Classical Monte Carlo and the Metropolis Algorithm: Revisiting the 2D Ising Model

Dominic Marchand
Department of Physics and Astronomy, University of British Columbia, Vancouver, BC, V6T 1Z1
(Dated: November 30, 2005)

Monte Carlo (MC) simulations are among the most powerful numerical tools to investigate very large systems. This review covers the fundamental principles of the Metropolis algorithm. Related topics like the determination of error bars and the autocorrelation time to measure how the algorithm will converge are also treated. The two dimensional Ising model and its behavior at the phase transition is then studied with a minimal implementation of the tools described herein.

I. INTRODUCTION

Monte Carlo (MC) simulations are powerful and versatile numerical tools used to study large systems. They are universal algorithms that can be applied to various systems, but they are especially useful to study systems with a large number of coupled degrees of freedom (liquids, disordered materials, ...). In fact, their relative efficiency with respect to other numerical methods increases with the dimension of the problem. MC simulations, like neural networks and genetic algorithms, are stochastic, that is nondeterministic and rely heavily on the use of random or pseudo-random numbers.

MC encompasses a very broad variety of algorithms and variations, from what is what is generically called statistical sampling, to the more recent diagrammatic quantum MC. The scope of this review only makes it possible to consider one of these: the Metropolis algorithm (MetA).

The Ising model in two dimensions will serve both as a case study and an example in the following review. The reasons for choosing this model are two-fold. First, it is an easy enough problem so that an analytical solution exists, thus allowing for a validation of our implementation. Second, despite its simplicity, the Ising model can still provide a good description of many real systems with a phase transition, especially crystals for which the exchange interaction is very anisotropic ($FeCl_2$ and $FeCO_3$). Nevertheless, it should be emphasized that a theoretical description of phase transitions is very difficult and can sometimes only be probed with numerical tools.

II. ISING MODEL

Before going on and introducing MC algorithms, a few words on the model considered are in order. This model was first proposed by Lenz (1920) to study the phase transition of ferromagnets at the Curie Temperature. It was later fully worked out by his pupil Ising for the one dimensional case (1925) and by Onsager 20 years later for the 2 dimensional case.

The Ising hamiltonian is given by

$$H = -J \sum_{<ij>} \sigma_i \sigma_j - \mu B \sum_i \sigma_i. \tag{1}$$

Only nearest-neighbor interactions are considered where J is the coupling constant and $\sigma_i = \pm 1$ are the two spin states allowed on each lattice site. For this study the magnetic field will be considered to be zero since no analytic solution exists when a field is present.

For a square lattice of size $N = L^2$ the magnetization and magnetization density are readily found to be

$$M = \sum_i \sigma_i; \quad \mathcal{M} = \frac{1}{L^2} \sum_i \sigma_i. \tag{2}$$

At a phase transition, some symmetry is broken and one need to introduce one or more variables to decribe the state of the system. Such a variable is an order parameter. For an infinite system, the 2D Ising model predicts a magnetic phase transition at T_c as shown on Figure 1 and the order parameter is the magnetization density.

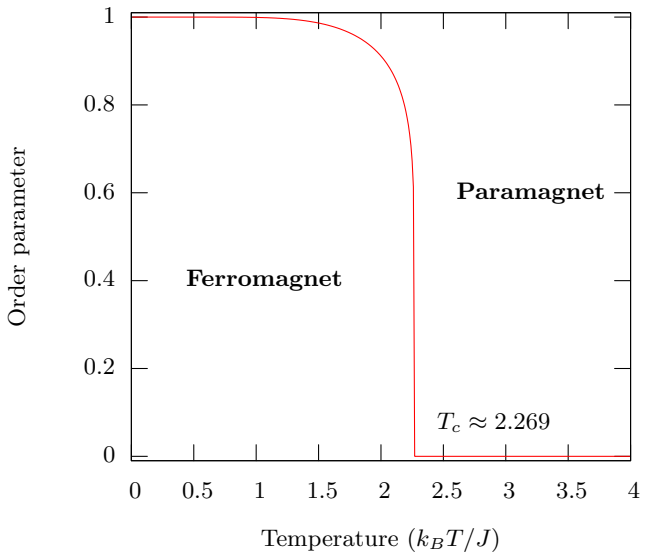

FIG. 1: Phase diagram of the 2D Ising model

We will assume that $k_BT \to T$ such that T has units of energy like J. The critical temperature can be calculated

analytically to be

$$T_c = \frac{2J}{\ln 1+\sqrt{2}} \approx 2.269J, \tag{3}$$

The magnetization density, like other thermodynamic quantities, behaves singularlyat T_c as

$$\mathcal{M} \sim |T - T_c|^{\beta}; \quad \beta = 1/8. \tag{4}$$

For a finite-size system, extrapolation to the thermodynamic limit by finite size scaling will be necessary to capture the critical behaviour. This will be explained later in this review.

III. CLASSICAL MONTE CARLO

In MC, we are interested in evaluating the ratio of two N-dimensional sums or integrals.

$$\begin{aligned} \langle A\rangle &= Z^{-1}\sum_{i_1}\cdots\sum_{i_N} A(i_1,\ldots,i_N)W(i_1,\ldots,i_N); \\ Z &= \sum_{i_1}\cdots\sum_{i_N} W(i_1,\ldots,i_N), \end{aligned}$$

where A and W are arbitrary. To simplify we define a configuration ν as the collection of all summation indices: $\nu \equiv \{i_1,\ldots,i_N\}$. We can then rewrite

$$\langle A\rangle = \sum_{\nu}\frac{A_\nu W_\nu}{Z}; \quad Z = \sum_{\nu} W_\nu. \tag{5}$$

We also can define the weight of a configuration ν as

$$p_\nu = \frac{W_\nu}{Z}. \tag{6}$$

In statistical mechanics, where we call Z the partition function, this weight is related to the energy of the configuration with $p_\nu = Z^{-1}\exp(-E_\nu)$ and $W_\nu = \exp(-E_\nu)$ is given by the normalized Boltzmann distribution.

A very demonstrative example of how MC works is the evaluation of the volume of a multidimensional body whose surface is defined as $F_\nu = 0$ and where $F_\nu < 0$ holds inside the body. One can then write an expression for the volume using the step function and a^N as the volume of the configuration space:

$$\begin{aligned} V_{body} &= \int dx_1 \ldots \int dx_N \Theta\big(-F_\nu\big), \\ V_{body} &\equiv a^N\big\langle\Theta\big(-F_\nu\big)\big\rangle; \qquad \nu \in V_0. \end{aligned} \tag{7}$$

So we have rewritten the volume in the form of (5) with $W_\nu = 1$. Most problems in MC can be viewed as an evaluation of the volume of a body in N-dimensional space. If W_ν is not constant, the problem can be viewed as evaluating the *mass* of a N-dimensional body where the mass density is allowed to vary. This is true even for the more advanced algorithms like diagramatic quantum Monte Carlo which is nothing more than summing weighted diagrams over the space of all possible diagrams considered.

In large problems, N can be relatively big making it impossible to add all the terms in the sum. Just consider the case where each index can take 2 values like for the magnetization. If $N = 100$, this makes $2^{100} \sim 10^{30}$ terms to sum making it virtually impossible to complete the sum over one lifetime. At this point getting an exact answer with this technique is hopeless. However, two important considerations make MC simulations still useful by ensuring that a good estimate can be calculated with high accuracy

1. Most of the time $\langle A\rangle$ does not contain a lot of information about $\{\nu\}$ and will only depend on the average of some quantity. If we have a representative set of $\{\nu\}$ we can get a very accurate estimate for $\langle A\rangle$. For the magnetization $\langle|M|\rangle$ for example, a system of N spins can only have $\frac{N}{2}-1$ different values. Therefore, a representative set for this system will be of order N.

2. Despite the fact that what we are summing is a product of two quantities $A_\nu W_\nu$, in many cases (in statistical physics at least) the structure of W will be finite only in a small region. That is, even if the number of configurations outside this region is very large, if their weight is exponentially small the answer will depend almost exclusively on a tiny fraction of the entire parameter space.

In other words, if we select a representative set of configurations to sum, picking them at random, we can get a good estimate for $\langle A\rangle$. Even better, if we can select configurations such that the ones with bigger weights are more likely to be chosen, then we can get an equivalent answer by summing fewer terms.

A. Metropolis algorithm

The Metropolis algorithm basically consists in replacing the sums in (5) by stochastic sums such that for an infinitely long computation we have

$$\frac{\sum'_\nu A_\nu}{\sum'_\nu} \to \frac{\sum_\nu A_\nu W_\nu}{\sum_\nu W_\nu}. \tag{8}$$

This means that random configurations are generated and are accepted or not in the sum according to some probability. Two conditions must be fulfilled for (8) to hold:

1. The probability to accept a configuration ν must be proportional to its weight W_ν

2. The stochastic generation of configurations must be ergodic. That is, in a very long simulation, all possible configurations must be eventually generated.

Defining N_ν to be the number of times ν was summed, (8) becomes

$$\frac{\sum'_\nu A_\nu}{\sum'_\nu} = \frac{\sum'_\nu A_\nu N_\nu}{\sum'_\nu N_\nu} \to \frac{\sum_\nu A_\nu W_\nu}{\sum_\nu W_\nu} \tag{9}$$

Metropolis Algorithm

1. Initialize $Sum\_A = 0$, $Z = 0$ and $N_\nu = 0$ for each ν
   Generate a first configuration ν
2. Include the configuration ν in the sum:
   $Sum\_A = Sum\_A + A_\nu$
   $Z = Z + 1 \qquad N_\nu = N_\nu + 1$
3. Suggest another configuration ν' from ν (update)
4. Accept ν' on average $\frac{W_{\nu'}}{W_\nu}$ times
   Accept: $\nu = \nu'$ Reject: $\nu = \nu$
   Go back to step 2

Step 4 of this algorithm needs elaboration. It should first be shown that accepting a configuration on average $\frac{W_{\nu'}}{W_\nu}$ times will enforce (9) after an infinitely long computation.

$$\begin{aligned}
N_{\nu'} &= N_\nu \tfrac{W_{\nu'}}{W_\nu} &&= W_{\nu'} \frac{N_\nu}{W_\nu}, \\
N_{\nu''} &= N_{\nu'} \tfrac{W_{\nu''}}{W_{\nu'}} &&= W_{\nu''} \frac{N_\nu}{W_\nu}, \\
N_{\nu'''} &= N_{\nu''} \tfrac{W_{\nu'''}}{W_{\nu''}} &&= W_{\nu'''} \frac{N_\nu}{W_\nu}, \\
&\cdots
\end{aligned}$$

and through this long series of ratios we get

$$\frac{N_\nu}{W_\nu} \to \text{const}, \tag{10}$$

which will enforce (9).

To implement this algorithm one will need to calculate probabilities to accept or reject a configuration as a function of weight ratios. This is given by the so-called balance equation. To derive this relation it is more convenient to consider an infinite number of parallel processors all doing one loop of the Metropolis algorithm instead of one processor doing an infinite number of loops. When the processors execute step 3 and 4 they all produce a configuration ν' and decide whether they accept or not this update. One can then look at the flow of processors between the configurations states $\{\nu\}$. Then (10) imposes that in equilibrium the number of processors in one state N_ν should remain constant. Equivalently, all flow must cancel. This condition translates into what is called the balance equation or master equation:

$$\begin{aligned}
&N_\nu \sum_{\nu'} \sum_{u \in (\nu \to \nu')} p_u P_u^{\text{acc}}(\nu \to \nu') \\
&= \sum_{\nu'} \sum_{\overline{u} \in (\nu' \to \nu)} N_{\nu'} p_{\overline{u}} P_{\overline{u}}^{\text{acc}}(\nu' \to \nu).
\end{aligned} \tag{11}$$

The terms in the l.h.s. of this equation are the decrease in the number of processors in state ν due to update u such that u is chosen in the set of updates going from ν to ν'. It is given by the number of processors in the state ν times the probability of applying update u times the probability of accepting the update suggested by u. The terms in the r.h.s. of this equation are the increase in the number of processors in state ν due to update $\overline{u}$ such that $\overline{u}$ is chosen in the set of updates going from ν' to ν. It is given by the number of processors in the state ν' times the probability of applying update $\overline{u}$ times the probability of accepting the update suggested by $\overline{u}$. Using (10) we can rewrite

$$\begin{aligned}
&W_\nu \sum_{\nu'} \sum_{u \in (\nu \to \nu')} p_u P_u^{\text{acc}}(\nu \to \nu') \\
&= \sum_{\nu'} \sum_{\overline{u} \in (\nu' \to \nu)} W_{\nu'} p_{\overline{u}} P_{\overline{u}}^{\text{acc}}(\nu' \to \nu).
\end{aligned} \tag{12}$$

There are many possibilities to ensure the balance equation (12) is fulfilled. A complicated loop update could be used, but an easier way consists in balancing the states in pairs. We can then get rid of the summations in (1). With $\overline{u}$ being the inverse of u we can rewrite

$$\frac{P_u^{\text{acc}}(\nu \to \nu')}{P_{\overline{u}}^{\text{acc}}(\nu' \to \nu)} = \frac{W_{\nu'}}{W_\nu} \frac{p_{\overline{u}}}{p_u} = R \tag{13}$$

where R is the acceptance ratio. The choice of $P_u^{\text{acc}}(\nu \to \nu')$ and $P_{\overline{u}}^{\text{acc}}(\nu' \to \nu)$ is not unique but a convenient choice, which will ensure a high acceptance probability, is

$$\begin{aligned}
P_u^{\text{acc}}(\nu \to \nu') &= \begin{cases} R & R < 1 \\ 1 & R \geq 1 \end{cases} \\
P_{\overline{u}}^{\text{acc}}(\nu' \to \nu) &= \begin{cases} 1 & R \leq 1 \\ R & R > 1 \end{cases}
\end{aligned} \tag{14}$$

Interestingly, it is possible in some cases to play the random numbers such that $R \equiv 1$. This is called the heat bath algorithm. Basically, it simply means adjusting $\frac{p_{\overline{u}}}{p_u}$ according to (13). This provides a interesting speedup in some cases but for a discrete two-level system it often means that the update does nothing and both are equivalent.

B. Estimation of the error bars and the blocking method

As discussed previously, probing only a small fraction of the configuration space can provide accurate results since A_ν is not sensitive to the details of ν. Granted that the result is not exact, how can the accuracy of a measure be evaluated? Assuming the measure after time Δ (in steps or updates) differs from $\langle A \rangle_{\text{exact}}$ by δA_Δ.

Repeating this measurement n times will give n different averages. They are called block averages since usually one will equivalently run a very long simulation and divide it into n blocks. The block averages are denoted

as

$$\langle A\rangle_i = \langle A\rangle_{\text{exact}} + \delta A_i. \tag{15}$$

The standard deviation, or population variance, for $\langle A\rangle_i$ can be expressed as

$$\sigma_\Delta = \frac{\sum_{i=1}^{n}(\delta_{A_i})^2}{n}. \tag{16}$$

The average over the entire simulation is simply

$$\langle A\rangle = \frac{\sum_{i=1}^{n}\langle A\rangle_i}{n}, \tag{17}$$

and it has a much smaller standard deviation from the exact answer:

$$\sigma_{n\Delta}^2 = \frac{\sum_{i=1}^{n}\sigma_\Delta^2}{n^2} = \frac{\sigma_\Delta^2}{n}. \tag{18}$$

Clearly, results become more accurate with time,

$$\sigma_t = \sigma_\Delta\sqrt{\frac{1}{n}} = \sigma_\Delta\sqrt{\frac{\Delta}{t}}. \tag{19}$$

Accuracies better than 10^{-4} can usually be achieved in most calculations and with state of the art implementations can be as good as 10^{-6}.

In practice, an honest error bar is calculated with the blocking method or an equivalent technique. They all share the idea of joining blocks together in various ways to get different measures of the deviation. In the blocking method, instead of keeping $\langle A\rangle_i$'s, we keep block numerators R_i's, with each block containing Z_B accepted configurations. We then calculate $\sigma_t^{(m)}$ which is the standard deviation calculated with m superblocks each of size n/m and therefore constituted by joining n/mZ_B small blocks together. The average $\langle B\rangle_i$ of a superblock is then given by

$$\langle B\rangle_i|_{size=n/m} = \frac{m}{nZ_B}\sum_{k=1+(i-1)n/m}^{ij} R_k \tag{20}$$

and the corresponding deviation is

$$\sigma_t^{(m)} = \frac{1}{m}\sqrt{\sum_{i=1}^{m}(\langle B\rangle_i - \langle A\rangle)^2} \tag{21}$$

For large n it is possible to construct larger and larger superblocks to study the behavior of $\sigma_t^{(m)}$. This will not depend on m if the superblocks are not correlated. This means that small blocks are expected to be correlated but $\sigma_t^{(m)}$ will eventually saturate with big enough blocks. This value can then be taken to be a realistic error bar.

C. Autocorrelation time

Since configurations are usually generated by modifying more or less dramatically the previous configuration, successive block averages will not necessarily be statistically independent, thus artificially reducing the error bars calculated. The autocorrelation function is useful to evaluate this correlation. To calculate it we keep a long record of M block averages

$$\Gamma_i = \Gamma\Big(\frac{t}{\Delta t}\Big) = \frac{\sum_{j=1}^{M-i}\Big(\langle A\rangle_j - \langle A\rangle\Big) * \Big(\langle A\rangle_{j+i} - \langle A\rangle\Big)}{\sum_{j=1}^{M-i}\Big(\langle A\rangle_j - \langle A\rangle\Big)^2}, \tag{22}$$

where $j = \frac{t}{\Delta t}$. Δt is usually chosen to be the size of the system.

We also define an integrated correlation time which is basically the integral of $\Gamma(t/\Delta t)$. If Γ is of the form $\Gamma^{-t/\tau}$, the integrated correlation time will be τ. For an arbitrary function it is defined as

$$\tau = \frac{\Delta t}{2} + \sum_{i=1}^{\infty}\Gamma_i\Delta t. \tag{23}$$

In a system with N degrees of freedom we expect the correlations to persist at least up to N updates if the procedure changes only one degree at a time. This introduces a natural time unit called a sweep which corresponds to Δt updates.

IV. IMPLEMENTATION AND MONTE CARLO STUDY OF THE ISING MODEL

The results presented in this section have been produced with my own C++ implementation of the Metropolis algorithm, block method and autocorrelation technique. Many optimizations used in serious MC simulation have been ignored to keep things simple. The update procedure used is the spin-flip alogorithm which has a very long integrated autocorrelation time. A proper thermalization process, that is, an initial period during which no data are collected to eliminate artifacts due to initial conditions and algorithm details, could have been used but was ignored due to computing resources. Also, the computation time was severely limited by the computer resources available and has been set lower on purpose to show the power of MC simulations. This being said, the results obtained are nevertheless surprising and enable us to look at the critical behaviour at the phase transition.

The pseudo-random number generator used is a simple shuffled generator based on a linear congruential generator. The parameters used are presented in Table I.

TABLE I: Parameters used by the shuffled linear congruential pseudo-random number generator

Parameter	Value
m	$2^{31}-1$
a	16807
c	0
seed	1

A. The spin-flip algorithm

This algorithm consists in flipping the spin of a randomly selected site. Clearly it satisfies the ergodicity requirement. Also $p_{\overline{u}} = p_u$ in (13) reduces R to the ratio of weights. Since only one spin is modified at each update, correlation times are expected to be very long.

Spin-flip algorithm

1. Select a site at random in the lattice
2. Suggest to flip the spin

B. Average modulus of magnetization, error bar and correlation time

Figure 2 presents the average modulus of the magnetization density for 3 different system sizes ($L = 16$, 32 and 64) and many temperatures going from 0.5 to 4 (J/k_b). The simulation time is $2^{16} = 65536$ sweeps which is really small. Figure 3 shows indeed that except for temperatures very close to the ferromagnetic ground state (absolute zero), the deviation does not saturate, indicating that we should allow more time to compute the magnetization.

The autocorrelation function of Figure 4 has been calculated for a system of size $L = 16$ and $T = 0.5$, 1.7 and 4.0. The integrated correlation time is respectively $\tau = 1884$, 2896 and 4727 sweeps. This clearly indicates that within 65536 sweeps we cannot reach a very accurate result. Nevertheless, the curves from Figure 2 are still very instructive and seem to indicate that the critical temperature is between $T = 2.2$ and 2.3. This can also be seen from Figure 5 since the convergence becomes very slow close to the phase transition. This is called a critical slowdown and makes it very difficult to look at what happens near T_c. Note that we get much smoother transitions than expected (difference between Figure 2 and Figure 1). This is due to finite-size effects, not too simulations errors. We see that as the size of the system is increased, we get closer to the expected behaviour.

As mentioned previously, to extrapolate to the thermodynamic limit we need to do a finite size scaling. In other words we use results for different sizes and extrapolate for an infinite system. First, consider the correlation length (with ν now the correlation length exponent not a configuration).

$$\xi \sim |T - T_c|^{-\nu}. \tag{24}$$

The relaxation time, or the typical time for a fluctuation to relax, is related to this length:

$$\tau \sim \xi^z, \tag{25}$$

with z being the dynamical critical exponent. We mention this because the divergence of τ is what causes the convergence to slow down and what is generally called the name critical slowdown.

More interesting to us is the order parameter of the Ising model:

$$\mathcal{M} \sim |T - T_c|^{\beta} \sim \xi^{\frac{-\beta}{\nu}}. \tag{26}$$

Usually $\xi \to \infty$ at T_c but in a finite system the largest length possible is the system size L. Therefore one has at T_c:

$$\mathcal{M} \sim L^{\frac{-\beta}{\nu}}, \tag{27}$$

or

$$\mathcal{M} L^{\frac{-\beta}{\nu}} = \text{const.} \tag{28}$$

To get the β and T_c we consider this an exact equality only at the transition,

$$\frac{\ln \mathcal{M}(L)/\mathcal{M}(L')}{\ln L/L'} = -\beta/\nu. \tag{29}$$

Hence, one only has to plot the l.h.s of (29) for different combinations of L and L'. These curves will intersect precisely at T_c and $-\beta/\nu$.

This has been done on Figure 6. To compensate for the short simulation times for each point calculated, 1000 points have been produced for each curve between $T = 2.2$ and 2.3. This allows to apply a curve smoothing by averaging over the 25 nearest neighbors of each point. Since these points have almost the same temperature this operation is *similar* to having a simulation time 25 times longer for each point while introducing a negligeable error compared to the deviation. We then get a value of $T_c \approx 2.26 \pm 0.02$ and a critical exponent $\beta \approx -0.11 \pm 0.03$ which are impressive results for the technique used.

[1] L.Onsager, Phys. Rev. 65, 177 (1944).
[3] B. Diu, C. Guthmann, D. Lederer, B. Roulet, Physique Statistique, Hermann diteurs des sciences et des arts, Paris (1989).
[3] N.V. Prokof'ev, Classical Monte Carlo and Metropolis Algorithm, Class notes, Massachusetts, (2004).
[4] Physics Letters A, Volume 238, Issue 4-5, p. 253-257.
[5] O. Edholm, M. Wallin, Monte Carlo Notes, Class notes for Computational Physics 5A1354, Royal Institute of Technology in Stockholm.

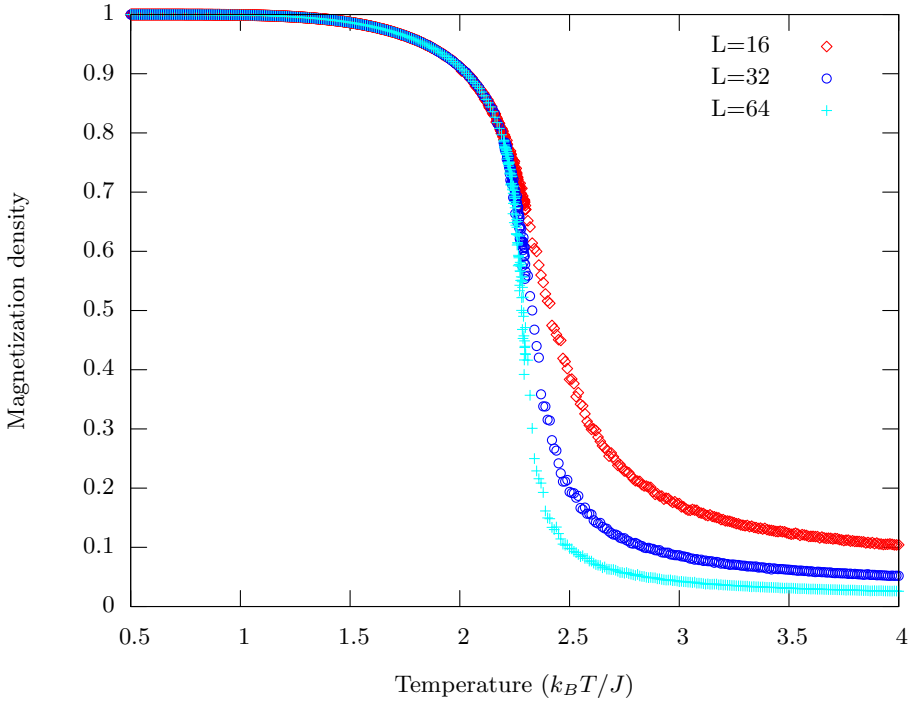

FIG. 2: Magnetization density for $L = 16, 32$ and 64 calculated with 2^{20} sweeps

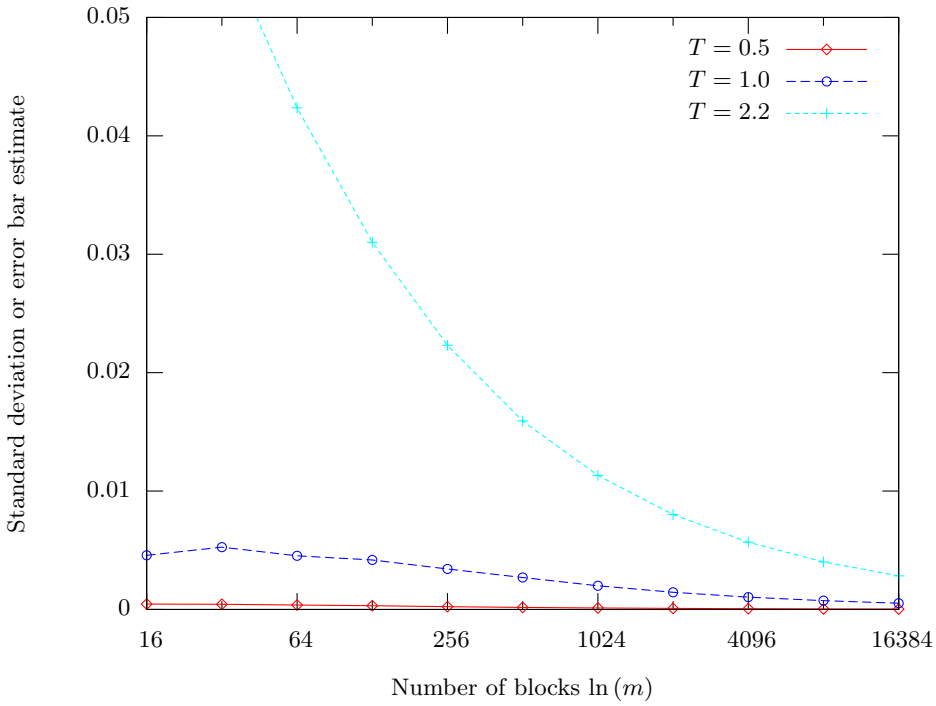

FIG. 3: Convergence of the error bars for $T = 0.5$, 1 and 2.2 and $L = 16$

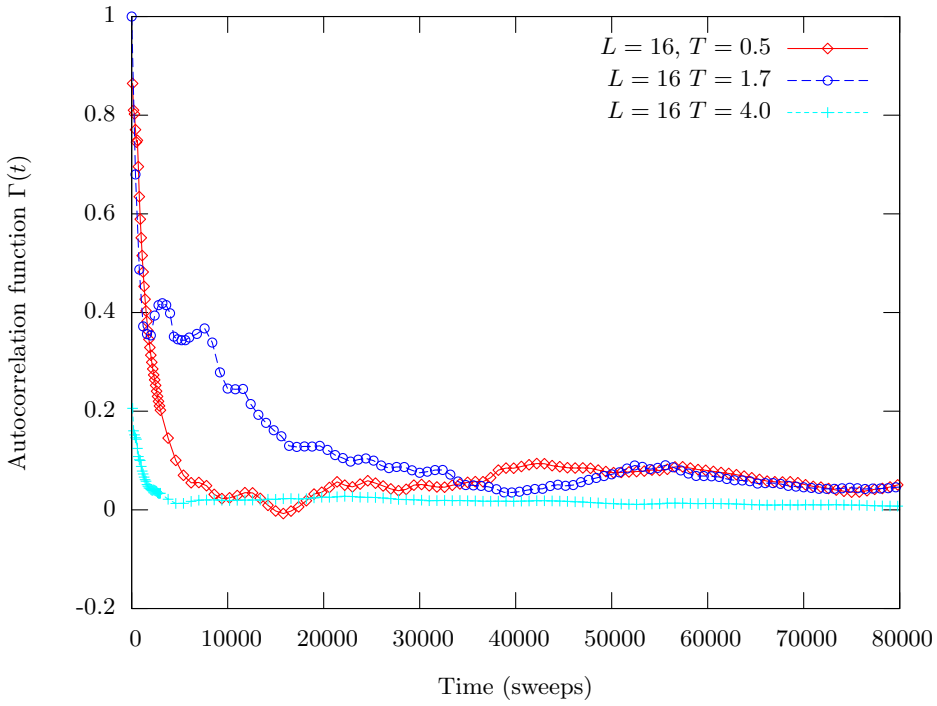

FIG. 4: Autocorrelation function for $T = 0.5$, 1.7 and 4.0 for $L = 16$

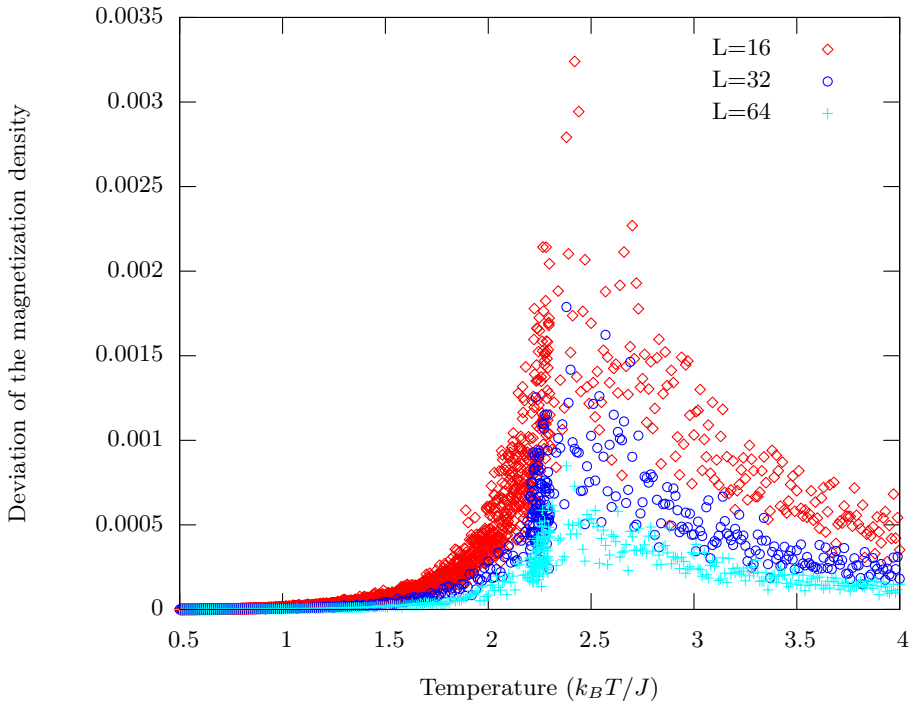

FIG. 5: Magnetization density deviation for $L = 16$, 32 and 64 calculated with 16 blocks of 2^{12} sweeps

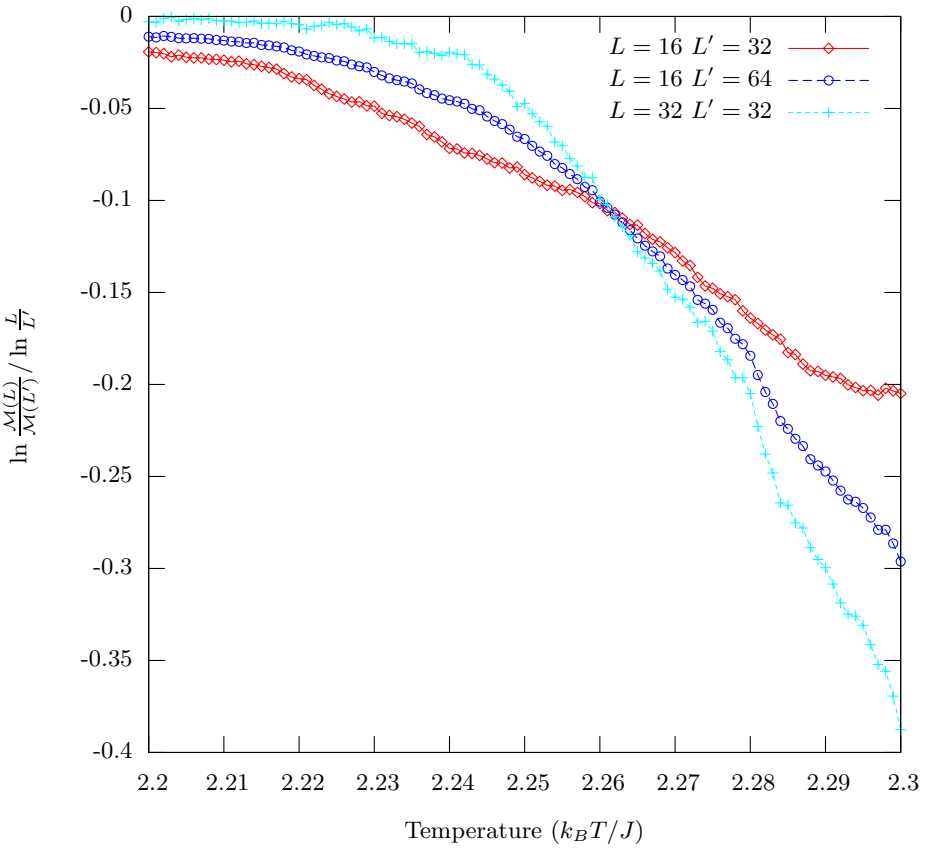

FIG. 6: Determination of T_c and the critical exponent β