Nathan Zhao
nzz2102
Physics 212

Cluster Monte Carlo Simulations of the Ising Model on a Cubic Lattice

Contents

I. Introduction:

The Ising model is one of the most well-studied models in statistical mechanics and rightly so for the rich insights it provides into phase transitions and universality. First introduced in the 1920s by Ernst Ising and formalized by Lenz, the model starts with the simple Hamiltonian.

$$H = -J \sum_{<i,j>} s_i s_j + \mu B \sum_i s_i \tag{1}$$

Where J represents the nearest-neighbor interaction strength, μ is the Bohr magneton, and B is the magnetic field strength. The partition function for this system is then simply written as:

$$Z = e^{-\beta H} = \exp\left(-\beta\left(-J \sum_{<i,j>} s_i s_j + \mu B \sum_i s_i\right)\right) \tag{2}$$

For many purposes, it is useful to define variables $K = \beta J$ and $h = \beta \mu B$ so that we can refactor the Hamiltonian as

$$H = K \sum_{<i,j>} s_i s_j + h \sum_i s_i \tag{3}$$

The constants K and h are unitless. For our analysis here, we set $h = 0$ (no external magnetic field). In one and two dimensions, the model has an analytic solution, which is to say the partition function Z can be written in terms of analytic functions and exact values of the critical exponents can be obtained (though in 1 dimension, there is no phase transition except at 0 temperature). To date, no analytic solution to the Ising model has been found in higher dimensions though approximation techniques, particularly mean field theory, have yielded relatively accurate if not precise predictions about the behavior of the model in higher dimensions.

In this paper, we will explore the Ising model in $d = 2,3,4,5$ to illustrate the power of Monte Carlo methods and use it as evidence of the accuracy of mean field approximations in dimensions greater than two. The remainder of the papers is structured as follows. Section II introduces the main results of mean field theory, as well as the definitions of the critical exponents. Some simple calculations are done to provide estimates

Section 3 introduces Monte Carlo techniques, specifically cluster techniques, which are the state of the art in simulating the Ising model. Section 4 provides the main numerical results of the paper and Section 5 concludes the paper.

II. Mean-Field-Theory of the Ising Model and the Critical exponents

In the usual analysis of the Ising model and similar variants, we are interested in calculating various descriptive quantities for the system, starting with the partition function Z.

$$Z = \sum_{\{s_i\}} \exp(-\beta H) = \sum_{\{s_i\}} \exp\left(-K \sum_{\langle i,j \rangle} s_i s_j\right) \tag{4}$$

Where H is the Hamiltonian and $\beta = 1/k_b T$ and we sum over potential configurations $\{s_i\}$ on a regular lattice. If we know the partition function, from there we can straightforwardly calculate quantities such as the expectation value of the spin (magnetization), the free energy, etc. For the purposes of this paper, we will analyze the magnetization and the energy. From there, we can find the susceptibility:

$$\chi = \frac{\partial m}{\partial h} \tag{5}$$

And the heat capacity:

$$C_v = \frac{\partial \langle E \rangle}{\partial T} \tag{6}$$

Of course, analytic expressions of Z don't exist for $d > 2$, so how can we estimate these quantities of interest?

Mean field theory is done based on coarse-graining the system with an order parameter. Heuristically, we can determine the form of Hamiltonian based on this order parameter by forcing it to satisfy three constraints: locality, rotational symmetry, and symmetry in the variable x.

The central idea of deriving the critical exponents is the Landau-Ginzburg Hamiltonian, which has the form:

$$\beta H = \int \left[\frac{t}{2}m^2(x) + um^4(x) + vm^6(x) + \cdots + \frac{K}{2}\left(\nabla m(x)\right)^2 + \left(\nabla m(x)\right)^4 + \cdots + \left(-hm(x)\right)\right] d^d x \tag{7}$$

This form can be derived by enforcing a set of reasonable constraints such as symmetry, which is discussed further in [1]. We can further approximate this by considering expansions about the most likely configurations of the order parameter, which means can drop the gradient terms in the expansion as they are all zero in the ground state

$$Z \sim Z_{sp} = e_0^{-\beta F_0} \int exp\left[-V\left(\frac{t}{2}m^2(x) + um^4(x) + vm^6(x) + \cdots - hm(x)\right)\right] dm \tag{8}$$

Where V is the system volume and the subscript sp denotes the saddle point approximation. For large volumes, we can perform a saddle point approximation, which effectively attempts to approximate the integral assuming the extremal values of the argument of the exponential dominate the integral:

$$Z_{sp} \sim \int exp\left[-V\left(\frac{t}{2}m^2(x) + um^4(x) + vm^6(x) + \cdots - hm(x)\right)\right] dm \tag{9}$$

which requires us to look at the extremal values of:

$$\Psi(m) = \frac{t}{2}m^2(x) + um^4(x) + vm^6(x) + \cdots - hm(x) \tag{10}$$

As we are interested in behavior only near the critical point, we expect the overall magnetization to be small and hence we will also selectively drop higher order terms in our analysis. The final form of the Landau free energy is then:

$$\Psi(m) = \frac{t}{2}m^2(x) + um^4(x) - hm(x) \tag{11}$$

This form has a strong dependence on the sign of the coefficients. Below, we show a plot of the Landau free energy for several different cases:

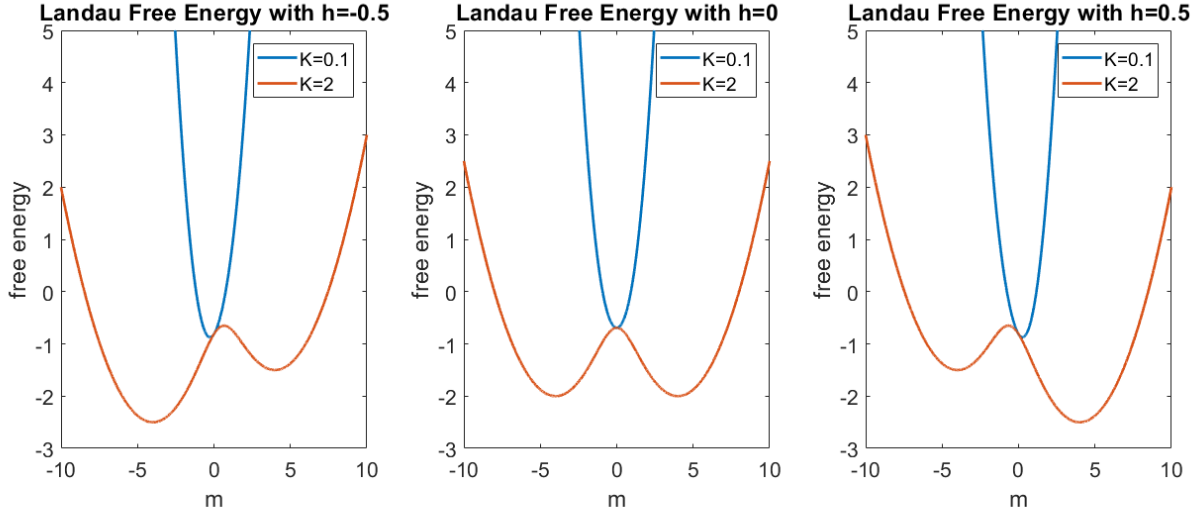


Figure 1: Landau Free Energy for different temperatures and different values of the magnetic field

Taking the derivatives of this leads us to solve an equation of the form:

$$\Psi'^{(m)} = tm + 4um^3 - h \ = 0 \tag{12}$$

First, we point out the fact that the parameter t has a specific importance, namely that $t \sim (T - T_c)$. To see where this comes from, consider that we know the coefficients are all analytic functions of temperature and hence be expanded as a Taylor series about the critical point.

Critical Exponent	Expression
Magnetization (M)	$m \sim \frac{1}{2}(T - T_c)^{\beta}$
Susceptibility (χ)	$\chi \sim \frac{1}{2}(T - T_c)^{-\gamma}$
Correlation length (Type equation here.)	$\xi \sim \frac{1}{2}(T - T_c)^{-\nu}$

Table 1: critical exponent expressions

It is these exponents which we wish to determine using the cluster Monte Carlo technique. Luckily, we can also analytically approximate the critical exponents using Mean field theory. Assuming $h = 0$, we can see that:

$$\mathrm{m} \sim |\mathrm{T} - \mathrm{T_c}|^{\beta} \tag{13}$$

Taking the derivative with respect to h gives:

$$\chi \sim |\mathrm{T} - \mathrm{T_c}|^{-\gamma} \tag{14}$$

Finally, we have the critical exponent for the heat capacity, which actually has different forms in different dimensions. In 2D:

$$\mathrm{C_V} \sim \log\left(\frac{1}{|T - T_c|}\right) \tag{15}$$

In this paper, we consider only the three critical exponents β, ν, and ξ. There exist scaling relations between the critical exponents so once these three are determined. We can effectively determine all other relevant critical exponents [2]. First, the literature (or very exact results) are displayed in Table 2.

Dimension	T_C	β	γ	ν
2	2.269	0.125	7/4	1
3	4.4511	0.33	1.25	0.63
4	6.668	0.5	1	0.5
5	8.7	0.5	1	0.5

Table 2: critical exponents for different dimensions of the Ising model on a regular lattice

In fact, for $d \geq 4$, the predictions of mean field theory are exact. From an intuitive standpoint, this is due to the increasing number of nearest neighbors per site in higher dimensions.

Cluster Monte Carlo:

The essential algorithm we will use to simulate our systems is Markov chain Monte Carlo. Essentially, we want to construct a sequence of configurations (specifically of the Ising lattice) of our problem space that, on average, represents the stationary distribution of our system (represented by the Boltzmann distribution in our case).

The classic approach to simulating the Ising model is the Metropolis-Hastings (based on Gibbs sampling) algorithm, which is a local update algorithm. Essentially at each iteration, it proposes a spin flip at one lattice site, calculates the energy change and accepts or rejects the proposal in such a way that it samples the Boltzmann distribution for the system.

However, it is well known that the Metropolis-Hastings algorithms does not work well near the critical point, where the correlation length diverges and hence it becomes more difficult to generate configurations which are independent configurations just by sampling single sites. A simple way to see this is to consider the magnetization or average spin on the lattice. On a lattice with N_s total sites, the magnetization should change by at most $O(1/N_s)$ per iteration, which is rather small number and almost definitely related to the magnetization calculated at the previous iteration (hence the two values are correlated).

An alternative approach called the cluster Monte Carlo was proposed in the late 1980s. The basic idea is to analyze clusters of spins instead of single spins at each iteration to circumvent the long correlation lengths near the critical point. There are two flavors of the algorithm. The first was proposed by Swendsen and Wang in 1987 [3]. Instead of single spin flips, we now form clusters. The clusters are formed via bonds connecting nearest neighbors with the same spins. Each bond is formed only with probability $p_{ij} = 1 - \exp(-2\beta J)$.

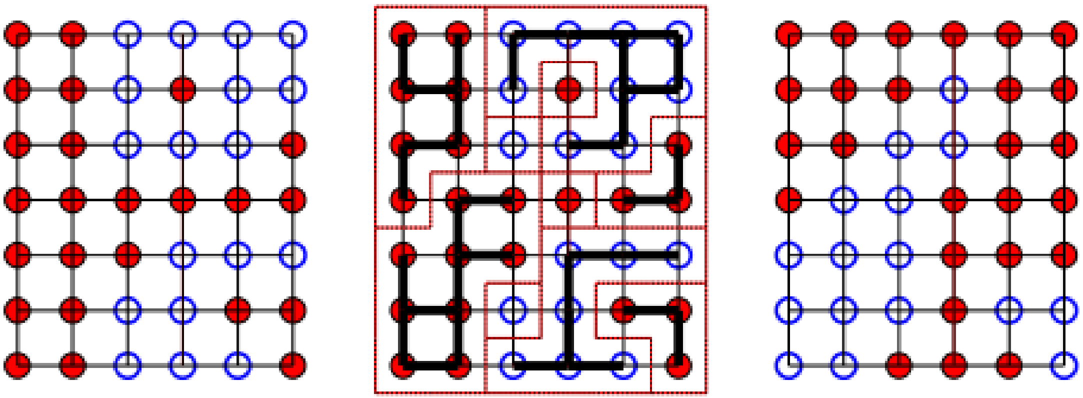


Figure 2: source: http://www.helsinki.fi/~rummukai/simu/cluster.pdf

All spins connected with the bond essentially form a cluster. A decision is made to either flip or retain the cluster with probability of one-half. Hence, in each step, we perform a global update of the lattice versus a local update.

The second version was proposed by Uli Wolff just a few years later [4]. It looks very similar to the SW algorithm but proposes and flips one cluster per iteration. We will use the Wolff algorithm to simulate the Ising model, so we will dive a little bit more into detail. The key to flipping a cluster versus a single spin is that the algorithm needs to be able to account for the statistical dynamics that affects the a priori probabilities of state (or configuration) a (lattice before cluster formation) and state b (lattice after cluster is formed), i.e., we want to construct an acceptance probability (much like the Metropolis-Hastings model) of moving from $a \to b$. Suppose beforehand, we know that we will pick a site at random and start looking at its nearest neighbors and potentially add them to the cluster with a probability p. We then recursively repeat this step until the cluster stops growing.

We can now look at the cluster and look at the links that must've been rejected, which we call n_s. We then look at the number of links where the spins are opposite and call this number n_d. Now suppose we make the transition to state b. With all the spins in the cluster flipped, we can make an analogous analysis of moving from state b back to state a and count links with spins in the flipped cluster with the outside spins. What one predictably finds is that n_d and n_s become switched. What becomes apparent is that the ratio of the transition from $a \to b$ can be captured with the expression:

$$P(a \to b) = \min\left\{1, \frac{exp(\beta n_d)\, exp(-\beta n_s)}{(1-p)^{n_s}}\left(\frac{(1-p)^d}{exp(-\beta n_d)\, exp(\beta n_s)}\right)\right\} \tag{16}$$

$$= \min\{1, \left[\frac{\exp(-2\beta)}{1-p}\right]^{n_s}\left[\frac{1-p}{\exp(-2\beta)}\right]^{n_d}\}$$

Here, we never specified the probability p, it is a free parameter. However, if we choose $p = 1 - \exp(-2K)$, then we see that the right-hand term is also one and hence, the algorithm becomes rejection free, which is one of the key advantages of the Wolff algorithm over both the Swendsen-Wang algorithm and the classical Metropolis-Hastings algorithm.

We see that the cluster algorithm can change the spin configurations drastically in one step, particularly at low temperatures. In fact, at low temperatures, the cluster algorithms are asymptotically the same cost as the Metropolis Hastings algorithm since the clusters will grow to be the size of the lattice in this limit. However, at low temperatures, we expect that the cluster algorithms to be significantly faster.

We want to know that this algorithm satisfies the necessary conditions of all Monte Carlo algorithms: ergodicity and detailed balance.

Below, we validate our cluster Monte Carlo algorithm by showing a few snapshots of a 100x100 Ising grid (K) at different iterations during the Wolff cluster algorithm for a temperature slightly higher (so K is slightly smaller than $K_c = 1/2.269$) than the critical temperature ($K = 0.42$).

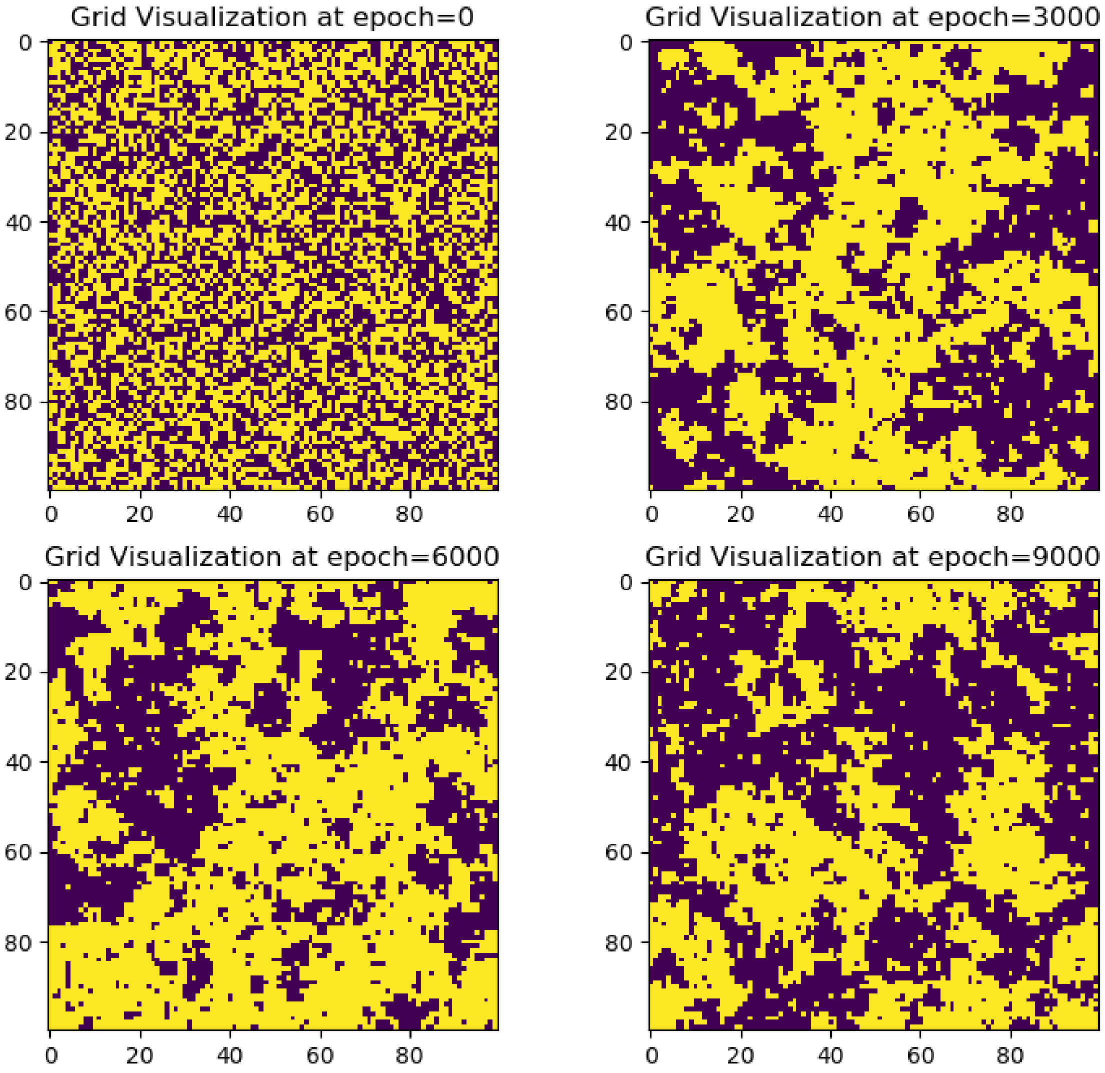


Figure 3 Ising Grid at different iterations of the Wolff Algorithm with $K = 0.42$. Yellow indicates spin up (+1) and dark blue indicates spin down (-1). The initial starting point (epoch = 0) is a lattice of random 1's and -1's.

Near low temperatures, we only need a few epochs before the lattice is completely uniform. We can also observe that sometimes, the spin orientation of the lattice switches (since we have no magnetic field)

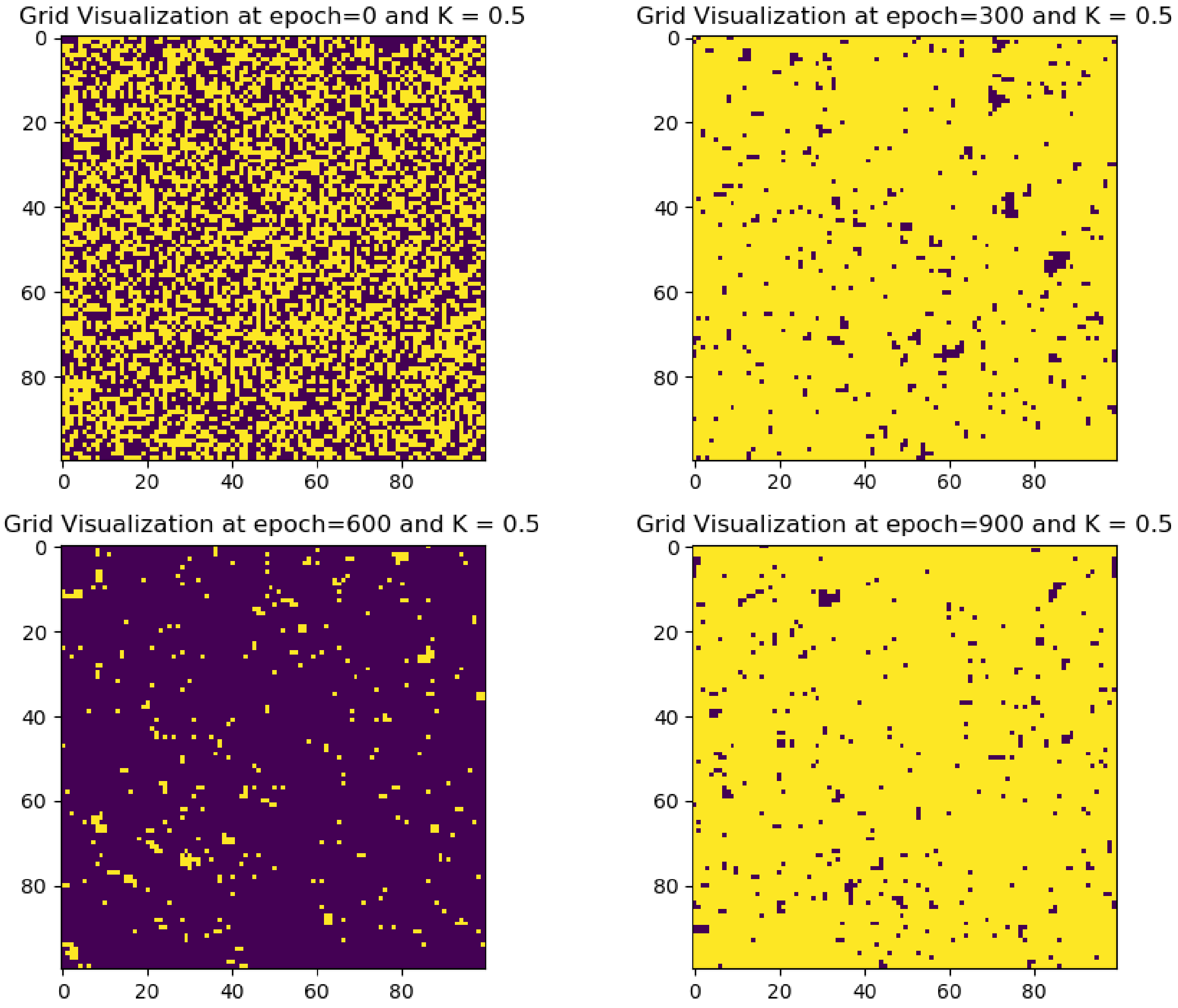


Figure 4 visualization of the Wolff algorithm for several epochs at low temperature ($K = 0.5$)

III. Numerical Calculations

1) Numerical Methods Description

The Monte Carlo simulations are run once in python, where we save all the lattice histories as an individual file (though this becomes inefficient for large iterations). To achieve good results, many references show iterations for the metropolis algorithm numbering in the tens of thousands up to millions [5], with significant accuracy only achieved at the higher end. For our purposes, we will run many fewer iterations due to time constraints (and with the hope that the

Wolff Cluster will be numerically superior in general) but it should be sufficient to resolve much of the behavior we expect to see. The lattice sizes we will run out simulations on will go as follows:

Dimension	Grid size	Total Sites (N_s)	# of nearest neighbors	Number of iterations
2	50×50	2500	4	10000
3	$30 \times 30 \times 30$	2700	6	1000
4	$10 \times 10 \times 10 \times 10$	10000	8	1000
5	$5 \times 5 \times 5 \times 5 \times 5$	3125	10	10000

Table 3: Simulation parameters for the Ising model simulations that we will demonstrate

In addition, we try to minimize the grid size we simulate. Minimizing the grid size will introduce finite size scaling effect near the critical point. To fit the critical exponents, we use the Levenberg-Marquadt algorithm (in Matlab). For every case, we will fit a function of the form:

$$\mathrm{f(T)} = \mathrm{a}|\mathrm{T} - \mathrm{T_c}|^{c} + d \tag{17}$$

Which contains four fitting parameters, a, b, d, and T_c. The two important fit parameters are c, which is the critical exponent and T_c, which is rough estimate of the critical temperature.

2) Calculation of Order Parameters

The simplest parameter to calculate is the magnetization of the lattice

$$\langle M \rangle = \frac{1}{N_S}\left(\sum_i s_i\right) \tag{18}$$

Where sum over N_S sites in the lattice ($N_s \sim N_i^d$) and we will take the absolute value since all spins up and all spins down are equally prefeerred low-temperature ordered phases. In the ordered phase, we expect the absolute value of the magnetization to be close to 1 but in the disordered phase, it should be close to 0. Analogously, we can calculate the expectation value of the magnetization squared:

$$\langle M^2 \rangle = \frac{1}{N_S}\left(\sum_i s_i\right)^2 \tag{19}$$

For the susceptibility, we calculate:

$$\chi = \frac{1}{T}(\langle M^2 \rangle - \langle M \rangle^2\ ) \tag{20}$$

Next is the spin correlation or the quantity $\langle s_o s_r \rangle$

$$\langle s_o s_r \rangle = \frac{1}{N_S}\left(\sum_i s_o s_i\right) \tag{21}$$

To measure correlation length, we have the expression for the spin-spin correlation function $C(r_i - r_j)$:

$$C_{or} = \langle s_o s_r \rangle - \langle s_o \rangle \langle s_r \rangle = e^{-\frac{r}{\xi}} = \frac{1}{N_S}\left(\sum_i s_o s_i\right) \tag{22}$$

To calculate energy, we can just directly read off the Ising Hamiltonian. For simplicity however, we first define the energy per site e_i, which is:

$$e_i = -J s_i \left(\sum_{j \in NN} s_j\right) \tag{23}$$

The total energy is then just: $E = \frac{1}{2}\sum_{i=1}^{N_s} e_i$, where the factor of 1/2 comes from the fact that we are double counting bonds in our definition of e_i. Next is the heat capacity, which is defined from the energy $C_V = \frac{\partial E}{\partial T}$

$$C_V = \frac{1}{T^2}(\langle E^2 \rangle - \langle E \rangle^2) \tag{24}$$

In practice, to calculate these order parameters, we will first 'equilibrate' the lattice by running 100 iterations without measuring any of the order parameters. After that we will run approximately 10000 additional iterations. For each iteration, we calculate the order parameters and aggregate them to generate a measurement for the given order parameters.

1) Finite Size Scaling

Finite size scaling phenomenon occur because the sharp phase transition only occurs in the thermodynamic limit $N_s \to \infty$. Hence, in any finite sized lattice where N is small, the free energy will be regular and the phase transition will not be sharp (i.e. there really isn't a phase transition). This can be seen in the power laws we derive for the correlation length (
$\xi \sim t^{-\nu}$) are for an infinite volume (so it diverges as the temperature approaches the critical point):

$$\xi(T) \sim \xi_0 + \left|1 - \frac{T}{T_c}\right|^{\nu} \tag{25}$$

However, simulations we conduct are on finite sized lattices, hence the correlation length maxes out at whatever the scale of the lattice is. As a result, we expect that:

$$\left|1 - \frac{T}{T_c}\right|^{\nu} \sim \xi(T)^{-1/\nu} \to \left|1 - \frac{T_c(L)}{T_c(\infty)}\right|^{\nu} \sim L^{-\frac{1}{\nu}} \tag{26}$$

Hence, we can say that system has an effective critical point, which is sometimes called a *pseudo-critical point.* We define

$$T_c(\infty) - T_c(L) = aL^{-\frac{1}{\nu}} \tag{27}$$

IV. Numerical Results

1) D = 2

First, we demonstrate some of the preliminary results using too few iterations as shown in Figure 5

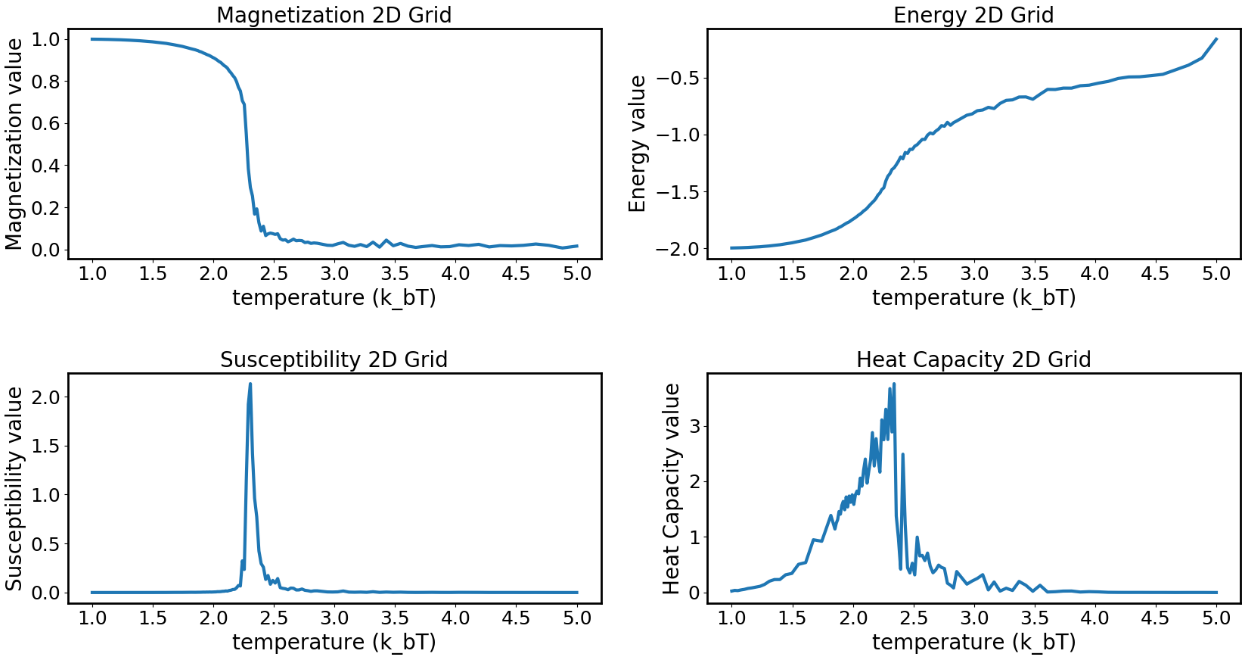


Figure 5: Order measures as a function of temperature on a 2D lattice with only ~200 iterations done. While, this is sufficient to resolve the critical point based on the magnetization, the heat capacity and susceptibility values that are calculated appear to be inaccurate.

In general, we observe that the heat capacity and susceptibility calculations are generally inaccurate for our broad-temperature scans. As a result, we start to run densely sampled simulations near the critical point with much longer iterations (>10000) in order to better resolve the behavior of these variables.

For D=2, figure a shows the fit to the magnetization near the critical point, showing a good fit. For 2D, the exact solution of the critical temperature is given in [6] as:

$$\mathrm{K}_c^{-1} = J\left(-\frac{1}{2}\log\left(\sqrt{2}-1\right)\right) = 2.269 \tag{28}$$

The fitted critical exponent β is given:

$$\beta_{2D} = 0.1136 \quad (0.1107, 0.1165) \tag{29}$$

The two numbers in the parentheses are the 95% confidence intervals. While this number is slightly off from the expected value of 0.125, it is remarkably close.

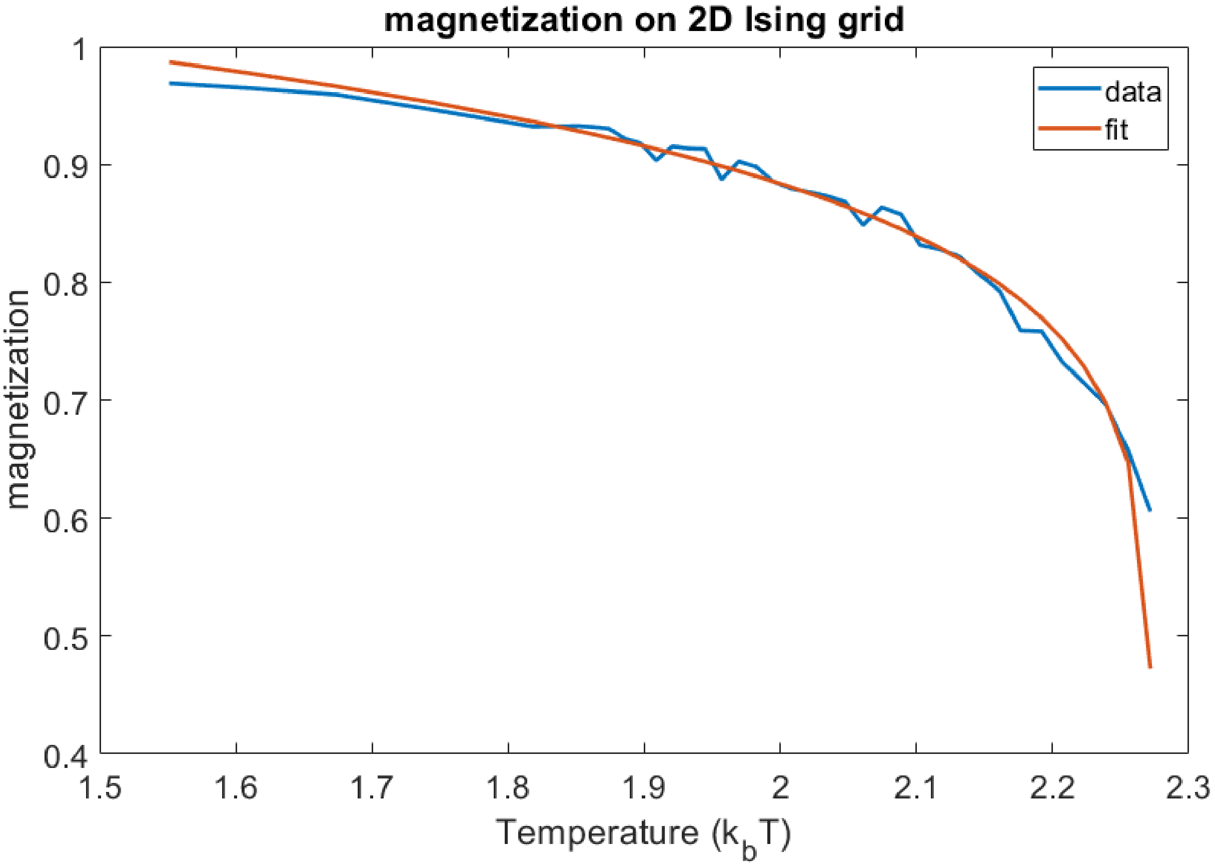


Figure 6: Fit of the critical exponent scaling law for magnetization in 2D

To perform the fit for the critical exponent, we densely sample near the critical temperature and fit the power law form. Clearly, we see significant differences in the plots for heat capacity and susceptibility which indicates more realistic results. We also observe that these were simulated on a 20x20 lattice so the smoothness can be attributed to finite scaling effects (as well as a slight shift of the critical point to what appears to be a higher temperature).

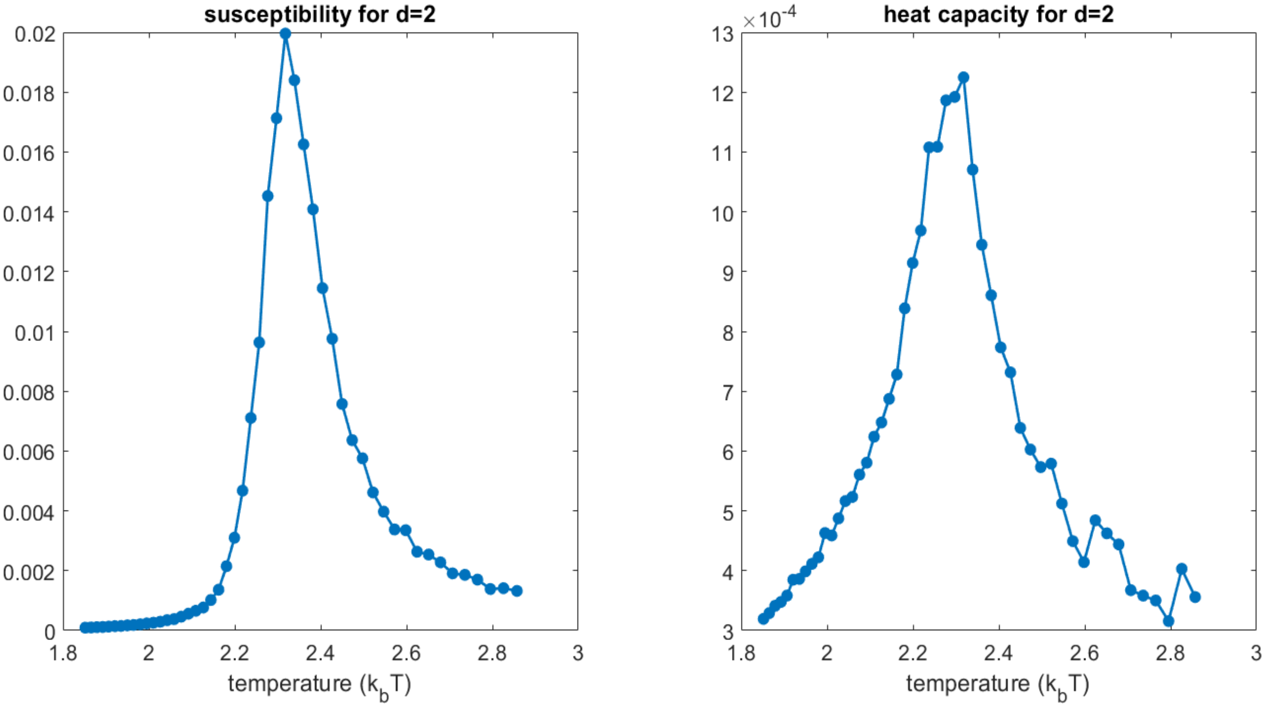
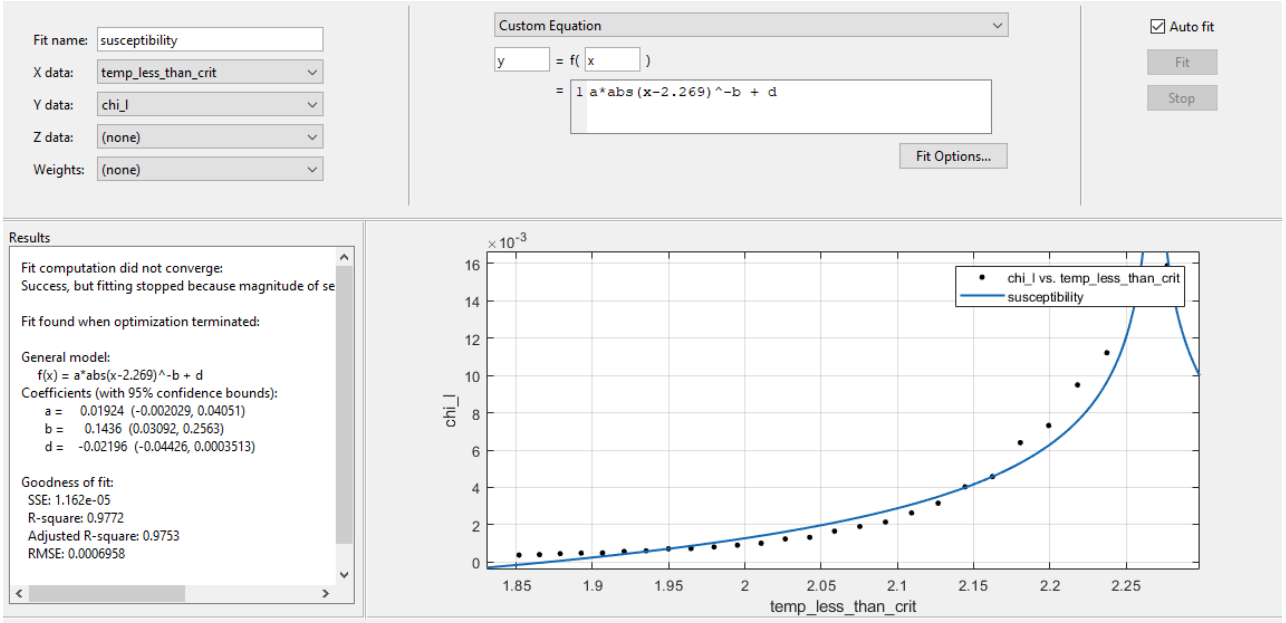


Figure 7 Densely sampled temperature scan of the susceptibility and the heat capacity for D = 2 Ising grid. As the overall lattice size is decreased, we can see the effects of finite scaling in the plot.

The fitted critical exponent for γ is

$$\gamma_{2D} = \ 0.1436 \ \ (0.03092, 0.2563) \tag{30}$$

Onsager's solution for these critical exponents are given in Table 2 as ($\beta = 0.125, \gamma = 1.75,$ $\nu = 1$). Clearly, we can see that the estimate for γ is quite off even though the location in terms of critical temperature is quite close.



Fit name: susceptibility
X data: temp_less_than_crit
Y data: chi_I
Z data: (none)
Weights: (none)

Custom Equation
y = f(x)
= a*abs(x-2.269)^-b + d

Fit Options...

☑ Auto fit

```
Fit computation did not converge:
Success, but fitting stopped because magnitude of se

Fit found when optimization terminated:

General model:
   f(x) = a*abs(x-2.269)^-b + d
Coefficients (with 95% confidence bounds):
     a =     0.01924  (-0.002029, 0.04051)
     b =      0.1436  (0.03092, 0.2563)
     d =    -0.02196  (-0.04426, 0.0003513)

Goodness of fit:
  SSE: 1.162e-05
  R-square: 0.9772
  Adjusted R-square: 0.9753
  RMSE: 0.0006958
```

Figure 8 fit for the suscepbitility in 2D

2) D = 3 Results

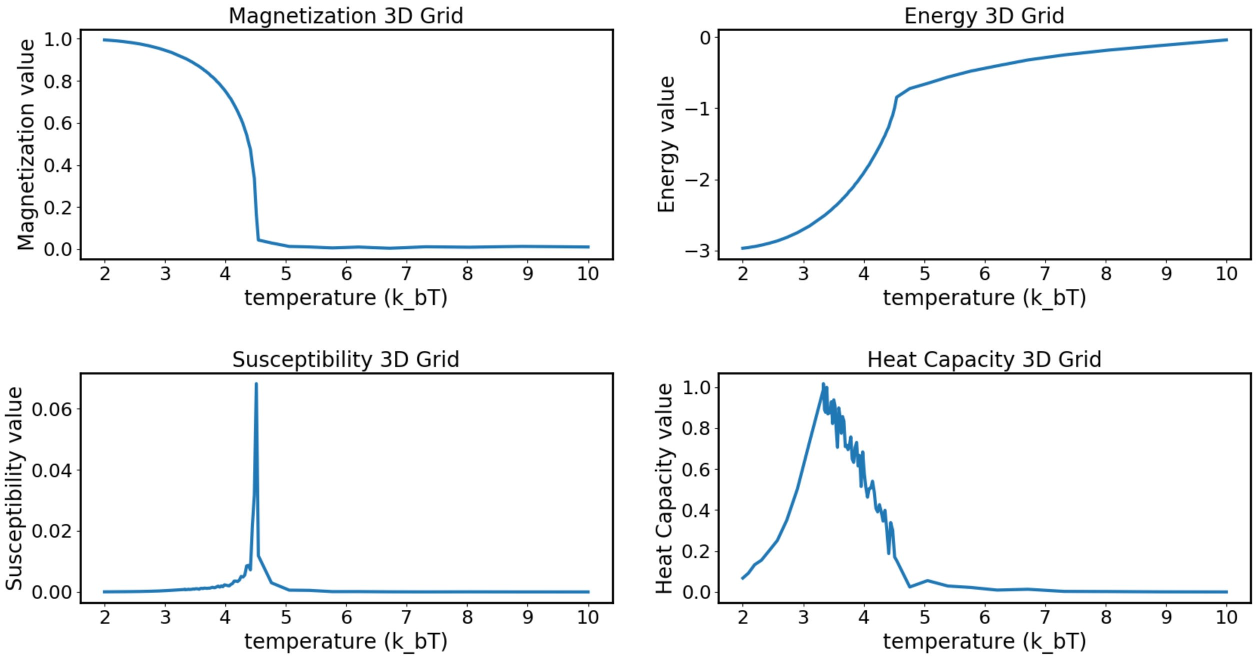


Figure 9 Broad temperature scan for energy magnetization, susceptibility and heat capacity for D = 3

An interesting observation that we can see in 3D that is not present in 2D is that the average energy function appears to develop a kink at the critical point, which is not true in the 2D case (energy is an analytic function). We are not sure if this is a numerical error, but we observe that there is no 3D analytic expression for the energy in the Ising model.

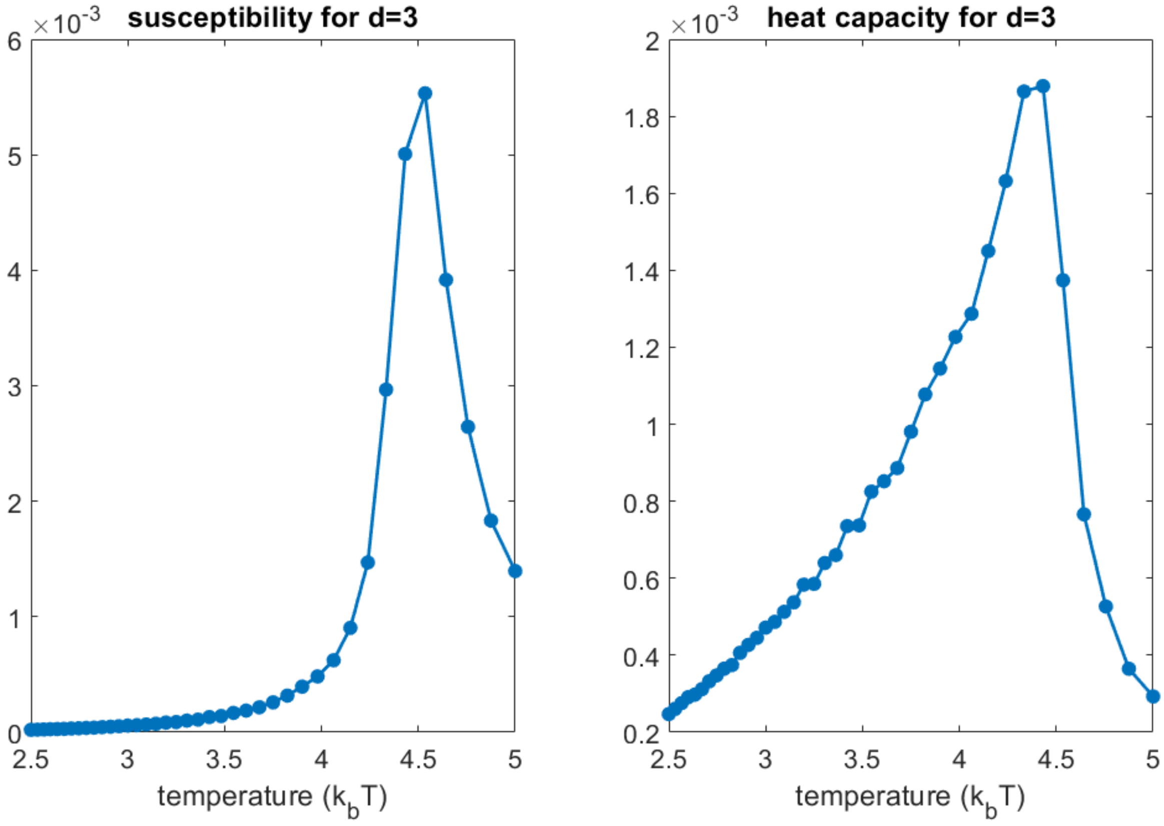


Figure 10 densely sampled temperature scan near the critical point for D=3 Ising lattice.

3) D = 4 Results

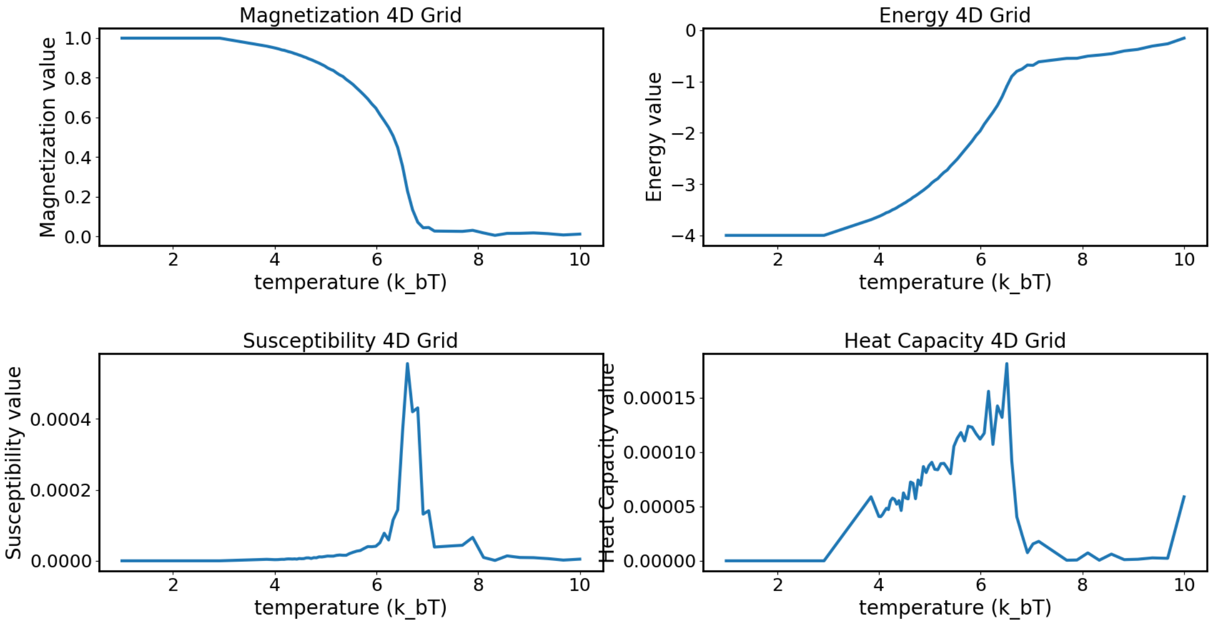


Figure 11 D = 4 temperature scan of the Ising model, showing relatively good results in terms of identifying the correct critical temperature of the phase transition.

The important part from our results is that we can replicate the location of the phase transition with extraordinary precision even accounting for finite size lattice effects

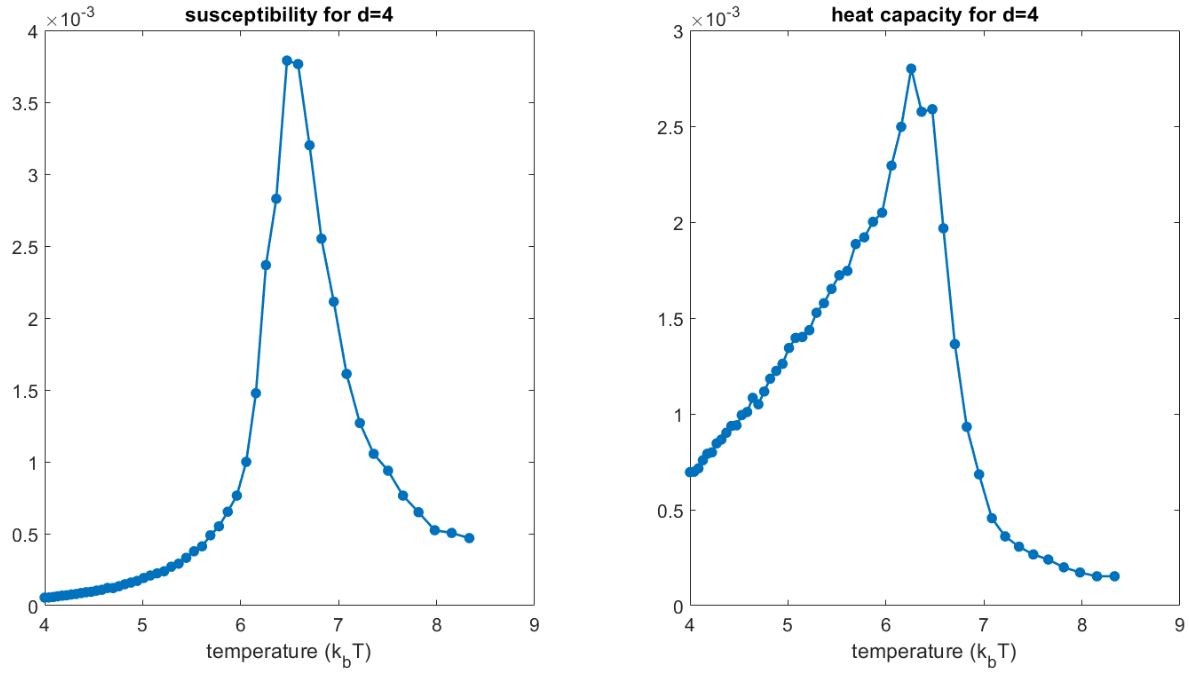


Figure 12 densely sampled temperature scan of susceptibility and heat capacity for D = 4.

4) D =5 Results

In five dimensions, we effectively expect mean field theory to produce exact results due to the aforementioned reason that the growing number of nearest neighbors per site effectively washes out finite scaling effects. Here, we see that the critical temperature ($k_b T_c$) is very close to 8.6.

For our densely sampled plots, we can see the correct qualitative behavior near the critical point, but unfortunately, a mistake in the selection of the initial sample points means we cannot resolve the entire phase transition correctly.

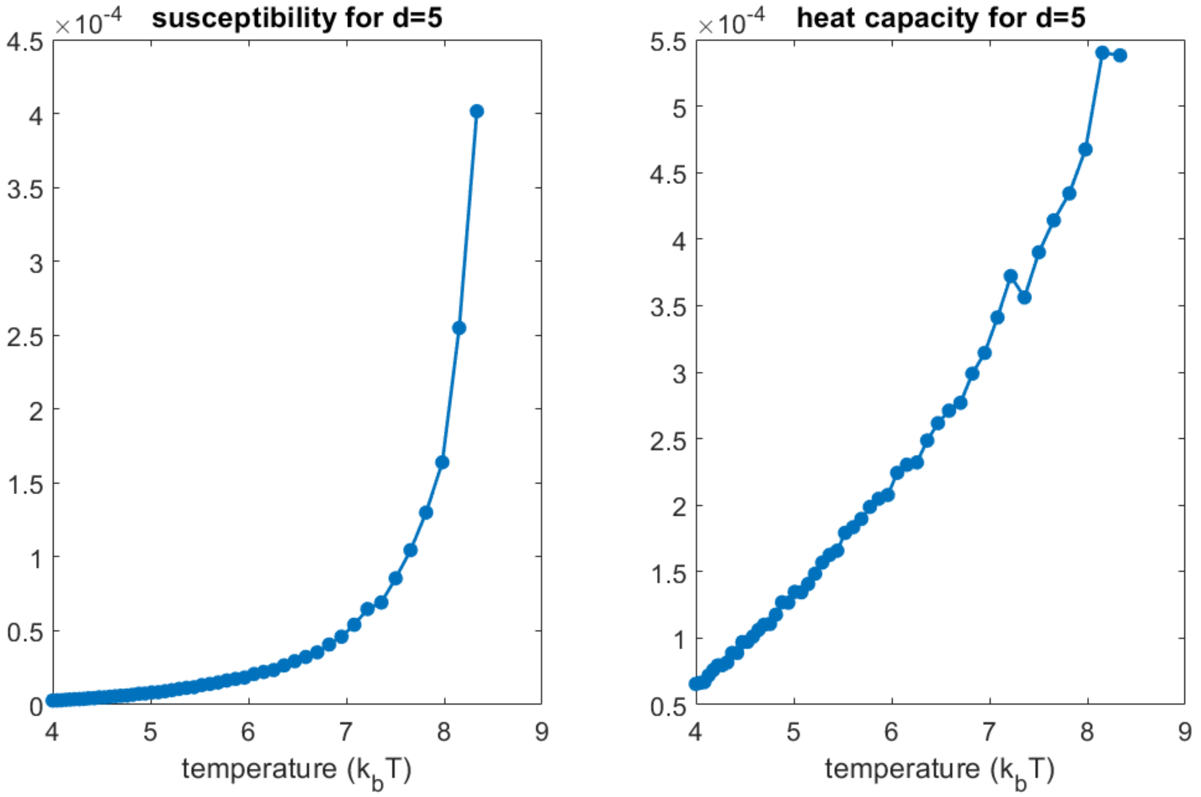


Figure 13 densely sampled temperature scan of susceptibility and heat capacity for D = 5.

Concluding Remarks

We have analyzed the Ising model using mean field theory as well as performed numerical simulations of the Ising model on a regular lattice for various dimensions using the cluster Monte Carlo technique proposed by Uli Wolff. Our numerical results show good agreement in finding the approximate location of the critical temperature in all dimensions. However, our estimation of the critical exponents in all dimensions (with the exception of $d = 2$) are not particularly accurate given two primary issues: insufficient iterations used to sample the lattice and finite size scaling issues.

Solving both of these issues can be accomplished by longer simulations on larger grids. To compensate the time consumption of these, various works have investigated the parallelization of Monte Carlo algorithms, both local and cluster types. For example, for a simple Metropolis Hastings algorithms, one can imagine proposing and analyzing updates in parallel on a ‘checkerboard’ subset of the square lattice. These sites are all effectively decoupled each other if we only consider nearest neighbors.

References

[1] Kardar M 2007 *Statistical Physics of Fields* (Cambridge: Cambridge University Press)

[2] Ibarra-García-Padilla E, Poveda-Cuevas F J and Malanche-Flore C G The Hobbyhorse of Magnetic Systems: The Ising Model

[3] Swendsen R H and Wang J-S 1987 Nonuniversal critical dynamics in Monte Carlo simulations *Phys. Rev. Lett.* **58** 86–8

[4] Wolff U 1989 Comparison between cluster Monte Carlo algorithms in the Ising model *Phys. Lett. B* **228** 379–82

[5] Anon Monte Carlo Simulation 8.1 Introduction

[6] Kramers H A and Wannier G H 1941 Statistics of the Two-Dimensional Ferromagnet. Part I *Phys. Rev.* **60** 252–62

Source Code

All the source code can be found on my github account:

https://github.com/zhaonat/cluster_monte_carlo

For completeness, we will insert samples of the core code snippet that is key to generating all the data, which is the Wolff algorithm:

```
def run_Wolff_epoch(Lattice, N, p):
    '''
    run one wolff epoch
    :param Lattice:
    :param N:
    :param p: 1-exp(-2K);
    :return:
    '''

    change_tracker = np.ones(N);
    visited = np.zeros(N);
    root = [];  # generate random coordinate by sampling from
```

```
uniform random...
    for i in range(len(N)):
        root.append(np.random.randint(0, N[i], 1)[0])
    root = tuple(root);
    visited[root] = 1;
    C = [root];  # denotes cluster coordinates
    F_old = [root];  # old frontier
    change_tracker[root] = -1;
    while (len(F_old) != 0):
        F_new = [];
        for site in F_old:
            site_spin = Lattice[tuple(site)]
            # get neighbors
            NN_list = getNN(site, N, num_NN=1);
            for NN_site in NN_list:  ## if we do the full
search, this is bad, because
                nn = tuple(NN_site)
                if (Lattice[nn] == site_spin and visited[nn] ==
0):
                    if (np.random.rand() < p):
                        F_new.append(nn);
                        visited[nn] = 1;
                        C.append(nn);
                        change_tracker[nn] = -1;
        F_old = F_new;
    Lattice = Lattice * change_tracker;

    return Lattice;
```

<u>Here is our main temperature scan script:</u>

```
import numpy as np
from Wolff_Algorithm.Wolff import *
import matplotlib.pyplot as plt
from variable_calculations.order_measures import *
from metropolis_hastings.metropolis import *
from fitting_module.critical_exponents import fit_power_law
import pickle
import pandas as pd

N = (5,5,5,5,5);
d = str(len(N));
lattice_size_name = '';
for i in range(len(N)):
    lattice_size_name +=str(N[i])+'x';
    if(i == len(N)-1):
        lattice_size_name += str(N[i]);

print('simulation at dimension: '+str(d));
mag_v_temp =list();
E_v_temp = list();

## beta_scans for different dimensions
beta_scan = np.linspace(0.35, 0.54, 50); #2D
#beta_scan = np.linspace(0.2, 0.4, 40); #3D
beta_scan = np.linspace(0.12, 0.25, 50); #4D
# beta_scan = np.linspace(0.1, 0.2, 50);

simulation_data = dict(); #keys will be betas

## ===========LATTICE INITIALIZATION ============##
# we claim that it is better to do the initialization once and
then run the simulation
# as we run to the next temperature, the previous lattice will
actually be not too far from thesteady state lattice of the next

##
np.random.seed(1); #lattice alwasy initializes to the same thing
when we run again
Lattice = 2 * np.random.randint(0, 2, N) - 1;

## ==============================================
K_counter = 0; avg_data = list();
for K in beta_scan:
    #Lattice = 2 * np.random.randint(0, 2, N) - 1; --
    Lattice = np.ones(N);

    lattice_history = list();
    epochs = 5000;
    print(K)
    p = 1 - np.exp(-2 * K);
    magn = list(); ene = list();
    m_avg = 0;
    m_2 = 0;
    E = 0;
    E_2 = 0;
    c_avg = 0;
    #run a wolff simulation
    for t in range(epochs):
        Lattice = run_Wolff_epoch(Lattice, N, p);
        if(t%1000 == 0):
            print(str(K)+', '+str(t));
        magn.append(magnetization(Lattice));
        ene.append(energy(Lattice, 1));  # J = 1
        m_avg += magnetization(Lattice);
        m_2 += magnetization(Lattice) ** 2;
        E += energy(Lattice, 1);  # constant should not be
beta....
        E_2 += energy(Lattice, 1) ** 2;
        c_avg += 1;
        if(t > epochs - 100): #only save the last 100 epochs...
            lattice_history.append(Lattice);
    chi = K*(m_2/c_avg - (m_avg/c_avg)**2);
    cv = K**2*(E_2/c_avg - (E/c_avg)**2)
    avg_data.append([m_avg/c_avg, E/c_avg, chi, cv])

    simulation_data[K_counter] = lattice_history;
    M = np.mean(magn);
    E = np.mean(ene);
    mag_v_temp.append(M);
    E_v_temp.append(E);
    K_counter+=1;

avg_data = np.array(avg_data);

## ============= SAVE LATTICE HISTORY DATA
=====================##
pickle.dump([simulation_data, epochs, beta_scan, N, avg_data],
open(lattice_size_name+'_Ising_'+d+'D_near_crit_Temp_Scan.p',
'wb'));
##
==============================================================

## save data as dataframe, eventually move to the temp scan
temp_scan_data = pd.DataFrame(avg_data, columns =
['magnetization per site', 'E per site', 'chi', 'heat
capacity'])
temp_scan_data['beta'] = beta_scan;
temp_scan_data.to_csv(d+'D_temp_data.csv')

vars = ['Magnetization', 'Energy', 'Susceptibility', 'Heat
Capacity'];
plt.figure(figsize = (20,15))
# fig, axes = plt.subplots(nrows=2, ncols=2)
# fig.tight_layout() # Or equivalently,  "plt.tight_layout()"
for i in range(len(vars)):
    plt.subplot(2,2,i+1);
    plt.plot(1/beta_scan[:], avg_data[:,i], linewidth=3);
    plt.title(vars[i]+ ' '+d+' Grid')
    plt.xlabel('temperature (k_bT)')
    plt.ylabel(vars[i]+' value')
```