Metropolis and Swendsen-Wang Simulation of the 2D Ising Lattice

Edward Jansen, Bas de Jong
Faculty of Applied Sciences, Delft University of Technology, Delft, the Netherlands
(Dated: April 9, 2016)

Monte Carlo (MC) methods allow us to simulate the behavior of the Ising model. Here, we investigated the behavior of the 2D Ising square lattice around the critical temperature with the Metropolis and Swendsen-Wang (SW) algorithm in Python. The critical temperature was determined with the fourth order Binder cumulant, while finite size scaling was used to determine β and γ, the critical exponents associated with the magnetization m and the magnetic susceptibility χ_M. The found values ($\beta = 0.122,\ \gamma = 1.617$) correspond reasonably well with the exact values ($\beta = 0.125,\ \gamma = 1.75$). Due to the limitations of the Python programming language, we implemented computationally expensive parts of the SW algorithm in Cython.

I. INTRODUCTION

The Ising model is a model that was originally formulated to describe the phenomenon of ferromagnetism. It consists of discrete variables that can take the values ± 1 and that represent atomic magnetic moments or spins. Here, we are concerned with the 2D Ising model, which was solved exactly by Onsager in 1944.

The model is widely used in physics since it describes the occurrence of second order phase transitions (except in 1D). A second order phase transition is a transition where the system goes from a state of order (the ordered/broken phase) to a state of chaos (the symmetric phase). Most second order phase transitions can be described by an order parameter, which is zero in the symmetric phase and non-zero in the ordered phase. Thus, it must be discontinuous at the transition point. In a ferromagnet, this corresponds to a transition from the non-magnetic phase (all spins oriented randomly) to the magnetic phase (all spins aligned). In nature, first and second order phase transitions exist, corresponding to discontinuous behavior of the first or second derivative of the free energy $F = -k_B T \ln Z$. On and around a second order phase transition, the system's properties can be described by so-called critical exponents. Apart from physics, the model is used in a variety of fields, such as cognitive neuroscience[1] and the social sciences[2].

Consider a 2D square lattice of spins that interact with each other and can assume values ± 1. The Hamiltonian that describes this Ising lattice is given by

$$\mathcal{H} = -J \sum_{\langle xy \rangle} s_x s_y + \sum_x h s_x, \tag{1}$$

where $s_x \in \{\pm 1\}$ denotes the spin value, $\langle xy \rangle$ represents all nearest neighbors, J is the coupling constant and h denotes a external magnetic field. Usually, h is set equal to zero. The corresponding probability distribution is

$$P(\mathbf{s}) = \exp\left(-\beta \mathcal{H}\right), \tag{2}$$

where $\beta = 1/k_B T$, with k_B being Boltzmann's constant and T the temperature, and $\mathbf{s} = \{s_1 s_2 \dots s_n\}$ denotes the state of the whole lattice[3]. To calculate the distribution $P(\mathbf{s})$, we should evaluate equation 2 for all possible $\mathbf{s}$. However, as the lattice size increases, we soon run into problems. A 'modest' lattice with dimensions 30×30 already requires $\sim 2^{1000}$ calculations. The relevant question is thus how to sample this distribution $P(\mathbf{s})$. The logical way to proceed is to consider Monte Carlo (MC) methods.

II. METHOD

In this section we first discuss critical exponents. Then, two Monte Carlo methods to simulate an Ising lattice and two methods to numerically approximate the critical exponents and temperature are discussed. The section is concluded with a consideration of the computational implementation of the simulation.

A. Critical Exponents

Second order phase transitions can be classified by critical exponents, which describe how various physical quantities behave on and around the critical point. Here, we consider the magnetization, magnetic susceptibility, heat capacity and correlation length:

$$M \propto (T_c - T)^{\beta}, \tag{3a}$$
$$\chi_M \propto |T - T_c|^{-\gamma}, \tag{3b}$$
$$C_V \propto |T - T_c|^{-\alpha}, \tag{3c}$$
$$\xi \propto |T - T_c|^{-\nu}, \tag{3d}$$

where T is the temperature and T_c is the temperature at the critical point. For the 2D Ising model, these exponents are known exactly: $\alpha = 0$, $\beta = 0.125$, $\gamma = 1.75$ and $\nu = 1$. Furthermore, the exact critical temperature is $T_c = 2J/(k_B \ln\left(1 + \sqrt{2}\right)) \simeq 2.269$ when we set $J = k_B = 1$.

To calculate these quantities, the system is simulated for a number of timesteps with either the Metropolis or the SW algorithm, before computing:

$$m = L^{-2} \sum_x s_x, \tag{4a}$$
$$\chi_M = L^2 T^{-1} \left(\langle m^2 \rangle - \langle |m| \rangle^2\right), \tag{4b}$$
$$C_V = (LT)^{-2} \left(\langle E^2 \rangle - \langle E \rangle^2\right), \tag{4c}$$

where E is given by the Hamiltonian $\mathcal{H}$ (where $h = 0$), L is the lattice size and T the temperature.

B. The Metropolis algorithm

A possible MC method is the Metropolis algorithm, which can be formalized as follows. Let $Q(s'; s(t))$ denote a proposal probability density. We then compute

$$p = \frac{P(s')}{P(s(t))} \frac{Q(s(t); s')}{Q(s', s(t))}, \tag{5}$$

where s' denotes the possible new state of the system and $s(t)$ denotes the current state of the system. Then,

If: $p \geq 1$, $s(t + \Delta t) = s'$.
Otherwise: $s(t + \Delta t) = s'$ with probability p.

It can be shown that, as $t \to \infty$, $P(s(t))$ converges to $P(\mathbf{s})$, under the condition that an appropriate choice of Q is made[4]. In this simulation we use the simplest possible proposal density - the spherically symmetric distribution - so that the latter fraction is unity. In this case, equation 5 reduces to $p = \exp\left[-\beta\left(\mathcal{H}_{s'} - \mathcal{H}_{s(t)}\right)\right]$.

We implement this procedure with the 'black-red' method. The lattice is divided up into a checkerboard-like grid of 'black' and 'red' sites, which all contain one spin. For each 'black' spin s_x, we calculate the Boltzmann weights that describe the coupling of all four nearest neighbors (above, below, left & right) to s_x, which are red spins. Then, knowing p, we either flip the spin immediately ($p \geq 1$) or flip the spin with probability p. This should then be repeated for the 'red' spins. This parallel updating of the lattice is possible because the red and black spins are independent of each other.

C. The Swendsen-Wang algorithm

When simulating around critical phase transitions, we encounter the problem of critical slowing down. This phenomenon occurs due to the divergence of the correlation length ξ and the correlation time τ around the critical point The slowdown is characterized by the parameter z: $\tau \sim \xi^z$. For the Metropolis method $z \simeq 2.125$, which means that the slowdown scales as $L^{2.125}$ since the correlation length at the critical point approaches the linear lattice size in a finite lattice[5]. The Swendsen-Wang (SW) method was developed to reduce critical slowing down. It has $z \simeq 0.35$[5].

The SW algorithm is a MC method that considers the links between the spins. For each link there are two options:

Either: the spins connected by the link are opposite, in which case the interaction between them is set to 0.

Or: the spins connected by the link are equal, in which case the interaction is either set to 0 with probability $p_d = \exp(-2\beta J)$ or made infinitely strong with probability $1 - p_d$.

In this way, the lattice is divided up in clusters of equal spins. Each cluster is then assigned a random new spin value after which the original Ising bonds are restored and the process is repeated[5].

D. Finite Size Scaling

To observe non-analytic behavior, one needs to consider an infinite lattice. A possible method to find the critical temperature and the critical exponents on a finite lattice is finite size scaling.

In an infinite system, the correlation length diverges near the critical point as $\xi \sim |T - T_c|^{-\nu}$. However, since a finite system has dimensions L, the system becomes effectively ordered when $\xi \sim L$. There is thus a 'pseudo-critical' point which satisfies $(T_c(\infty) - T_c(L))^{-\nu} \sim L$.

Now, consider one of the relevant physical quantities - e.g. χ_M. χ_M has a maximum at the critical point, because it diverges at T_c. Since $\chi_M \sim |T - T_c|^{-\gamma}$, we find that

$$\chi_{M,\max} \sim (T_c(L) - T_c(\infty))^{-\gamma} \sim L^{\gamma/\nu}. \tag{6}$$

So, to find T_c and the critical exponents, one

1: locates the maxima of χ_M for various lattice sizes L,
2: fits $T_c(L) = T_c(\infty) - c_1 L^x$ through the data, where $T_c(\infty)$ and $x = -1/\nu$ are the parameters of interest, and
3: estimates the exponent γ/ν from the maximum value of $\chi_{\max}$ using equation 6.

E. Binder Cumulant

Another frequently used method to determine the critical point is to use the intersection points of the Binder cumulants. This is often better than the maximum location of physical quantities, because the finite size effects are usually much reduced. The fourth order Binder cumulant is defined as $U_L = 1 - \frac{\langle s^4 \rangle}{3\langle s^2 \rangle^2}$, where $s = L^{-d} \sum_i s_i$. In the symmetric phase, $U_L = 0 + \mathcal{O}(1/L)$ as $L \to \infty$. In the broken phase, $U_L = 2/3 + \mathcal{O}(1/L)$ as $L \to \infty$. The larger L, the steeper this transition will be. For $L = \infty$, this behavior is described by a step function. The strategy is now to calculate U_L for various L as function of T. Usually, one finds the crossings using ascending pairs of volumes L_1/L_2, L_2/L_3, ..., where $L_1 < L_2 < L_3$. The crossing point of the ascending pairs then converge to the critical temperature.

F. Computational Implementation

The methods discussed above are implemented using the Python programming language. Both methods used to simulate the Ising lattice have their own merits and demerits. The Metropolis algorithm can quickly compute singe steps due to the fact that Numerical Python (Numpy) allows for the vectorization of the lattice. The computational complexity of these operations scales as $\mathcal{O}(L^x)$, where $x \approx 1.5$ as discussed in the Appendix. However, the dynamic critical exponent, $z = 2.125$, causes this algorithm to suffer from critical slowdown.

The SW method does not suffer from critical slowdown as severely, since $z = 0.35$. However, the algorithm does require the identification of the clusters in the lattice.

The algorithm used to do this is the Hoshen-Kopelman algorithm, which is a specific application of the union-find algorithm, well-known to computer scientists. This algorithm requires that each link is considered at least once as well as the merging of clusters, causing a computational complexity of $\mathcal{O}(L^x)$, with $x \approx 3.2$ as discussed in the Appendix. It might be possible to reduce this slightly by finding a clever way to reduce the operations needed to merge clusters.

When only considering the computational complexity it seems like the two methods are on par, both scaling as $\mathcal{O}(L^x)$ with $x \approx 3.5$. While we will see that the Swendsen-Wang implementation is able to provide better results when both algorithms are given equal computational time.

III. RESULTS

In this paper, we have simulated the Ising model with the Metropolis MC algorithm as well as with the Swendsen-Wang (SW) method in the Python programming language. Due to the limitations of interpreted languages such as Python we used Cython, an optimising static compiler, for the computationally expensive parts of the Swendsen-Wang algorithm.

A. Metropolis simulation

Critical slowing down is one of the main limitations of the Metropolis MC method when applied to the Ising model. Below we present some of the data obtained using this method. Due to the limited computational power available the estimated error is rather large on all of the data making it ill-suited for estimating the critical exponents. We do, however, choose to present the data to illustrate the implications of slowing down and the power of the SW method.

In figure 1 the absolute value of the magnetization is shown for each thousand computed steps of the Metropolis MC method and the SW method. The computations take place at the critical temperature and for a linear lattice size ($L = 100$). It is clear from this figure that the fluctuations of the Metropolis method occur much slower than those computed with the SW method.

To further demonstrate the severity of critical slowing down we show the computed results of the magnetic susceptibility, χ_M, as a function of temperature for different lattice sizes in figure 2. Critical slowing down has two serious consequences in this case. Firstly, due to the large time needed to gather enough fluctuations in the magnetization the error increases quickly for increasing lattice sizes. Secondly, as the lattice size increases the limited computational time available causes the fluctuations to slow down so dramatically that the peak height, χ_{max}, remains constant as L increases. Both of these problems can be solved by allocating more computation time, although the approximately quadratic scaling of the correlation time will make this highly impractical.

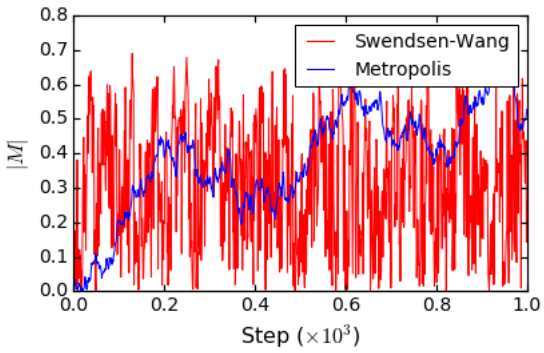

FIG. 1. The fluctuations of $|M|$ at the critical temperature for a linear lattice size, $L = 100$. Note that $|M|$ fluctuates much slower when computed with the Metropolis method than with the SW method.

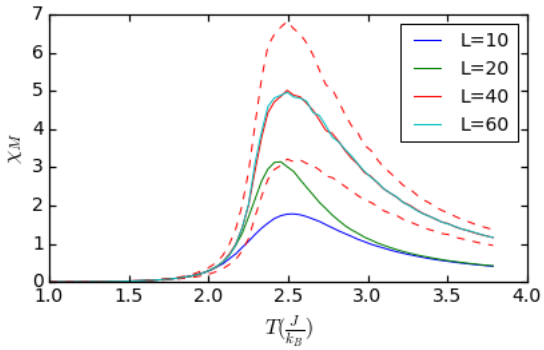

FIG. 2. The magnetic susceptibility, χ_M, as function of temperature for different lattice sizes and using the Metropolis method. For clarity the uncertainty is only shown for $L = 40$. Note the large uncertainty and the fact that the peak size does not keep on growing for larger L.

B. Swendsen-Wang simulation

The Swendsen-Wang method, although less severely hindered by critical slowing down than the Metropolis MC method, still suffers due to the fact that the algorithm is computationally expensive. This, coupled with a limited time-frame, means that the data presented here is limited in linear lattice size.

First, we determine the critical temperature of the system by using the Binder cumulant method. Figure 3 shows the fourth order Binder cumulant, U_4, for different lattice sizes. The crossing points of ascending pairs of sizes are all very close the theoretical critical point. The highest pair has its crossing point at $T \simeq 2.266$, which is very close to the exact value of T_c.

To determine β the value of M_{RMS} (shown in figure 4) is taken at T_c for each lattice size. Using the scaling relation, $M_{RMS}(T_c) \propto L^{-\beta/\nu}$ we find that $\beta/\nu \simeq 0.138$ which decreases towards $\beta/\nu \simeq 0.122$ when not accounting for the smallest lattice sizes. This is quite close to the exact value $\beta_{exact} = 0.125$, with an error of approximately 10%. Using larger lattice sizes will probably improve this result.

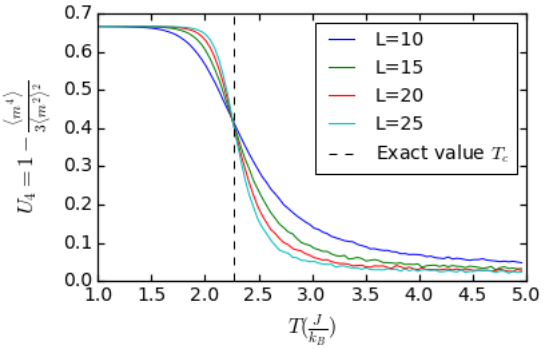

FIG. 3. The Binder cumulant for different lattice sizes. The crossing point of ascending pairs (e.g. $L = 15$ and $L = 20$) converge towards the critical temperature, the highest pair ($L = 20/L = 25$) has a crossing point at $T \simeq 2.266$, which is a very good estimate of T_c.

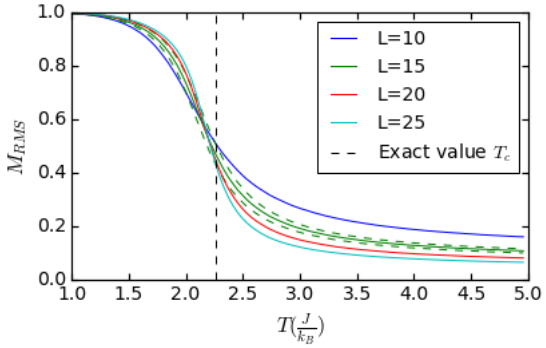

FIG. 4. The root mean square value of the magnetization for different lattice sizes, $M_{RMS} = \sqrt{\langle m^2 \rangle}$. For clarity the uncertainty is only shown for $L = 15$, furthermore the critical temperature is indicated in the figure.

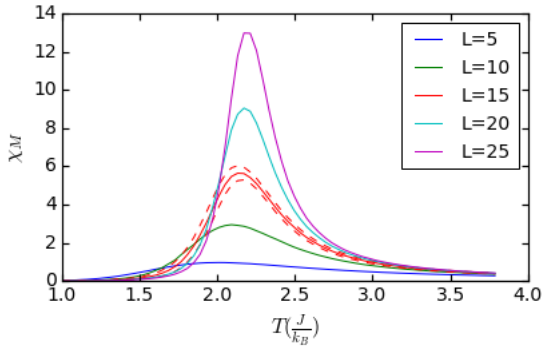

FIG. 5. The magnetic susceptibility, χ_M, as a function of temperature for different lattice sizes. For clarity the uncertainty is only shown for $L = 15$.

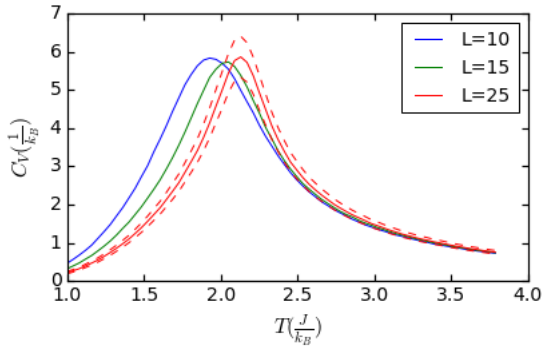

FIG. 6. The heat capacity, C_V, as a function of temperature for different lattice sizes. For clarity the uncertainty is only shown for $L = 25$.

To determine γ, the critical exponent associated with χ_M, we use the finite size scaling relation, $\chi_{\max} \propto L^{\gamma/\nu}$. Figure 5 shows χ_M for different lattice sizes, where the height of the peaks provides us with $\chi_{\max}$. We find that $\gamma/\nu \simeq 1.617$ which again agrees reasonably well with the exact value ($\gamma/\nu = 1.75$). The discrepancy between the simulation and the exact value is probably due to using too small lattice sizes.

For the critical exponent α, we expect to find the value zero since the specific heat diverges logarithmically at the critical point. To find this divergence with the limited lattice sizes available is not feasible with an acceptable error. In figure 6 the specific heat is shown for a few lattice sizes. From this figure it is clear that the height of the peaks only shifts towards the critical point and does not grow with lattice size. This is an indication that indeed $\alpha \simeq 0$.

IV. CONCLUSION

We investigated the behavior of the 2D Ising lattice around the critical temperature with the Metropolis and Swendsen-Wang (SW) algorithm in Python. The critical temperature was determined with the fourth order Binder cumulant, while finite size scaling was used to determine β and γ. The found values ($\beta = 0.122,\ \gamma = 1.617$) correspond reasonably well with the exact values ($\beta = 0.125,\ \gamma = 1.75$). Furthermore, indications were found that α indeed tends to zero for $L \to \infty$. The discrepancy between the simulation results and the exact values is probably due to using too small lattice sizes. We also found that the SW algorithm indeed performs much better than the Metropolis algorithm around the critical temperature, although it under performs in Python. To overcome that problem, the computationally expensive parts of the SW algorithm were implemented in Cython.

To improve this simulation, future work could focus on using bigger lattices. This is however limited to the amount of computertime available. Also, it can be advised to implement the simulation in a compiled language like C or Fortran.

To extend the analysis of the Ising lattice, future work could focus on determining critical exponents α and η. Finally, simulations of other and/or higher dimensional configurations, such as the triangular lattice or the 3D square lattice, could be performed.

Appendix A: A short introduction to Cython

The objective of this appendix is to provide a quick introduction to Cython. The implementation of the SW algorithm was one of the main obstacles when writing this paper. The problem is that the Hoshen-Kopelman (HK) algorithm is computationally expensive and both Python and NumPy are ill-suited to deal with this. The computationally expensive part of the problem cannot be vectorized and to access each element in the arrays nested for-loops must be used. Interpreted languages such as Python perform much worse with for-loops than compiled languages such as C or Fortran. This is due to the fact that the compilers of compiled languages optimize the code and translate it to commands which the processor can follow as efficiently as possible. In general, using compiled languages in these situations will result in shorter computation times.

Although it is possible to implement the whole simulation in a compiled language, this removes the advantages of using a language like Python, which has a lot of useful packages such as NumPy, Matplotlib, SciPy and IPython. Python is also much better readable than many compiled languages and easier to adjust and run since there is no need to compile. A compromise is Cython. Cython is an optimizing static compiler for both the Python programming language and the extended Cython programming language[6]. Typically only a small piece of the code is computationally expensive. IPython magics allows you to call the Cython compiler. The compiler will then compile the code that you want to optimize. When these are compiled the functions can be called throughout the code.

When compiling a piece of code written completely in Python there will not be any major improvements. Most improvements can be gained by declaring static types of the variables used. These declarations allow the compiler to take the data type of the variables into account. When using NumPy arrays it is also possible to specifically import NumPy into Cython and declare the static types of arrays. These simple adjustments already have the potential to increase the speed of the code by orders of magnitude. A simple, but insightful, example of this is shown at docs.cython.org.

By comparing the Python and Cython functions used for this paper it is possible to see how Cython has helped to improve the performance of the code without the need for additional programming languages. For more information and documentation it is advised to visit: `http://cython.org` and/or read *Cython Tutorial*[7] which can be found here.

Appendix B: Performance of used algorithms

As mentioned before, the performance of both the Metropolis and SW algorithm was determined in terms of how their speed scales with the linear lattice size. In figure 7 the linear fit indicates that algorithms scale as $\mathcal{O}(L^x)$ where $x \approx 1.5$ for the Metropolis algorithm and $x \approx 3.2$ for the SW algorithm. Furthermore, an estimate was made of the performance gained by using Cython. In this case Cython improved the performance of the SW algorithm by a factor 20-25.

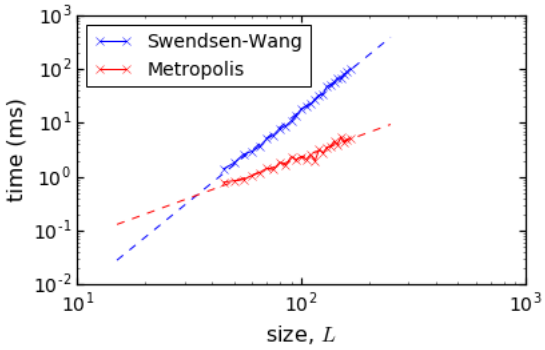

FIG. 7. The time needed for one step as a function of lattice size for both the Metropolis algorithm and the SW algorithm in Cython. Note the log-log scale, which allows a linear fit (dashed) to determine x in $\mathcal{O}(L^x)$.

[1] Elad Schneidman, Michael J Berry, Ronen Segev, and William Bialek. Weak pairwise correlations imply strongly correlated network states in a neural population. *Nature*, 440(7087):1007–1012, 2006.

[2] Dietrich Stauffer. Social applications of two-dimensional ising models. *American Journal of Physics*, 76(4):470–473, 2008.

[3] J.M. Thijssen. Lecture notes in advanced statistical mechanics, September 2014.

[4] D.J.C. MacKay. Introduction to monte carlo methods.

[5] J.M. Thijssen. *Computational Physics*. Cambridge University Press, Cambridge, United Kingdom, 1999.

[6] S. Behnel, R. Bradshaw, C. Citro, L. Dalcin, D.S. Seljebotn, and K. Smith. Cython: The best of both worlds. *Computing in Science Engineering*, 13(2):31 –39, 2011.

[7] Stefan Behnel, Robert W. Bradshaw, and Dag Sverre Seljebotn. Cython tutorial. In Gaël Varoquaux, Stéfan van der Walt, and Jarrod Millman, editors, *Proceedings of the 8th Python in Science Conference*, pages 4 – 14, Pasadena, CA USA, 2009.