finite number of states from any other possible state. If this property was not fulfilled, the average over time would differ from the average over the phase space of the physical system we want to simulate.

On the other hand, the property of *detailed balance* ensures that the system generates the Boltzmann probability when it comes to equilibrium. The concept of equilibrium in this context is not an explicit staticity of the system, but rather that the rate at which the system enters and exits a state μ should be equal:

$$\sum_{\nu \in S} p_\mu^i \, P_i(\mu \to \nu) = \sum_{\nu \in S} p_\nu^i \, P_i(\nu \to \mu) \tag{4.1}$$

where p_λ^i is the probability of being in state λ at time i. Using the properties of probability functions, we get

$$p_\mu^i = \sum_{\nu \in S} p_\nu^i \, P_i(\nu \to \mu) \tag{4.2}$$

This however could lead us to a situation in which the system reaches a dynamic equilibrium, i.e. the probability distribution rotates around a limit cycle. What we are looking for is a simple limit for the distribution function. Here is where the property of detailed balance comes into play. This property can be expressed as

$$p_\mu^i P_i(\mu \to \nu) = p_\nu^i \, P_i(\nu \to \mu) \tag{4.3}$$

Obviously, a set of transition probabilities fulfilling (4.3) will in particular fulfill (4.1). Furthermore, it can be easily shown that this property eliminates the possibility of limits cycles. Now, since the desired final distribution is the Boltzmann distribution, whose probability function is given by (2.3), the detailed balance condition results on the following imposition for transition probabilities:

$$\frac{P(\mu \to \nu)}{P(\nu \to \mu)} = \frac{p_\nu}{p_\mu} = e^{-\beta(\mathcal{H}_\nu - \mathcal{H}_\mu)} \tag{4.4}$$

where the index i has been omitted for readability.

At this point, the development of an algorithm is based on finding a way to obtain this relation between transition probabilities. In this study, the *Metropolis algorithm*, which is explained below, is used.

4.4 Metropolis Algorithm

4.4.1 Idea

The Metropolis algorithm was introduced by Nicholas Metropolis and his co-workers in 1953 on simulations of hard-sphere gases. It is a very intuitive algorithm and relatively easy to implement. It is not as efficient as some posterior algorithms, but enough for our purpose.

The idea of the algorithm is to separate the transition probabilities, which are time-independent, in a product of two functions: a *selection probability* $g(\mu \to \nu)$ and an *acceptance probability* $A(\mu \to \nu)$. In each step, the algorithm chooses to switch to a random state ν with probability $g(\mu \to \nu)$, and then randomly decides whether to accept or not this state transition, with probability $A(\mu \to \nu)$ of accepting it.

A physical system in thermal equilibrium will spend most of its time in states with a narrow range of energies. To avoid putting too much computation effort on states energetically too distant from the current state, the proposed state transitions consist just on the flip of a single spin. Then, the selection probabilities are usually equal for every state and every spin, and the importance lies in the acceptance probabilities. With these considerations, Eq. (4.4) turns into

$$\frac{P(\mu \to \nu)}{P(\nu \to \mu)} = \frac{g(\mu \to \nu)A(\mu \to \nu)}{g(\nu \to \mu)A(\nu \to \mu)} = \frac{A(\mu \to \nu)}{A(\nu \to \mu)} = e^{-\beta(\mathcal{H}_\nu - \mathcal{H}_\mu)} \tag{4.5}$$

To make the algorithm more efficient by saving computation time, we will prefer to have the maximum number of steps in which a transition occurs possible. This is achieved by setting the largest of both transition probabilities between two states as 1. An interpretation of this is that the system always accepts to drop from one state to a less energetic one, while it stochastically jumps to a more energetic one with a probability that depends on the temperature, i.e. on the available energy. The resulting probabilities are:

$$g(\mu \to \nu) = \tfrac{1}{N}\ ; \quad A(\mu \to \nu) = \begin{cases} e^{-\beta(\mathcal{H}_\nu - \mathcal{H}_\mu)} & \text{if } \mathcal{H}_\nu - \mathcal{H}_\mu > 0 \\ 1 & \text{otherwise} \end{cases}$$

where N is the total number of lattice points.

4.4.2 Implementation

In this case, we have implemented the Metropolis algorithm on the C programming language. The first step for the program is the initialization of the acceptance probabilities. Since the transitions consist only on single spin flips, the energy difference between two consecutive states depend only on the orientation spin of the neighbours of the spin to be flipped. Then, it is only necessary to compute a table of $\frac{1}{2}z$ probabilities at the beginning of the program, where z is the coordination number, i.e. the number of neighbours that a lattice site has. The coordination number is related to the dimension of the lattice as $z = 2^D$. Thus, for our two-dimensional model we only need to compute 2 acceptance probabilities. This 2 acceptance probabilities are the ones corresponding to $\mathcal{H}_\nu - \mathcal{H}_\mu \in \{4, 8\}$. The possibilities for the acceptance probability

are shown in Table 1. We have chosen $A(\mu \to \nu) = 1$ when the energy difference is null, so that again the number of transitions is maximized while being consistent with the model.

$\mathcal{H}_\nu - \mathcal{H}_\mu$	$A(\mu \to \nu)$
8	$e^{-8\beta}$
4	$e^{-4\beta}$
0	1
-4	1
-8	1

Table 1: Acceptance probabilities for the Metropolis algorithm on the two-dimensional Ising Model

The next step for the program is to initialize the spins in the lattice. There are several ways to do it. When run, our program offers the possibility of either a *cold start* or a *hot start*. The cold start simulates that the system is "heated" from $T = 0$ by initializing all the spins in the same direction, i.e. from an state of total order. The hot start simulates that the system is "cooled" from $T = \infty$ by initializing each spin in a random orientation, both orientations with equal probability, i.e. from a state of total disorder. All the random number are generated in this program with the default C function *rand48()*. The spins are included in a vector that goes from 0 to $N - 1$, being $N = L \cdot D$ the total number of sites, where L and D are the sizes of the lattice on the x and y axis, respectively. The simulations were made with $D = 100$, $L \in \{10n;\ 4 \leq n \leq 20\}$ and $\beta = 0.46905$. This choice of β was made so the correlation length would be suitable for the lattice dimensions ($\xi \approx 4.5$).

The number of steps is also predetermined. The computer time is actually not measured in steps, but sweeps. A sweep is a set of N consecutive steps, so that the algorithm can theoretically process a spin flip for every lattice site. In this case, the computer time was set to 70000 sweeps, which was considered to be a good balance between having enough measures and a decent computation time ($\sim 2 - 5$ minutes).

Actually, the algorithm does not necessarily process every lattice site in each sweep, since in every step it chooses randomly which spin to flip with probability $\frac{1}{N}$, as stated in 4.4.1. Next, it computes its neighbouring relations, and calculates the sum the adjacent spins. This way, the energy difference can be calculated by multiplying this sum by the flipped spin -see Eq. (4.6)-, and the acceptance probability is got by the values in Table 1. A random number between 0 and 1 is generated and compared with this probability, so the spin will be in fact flipped if and only if the random number is smaller than $A(\mu \to \nu)$. Let's parametrize the lattice as points $\{(x, y);\ 0 \leq x \leq L-1,\ 0 \leq y \leq D-1\}$. Then, if we were to flip a spin $s_{x,y}$ the energy difference would be given by

$$\Delta\mathcal{H} = \mathcal{H}_\nu - \mathcal{H}_\mu = -\sum_{i=\pm1}(-s_{x,y})s_{x+i,y} - \sum_{j=\pm1}(-s_{x,y})s_{x,y+j} + \sum_{i=\pm1} s_{x,y}s_{x+i,y} + \sum_{j=\pm1} s_{x,y}s_{x,y+j}$$

$$\Delta\mathcal{H} = 2\, s_{x,y} \left(\sum_{i=\pm 1} s_{x+i,y} + \sum_{j=\pm 1} s_{x,y+j} \right) \tag{4.6}$$

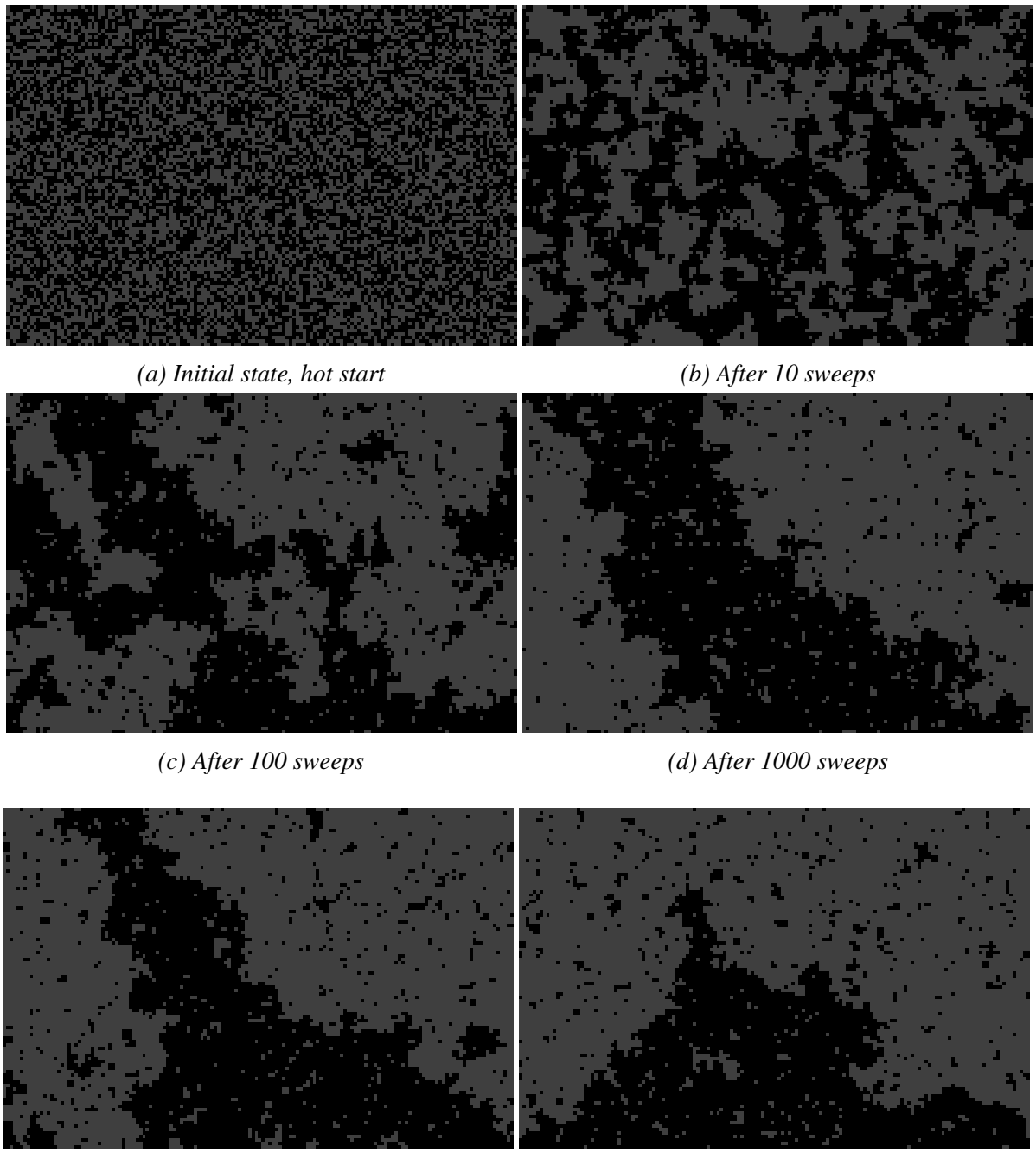
When calculating the adjacency, it is essential to have in mind the *boundary conditions*. A very usual choice is periodic boundary conditions, in which the values (L, y), $(-1, y)$, (x, D), $(x, -1)$ are respectively identified with the sites $(0, y)$, $(L-1, y)$, (x, D), $(x, -1)$, i.e. the spins on the edges are adjacent to the ones in the opposite edge. As the name indicates, this gives a periodicity to the lattice and ensures translational invariance.

Nonetheless, we are not looking for this translational invariance in the y-axis. Instead, we seek to break this symmetry. This is done by applying *antiperiodic boundary conditions* in this direction. The points (x, D), $(x, -1)$ are then identified with (x, D), $(x, -1)$, but with an inverted spin. Thus, for example, in the sum of Eq. (4.6), $s_{2,D} = -s_{2,0}$. This means that the Hamiltonian includes a term with an inverted sign, due to the interaction between the spins in the horizontal edges. A schematic of the lattice with the boundary conditions is displayed on Fig. 3

After every sweep, the program writes a file with an vector of size D including the *local magnetization*, which main purpose is introduced in section 4.4.5. This local magnetization $\mathfrak{m}(y)$ is defined as

$$\mathfrak{m}(y) = \frac{1}{L} \sum_{x=0}^{L-1} s_{x,y} \tag{4.7}$$

The program also writes a file that includes an array with the value of the spins in every lattice point, so the state of the system after every sweep can be represented. These files can then be read to perform the data analysis. In our case, it was made using the free software *Octave*. Below, in Fig. 2, some examples of lattice configurations for the simulation of a 150x100 lattice are displayed. The C code for the implementation of the simulation is showed on Appendix A.



(a) Initial state, hot start

(b) After 10 sweeps

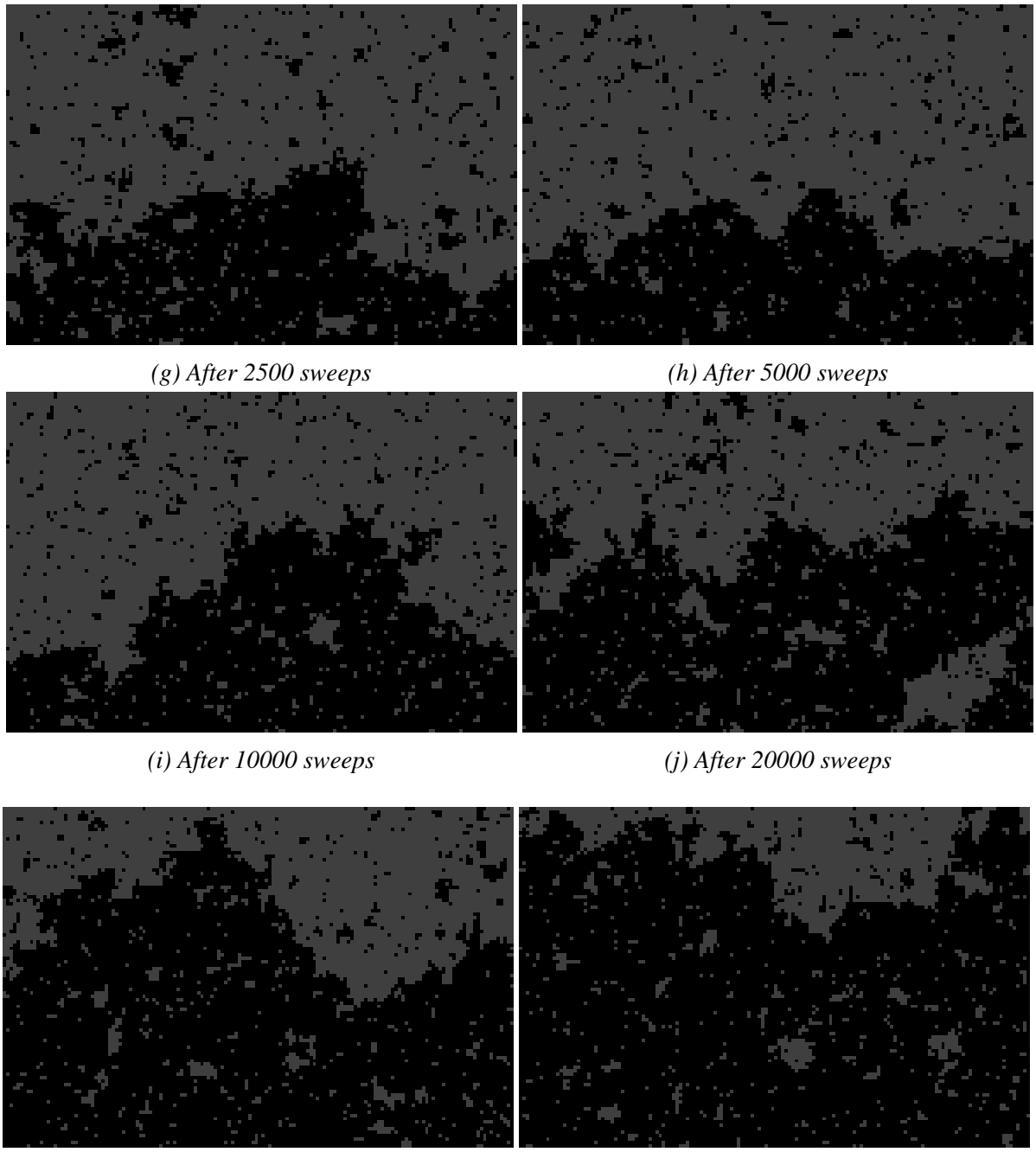
(c) After 100 sweeps

(d) After 1000 sweeps

(e) After 1500 sweeps

(f) After 2000 sweeps

Figure 2: Lattice evolution on a 150x100 lattice using antiperiodic boundary conditions, simulated with $\beta = 0.46905$ by the Metropolis algorithm. One can observe how the interface is formed.



(g) After 2500 sweeps *(h) After 5000 sweeps*

(i) After 10000 sweeps *(j) After 20000 sweeps*

(k) After 40000 sweeps *(l) After 50000 sweeps*

Figure 2: Lattice evolution on a 150x100 lattice using antiperiodic boundary conditions, simulated wih $\beta = 0.46905$ by the Metropolis algorithm. Here, the movement of the interface position can be observed.

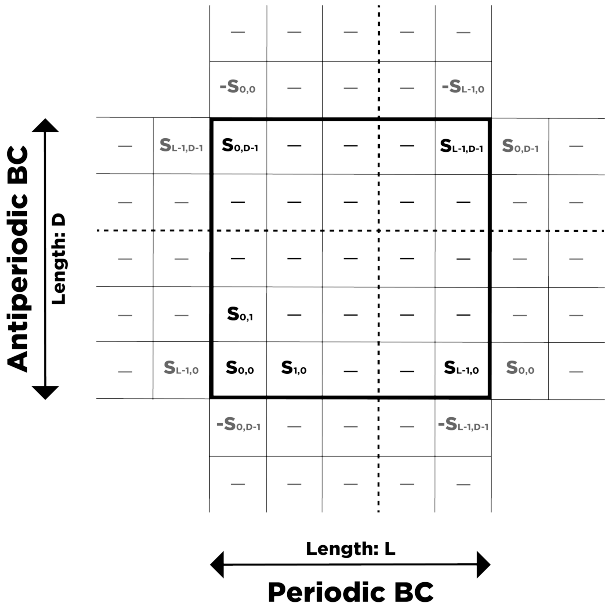


Figure 3: Schematic of the lattice disposition, including antiperiodic boundary conditions

4.4.3 Thermalization

Before performing computation over our data, the simulation has to be run long enough for the system to achieve an equilibrium, in the sense of, as explained in 4.4.1, having the states distributed following the Boltzmann distribution 2.3. The process of reaching equilibrium is indistinctly called *equilibration* or *thermalization*, because the equilibrium is considered as a thermal equilibrium. As we cannot directly measure the distribution of the states, one has to find an indirect way to determine whether the system has been *thermalized*.

Since a system in thermal equilibrium spends most of its time in a narrow range of state energies, one would expect certain observables to be relatively stationary when thermalization is reached. For periodic boundary conditions, the total magnetization provides with an excellent measure of thermalization time, since its value for a certain temperature is known.

However, we are considering antiperiodic boundary conditions. Here, the position of the interface is arbitrary and equiprobable. Due to this, the interface will move up and down the lattice, thus changing the size of the domains it separates. This means, that the magnetization gives us no information about the system. However, since the width of the interface should be almost stationary in equilibrium, one can use the following observable:

$$\mathcal{M} = \sum_{y=0}^{D-1} \left| \mathfrak{m}(y) \right| = \frac{1}{L} \sum_{y=0}^{D-1} \left| \sum_{x=0}^{L-1} s_{x,y} \right| \tag{4.8}$$

This is far from being perfect, since fluctuations in the shape of the interface result in bigger fluctuations in $\mathcal{M}$ than the one one finds in the magnetization with periodic boundary conditions. However, as showed on Fig. 4, it is not difficult to decide when the system has reached equilibrium. The equilibration time t_{eq} grows monotonically with lattice size due to

the increasing fluctuations. In our case, t_{eq} has been taken as different values depending on L, from $t_{eq} = 5000$ for $L = 40$ to $t_{eq} = 20000$ for $L = 200$. It is possible for a system, specially for those with bigger size, to get trapped in a metastable region of states, making t_{eq} much larger, even to the point of not reaching thermalization during the simulation. To avoid this, for each lattice size multiple simulations were run, and then the one with smaller t_{eq} was chosen.

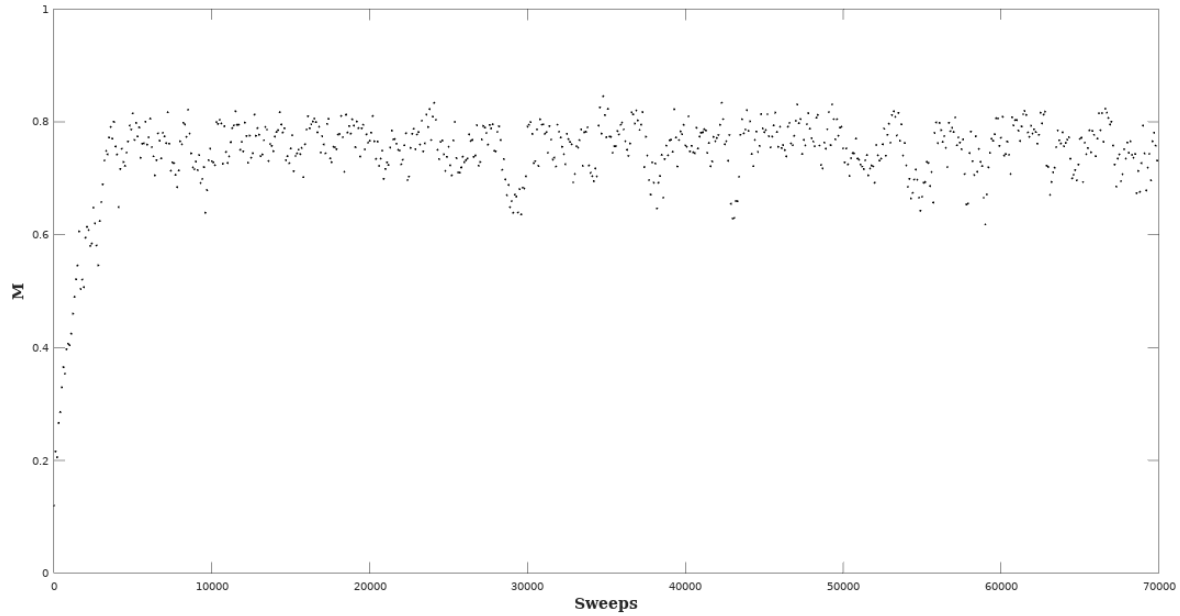


Figure 4: Evolution of $\mathcal{M}$ with computer time for $L = 80$, $D = 100$, $\beta = 0.46905$. The thermalization can be observed around 5000 sweeps, a $t_{eq} = 10000$ was taken for confidence.

4.4.4 Autocorrelation

After thermalization has been achieved, observables can be estimated by averaging in time. However, it is important to have in mind that the state at time t is strongly correlated with the state at time $t + 1$ (measured in sweeps). Then, an average over the whole computation time would take us to consider redundant correlated states. To work around this, we have to figure out a time interval between measurements. Again, we cannot directly see the correlation between states, so we need to consider the correlation between measurements of an observable. Again, a usual choice for periodic boundary conditions is the magnetization per spin, but we will use the magnitude $\mathcal{M}$, introduced in (4.8).

The tool used for solving this problem is the *time-displaced autocorrelation function*. This function is given by [5]

$$\chi(t) = \int dt' \, [\mathcal{M}(t') - \langle \mathcal{M} \rangle][\mathcal{M}(t' + t) - \langle \mathcal{M} \rangle] =$$

$$= \int dt' \, [\mathcal{M}(t')]\mathcal{M}(t' + t) - \langle \mathcal{M} \rangle^2] \tag{4.9}$$

This function gives an estimation of the correlation between two states of the simulation, one at time t later than the other one. In principle, one would expect the correlation to decrease exponentially with time:

$$\chi(t) \sim e^{-t/\tau} \tag{4.10}$$

Here, τ is called the *autocorrelation time*. A multiple of this time can be used as an estimate of the time interval that we should wait between measurements. Usually, a time 2τ is taken, so that we will estimate the expectation of an observable f as

$$\langle f \rangle = \sum_{i=0}^{\frac{t_{max}-t_{eq}}{2\tau}} f(t_{eq} + 2i\tau) \tag{4.11}$$

which means that we take $n = \frac{t_{max}-t_{eq}}{2\tau} + 1$ measurements for the estimation. A naive estimate of the autocorrelation time could be the value for which $\chi(\tau) = \frac{1}{e}$. However, a better alternative is to use the *integrated autocorrelation time*. If we suppose that the autocorrelation function follows the shape (4.10), then:

$$\int_0^\infty \frac{\chi(t)}{\chi(0)} dt = \int_0^\infty e^{-t/\tau} dt = \tau \tag{4.12}$$

where we have displaced $t \to t - t_{eq}$, so that we start the calculation of χ after the thermalization. Note that we have normalized the correlation function to its initial value, which is actually 1. Of course, one would actually integrate from 0 to t_{max} instead of ∞. However, as we can already see at Fig. 5, the real behaviour of the autocorrelation function is a divergence for high values. Then, a cutoff M is set. In this case, for a rough estimate, we use $M = 200$, and the integrated correlation time can be calculated as:

$$\tau_{int} = \int_0^M \frac{\chi(t)}{\chi(0)} dt \tag{4.13}$$

This integration has of course to be estimated numerically, since the data is actually discrete. We have estimated it with the trapezoid rule. Of course, the autocorrelation function (4.9) also requires numerical integration for every t between t_{eq} and M. However, the Fourier transform can be used to diminish computation time, since

$$\widetilde{\chi}(\omega) = \int dt\, e^{i\omega t} \int dt' \, [\mathcal{M}(t') - \langle \mathcal{M} \rangle][\mathcal{M}(t'+t) - \langle \mathcal{M} \rangle] =$$

$$= \int dt \int dt' \, e^{-i\omega t'} \, [\mathcal{M}(t') - \langle \mathcal{M} \rangle] \, e^{i\omega(t+t')} \, [\mathcal{M}(t'+t) - \langle \mathcal{M} \rangle] =$$

$$= \widetilde{\mathcal{M}}'(\omega) \, \widetilde{\mathcal{M}}'(-\omega) = \left| \widetilde{\mathcal{M}}'(\omega) \right|^2 \tag{4.14}$$

where $\widetilde{\mathcal{M}}'(\omega)$ is the Fourier transform of $\mathcal{M}'(t) \coloneqq \mathcal{M}(t) - \langle \mathcal{M} \rangle$. Then, we can use the Fast Fourier Transform (FFT) to calculate $\widetilde{\mathcal{M}}'(\omega)$, plug it in on (4.14) to calculate $\widetilde{\chi}(\omega)$ and then

get $\chi(t)$ by using the inverse FFT.

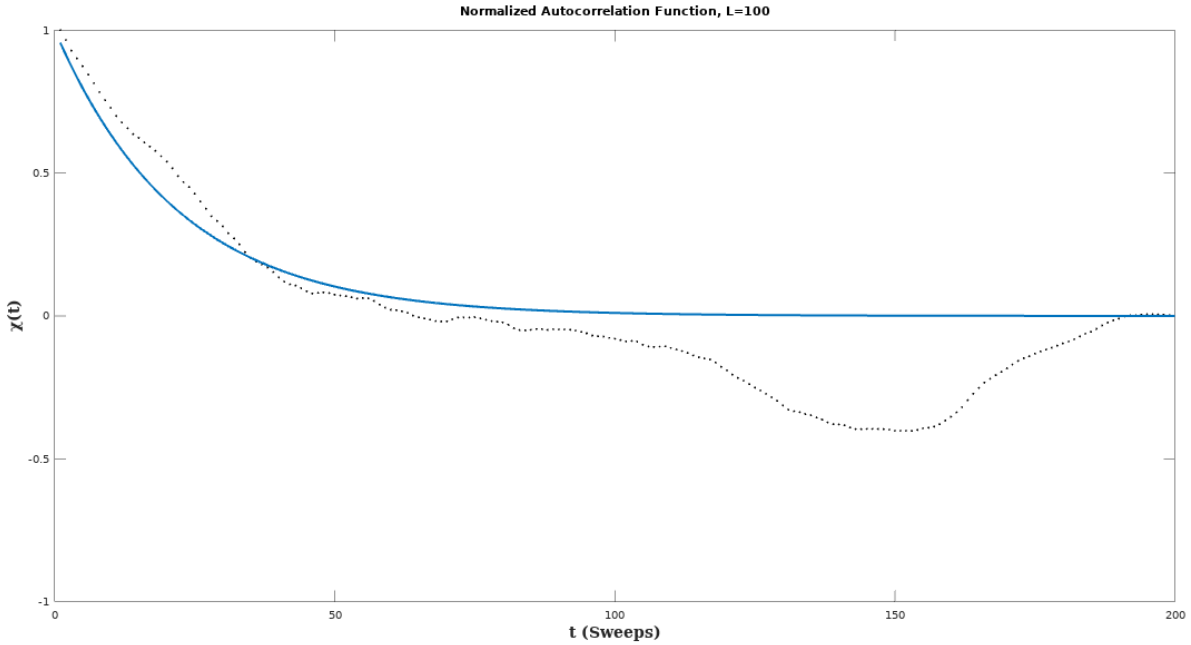


Figure 5: Normalized autocorrelation function for $L = 100$, $D = 100$, $\beta = 0.46905$. The markers represent its value at the data points, whose deviation from exponential behaviour can be observed when time is incremented. The line is the representation of $\exp(t/\tau_{int})$. For values around $\tau = 43.97$, the exponential behaviour of $\chi(t)$ is visible.

4.4.5 Interface Shift Method

In section 3.3, we introduced the width of the interface as (3.18). To determine this width with Monte Carlo methods, we need a parameter that defines the interface profile. The notation is not accidental, and the local magnetization $\mathfrak{m}$ is the most suitable one. Nonetheless, as stated in 4.4.3, the interface position is arbitrary and, since it will move up and down during the simulation (as seen on Fig. 2), averaging over time would make no sense and would produce mistaken results.

Because of this, the following *interface shift method* is applied. First, the position of the center of the interface has to be found. We can say that this center is the point in which the local magnetization is crossing 0. We would expect that all the y-points above the center of the interface have positive (or negative) local magnetization and that the y-points below it all have negative (or positive) local magnetization. Then, inverting the local magnetization of the points below it and then summing over all points would actually give a maximum for the following function:

$$g\left(\underline{k}\right) = \sum_{j=0}^{D-1} k_j \mathfrak{m}(j) \tag{4.15}$$

where $\underline{k} = \left(k_j\right)_{j=0}^{D-1}$ is a vector with $k_j \in \{-1, 1\}$ that determines whether the sign of the local magnetization is inverted in each point. The maximum of this function would ideally give $\mathcal{M}$, as in (4.8). Then, the first step of the method actually consists on evaluating the function (4.15) for vectors $\underline{k}^i$, with $i \in 0, 1, ..., D-1$, where the i first elements of $\underline{k}^i$ are -1, and the rest are 1. For this set of vectors, we look for l such that

$$g\left(\underline{k}^l\right) = \max_{0 \leq i \leq D-1} \left| g\left(\underline{k}^i\right) \right| \tag{4.16}$$

The absolute value is taken because we may have switched every point either to negative or positive values, depending on whether the positive domain is below or above the interface in our system.

The position of the interface will then be $l + \frac{1}{2}$, and we can then shift the points $\frac{D}{2} - l$ steps. Of course, those points y shifted beyond $D - 1$ or 0, would be identified with the points $y \pmod{D-1}$. Below, in Fig. 6, an example of the application of this method can be seen.

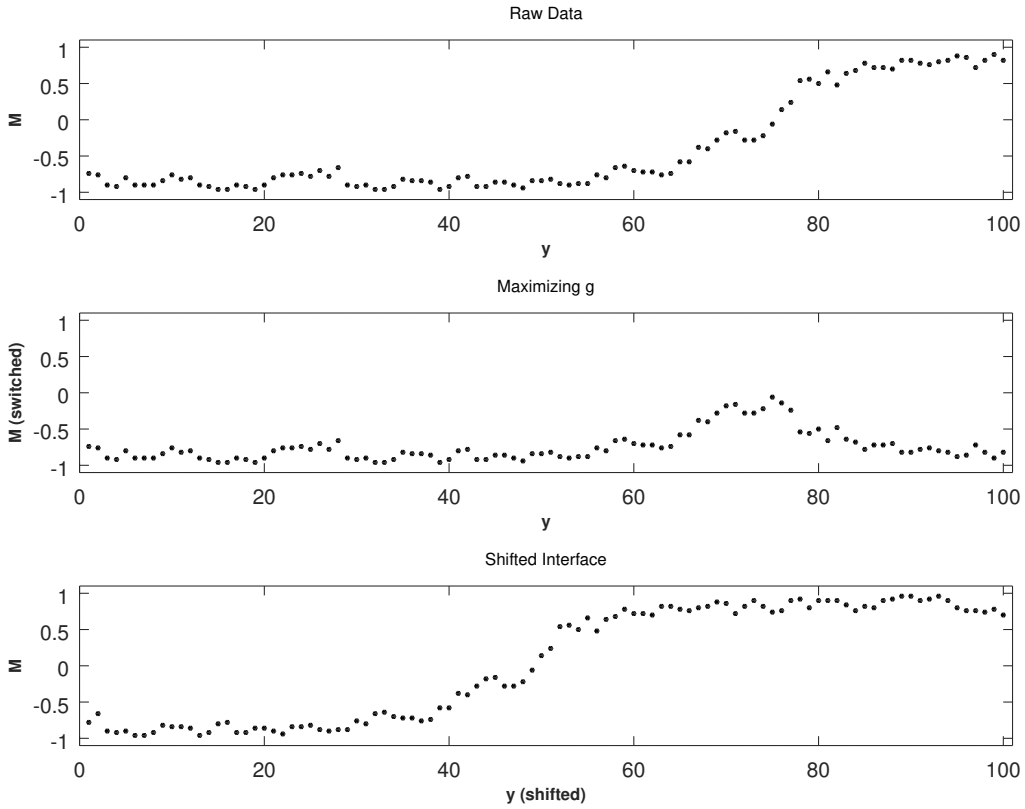


Figure 6: An example of the interface shift method, performed on the 150x100 lattice, $\beta = 0.46905$, for the data obtained after 50000 sweeps.

Once this shift is performed, the local magnetization can be averaged for each y, as

$$\langle \mathfrak{m}(y) \rangle = \sum_{i=0}^{n} \mathfrak{m}(t_{eq} + 2i\tau) \tag{4.17}$$

After the profile is "smoothened" by this method, the width can be computed. First, the weight function is calculated as

$$P(y) = \langle \mathfrak{m}(y+1) \rangle - \langle \mathfrak{m}(y) \rangle \tag{4.18}$$

Then, the squared width will be the second central moment associated to this weight function

$$w^2 = \frac{\sum_{y=0}^{D-1} y^2 \, P(y)}{\sum_{y=0}^{D-1} P(y)} - \left[\frac{\sum_{y=0}^{D-1} y \, P(y)}{\sum_{y=0}^{D-1} P(y)} \right]^2 \tag{4.19}$$

4.4.6 Error Analysis. Jackknife Method.

The most straight-forward way to calculate the error of a set of samples is to calculate the standard deviation on the mean as

$$\sigma(y) = \sqrt{\frac{1}{n-1} \left(\langle \mathfrak{m}^2(y) \rangle - \langle \mathfrak{m}(y) \rangle^2 \right)} \tag{4.20}$$

We would calculate this error for every point y, and then propagate the error to get the error of the width. However, this expression is not correct at all. First, the data samples are not completely uncorrelated, although our estimation of the autocorrelation time diminished the correlation significantly. Second, and most important, the value of $\langle \mathfrak{m}(y) \rangle$ will be correlated to the value of every other point $\langle \mathfrak{m}(\psi) \rangle$. Then, the propagation of uncertainty should include the covariance, a magnitude we cannot easily calculate.

As an alternative, we use the *jackknife method* [5]. This method consists on calculating the n parameters $\hat{w}_k^2$, $k \in \{0, 1, ..., n-1\}$, defined as the value of the width if we leave out the data point k:

$$\langle \hat{\mathfrak{m}}_k(y) \rangle = \left(\sum_{i=0}^{n-1} \mathfrak{m}(t_{eq} + 2i\tau) \right) - \mathfrak{m}(t_{eq} + 2k\tau) \tag{4.21}$$

$$\hat{P}_k(y) = \langle \hat{\mathfrak{m}}_k(y+1) \rangle - \langle \hat{\mathfrak{m}}_k(y) \rangle \tag{4.22}$$

$$\hat{w}_k^2 = \frac{\sum_{y=0}^{D-1} y^2 \, \hat{P}_k(y)}{\sum_{y=0}^{D-1} \hat{P}_k(y)} - \left[\frac{\sum_{y=0}^{D-1} y \, \hat{P}_k(y)}{\sum_{y=0}^{D-1} \hat{P}_k(y)} \right] \tag{4.23}$$

This method provides us with both an estimate of the statistical error and a bias for the mean. If we define the jackknife estimator $\hat{w}_{(.)}^2$ as follows,

$$\hat{w}^2_{(.)} = \frac{1}{n}\sum_{k=0}^{n-1} \hat{w}^2_k \tag{4.24}$$

the variance (statistical error squared) is given by

$$\hat{\sigma}^2 = \frac{n-1}{n}\sum_{k=0}^{n-1}\left(\hat{w}^2_{(.)} - \hat{w}^2_k\right)^2 \tag{4.25}$$

and the bias for the mean is given by

$$\hat{B} = (n-1)\left(\hat{w}^2_{(.)} - w^2\right) \tag{4.26}$$

so that the final jackknife estimate for the mean would be:

$$w^2_J = w^2 + \hat{B} = n\,w^2 - (n-1)\,\hat{w}^2_{(.)} \tag{4.27}$$

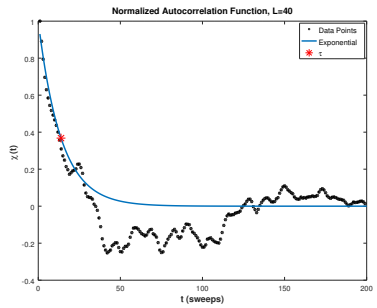
Nonetheless, the bias $\hat{B}$ will end up being so small, in comparison with the first estimation of the width, w^2, that the jackknife estimation will have no practical difference with the first estimation. The real purpose for the jackknife method is the estimation of the variance as in (4.25).

5 Numerical Results

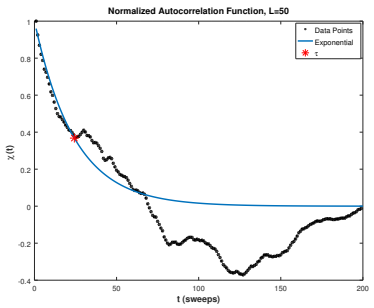
As we have stated in previous sections, our simulations are performed with the Metropolis algorithm, at an inverse temperature $\beta = 0.46905$. In the vertical direction, the lattice has a length of $D = 100$, and antiperiodic boundary conditions are implemented in this direction. In the horizontal direction, the length of the lattice varies from $L = 40$ to $L = 200$ in steps of 10. The number of sweeps was 700000 for every simulation. However, in some cases, as discussed in 4.4.3, the system would get to a metastable state, which would not allow for it to get to thermalization fast enough to have a decent amount of data. In the case that this happened, the simulation would be repeated.

In Fig. 7, the autocorrelation function is plotted for every value of L, both the experimental data points and the function $\exp(-t/\tau_{int})$ can be seen, as well as the point corresponding to τ_{int}. We can observe the good behaviour around this points, and the big deviations with increasing computer time.

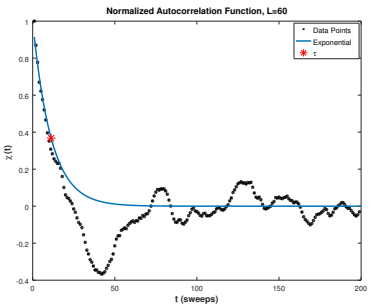
In Fig. 8, the weight function, $P(y)$ as in 4.18 used for the calculation of the squared width is also plotted for every value of L. In the plots, the function is normalized so $\sum_{y=0}^{D-1} P(y) = 1$. We can observe how the function widens out when L increases.



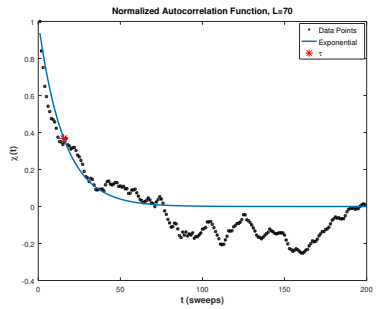
(a) $L = 40$



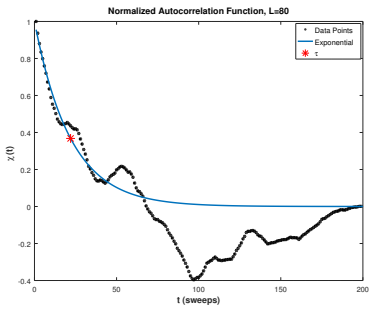
(b) $L = 50$



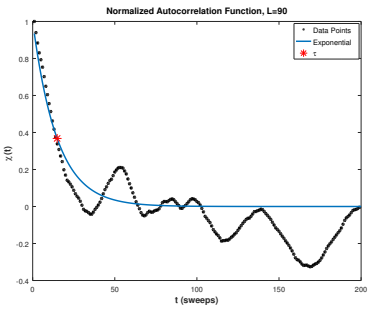
(c) $L = 60$



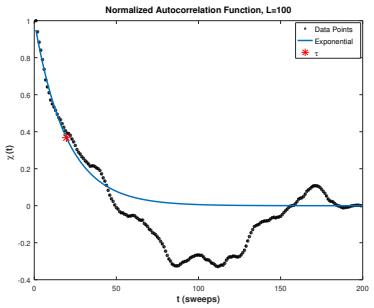
(d) $L = 70$



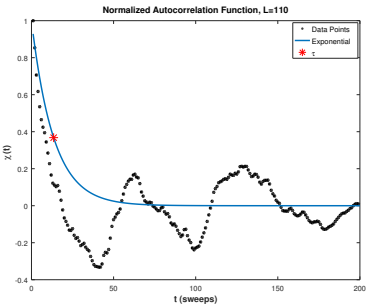
(e) $L = 80$



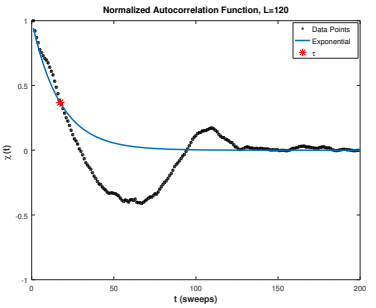
(f) $L = 90$



(g) $L = 100$

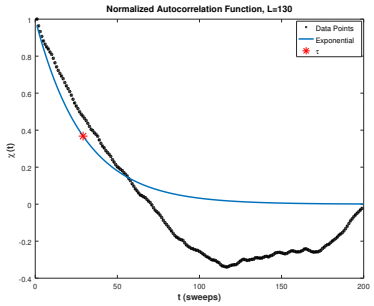


(h) $L = 110$

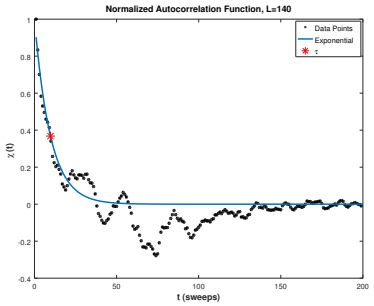


(i) $L = 120$

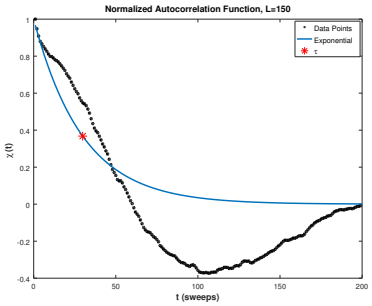
Figure 7: Autocorrelation function for different values of L, with $D = 100$ *and* $\beta = 0.46905$*. Both the experimental data points and the function* $\exp(-t/\tau_{int})$ *can be seen, as well as the point corresponding to* τ_{int}*. We can observe the good behaviour around this points, and the big deviations with increasing computer time.*



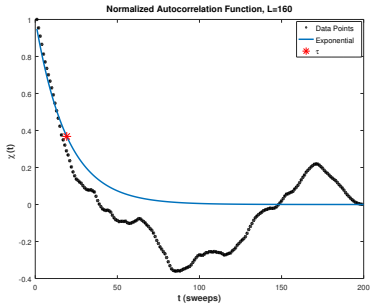
(j) $L = 130$



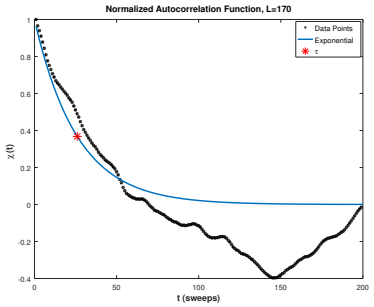
(k) $L = 140$



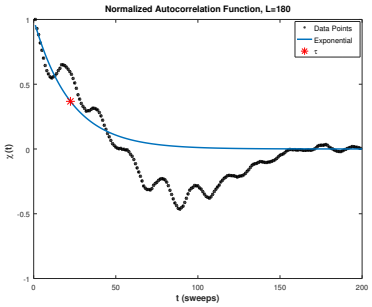
(l) $L = 150$



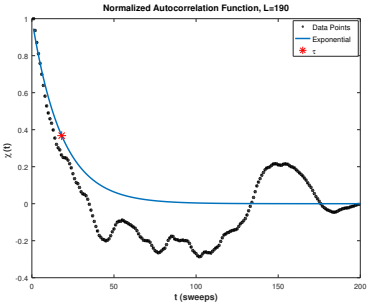
(m) $L = 160$



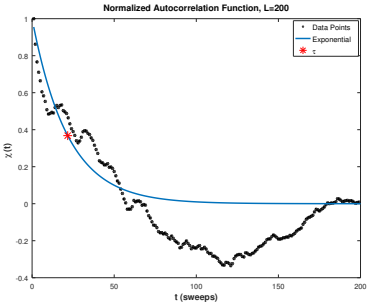
(n) $L = 170$



(o) $L = 180$

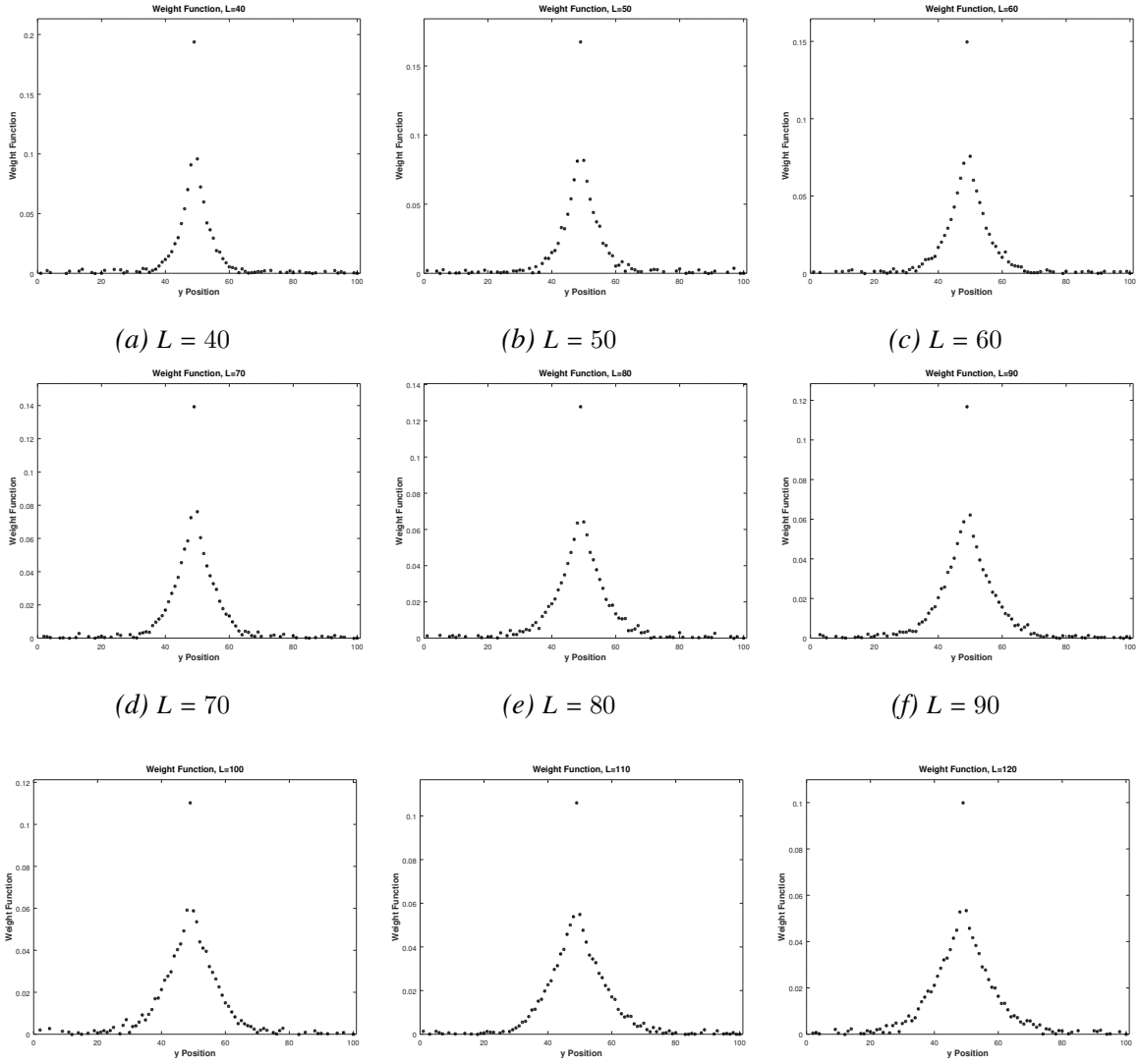


(p) $L = 190$



(q) $L = 200$

Figure 7: Autocorrelation function for different values of L, with $D = 100$ *and* $\beta = 0.46905$*. Both the experimental data points and the function* $\exp(-t/\tau_{int})$ *can be seen, as well as the point corresponding to* τ_{int}*. We can observe the good behaviour around this points, and the big deviations with increasing computer time.*

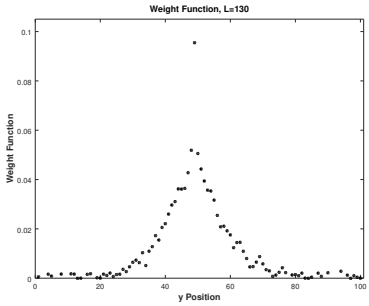


(a) $L = 40$ *(b)* $L = 50$ *(c)* $L = 60$

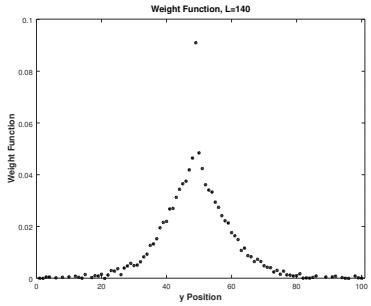
(d) $L = 70$ *(e)* $L = 80$ *(f)* $L = 90$

(g) $L = 100$ *(h)* $L = 110$ *(i)* $L = 120$

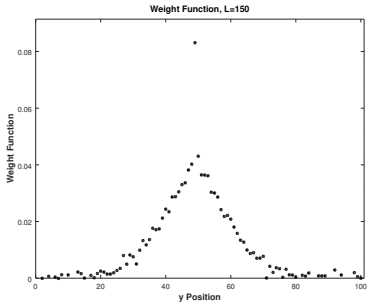
Figure 8: Normalized weight function for different values of L, with $D = 100$ *and* $\beta = 0.46905$*. We can observe how the function widens out when L increases.*



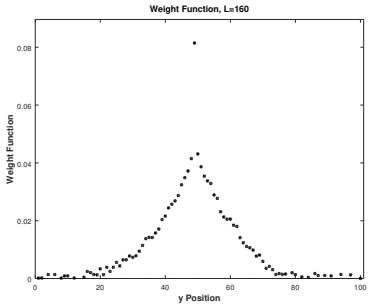
(j) $L = 130$



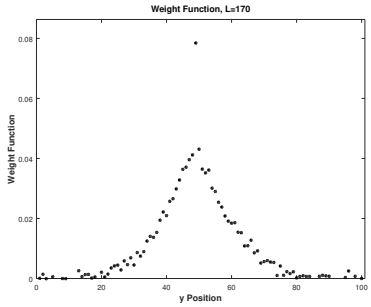
(k) $L = 140$



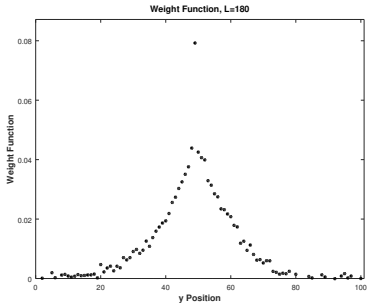
(l) $L = 150$



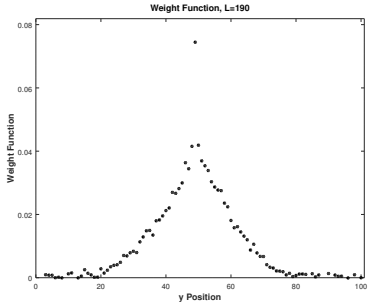
(m) $L = 160$



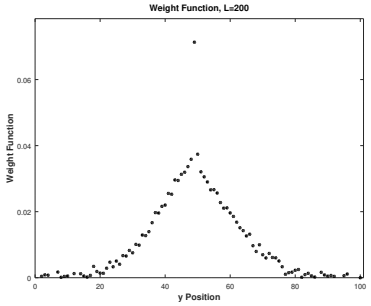
(n) $L = 170$



(o) $L = 180$



(p) $L = 190$



(q) $L = 200$

Figure 8: Normalized weight function for different values of L, with $D = 100$ *and* $\beta = 0.46905$*. We can observe how the function widens out when L increases.*

The results for the values of w^2, with their errors calculated with the jackknife method (Section 4.4.6), can be found in Table 2.

In Fig. 9, the averaged interfaces are shown for different values of L. They have already been "smoothened" by applying the interface shift method described in 4.4.5, and then the shape has been averaged point-wise in computer time. We have represented the center of the interface with a green line, and its limits with red lines. These limits are taken as $C \pm w$, where $C = \frac{D}{2}$ is the position of the center. One can observe how the squared with grows monotonically, if we consider the uncertainty interval.

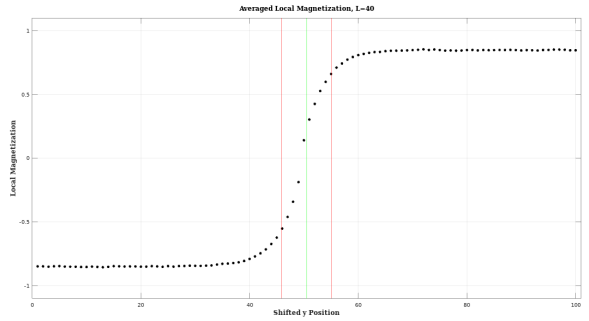
L	w^2	**L**	w^2
40	21 ± 7	130	99 ± 6
50	32 ± 8	140	98 ± 5
60	42 ± 5	150	113 ± 6
70	36 ± 5	160	116 ± 5
80	45 ± 6	170	120 ± 5
90	65 ± 5	180	127 ± 5
100	63 ± 5	190	132 ± 5
110	71 ± 4	200	148 ± 5
120	86 ± 5	-	-

Table 2: Results for the squared width on a lattice with $D = 100$, $\beta = 0.46905$. The errors are calculated with the jackknife method.

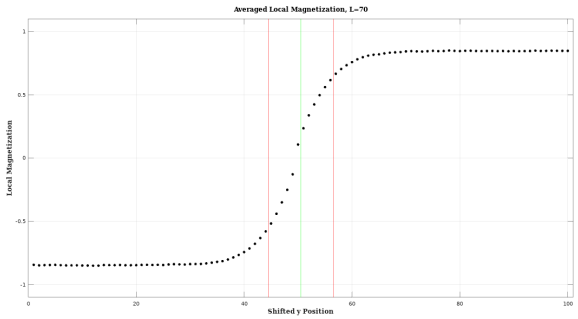
In Fig. 11, the squared width w^2 is plotted against the length of the horizontal edge L. The uncertainty of most of the values intersect the fitting line, thus we can affirm that the results have a correct behaviour. A linear regression is performed by the method of least squares method, and the results of this linear fit are shown on Table 3.

a_1	b_1	R^2
0.78 ± 0.03	-11 ± 4	0.98

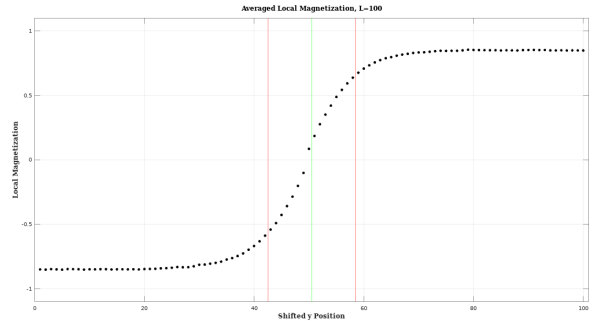
Table 3: Coefficients of the linear fit for w^2 v.s. L,, with $\beta = 0.46905$ and $D = 100$. The coefficients correspond to the equation $w^2 = a_1 L + b_1$, and R^2 is the coefficient of determination.



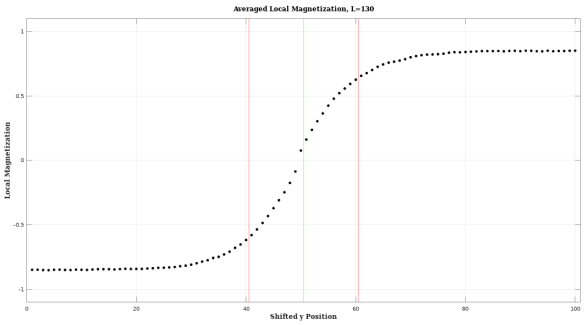
(a) $L = 40$



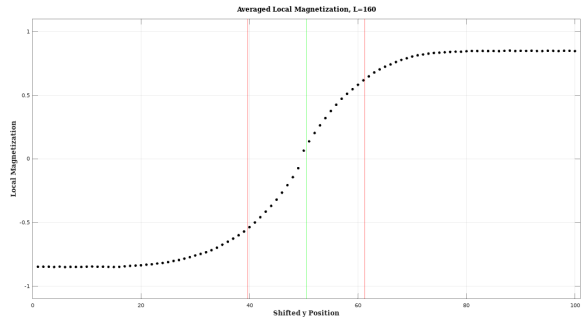
(b) $L = 70$



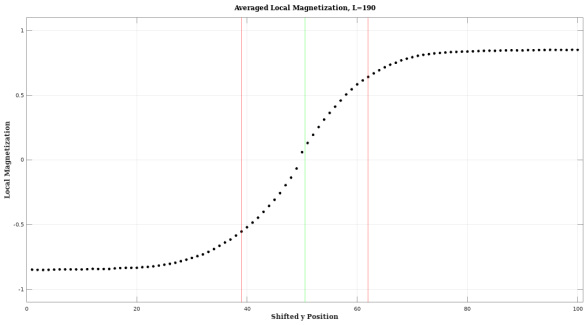
(c) $L = 100$



(d) $L = 130$



(e) $L = 160$



(f) $L = 190$

Figure 9: Averaged interfaces for different values of L, with $\beta = 0.46905$*. We can observe how the interface monotonically widens with the increase of L. We have represented the center of the interface with a green line, and its limits with red lines. Note that not every value of L has been represented.*

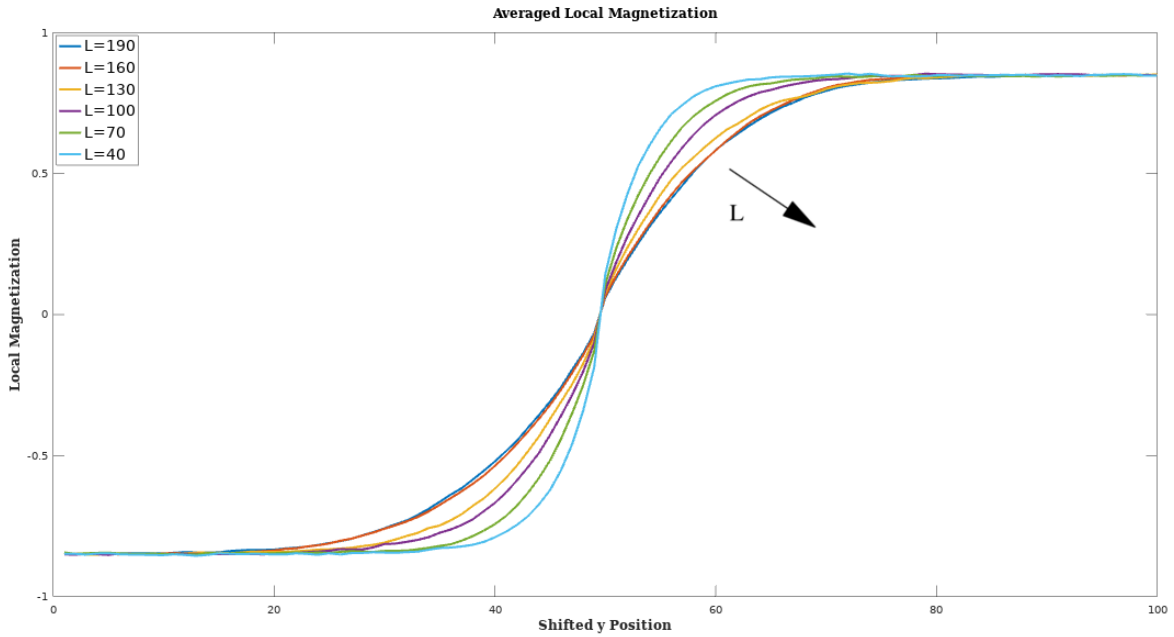


Figure 10: Comparison between the shape of interfaces with L from 40 to 190 in steps of 30. $D = 100$, $\beta = 46905$. The widening of the interface can be observed easily in this plot.

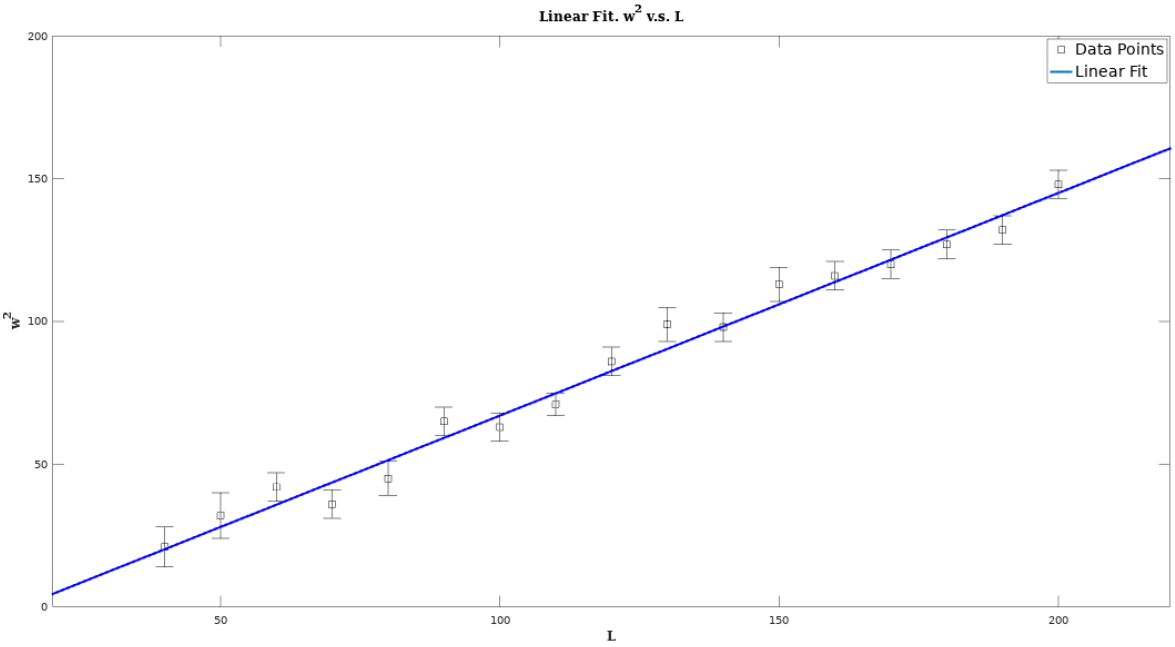


Figure 11: Linear fit of w^2 v.s L, with $\beta = 0.46905$ and $D = 100$. The uncertainty of most of the values intersect the fitting line, thus we can affirm that the results have a correct behaviour.

If we look back at (3.20), we can easily see that:

$$\frac{w^2}{\xi^2} = \frac{1}{6}\frac{L}{\xi} + \frac{\pi^2}{3} - \frac{2}{\pi q_{\max}\xi} \tag{5.1}$$

Then, if we represent $\frac{w^2}{\xi^2}$ against $\frac{L}{\xi}$ and perform a linear regression, we should get a slope equal to $\frac{1}{6}$, and an intercept equal to $\frac{\pi^2}{3} - \frac{2}{\pi q_{\max}\xi}$. The plot of this linear fit can be seen on Fig. 12, and the coefficients on Table 4, below.

a_2	b_2	R^2
0.173 ± 0.007	-0.6 ± 0.2	0.98

Table 4: Coefficients of the linear fit for $\frac{w^2}{\xi^2}$ v.s. $\frac{L}{\xi}$, with $\beta = 0.46905$ and $D = 100$. The coefficients correspond to the equation $\frac{w^2}{\xi^2} = a_2\frac{L}{\xi} + b_2$, and R^2 is the coefficient of determination.

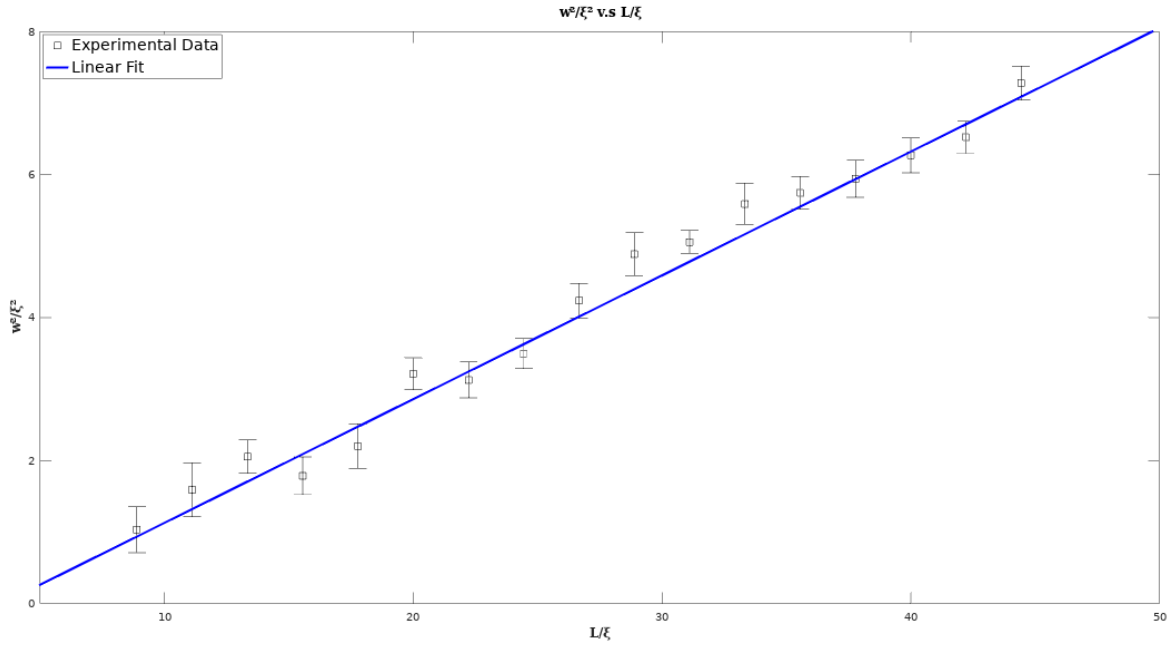


Figure 12: Linear fit of $\frac{w^2}{\xi^2}$ v.s $\frac{L}{\xi}$, with $\beta = 0.46905$ and $D = 100$. Of course, the uncertainty of most of the values keep intersecting the fitting line.

We can compare the value of a_2 with the value of the theoretical slope, which is:

$$\boxed{\frac{1}{6} \approx 0.167}$$

This is a result that lies inside our uncertainty, which is smaller than a 5% of the value of the measure. Thus, we can affirm that the estimation and the analytic result have a good concordance. Now, the value of b_2 on Table 4 can be used to estimate the cutoff q_{max}, as

$$q_{max} = \frac{6}{\pi(3b - \pi^2)}\frac{1}{\xi} = c\frac{1}{\xi} \tag{5.2}$$

and this computation gives:

$$\boxed{c \approx 0.164 \pm 0.008}$$

This provides us with a cutoff of our convolution approximation between mean field theory and capillary wave theory, inversely proportional to ξ, which gives us an idea of the scale at which we can apply this approximation.

6 Conclusion

During the thesis, we were hopefully able to make a brief and concise introduction to the Ising model and Monte Carlo methods, two important pillars that anybody who intends to work on statistical physics and computer simulations should be familiar with. We discussed all the main points of the implementation of the Metropolis algorithm, specially applying it to the Ising model, making the comments necessary to implement antiperiodic boundary conditions.

An approach to the calculation of profile interfaces using mean field approximation and capillary wave theory in the context of Quantum Field Theory was also made. This whole theory falls a little bit out of the scope of a bachelor thesis, and it was made without numerous details. Nonetheless, it will positively encourage the reader to deepen its knowledge in this field, as it has encouraged the author.

Finally, after implementing and running the simulations, the analytic estimation has proven to be in accordance with the numerical results, and a scale of its validity has been found. In the process, we have learned the actual difficulties of writing and running a simulation, and the procedure used to process data in order to get the results.

References

[1] B. McCoy, T.T. Wu: THE TWO-DIMENSIONAL ISING MODEL,
Harvard, 1973

[2] M. Müller, G. Münster: PROFILE AND WIDTH OF ROUGH INTERFACES,
J. Stat. Phys. 118 (2005) 669.

[3] G. Münster: Private communication.

[4] S. Friedli, Y. Velenik: STATISTICAL MECHANICS OF LATTICE SYSTEMS, **CH.2:** THE CURIE-WEISS MODEL,
Cambridge University Press, 2017

[5] M.E.J. Newman, G.T. Barkema: MONTE CARLO METHODS IN STATISTICAL PHYSICS
Oxford University Press, 1999

[6] V. Privman, P.C. Hohenberg, A. Aharony: UNIVERSAL CRITICAL-POINT AMPLITUDE RELATIONS, in
PHASE TRANSITIONS AND CRITICAL PHENOMENA, VOL. 14, eds.: C. Domb, J.L. Lebowitz,
Academic Press, 1991

[7] I.S. Chappel II: CALCULATION OF INTERFACE TENSION AND STIFFNESS IN A TWO DIMENSIONAL ISING MODEL BY MONTE CARLO SIMULATION,
MIT, 1998

[8] W. Krauth: STATISTICAL MECHANICS: ALGORITHMS AND COMPUTATIONS,
Oxford University Press, 2006

[9] M.P Gelfand, M.E. Fisher: FINITE SIZE EFFECTS IN FLUID INTERFACES,
PHYSICA A 166 (1990), 1-75

Appendix A C Code

```
#include <limits.h>
#include <stdio.h>
#include <stdlib.h>
#include <time.h>
#include <math.h>
#include <string.h>

//Metropolis algorithm for J=1, 2-dimensional lattice

#define T 100 //length of the lattice y-edge
#define L 190  //length of the lattice x-edge
#define N L*T //Number of spins ofc
#define tmax 70000 //Number of sweeps

int s[N]; //Array of spins
float beta; //Beta ~ inverse temperature
float aprob[5]; //acceptance probabilities
char optinf, optbc; //Options
char direct[3];
unsigned short int seed16v[3]; //Seed for random numbers
int mag[tmax+1]; //Array of magnetizations
int t, n, m, i; //Aux variables
int tstep=1; //Steps to write magnetization and represent lattice
FILE *p;

void spins0(){ //Initialization of the spins. T=0
  int i;
  double p;
    for(i=0;i<N;i++){
      s[i]=1;
    }
}

void spinsinf(){ //Initialization of the spins. T=inf
  int i;
  double p;
  for(i=0;i<N;i++){
    p=drand48();
    if(p<0.5) {
      s[i]=1;
    } else s[i]=-1;
```

```
  }
}

void initialize() //Creates the table of acceptance probabilities exp(-be
{
  int i;
  for (i=2; i<5; i+=2) aprob[i] = exp(-2*beta*i);
}

// double drand(){ //Gives us a random number in [0,1]
//
//      return (double)rand() / (double)RAND_MAX ;
//}

void sweep() // Makes a sweep, i.e., flips N random spins
{
  int i, k;
  double ra, rara;
  int nn, sum, delta;
  double prob;
  double randi;

  for (k=0; k<N; k++){
    // Choose a site
    randi=drand48();
    ra=drand48();
    rara=trunc(N*ra);
    i=(int) rara;
    //Calculate sum of neighbourhood spins

    nn=i+L;             //Upper neighbour
    if (nn>=N) {
      nn-=N;
      if (optbc=='y'){
        sum=-s[nn];
      } else {
        sum=s[nn];
      }
    } else {sum=s[nn];
    }

    nn=i-L;             //Lower neighbour
    if (nn<0) {
```

```
      nn+=N;
      if (optbc=='y'){
        sum-=s[nn];
      } else {
        sum+=s[nn];
      }
    } else {sum+=s[nn];
    }

    nn=i+1;       //Right neighbour
    if ((nn%L)==0) nn-=L;
    sum+=s[nn];

    nn=i-1;                       //Left neighbour
    if ((nn%L)==(L-1)) nn+=L;
    sum+=s[nn];

    //Calculate the change in energy

    delta=sum*s[i];
    prob = aprob[delta];

    //Decide whether to make the flips

    if (delta<=0) {
      s[i] = -s[i];
    } else if (randi<prob) {
      s[i]= -s[i];
    }
  }
}

void writemag(){  //Write magnetization in file, with T_0=inf
  FILE *p;
  remove("mag.txt");
  p=fopen("mag.txt","w+");

  fprintf(p,"[");

  fprintf(p,"%i",0);
  for(t=0;t<tmax;t+=tstep){
    fprintf(p,",%i",t+1);
  }
```

```
  fprintf(p,"\n");

  fprintf(p,"%i",mag[0]);
  for(t=0;t<tmax;t+=tstep){
    fprintf(p," %i",mag[t+1]);
  }
  fprintf(p,"]");
  fclose(p);
}


void writelatt(int num){ //Write lattice in FILE
  FILE *p;
  char name[20];
  sprintf(name,"%i",num);
  strcat(name,".txt");

  remove(name);
  p=fopen(name,"w+");
  for (n=T;n>0;n--){
    for (m=0;m<L;m++){
      fprintf(p,"%i ",s[(n-1)*L+m]);
    }
   fprintf(p,"\n");
  }
  fclose(p);

}

void localmag(int num){ //Write local magnetization
  FILE *p;
  char name[20];
  int lmag[T];
  sprintf(name,"%i",num);
  strcat(name,"mag.txt");

  remove(name);
  p=fopen(name,"w+");
  for (n=0;n<T;n++){
    lmag[n]=0;
    for (m=0;m<L;m++){
      lmag[n]+=s[L*n+m];
    }
```

```
    fprintf(p,"%i ",lmag[n]);
  }
  fclose(p);

}

  // MAIN

int main(){

  seed16v[0]=(unsigned)time(NULL);  //Seeding the random number generator
  seed48(seed16v);

  printf("Beta?\n>");
  scanf("%f",&beta);
  printf("\n");

  printf("Enter y for Hot Start, otherwise Cold Start\n>");
  scanf(" %c",&optinf);
  printf("\n");

  printf("Enter y for Antiperiodic BC, otherwise Periodic BC\n>");
  scanf(" %c",&optbc);
  printf("\n");

       //Initialization of the spins
  if(optinf=='y'){
    spinsinf();
  } else{
    spins0();
  }

  mag[0]=0;
  writelatt(0);
  for (n=T;n>0;n--){
    for (m=0;m<L;m++){
      mag[0]=mag[0]+s[(n-1)*L+m-1];
    }

  }


                    //Initialization of the acceptance probabilities
```

```
    initialize();

                                //Sweeping

    for(t=0;t<tmax;t+=1){
      sweep();
      //Write lattice state and calculate magnetization

      if (t%tstep==0){
        writelatt(t+1);
        localmag(t+1);
        mag[t+1]=0;
        for (n=T;n>0;n--){
          for (m=0;m<L;m++){
            mag[t+1]+=s[(n-1)*L+m];
          }
        }
     }

    }

    //CREATING A FILE WITH THE MAGNETIZATION

    writemag();
    return 0;

}
```