# 2-$d$ Ising Model

Victor Muñoz-Vitelly

The 2-dimensional Ising Model is one of the simplest model exhibiting a phase transition, which is of second order. The model consists of classical spins $\sigma$, which can take the values $\pm 1$. The Hamiltonian is given by

$$H = -J \sum_{\langle \vec{i}, \vec{j} \rangle} \sigma_{\vec{i}} \sigma_{\vec{j}} = -\frac{J}{2} \sum_{i_1} \sum_{i_2} \sigma_{i_1,i_2} (\sigma_{i_1+1,i_2} + \sigma_{i_1-1,i_2} + \sigma_{i_1,i_2+1} + \sigma_{i_1,i_2-1}).,\qquad (1)$$

where $\langle \vec{i}, \vec{j} \rangle$ denotes the nearest neighbors, and $\sigma_{\vec{i}}$ is the spin at position $\vec{i}$, and we have taken the external magnetic field to zero $h = 0$. Below a critical temperature $T_c$, the discrete symmetry $\mathbb{Z}_2$ is spontaneously broken, and the system is in a ferromagnetic phase (for $J > 0$). For simplicity, we can work in units where the Boltzmann constant is $k_B = 1$ and we take $J = 1$.

An alternate expression to the previous, defined on a finite lattice with periodic boundary conditions, is

$$H = -\sum_{i_1=1}^{L} \sum_{i_2=1}^{L} \sigma_{i_1,i_2} (\sigma_{i_1+1,i_2} + \sigma_{i_1,i_2+1}),\qquad (2)$$

which halves the computing time compared to eq. (1). Or, alternatively, replacing $i + 1 \to i - 1$. The periodic boundary conditions are $i = L + 1 \to 1$, and $i = 0 \to L$.

To simulate the Ising model on a finite lattice, we start from an initial configuration. It can be a cold-start, where all the spins are aligned, or a hot-start, where the spins are randomly distributed, for example, in a $L \times L$ square lattice with $L = 8$, we can have the initial configurations

$$\begin{pmatrix} -1 & -1 & -1 & -1 & -1 & -1 & -1 & -1 \\ -1 & -1 & -1 & -1 & -1 & -1 & -1 & -1 \\ -1 & -1 & -1 & -1 & -1 & -1 & -1 & -1 \\ -1 & -1 & -1 & -1 & -1 & -1 & -1 & -1 \\ -1 & -1 & -1 & -1 & -1 & -1 & -1 & -1 \\ -1 & -1 & -1 & -1 & -1 & -1 & -1 & -1 \\ -1 & -1 & -1 & -1 & -1 & -1 & -1 & -1 \\ -1 & -1 & -1 & -1 & -1 & -1 & -1 & -1 \end{pmatrix}, \quad \begin{pmatrix} +1 & -1 & -1 & +1 & -1 & +1 & -1 & +1 \\ -1 & +1 & +1 & -1 & -1 & -1 & +1 & -1 \\ +1 & -1 & -1 & +1 & +1 & +1 & +1 & -1 \\ +1 & -1 & +1 & -1 & +1 & -1 & -1 & +1 \\ -1 & -1 & +1 & -1 & +1 & -1 & +1 & +1 \\ -1 & +1 & -1 & +1 & -1 & +1 & -1 & -1 \\ +1 & -1 & -1 & +1 & +1 & -1 & -1 & +1 \\ -1 & +1 & +1 & +1 & -1 & +1 & +1 & -1 \end{pmatrix},$$

for cold-start and hot-start, respectively. From eqs. (1) or (2) we can see that the energy of the cold-start is the lowest energy since all the spins are aligned, while the energy of the hot-start is higher than it.

## Contents

## 1 Metropolis Algorithm

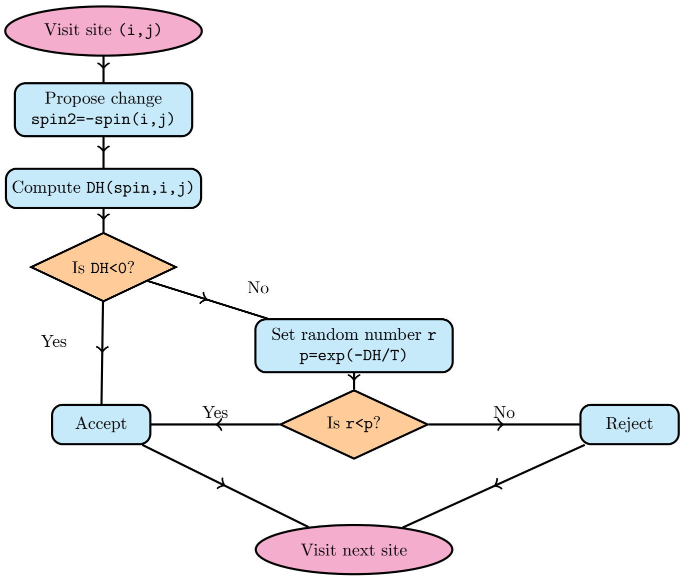
Then, we apply the Metropolis algorithm. We visit each site $\vec{i} = (i_1, i_2)$ and flip the sign of the spin if it decreases the energy, that is, if $\Delta H < 0$ then we take $\sigma'_{\vec{i}} = -\sigma_{\vec{i}}$. But if the change does not decrease the energy, $\Delta H > 0$, then we flip the sign with probability $p = e^{-\Delta H/T}$. A sweep is defined when we have visited all the spins on the lattice. The energy change for flipping the sign of a spin on the site $\vec{i}$, is

$$\Delta H = 2\sigma_{i_1,i_2}(\sigma_{i_1+1,i_2} + \sigma_{i_1-1,i_2} + \sigma_{i_1,i_2+1} + \sigma_{i_1,i_2-1}), \tag{3}$$

where $\sigma_{i_1,i_2}$ is the spin before flipping its sign.

The next flowchart shows the architecture of the algorithm, where we visit all spins in lexicographic order, from 1 to $L$. The periodic boundary conditions are implemented inside the function `DH(spin,i,j)`, which takes the form of eq. (3) and considers the neighboring spins of the site $(i, j)$. To implement the boundary conditions, we define a function of the indexes `iv(i)`, that takes the form

$$\texttt{iv(i)} = \begin{cases} \texttt{i} \ \text{ if } \texttt{i} \in [1, L] \\ 1 \ \text{ if } \texttt{i} = L + 1 \\ \text{L} \ \text{ if } \texttt{i} = 0 \end{cases} . \tag{4}$$



The first step is to visualize the thermalization, as shown in Fig. 1, for different temperatures. Thermalization can be studied, for example, with a plot of the energy density v.s. sweeps. In order to take measurements, we want to avoid the first sweeps that lead to thermalization, that is, when the system reaches equilibrium. Measurements should also be separated by a specific number of sweeps since they can be autocorrelated. We can also see in Fig. 1 that for $T = 1$, the system has a preference for the spins to align, so the energy fluctuates rarely. When the temperature is increased, this behavior is lost and the energy of the system varies around a determined value.

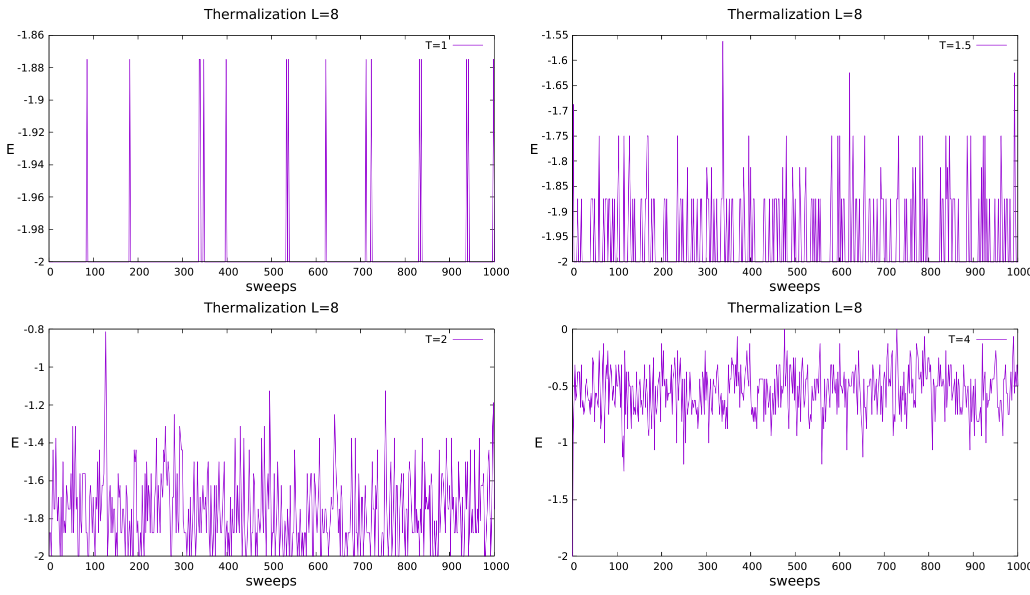


Figure 1: Thermalization for different temperatures on a square lattice of size $L = 8$.

To determine the value of an observable, we make a large number of measurements $N_m$. Several sweeps, $N_{\text{sweeps}}$, must be taken to separate these measurements, so that the statistic turns out to be independent. Otherwise, it might be the case that the measurements are autocorrelated and do not provide a reliable result. The autocorrelation time $\tau_{\text{exp}}$ is a good indicator, so we can take $N_{\text{sweeps}} \geq 2\tau_{\text{exp}}$. To determine this time, we use the connected autocorrelation function $C(t)$, for example, for the hamiltonian $H$,

$$C_H(t) = \langle H_{t_0} H_{t_0+t} \rangle - \langle H \rangle^2, \tag{5}$$

which has an exponential behavior of the form

$$C_H(t) = ce^{-t/\tau_{\text{exp}}}. \tag{6}$$

We measure this function on the lattice and determine the autocorrelation time by means of a fit. This procedure is shown in Fig. 2 for different lattice sizes and distinct values of $T$, around the expected critical value $T_c$. We can see the autocorrelation time $\tau_{\text{exp}}$ increases with the lattice size. After thermalization, we take measurements of the action every one sweep and measure eq. (5), taking $t_0$ as each sweep in the history.

In order to fit the exponential function to the data, we select a region $t \in [t_i, t_f]$ for each curve. In a logarithmic scale, the region of interest is where the curve looks asymptotically like a straight line. Before $t_i$, the autocorrelation might look curved or like a straight line with a different slope. After $t_f$ there might be fluctuations, so for each curve we have to determine $t_i$ and $t_f$, taking into account the reduced $\chi^2$/dof as an indicator. We can see in

Fig. 2 that $t_i \gtrapprox 3$ but a good $t_f$ depends on the temperature. For example, for $T = 2.8$ and $T = 3.0$, after $t = 12$ the points are statistically compatible with zero, as the uncertainties indicate. As a matter of fact, there are missing points since they take negative values, which are not defined in a logarithmic scale.

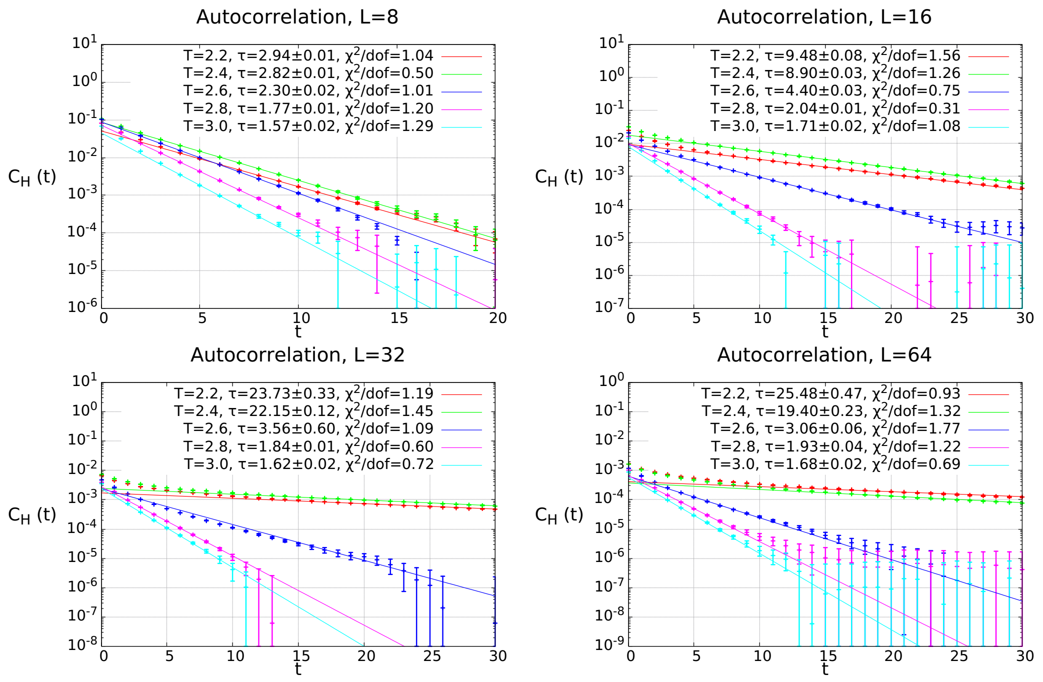


Figure 2: Autocorrelation function fits for $\tau_{\text{exp}}$ in logarithmic scale, for different temperatures at different lattice sizes (Metropolis algorithm).

In this report, we set the thermalization as 1,000 sweeps and take $N = 10,000$ measurements every 10 sweeps. A number of measurements $N$ between 1,000 and 10,000 is enough, but with this large value, the errors are small for all the lattice sizes explored. For the special case of a lattice size of $L = 64$, we take measurements every 100 sweeps, since the errors become larger, particularly for the susceptibility. According to the autocorrelation results, we should had taken measurements every 10 sweeps for $L = 8$, but every 20 sweeps for $L = 16$, 50 sweeps for $L = 32$. But the autocorrelation was computed after the subsequent results.

The first quantity that can be measured is the energy density $E$, shown in Fig. 3 (left) and given by the expression

$$E = \frac{1}{L^2}\langle H \rangle = \frac{1}{L^2}\frac{1}{N}\sum_{k=1}^{N} H_k, \tag{7}$$

where again, $N$ is the number of measurements and the index $k$ labels the lattice configuration every 10 sweeps (100 for $L = 64$) as mentioned above. We divide by $L^2$ so that we can compare between different lattice sizes. Another quantity we can measure is the magnetization, shown in Fig. 3 (right), and given by the expression

$$M = \frac{1}{L^2}\left\langle \left| \sum_{i_1=1}^{L}\sum_{i_2=1}^{L} \sigma_{i_1,i_2} \right| \right\rangle = \frac{1}{L^2}\frac{1}{N}\sum_{k=1}^{N}\left| \sum_{i_1=1}^{L}\sum_{i_2=1}^{L} (\sigma_{i_1,i_2})_k \right|. \tag{8}$$

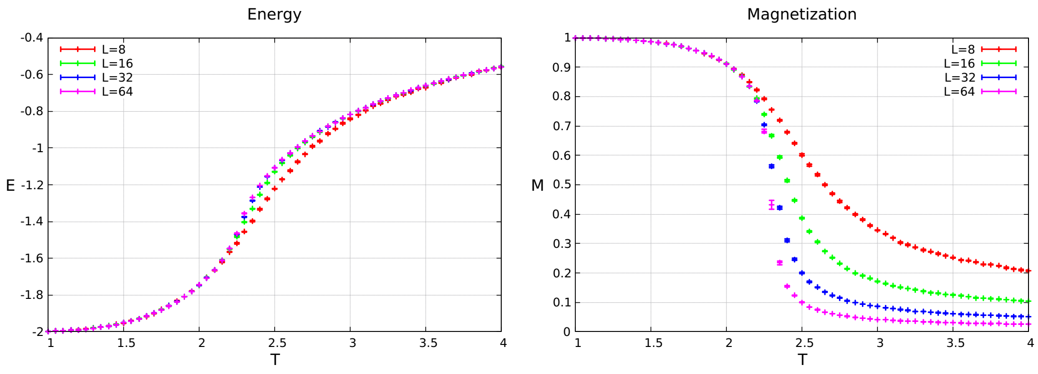


Figure 3: Energy (left) and magnetization (right) dependence on the temperature for different lattice sizes, 8, 16, 32 and 64.

Each point in these figures is calculated at a fixed temperature and fixed lattice size. We vary the temperature from $T = 1$ to $T = 4$ for aesthetic reasons, but the relevant information from the model might be around the critical temperature. From the last two equations, we can see that these points are averages,

$$\langle x \rangle = \frac{1}{N}\sum_{k=1}^{N} x_k \tag{9}$$

with $N = 10,000$ samples. A simple way to define their associated error is using the standard error, given by

$$\delta x = \sqrt{\frac{\text{Var}}{N}} = \sqrt{\frac{1}{N(N-1)}\sum_{k=1}^{N}(x_k - \langle x \rangle)^2}, \tag{10}$$

where Var is the variance. The standard error for these two quantities is easy to implement. There is an alternative known as the Jackknife error, which is more convenient for other quantities such as susceptibility, specific heat, and the correlation function. Because of the structure of these quantities, using the propagation of the standard error is inaccurate and leads to errors larger than what they truly are. The Jackknife error consists in dividing the samples in $M$ bins; for example, the $N = 10,000$ samples can be divided in $M = 10$ bins, giving 10 sets of samples with 1,000 measurements each. Then, the error is computed with

$$\delta x_{\text{Jackk}} = \sqrt{\frac{M-1}{M}\sum_{m=1}^{M}(\langle x \rangle_m - \langle x \rangle)^2}, \tag{11}$$

where $\langle x \rangle$ is the average of $x$, and $\langle x \rangle_m$ is the average of $x$ of all measurements that exclude the $m$-th set of samples.

## 1.1 Susceptibility

A third quantity we can measure is the magnetic susceptibility, given by

$$\chi_M = \frac{1}{L^2 T} \left( \langle M^2 \rangle - \langle M \rangle^2 \right), \tag{12}$$

and shown in Fig. 4 in logarithmic scale, for different lattice sizes and with jackknife errors. Notice that from eq. (12) there is a unique value for $\chi_M$ and the statistic is done on $M^2$ and $M$. This is why we use jackknife and generate a set of different $\chi_M$'s that allow us to perform the error statistics.

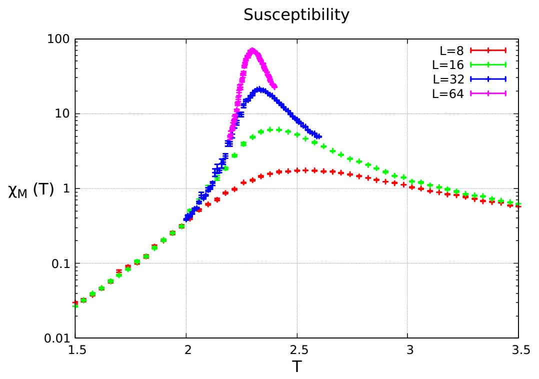


Figure 4: Susceptibility in logarithmic scale for different lattice sizes. The temperature range varies depending on the lattice size.

As mentioned above, the 2-dimensional Ising model has an analytic solution. In the thermodynamic limit, the susceptibility is a divergent quantity at the critical temperature. But on a finite lattice, the susceptibility is a finite function with a peak, that increases its size and reduces its width with the lattice size. There is no formal phase transition in a finite volume, the maximum susceptibility depends on the lattice size, and we expect that taking the continuum limit $L \to \infty$ leads to the analytical result. That is why we fit the susceptibility peak using a Gaussian distribution

$$f(T) = A e^{-b(T - T_{max})^2} \tag{13}$$

to find the values of $T_{\max}(L)$. These fits are shown in Fig. 5, which consider a small set of points in a region around the maxima since the full susceptibility is not a Gaussian function, but a neighborhood around the maximum can be approximated by one. These fits lead to the results presented in Table 1, along with the $\chi^2$/dof of the fits, which we verified to be around one.

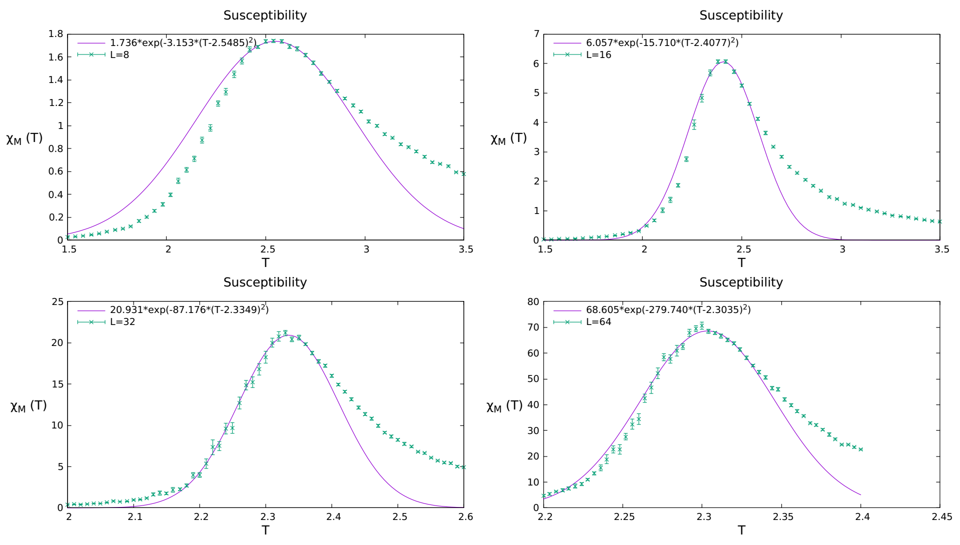


Figure 5: Gaussian fits for the susceptibility at different lattice sizes.

| | $L = 8$ | $L = 16$ | $L = 32$ | $L = 64$ |
|---|---|---|---|---|
| $A$ | 1.736(4) | 6.06(6) | 20.9(2) | 68.6(4) |
| $b$ | 3.2(1) | 16(1) | 87(15) | 280(24) |
| $T_{\max}$ | 2.548(4) | 2.408(4) | 2.335(2) | 2.3035(7) |
| $\chi^2$/d.o.f. | 0.91 | 2.27 | 0.96 | 1.17 |

Table 1: Gaussian fit parameters for susceptibility on different lattice sizes.

With these values of $T_{\max}(L)$, we want to interpolate the final result at $L \to \infty$. To do so, we take another fit assuming $|T_{\max}(L) - T_c|^{-\nu} \sim L$. So the function to fit becomes

$$T_{\max}(L) = T_c + \frac{C}{L^{1/\nu}}, \tag{14}$$

where $T_c$ is the critical temperature, $\nu$ is a critical exponent and $C$ is a proportionality constant. The results are shown in Fig. 6. We get a critical temperature of $T_c = 2.274(4)$, while its theoretical value is $T_c \approx 2.269$. We also get a critical exponent $\nu = 0.93(4)$, whose theoretical value is $\nu = 1$.

| | |
|---|---|
| $T_c$ | 2.274(4) |
| $\nu$ | 0.93(4) |
| $C$ | 2.6(3) |
| $\chi^2$/d.o.f. | 0.88 |

Table 2: Fit values for critical temperature $T_c$, critical exponent $\nu$ and proportionality constant $C$.

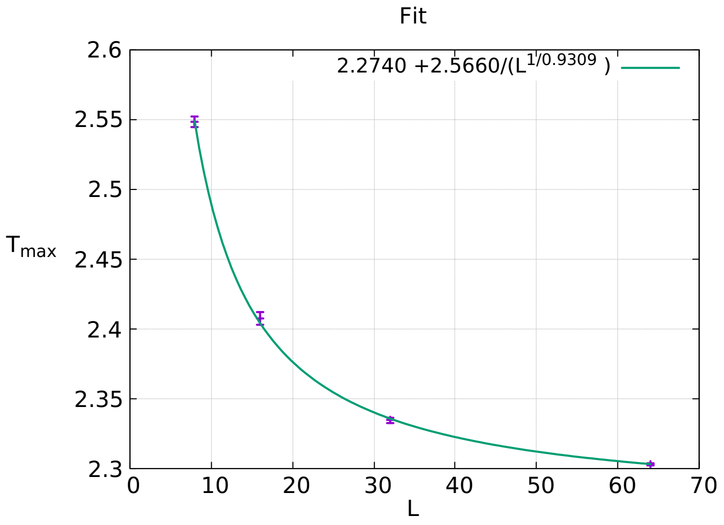


Figure 6: Fit for $T_{\max}(L)$ for different lattice sizes, 8, 16, 32, and 64.

## 1.2 Correlation Length

The correlation function is an exponentially decaying function with the distance, but on a finite lattice, because of the boundary conditions, we can fit it to a hyperbolic cosine of the form

$$C(x-y) = A\cosh\left(\frac{|x-y|-L/2}{\xi}\right), \tag{15}$$

where $A$ is an amplitude and $\xi$ is the correlation length. To measure this function in practice, we can define a spin vector $S_{i_1}$, as

$$S_{i_1} = \sum_{i_2=1}^{L} \sigma_{i_1,i_2}, \tag{16}$$

and then, the correlation function will be

$$C(i-j) = \langle S_i S_j\rangle - \langle S_i\rangle\langle S_j\rangle. \tag{17}$$

Figure 7 shows the correlation function's behavior for different temperatures, on a lattice with size $L = 64$. To obtain their respective correlation lengths, we fit the hyperbolic cosine from eq. (15). Applying this procedure to different temperatures gives a temperature-dependent correlation length $\xi(T)$, shown in Fig. 8 for lattice sizes $L = 8$ and $L = 64$.

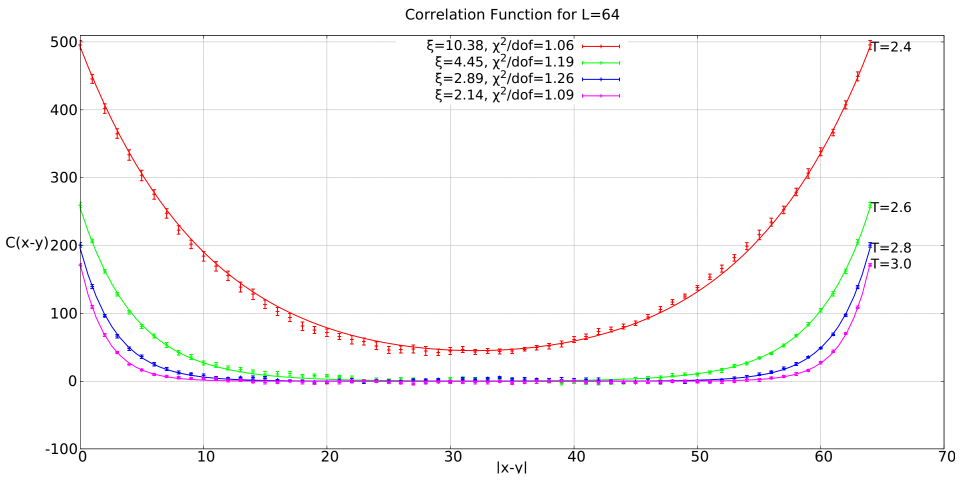


Figure 7: Correlation function fit on a lattice size $L = 64$ for different temperatures.

Again, in the thermodynamic limit, the correlation length is a divergent quantity at the critical temperature. In Fig. 8 we can see that the correlation length also depends on the lattice size. We identify that the correlation length increases towards the critical temperature, but the increment is more notorious as the lattice size is bigger. We plot for two different lattice sizes to determine the region without volume dependence. Around $T = 3$, the results seem to be independent of the lattice size, but after this temperature, the correlation length decreases slower than expected theoretically. So we select the region $T \in [2.50, 2.82]$.

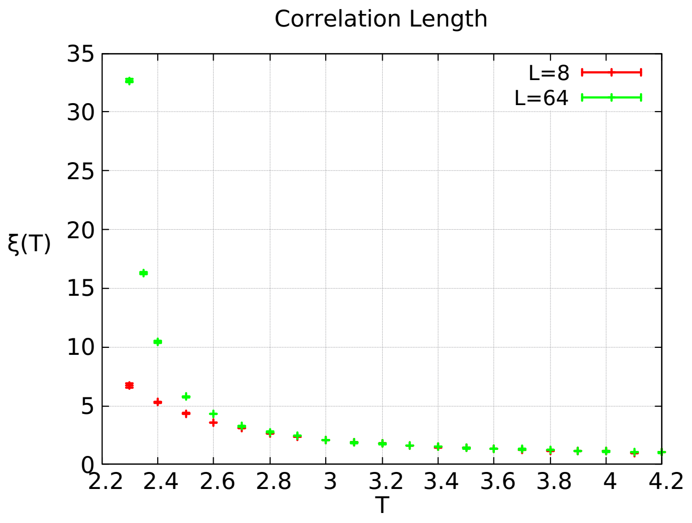


Figure 8: Temperature dependence of the correlation length for two different lattice sizes.

Figure 9 shows the temperature region chosen for the correlation length in a lattice size of $L = 64$. We fit the function

$$\xi(T) = \frac{D}{(T - T_c)^{\nu}}, \tag{18}$$

where $D$ is a constant, $T_c$ is the critical temperature and $\nu$ is the critical exponent. We also fit the same function, but introducing the theoretical value of $T_c \approx 2.269$, so that the precision can be increased only by having two fitting parameters $D$ and $\nu$. For the first fit, we get $T_c = 2.25(8)$ and $\nu = 1.0(2)$, while for the second we get $\nu = 0.96(2)$.

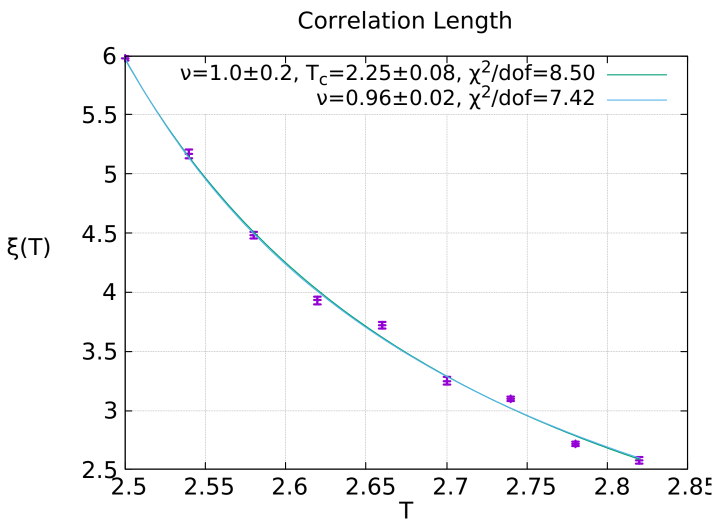


Figure 9: Correlation length fit.

Figure 9 also shows that $\chi^2$/dof is of higher order than one. This might be because the error for the correlation length is underestimated by the fitting method. To avoid this issue, we repeat the procedure by generating a set of 5 to 10 correlation lengths for each temperature. We run our program 5 to 10 times, to generate sets of correlation functions to be fitted, one set is shown in Fig. 10 as an example. Then we can obtain a better mean correlation length with its associated standard error. As a first attempt, we can simply fit the correlation lengths of each set of correlation functions. Figure 11 shows this procedure, which provides us with a set of critical exponents and temperatures, which can be averaged with the results shown in Table 3.

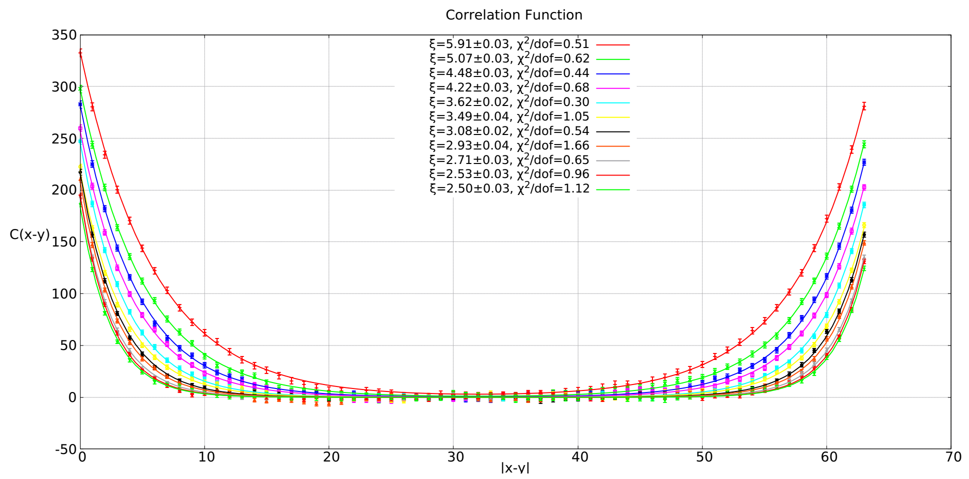


Figure 10: One set of fits for the correlation functions displaying their correlation lengths, on a lattice size of $L = 64$. Each correlation function corresponds to a temperature $T \in [2.50, 2.82]$, in steps of $\Delta T = 0.04$

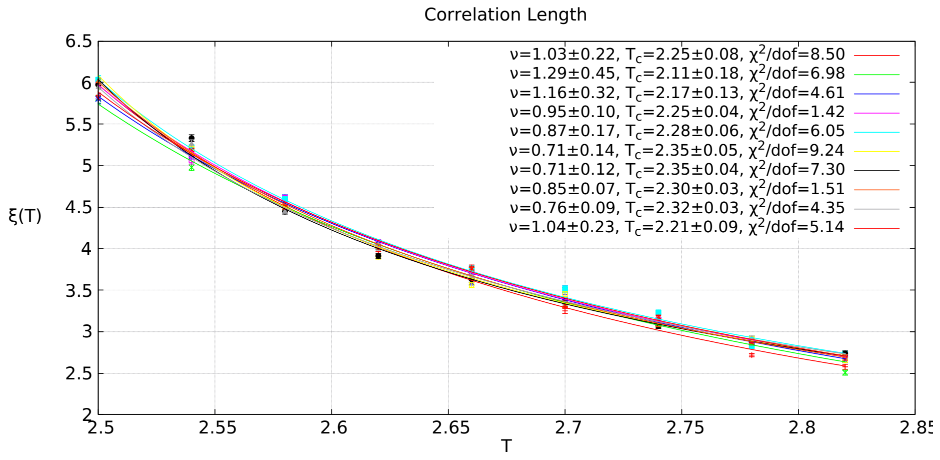


Figure 11: Correlation length fit for each one of the 10 sets of correlation functions.

| | |
|---|---|
| $T_c$ | 2.26(2) |
| $\nu$ | 0.94(6) |

Table 3: Average values for critical temperature $T_c$ and critical exponent $\nu$, using fits from Fig. 11.

As an alternative, we can take the sets of correlation functions and fit them to their respective correlation length. Then we obtain a set of correlation lengths and we average them, instead of simply fit them. The results of this procedure are shown in Figs. 12 and 13, where we can see that the averaged correlation length is better behaved and with plausible errors.

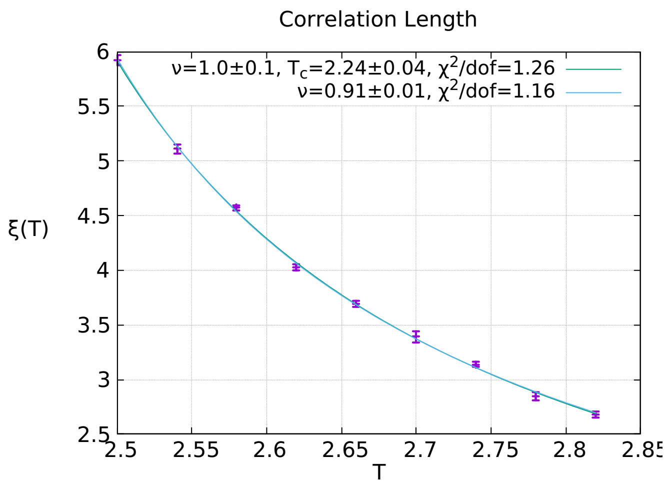


Figure 12: Correlation length fit using 5 sets of correlation functions.

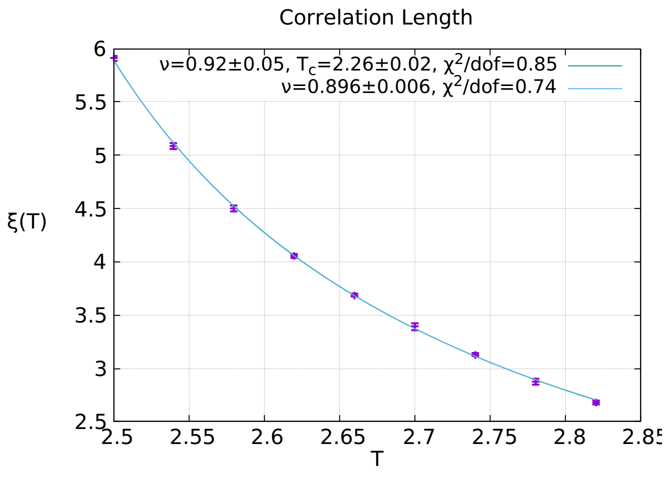


Figure 13: Correlation length fit using 10 sets of correlation functions.

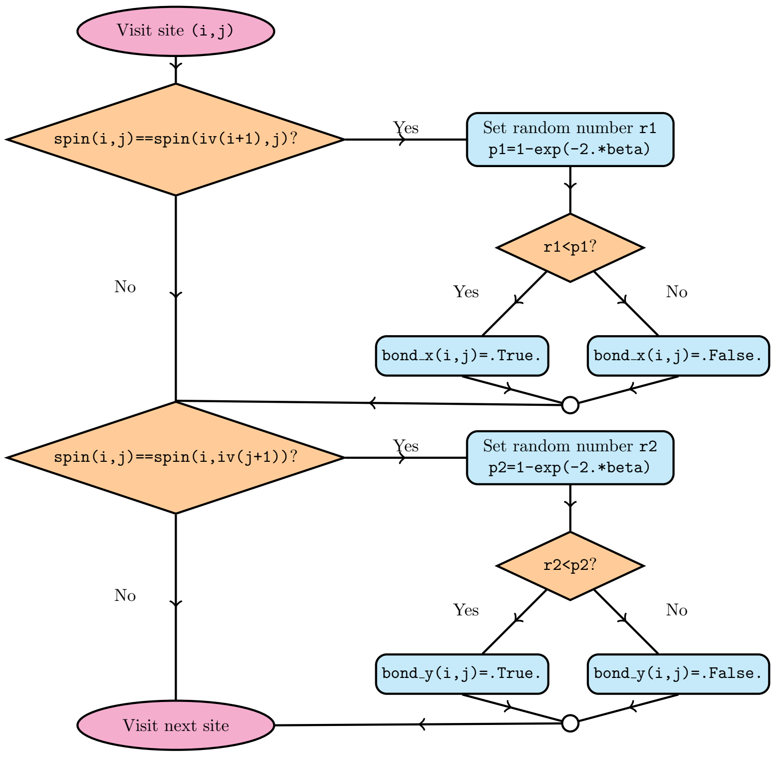
## 2 Swendsen-Wang Algorithm

Metropolis algorithm is no the only Monte Carlo algorithm we cant test. Another algorithm we can use is the Swendsen-Wang, which is a multi-cluster algorithm developed in Ref. [2]. It consists in creating bonds between parallel neighboring spins with probability

$$p = 1 - e^{-2\beta}. \tag{19}$$

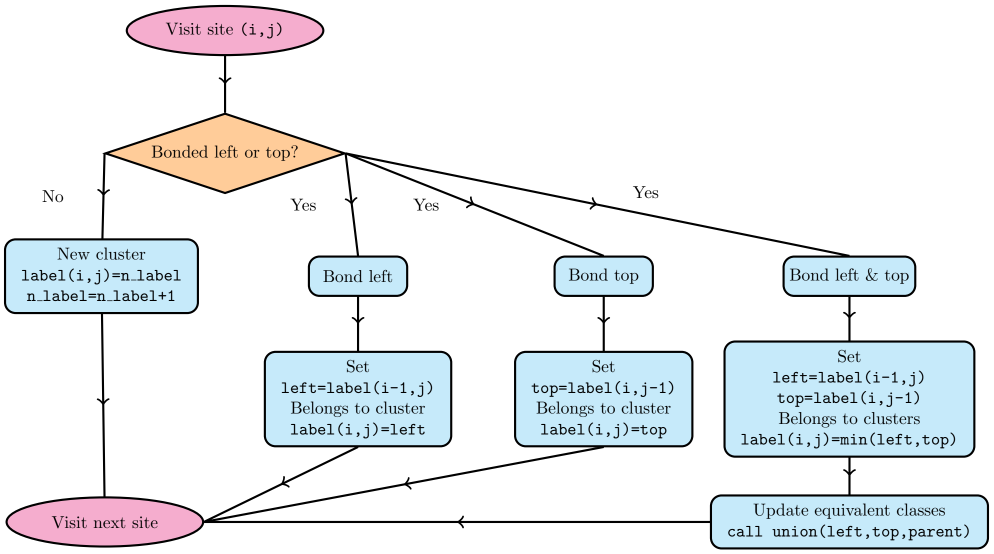
This procedure creates clusters of aligned spins. After identifying them, we flip the sign of all the spins in the same cluster with probability $p = 0.5$.

As we have seen, each spin has 4 nearest neighbors, but due to boundary conditions, we only need to create bonds for the right and bottom neighbors. Then, we visit each site in lexicographic order and store the bonds in two arrays `bond_x(i,j)` and `bond_y(i,j)`, for the right and bottom bonds, respectively. The flowchart of our code for the Swendsen-Wang algorithm is shown in the next diagram, where, again, boundary conditions are implemented through the function `iv(i)`.

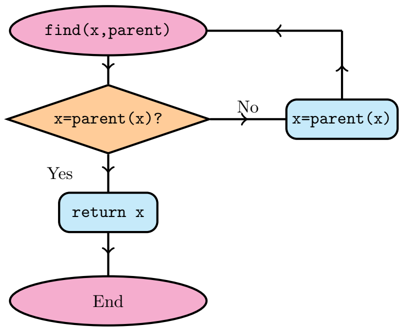


### 2.1 Hoshen-Kopelman Algorithm

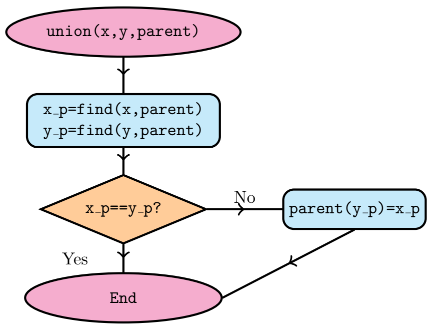
Once the bonds have been created, we need to identify the clusters, that is, the groups of spins that are connected by bonds. An efficient way to do this is with the Hoshen-Kopelman algorithm [3], which is shown in the next flowchart. We label the clusters with `label(i,j)`. This matrix tells us the number of the cluster to which `spin(i,j)` belongs. We initialize the algorithm with an integer `n_label=1`, which counts and labels when a new cluster is identified, and `parent(i)=i`, which is a vector of size $L^2$, defining the class of equivalence of the spins. With this initialization, `parent` tells us that every spin is in its own class of equivalence, and with the Union-Find algorithm, `parent` will be updated.



The Union-Find algorithm consists of a recursive function `find` that returns a representative member of the equivalence class and a subroutine `union`. Consider a set of elements $a_1, a_2, a_3, \ldots$. The statement `parent(`$a_2$`)=`$a_1$, means that $a_2$ and $a_1$ belong to the same equivalence class. And if `parent(`$a_1$`)=`$a_1$, then $a_1$ is the representative of the equivalence class. The other elements can be bonded by joining the equivalence class, for example, for $a_3$ we can set `parent(`$a_3$`)=`$a_1$ or `parent(`$a_3$`)=`$a_2$. The function `find(x, parent(x))` will recursively find the representative, as shown in the next flowchart.



The `union` subroutine is used to merge two different equivalent classes. When a spin has a bond with its left and top neighbors, it can be the case that the neighbors belong to different clusters, but the spin at `(i,j)` must merge them. To describe the subroutine, let us consider two sets of elements $a_1, a_2, \ldots,\ b_1, b_2, \ldots$ that belong to different equivalence classes, with representatives `parent(`$a_1$`)=`$a_1$, `parent(`$b_1$`)=`$b_1$. The new element will merge them by setting `parent(`$b_1$`)=`$a_1$. A flowchart is shown in the next diagram.



For simplicity, we haven't implemented the periodic boundary conditions yet, as can be noted in our description of the Hoshen-Kopelman algorithm. We can impose them now, simply by visiting the spins at the boundary, `spin(1,j)` and `spin(i,1)` and using `union` if they are bonded to their respective neighbor `spin(L,j)` and `spin(i,L)`. The last step is to merge the clusters. The structure of the equivalence classes is stored inside `parent`, then we just redefine `label(i,j)=find(label(i,j),parent)`, and all the clusters are now labeled with a unique identifier. We proceed with the last part of the Swendsen-Wang algorithm and flip them with a probability of $p = 0.5$.

The computing time of the algorithm is proportional to the volume $L^2$. We tested 100,000 iterations of the cluster algorithm at different lattice sizes and measured the computing time, as shown in Fig. 14. To define errors, we ran the 100,000 iterations, 10 times for each point. Analogously, in the same figure we can find the computing time for the Metropolis algorithm.

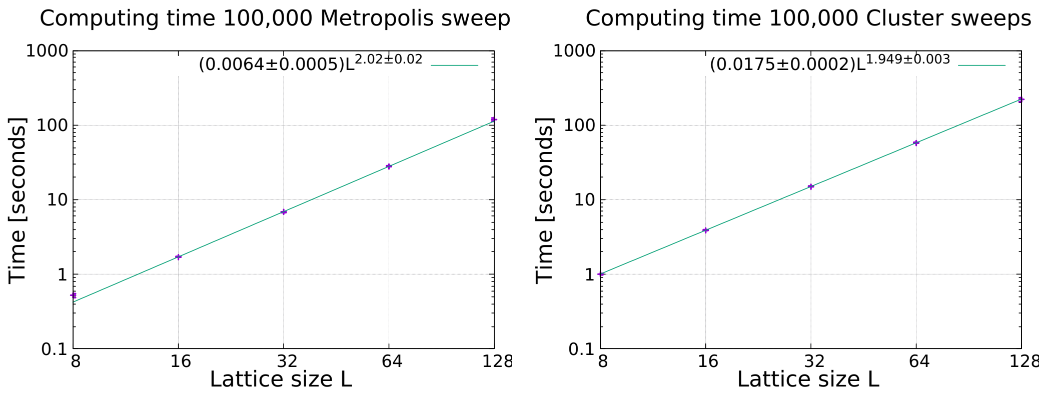


Figure 14: Computing time of 100,000 sweeps of only Metropolis algorithm and 100,000 sweeps of only cluster algorithm at different lattice sizes, fits with $\chi^2/\text{dof} = 3.90$ and $\chi^2/\text{dof} = 0.17$, respectively.

The clusters created with this algorithm have a well-defined average size, as we can see in Fig. 15. Around the critical temperature and above, the size of the clusters appears to be constant as we increase the volume of the system. At lower temperatures than the critical temperature, the size of the cluster increases with volume. In this region, we expect the average size to be of the order of the volume $L^2$, since in this phase all the spins prefer to be aligned.

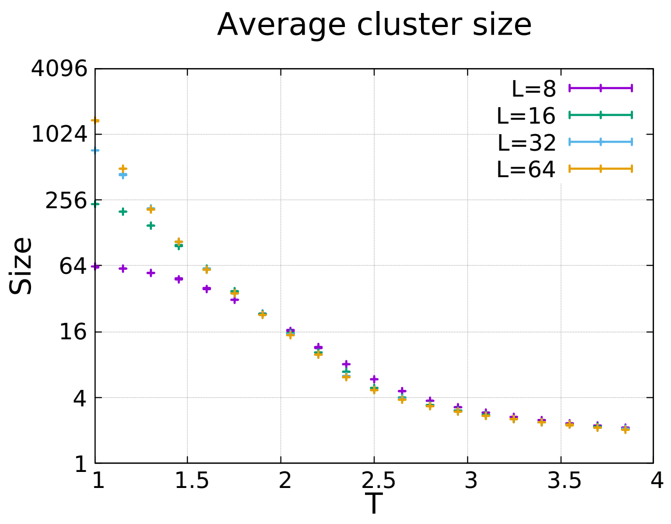


Figure 15: Cluster sizes dependence on temperature for different lattice sizes.

We measure the autocorrelation function following the procedure described above, and we obtain the data from Fig. 16. In this figure, we also find the fit values for the exponential function from eq. (6), and obtain the exponential autocorrelation time $\tau_{\text{exp}}$. If we compare the autocorrelation times from Fig. 2 for the Metropolis algorithm and Fig. 16 for the Cluster algorithm, we see an improvement. Typical values for $\tau_{\text{exp}}$ for the Metropolis algorithm are around $\tau_{\text{exp}} \sim 2$ for $L = 8$ and increase to order $\tau \sim 20$ for $L = 64$ near the critical temperature. However, for the Cluster algorithm, the typical values of $\tau_{\text{exp}}$ are in the region $\tau_{\text{exp}} \sim (1, 3)$ for all volumes explored at the indicated temperatures.

We conclude that the Swendsen-Wang algorithm is an improvement for the 2-dimensional Ising model studied here, since a smaller autocorrelation time will reduce the computing time of our code. This result is expected, since originally in Ref. [2], Swendsen and Wang discuss the critical slowing down and the efficiency of their algorithm near criticality for a variety of spin systems, including the 2-dimensional Ising model.

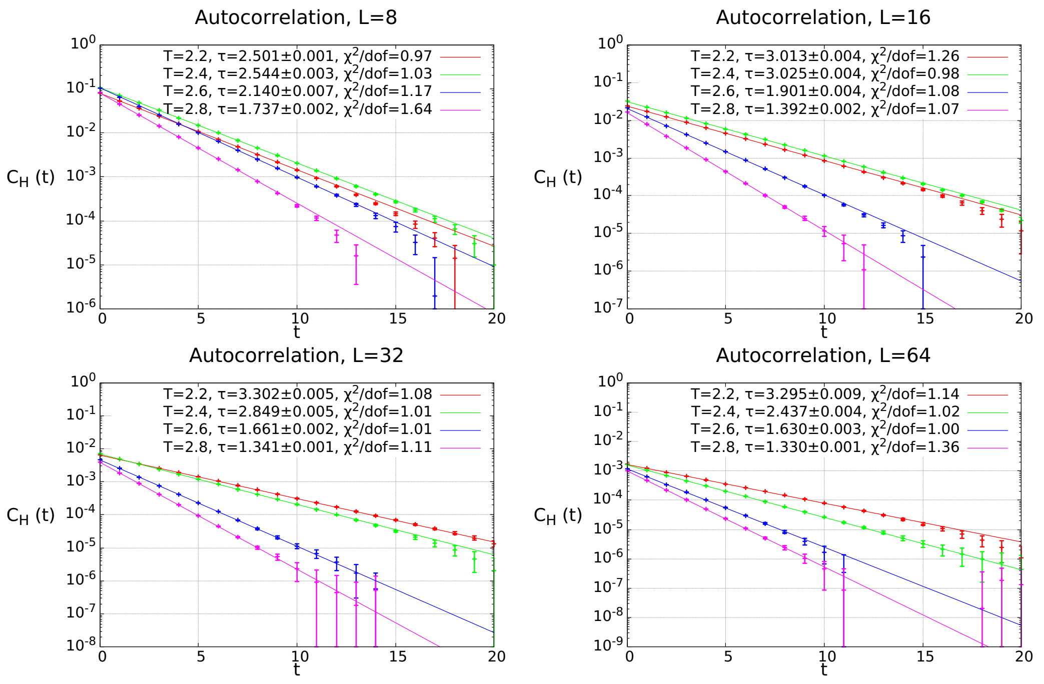


Figure 16: Autocorrelation function fits for $\tau_{\text{exp}}$ in logarithmic scale, for different temperatures at different lattice sizes (Cluster algorithm).

The Wolff algorithm, developed in Ref. [4], is a single-cluster algorithm. The difference with Swendsen-Wang is that there is only one cluster in each iteration. In this reference, Wolff relates the size of his cluster $|c|$ with the size of Swendsen-Wang clusters $|c_{\text{SW}}|$ as

$$\langle |c| \rangle = \frac{1}{L^2} \Big\langle \sum_{c_{\text{SW}}} |c_{\text{SW}}|^2 \Big\rangle, \tag{20}$$

and reports values of $\langle |c| \rangle / L^2$ at the critical temperature. He also relates

$$\langle |c| \rangle = \chi_M = \frac{1}{L^2} \langle M^2 \rangle, \tag{21}$$

as an improved estimator for the magnetic susceptibility. We measure these quantities in the lattice and compare them with the results of Ref. [4]. This is shown in Table 4, which we take as a consistency check of our implementation of the algorithm.

| $L$ | Wolff $\langle \lvert c \rvert \rangle / L^2$ | Our $\langle \lvert c \rvert \rangle / L^2$ | Wolff $\chi_M / L^2$ | Our $\chi_M / L^2$ |
|---|---|---|---|---|
| 32 | 0.4602(7) | 0.458(1) | 0.4598(7) | 0.457(2) |
| 64 | 0.3858(11) | 0.3866(14) | 0.3852(10) | 0.3859(16) |
| 128 | 0.3225(11) | 0.3216(8) | 0.3229(10) | 0.3226(15) |

Table 4: Average values of cluster sizes $\langle |c| \rangle / L^2$ and susceptibility $\chi_M$, from Ref. [4] and our results with 10,000 measurements using eq. (20), for the sizes of the Swendsen-Wang clusters, at temperature $T \approx 1/0.4406$.

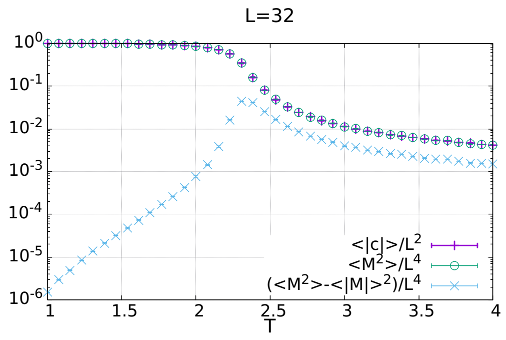


Figure 17: Average cluster size as an improved estimator of squared magnetization and susceptibility.

Another consistency check is that the results must be independent of the algorithm, as we can see from Figs. 18 and 19. These figures compare the energy, magnetization, susceptibility, and specific heat using the two different algorithms.

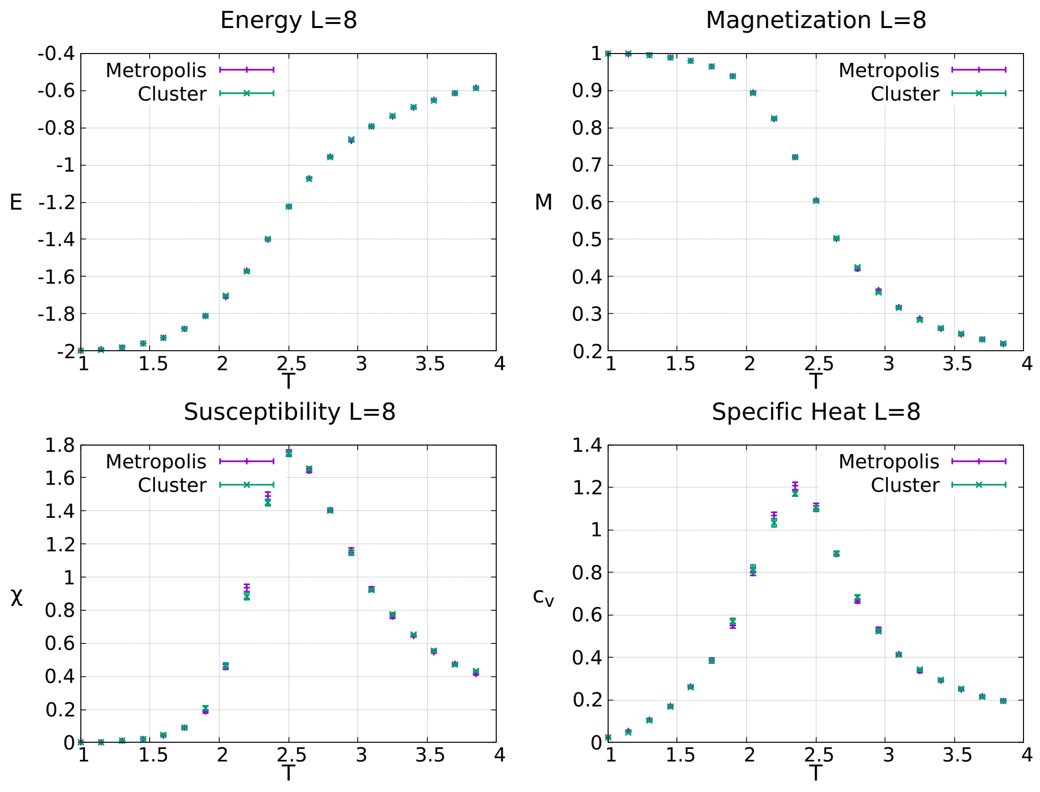


Figure 18: Energy, magnetization, susceptibility and specific heat for $L = 8$, obtained from Metropolis and Cluster algorithms.

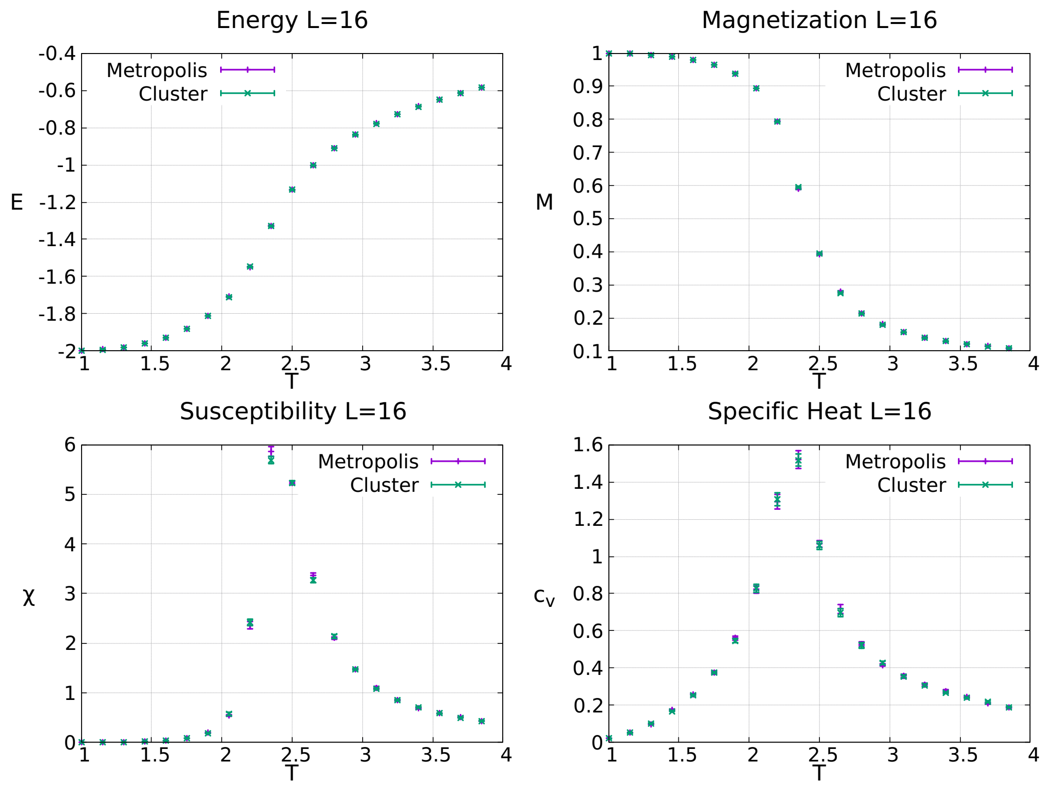


Figure 19: Energy, magnetization, susceptibility and specific heat for $L = 16$, obtained from Metropolis and Cluster algorithms.

## References

[1] N. Metropolis, A. W. Rosenbluth, M. N. Rosenbluth, A. H. Teller & E. Teller, *Equation of State Calculations by Fast Computing Machines Available*. J. Chem. Phys. 21, 1087–1092 (1953). DOI: 10.1063/1.1699114.

[2] R. H. Swendsen & J. S. Wang, *Nonuniversal critical dynamics in Monte Carlo simulations*. Phys. Rev. Lett. 58, 86 (1987). DOI: 10.1103/PhysRevLett.58.86.

[3] J. Hoshen & R. Kopelman, *Percolation and cluster distribution. I. Cluster multiple labeling technique and critical concentration algorithm*. Phys. Rev. B 14, 3438 (1976). DOI: 10.1103/PhysRevB.14.3438.

[4] U. Wolff, *Collective Monte Carlo Updating for Spin Systems*. Phys. Rev. Lett. 62, 361 (1989). DOI: 10.1103/PhysRevLett.62.361.