Monte-Carlo simulation of the 2D Ising model

Karlo Krakan*
UBC Department of Physics and Astronomy, Vancouver, Canada
(Dated: November 8th, 2017)

I.

I first converted the provided code for the one-dimensional Ising model from MATLAB to Python 3. Then I generalized the provided code to work in two dimensions.

The biggest difference between the two implementations is the two-dimensional square grid rather than a single dimension. I generate an $N \times N$ square grid with a random initial configuration. To calculate the initial energy of the grid I note that going down and to the right on the grid that each spin will have a nearest neighbour to the left and above (with periodic boundary conditions). Therefore to calculate the total energy of the configuration, I loop through each point in the grid and calculate the contribution of each nearest neighbour interaction for every spin in the grid. In other words

$$E_{tot} = \sum_{i,j} E_{i,j} = \sum_{i,j} -J \cdot S^{(z)}_{i,j} \cdot (S^{(z)}_{i-1,j} + S^{(z)}_{i,j-1}) \quad (1)$$

where E_{tot} is the total energy, $E_{i,j}$ is the energy contribution of each spin's two nearest neighbour interactions, J is the interaction energy, and $S^{(z)}_{i,j}$ is the value of the spin at the grid point (i, j).

The next difference is that, since the position of each spin is defined in two-dimensions rather than one, I need to generate a random sequence of spins to be flipped in two dimensions.

Finally, the last difference is the calculation of the energy difference at each time step. In order to calculate the energy difference I find the location of each nearest neighbour (with periodic BCs) to the spin which is flipped and calculate the energy difference ΔE with a simple formula

$$\Delta E = 2J \cdot S^{(z)}_{i,j} \cdot (S^{(z)}_{i-1,j} + S^{(z)}_{i,j-1} + S^{(z)}_{i+1,j} + S^{(z)}_{i,j+1}) \quad (2)$$

Everything else about the algorithm essentially remains unchanged from the provided code aside from some things I have implemented for thermalization which I will now expand upon.

II.

To decide when the system thermalizes I note that the system will always converge to the thermalized configurations whether it starts from a cold start or a hot start. Therefore, I thought the best way to test for thermalization was to run the MCMC algorithm simultaneously for a hot start and a cold start and try to find when the two have converged. The way I have found to do this was to compare the average magnetization and energy of the hot and cold starts and declare the system thermalized when the averages do not differ by much.

Specifically, every 10 sweeps (ie. every $10 \cdot N^2$ iterations of the MCMC algorithm) I calculate the average energy and magnetization over the last 10 sweeps and declare the system thermalized when the combined relative difference, ϵ, between the average energy and magnetization for the hot and cold starts is less than 5%. Specifically, this condition may be stated as

$$\epsilon = \sqrt{\left|\frac{M_c - M_h}{M_h}\right|^2 + \left|\frac{E_c - E_h}{E_h}\right|^2} < 0.05 \quad (3)$$

where M_c,M_h are the average magnetizations over the last 10 sweeps for the cold and hot starts respectively, E_c,E_h are the average energies over the last 10 sweeps for the cold and hot starts respectively. So when $\epsilon < 0.05$ we declare the system thermalized and take the last thermalized grid configuration and feed it into the main MCMC algorithm that will find all the thermalized configurations for which we will actually average over.

Both the use of 10 sweeps and 5% for the relative difference is completely arbitrary and chosen because it seems to work well (after much testing) not only for the temperature interval we are interested in but for temperatures even outside the interval of interest. For instance, in the low temperature limit this condition for thermalization works very well, as seen in Figure 1.

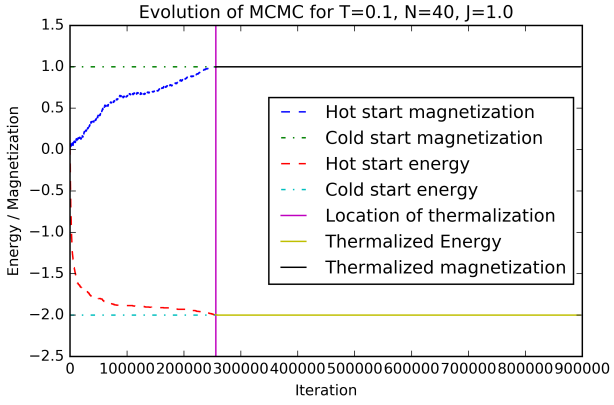

FIG. 1.

In Figure 1 we see the evolution of our MCMC algo-

* karlo@krakan.ca

rithm with temperature $T = 0.1$, grid size $N = 40$, and interaction energy $J = 1$. To the left of the vertical line, which is the location of the iteration at which our system thermalizes, we see the simultaneous evolution of the hot and cold and starts. The blue dashed line corresponds to the magnetization of the hot start at each iteration and the green dash-dotted line corresponds to the magnetization of the cold start at each iteration. Similarly, the red dashed line corresponds to the energy of the hot start at each iteration and the cyan dash-dotted line corresponds to the energy of the cold start at each iteration.

At this low temperature limit, we expect that the system will most often find itself in a state of full magnetization and minimal energy. What we see is that, for the hot start, the system begins at magnetization $M = 0$ and energy $E = 0$ (which corresponds to $T = \infty$, a "hot start") and slowly the system evolves to maximal magnetization and minimal energy at $M = 1.0$ and $E = -2.0$. We also see that the for the cold start, the system already was in the maximal magnetization and minimal energy state. In Figure 1 we see how the algorithm has performed the simulation simultaneously and then decided that the system was thermalized when the cold start and the hot start converged. To the right of the vertical line are the thermalized configurations we use to actually calculate the final estimate of the energy and magnetization. How this is done is that when the algorithm has decided it has reached thermalization, it takes the last (thermalized) grid configuration from the simultaneous MCMC algorithm and uses it as a starting point for the main MCMC algorithm which will always have $10^4 \cdot N^2$ iterations (ie. 10^4 sweeps) so that we always average over $10^4 \cdot N^2$ thermalized configurations regardless of how long it takes to thermalize.

To further demonstrate this algorithm I will show the evolution of the algorithm for the systems with $N = 20$ and $N = 40$, both with $T = 3$ and $J = 1$. In Figure 2, I show the evolution for the system with $N = 40$, but with a smaller number of total iterations so that we may more clearly see the evolution of the first number of iterations before thermalization.

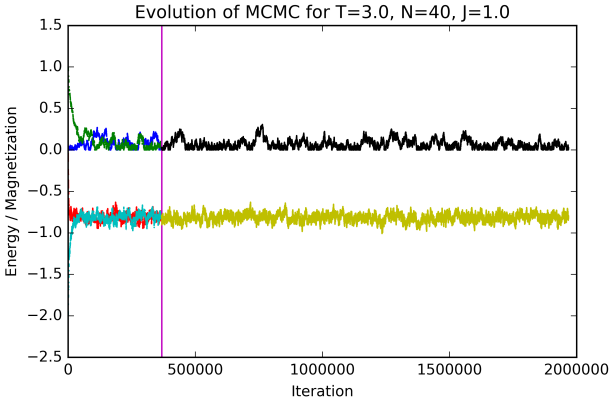

FIG. 2.

So we again that the cold and hot starts begin where we expect and the algorithm seems to stop at a good spot. In Figure 3, we see the evolution of the same system with $N = 40$ but with the normal number of iterations ($10^4 \cdot N^2$), which makes it look much more convincing that the thermalization condition really does do a good job in deciding when the system has thermalized. Since the systems always start in either the hot or cold states, at the worst we will at least always exclude the first number of configurations before the hot and cold starts begin to converge.

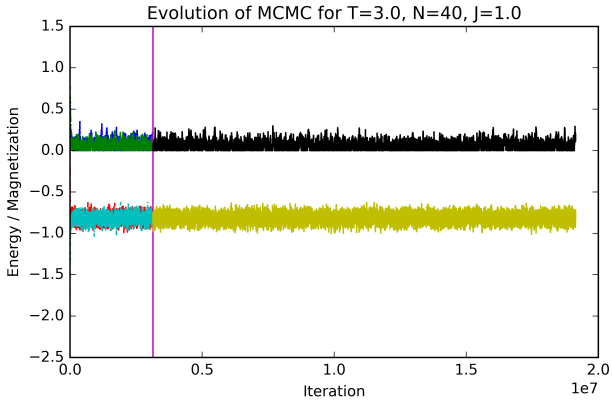

FIG. 3.

In Figure 4 we see the evolution of the system with $N = 20$ (with the normal number of iterations).

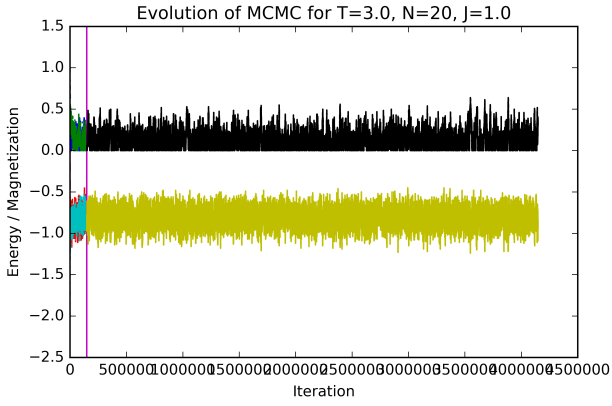

FIG. 4.

I want to investigate how thermalization time depends on the lattice size. I will do this by studying two systems of lattice size $N = 20$ and $N = 40$, both with $T = 3$ and $J = 1$. I have a naive suspicion that the time to thermalization relates linearly with the the lattice size, in other words I find it likely that this relationship may be described

$$\tau_{therm} \propto N \times N \tag{4}$$

where τ_{therm} is the time to thermalization. So I expect that the system with lattice size $N = 40$ will take 4 times

as long, on average, to thermalize than the system with lattice size $N = 20$, since the system with $N = 40$ has 4 times as many grid points. To check this, I have used my algorithm to calculate the average time to thermalize (or more correctly, the average time until my thermalization condition in (3) is met) over 10 runs for the systems with $N = 20$ and $N = 40$. The results are shown in Figure 5 below.

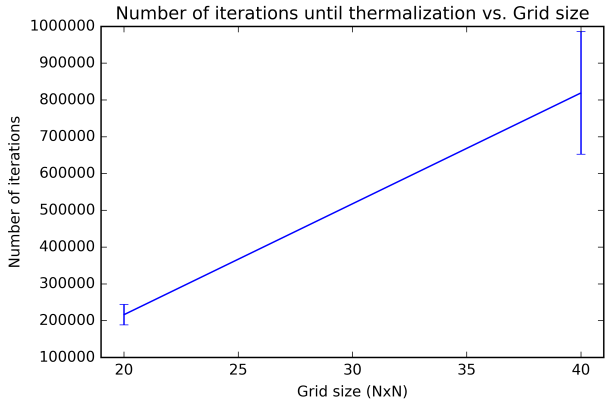

FIG. 5.

What we see is an indication that my naive suspicion is correct. With $N = 20$, the system thermalized on average after $\sim 200,000$ iterations whereas with $N = 40$, the system thermalized on average after $\sim 800,000$ iterations. This is means that it took the system with $N = 40$ on average four times longer (the exact number was ~ 3.78) to thermalize than the system with $N = 20$, as I expected. This scaling makes sense, it always takes a certain number of sweeps over the grid before thermalization, if you increase the grid size, the time to thermalization just scales linearly with the increase in the grid size.

III.

Now I will be investigating how the magnetic susceptibility and the heat capacity of the 2D Ising model. To calculate these quantities, I use the following equations

$$C_V = \beta^2(< E^2 > - < E >^2) \tag{5}$$

$$\chi_M = \beta(< M^2 > - < M >^2) \tag{6}$$

where C_V is the heat capacity, χ_M is the magnetic susceptibility, $\beta = 1/(k_B T)$ where $k_B (= 1)$ is the Boltzmann's constant, T is the temperature, and $(< M^2 > - < M >^2)$, $(< E^2 > - < E >^2)$ are the variances in the thermalized configurations for magnetization and energy, respectively. This is easily calculated with Python. The following figures shows plots of the energy E (Figure 6), magnetization M (Figure 7), heat capacity C_V (Figure 8), and magnetic susceptibility χ_M (Figure 9) for systems with $N = 10, 20, 50$, $J = 1$, between $2.0 < T < 2.8$ with a step size of 0.02. These quantities were calculated using $3 \times 10^4 N^2$ thermalized configurations

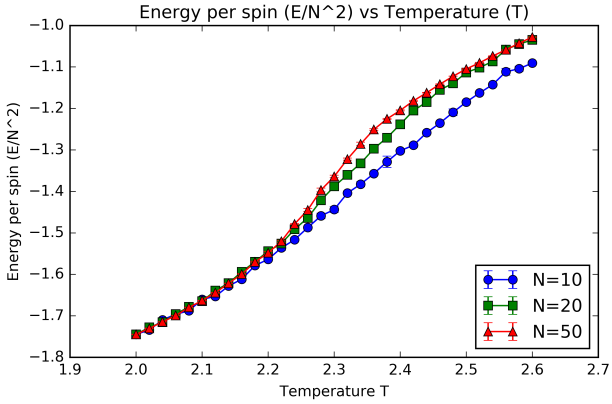

FIG. 6.

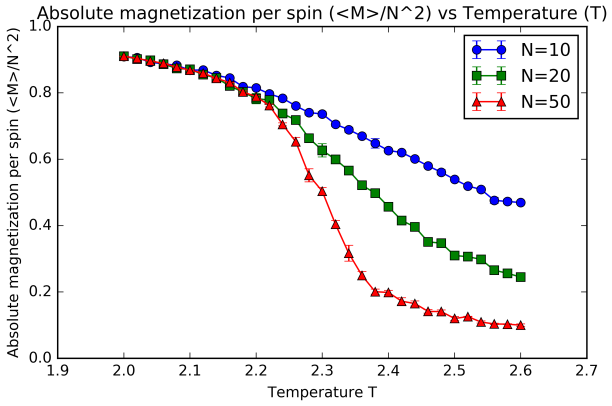

FIG. 7.

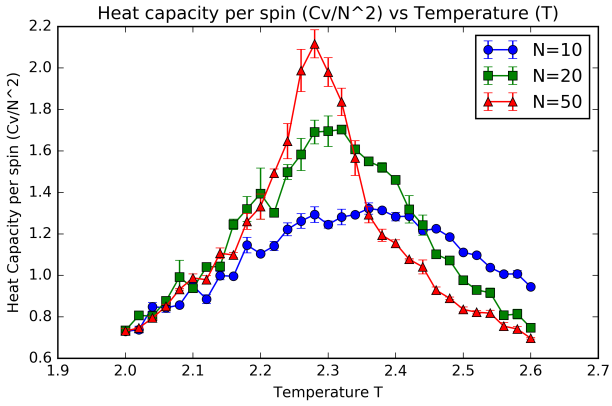

FIG. 8.

What we see in the previous figures is a hint of phase transition near $T = 2.3$. The properties of the system

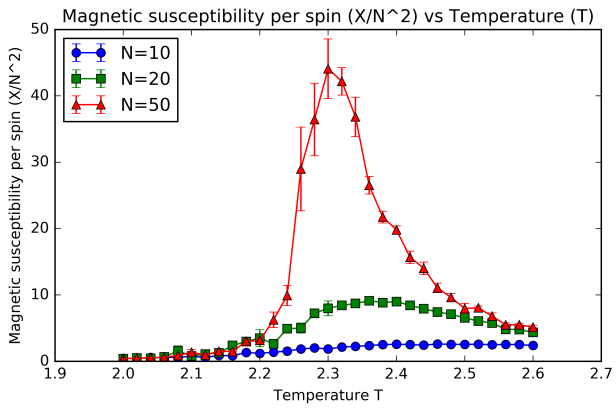

FIG. 9.

change drastically at some point and when they do it represents the system going from ordered state to an un-ordered state. As can be seen in Figure 7 (which shows the magnetization as a function of temperature), we see that initially the systems are in a typical "cold" state at $T = 2.0$, meaning that all (or most) of the spins are aligned. We see as the temperature raises that we are quickly entering the un-ordered state where the spins are mostly mis-aligned. This is the phase transition. We also see that the energy gets larger as we increase the temperature, as expected.

IV.

In the previous Figures 6, 7, 8, and 9, I noted that each quantity E, M, C_V, χ, was calculated using $3 \times 10^4 N^2$ thermalized configurations which is not exactly correct. I actually ran the simulation 3 times, each time using $1 \times 10^4 N^2$ thermalized configurations, and then calculated the average of the three runs and used the standard error to represent the uncertainty in the mean. Doing this helped smooth out the large variations near the critical temperature T_C, where the phase transition occurs and where the system experiences critical slowdown.

It was not that the system was not "thermalizing" using my criteria in (3), but rather that each complete run of the Ising simulation had a large variance as to what the calculated properties actually were. This is not so much true for the energy and the magnetization (note the very small error bars in Figures 6 and 7), but true for C_V and even more so χ.

I could have instead used $5 \times 10^4 N^2$ configurations rather than averaging the quantities over 3 complete runs but I opted to use the averaging method since its still averaging over many configurations but it also gives you an idea of the variation between runs in the areas near the T_C through the standard error. Since near the critical temperature we expect (and confirmed through testing) that there will be a much bigger variation in the energy and magnetization for each thermalized configuration and since χ, C_V are calculated using the variances $(< M^2 > - < M >^2)$, $(< E^2 > - < E >^2)$ of the thermalized configurations, what we expect near the critical temperature is that there will be a big difference between the calculated χ and C_V even between runs. I also thought it could have been the case that even with a single run of $5 \times 10^4 N^2$ configurations that there still could have been a large variance in the calculated quantities between runs near the critical temperature however I did not find the time to test whether this was true. In the Figures 8 and 9 we see that the biggest errorbars on the calculated quantities are the points near $T = 2.3$.

Another minor reason I never used $5 \times 10^4 N^2$ thermalized configurations was that this took a very long time. Using $1 \times 10^4 N^2$ configurations was much faster and seemed to work well so I stuck with it. When I actually did this averaging method over three runs (it took a few hours to complete) I at first got unsatisfactory results near the critical temperature (for $N = 50$. To ameliorate this, I redid the simulations near the critical temperature (for $N = 50$ only) but instead of averaging it over 3 runs, I averaged it over 6 runs. That is how I produced the data for the Figures 6, 7, 8, and 9.

As I have said, in Figures 8 and 9 we see that the biggest error bars on the calculated quantities are near the critical temperature. I will estimate the critical temperature by fitting a Gaussian over the data for the heat capacity C_V for the system with $N = 50$. I do this because it looks like a better data set than for χ but not only that it is more symmetrical, it looks like it captures the whole phase transition, at least for $N = 50$, since the property starts at one quantity, reaches a maximum, and then returns to where it was. For χ this is not so true, its hard to tell where the maximum even should be for χ. I think this is because for $N \to \infty$ that $\chi \to \infty$ near T_C. Note in Figure 7 that for $N = 50$, the magnetization more rapidly approaches 0 than for $N = 10$ and $N = 20$. As N gets larger and larger not only does the magnetization more rapidly approach 0 and so therefore χ, which is also defined as the derivative of the magnetization with respect to temperature, will get larger and larger. On the other hand, the energy increases much more gradually than the magnetization, so that C_V, which is also defined as the derivative of the energy with respect to temperature, will also change much more gradually and not get increasingly large with N as fast as χ does. That is why I chose to use C_V to estimate T_C.

In Figure 10, I have fit a Guassian over the data for $C_V(T)$ for $N = 50$.

The mean of the fitted Guassian is my estimate for the critical temperature. I used this method because I thought it would give a best estimate of the midpoint, not necessarily because the data should fit to a Guassian (I have a suspicion it would fit better for larger N). Calculating the critical temperature in this way I find that $T_C \approx 2.2756$

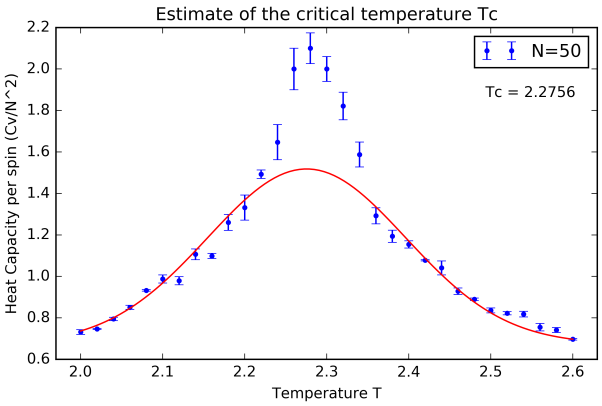

FIG. 10.

V.

Please run "plotscript.py" to generate and save two plots for $N = 20, 40$ at $T = 3$ and $J = 1$. The file plotscript.py also contains the code for generating all the other plots used in this work. Uncomment them to run but be aware they may take a long time to run. The main Metropolis-Hastings algorithm for the 2D Ising model may be found in the file "ising2D.py".