2D and 3D Ising Model

Zachary Robertson
Physics 401

***Abstract*—This paper will explore numerical solutions to the 2 and 3 dimensional Ising model in the absence of a magnetic field**

INTRODUCTION)

The Bohr-van Leeuwen theorem was discovered by Niels Bohrn in 1911. It proves that things such as paramagnetism, diamagnetism and ferromagnetism could not arise purely from classical mechanics and thus must be quantum mechanical phenomenon. He showed that there was only one magnetization state once thermal equilibrium was met, and it is identically 0, but of course the existence of magnets proves this false, so we need some model that explains the temperature dependence of these materials' magnetization.

These types of magnetism's arise from the spins of each atoms and how they interact with their neighbors, so to find an exact solution using quantum mechanics we would need to consider the rules of spin angular momentum and many other effects. To simplify the Ferromagnetism problem Wilhelm Lenz invented the Ising model in 1920. This model assumes each lattice point only has one discrete parameter, its spin, that can either be up or down(+1 or -1) and that each atom only interacts with its closest neighbors. The Hamiltonian of an Ising model lattice of N points in a magnetic field H is give by;

$$H = -J\sum_{\langle i,j\rangle}^{N} s_i s_j - \mu H \sum_{i}^{N} s_i = E \quad (1)$$

Here $s_{i,j}$ refers to the spin at sit (i, j), the index $\langle i, j\rangle$ in the first summation refers to a sum over the nearest neighbors, J is the interaction strength between spins, and μ is the permeability(a property of the type of material). The second term comes from the effects of an external magnetic field on a magnetic dipole.

From the Hamiltonian above we could try to work out a mathematical solution, but this is still very complex. Lars Onsager solved the 2D case for a square lattice in the absence of a magnetic field in 1944. The shape of the lattice will effect the number of nearest neighbors and their interaction strength and a square/cubic lattice of the 2D/3D case is the least complex. This solution is still very complicated and I will not delve into the mathematics of it here, but its' biggest take away is that there is a phase transition of the magnetization at the critical temperature ($T = T_c$). A phase transition means that above T_c there is some ordered phase in the magnetism(the spins are aligned) and below T_c there is a disordered phase(spin are anti-aligned).

It is clear we need some other way to solve this problem, the rest of this paper will explore a way of using stochastic processes to solve the 2 and 3 dimensional Ising model on square/cubic lattices in the absence of an external magnetic field.

MEAN FIELD THEORY)

Before we use numerical methods to solve this problem we need to understand how to measure different properties of the system in order to observe the phase transition, the mean field theory gives us an approximate solution to the Ising model. First we must understand that the magnetization of the system is related to the average spin aliment $\langle s_i\rangle$ where the brackets

indicate an average in thermal equilibrium, or it can be thought of as a time average of a system in equilibrium with a heat bath. The heat bath will allow the spins in the system to flip from +1 to -1 and vice versa while equilibrating. For an infinitely large system in an heat bath the spins will all have the same average alignment. This is because each spin interacts only with its neighbors and for an infinitely large lattice all particles will be infinitely far from the boundaries making each spin identical mathematically. The total magnetization will then be given by;

$$M = \sum_{i}^{N} \langle s_i \rangle = N \langle s_i \rangle \tag{2}$$

From statistical mechanics we know the probability of finding a configuration of spins with energy = E_α at temperature T will be $P_\alpha = Ce^{\frac{E_\alpha}{k_b T}}$ where C is a constant that can be found by normalization. For an Ising system with only 1 spin(s_i) the Hamiltonian is;

$$H = -\mu H s_i = E \tag{3}$$

so the probabilities for the two possible states of s_i are;

$$\begin{aligned} P_+ &= Ce^{\frac{+\mu H}{k_b T}} \\ P_- &= Ce^{\frac{-\mu H}{k_b T}} \end{aligned} \tag{4}$$

and normalizing yields;

$$C = \frac{1}{e^{\frac{+\mu H}{k_b T}} + e^{\frac{-\mu H}{k_b T}}} \tag{5}$$

so that;

$$\langle s_i \rangle = \sum_{s_i = \pm 1} s_i P_\pm = P_+ - P_- = \tanh(\frac{\mu H}{k_b T}) \tag{6}$$

This is the exact result for a single spin in a magnetic field, but for our approximate solution we will make the assumption that the interaction of a spin s_i with its neighbors is equivalent to an effective magnetic field acting as the spin s_i.

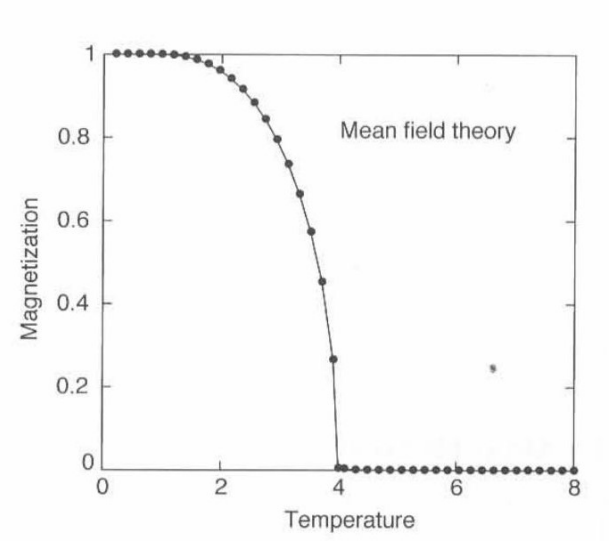


Fig. 1: From [1]

The Hamiltonian can be rewritten as;

$$E = -(J \sum_{\angle i,j\rangle}^{N} s_j) s_i - \mu H s_i \tag{7}$$

This means that we can write the term involving J in the form of $\mu H_{eff} = J \sum s_j$, and making the assumption that all spin variables s_j can be replaced with their thermal averages since all spins have the same average alignment, gives us;

$$H_{eff} = \frac{J}{\mu} \sum \langle s \rangle = \frac{zJ}{\mu} \langle s \rangle \tag{8}$$

where z is the number of nearest neighbors, combing this with the above equation for a single spin gives;

$$\langle s \rangle = \tanh(\frac{zJ\langle s \rangle}{k_b T}) \tag{9}$$

This is an implicit equation for $\langle s \rangle$ which cannot be solved analytically so numerical solutions are required to solve this which I will not delve into in this paper. The textbook gives a graph of magnetization versus temperature for this relationship which looks like;

This solution tells us that there is a ferromagnetic ordered phase at $T < T_c$, and a paramagnetic disordered phase at $T > T_c$. In the limit that $T \to T_c$ we see that $\langle s \rangle \sim (T_c - T)^\beta$ where β is the critical exponent.

MONTE CARLO METHOD)

To numerically solve the Ising model we will use the Monte Carlo Method. In general we will consider a spin field in thermal contact with a heat bath at temperature T, according to statistical mechanics the heat bath will exchange energy with the spin field by flipping the spins and causing the system to change microstates. The observable values will then depend on the probability of finding the system in its different microstates. The Monte Carlo method uses stochastic processes to simulate the interaction between the heat bath and the spin field. In general our code will pick 1 spin at random from the spin field and based on the energy and temperature it will flip the spin. The rules for flipping a spin are as follows;

- dE is the energy of the randomly chosen spin given by equation (1)
- **If** $dE < 0$ the spin is flipped
- **If** a random number between 0 and 1 $< \exp(-dE/k_bT)$ the spin is also flipped

This particular process is called the *Metropolis algorithm*. We repeat this algorithm a large number of times so that each spin is given the chance to flip multiple times. This method gives us the correct solution because the dE criteria will put the spin field in the lowest possible energy state, which is the ferromagnetic phase will all spins aligned. But of course we need some way to transition to states of higher energy based on the temperature of the heat bath, and from statistical mechanics we know that Boltzmann factor $\exp(-dE/k_bT)$ represents the probability of these transitions for $dE > 0$. We can see that at low T the Boltzmann factor is very small so the system will still be ferromagnetic, but as we increase T the Boltzmann factor gets closer and closer to 1 and thus the probability of a spin flip is increase greatly.

From the above discussion in the introduction we know how to calculate the total energy and magnetization of a spin field, and with the Monte Carlo method we have a way to equilibrate a spin field at a given temperature. All that is left now is to measure different observables to see if we are reproducing the ising model numerically and to probe for the value of T_c. From the fluctuation-dissipation theorem , the variance of the energy is related to the specific heat with the relation;

$$C = \frac{\sigma_E^2}{k_b T^2} \qquad \sigma_E^2 = \langle E^2 \rangle - \langle E \rangle^2 \tag{10}$$

Also from the fluctuation-dissipation theorem;

$$\chi = \frac{\sigma_M^2}{k_b T} \qquad \sigma_M^2 = \langle M^2 \rangle - \langle M \rangle^2 \tag{11}$$

2D ISING MODEL)

We will start by exploring the 2D square lattice Ising model in the absence of an external field using the Monte Carlo method. I decided to use python to implement all of this code. First we begin with a random spin of N particles created by choosing either +1 or -1 at random for each lattice site. This field is then equilibrated using the *Metropolis algorithm*. After the equilibration steps, we continue to loop through the Metropolis algorithm and at each loop we calculate all of the observables. Then the average of these values is taken and the data is exported to csv files. Below is the code I wrote to carry out this process as well as a list of the methods and what they do;

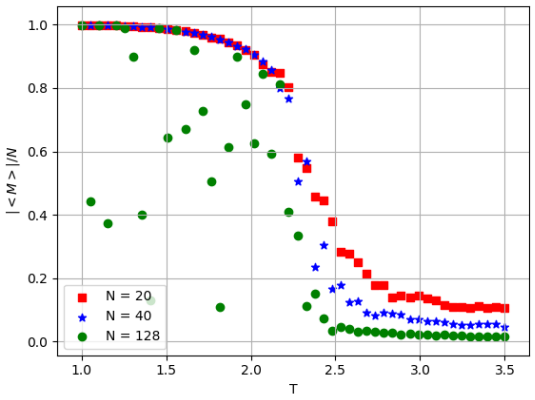
- init(nSites): creates a random spin field of side length nSites. So the number of particles is nSites^2.
- ising_update(field, T): This is one Monte Carlo sweep, there are boundary conditions implemented in the variable nb. This

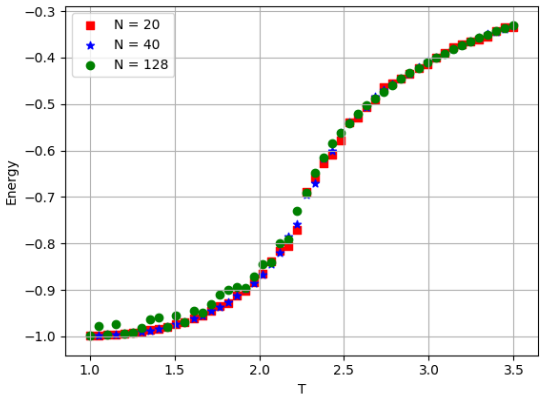
is because at the edges of the lattice these particles will not have 4 neighbors, my boundary conditions effectively wrap the ends of the field onto each other.

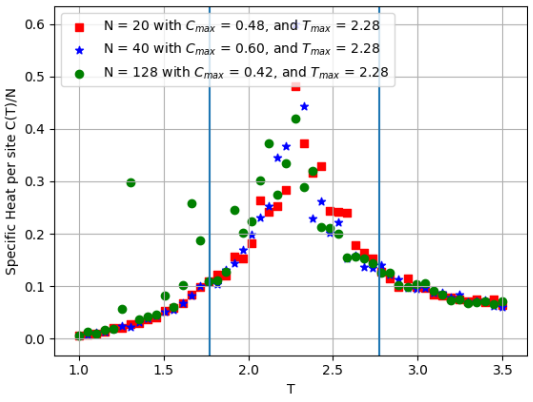
- energy(field) and mag(field) calculate the energy and magnetization of the spin field respectively.
- data() brings a spin field to equilibrium through 2400 Monte Carlo steps, then calculates the observables for 1240 Monte Carlo steps then outputs these values in a csv, it does this for three different lattice sizes and a temperature range from 1 to 3.5 with .05 step size.

```
import numpy as np
from random import randint
import matplotlib.pyplot as plt
from matplotlib import colors
from numpy.random import rand
import pandas as pd

J = 1

def init(nSites):
    return np.random.choice([-1, 1],
size=(nSites, nSites))

def ising_update(field,  T):
    N, M = field.shape
    for _ in range(N):
        for _ in range(M):
            x = np.random.randint(N)
            y = np.random.randint(M)
            spin = field[x,y]
            nb = field[(x+1)%N, y] +\
             field[x, (y+1)%M] +\
                field[(x-1)%N, y] +\
                 field[x, (y-1)%M]
            dE = 2*J*spin*nb
            if dE < 0:
                spin *= -1
            elif rand() < np.exp(-dE/T):
                spin *= -1
            field[x,y] = spin
    return field

def energy(field):
    energy = 0
    N, M = field.shape
    for i in range(N):
        for j in range(M):
            s = field[i, j]
            nb = field[(i+1)%N, j] +\
             field[i,(j+1)%M] +\
              field[(i-1)%N,j] +\
               field[i,(j-1)%M]
            energy += -J*nb*s
    return energy/4.

def mag(field):
    mag = np.sum(field)
    return abs(mag)

def data():
    nSites = [20,40,128]
    for k in range(len(nSites)):
        start = 1.0
        stop = 3.5
        step = .05
        nt = int((stop-start)/step)
        N = nSites[k]
        nEquil = 2400
        nSteps = 1240
        T = np.linspace(start, stop, nt)
        E, M, C, X, LOGM, U =
        np.zeros(nt), np.zeros(nt), np.zeros(nt),
        np.zeros(nt), np.zeros(nt), np.zeros(nt)
        n1, n2=
        1.0/(nSteps*N*N), 1.0/(nSteps*nSteps*N*N)
        for tt in range(nt):
            E1 = M1 = E2 = M2 = M4 = 0
            iT = 1.0/(T[tt]); iT2 = iT*iT
            E1: np.float64
            E2: np.float64
            M1: np.float64
            M2: np.float64
            field = init(N)

            for i in range(nEquil):
                ising_update(field, T[tt])

            for i in range(nSteps):
                ising_update(field, T[tt])
                Ene = energy(field)
                Mag = mag(field)

                E1 = E1 + Ene
                M1 = M1 + Mag
                M2 = M2 + Mag*Mag*n1
                E2 = E2 + Ene*Ene*n1
                M4 = M4 + pow(Mag, 4)*n1

            E[tt] = n1*E1
            M[tt] = n1*M1
```

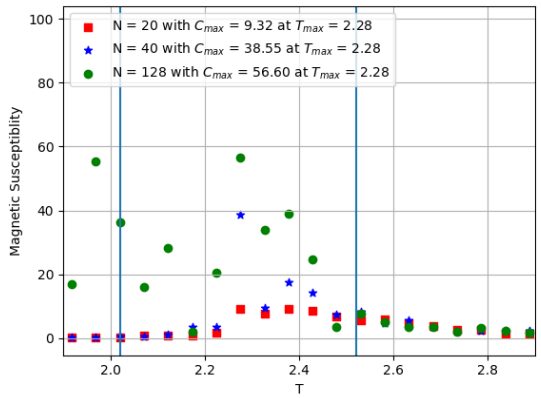
```
            C[tt] = (E2 - n2*E1*E1)*iT2
            X[tt] = (M2 - n2*M1*M1)*iT
            U[tt] = 1 - (M4/3*(M2*M2)*(N*N
            LOGM[tt] = np.log10(n1*M1)
        data = np.column_stack((T,E,M,C,X,
        df = pd.DataFrame(data, columns=['
        if k == 0:
            df.to_csv('20_2D.csv', index=F
        elif k == 1:
            df.to_csv('40_2D.csv', index=F
        else:
            df.to_csv('128_2D.csv', index=
print("All Done!")
```







The csv files were graphed in a program not listed here, as can be seen we clearly have a phase transition at $T_c \approx 2.28$, which is very close to the actual value of $T_c = 2.27$. We can see that this transition becomes more well defined as N increases(N is the side length so we have N^2 spins) but we also have more random perturbations from the expected value as N increases. This effect could be minimized by increasing the number of Monte Carlo sweeps as N increases. Unfortunately nested for loops in python are quite slow as it is a high level language and requires a lot of interpenetration by the computer to run, so if we increase the number of Monte Carlo sweeps the run time will exponentiate and make this process not useful. The run time problem can be solved by just using a different language, specifically a lower level language like C that is better at computations. To help visualize the spin field I have also included three graphs of the spin field for N = 128 after they have reached equilibrium for three different temperature. The first graph is $T = 1.1$ which is less than T_c then a metastable state closer to T_c at $T = 2.1$, and finally a state a $T = 5.0$ well above T_c.



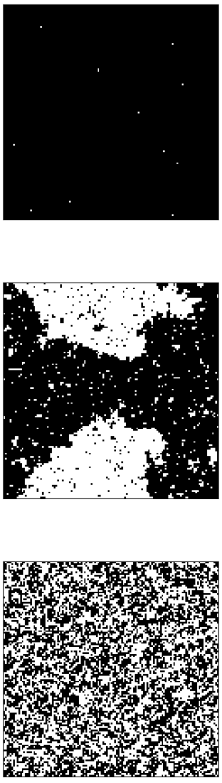
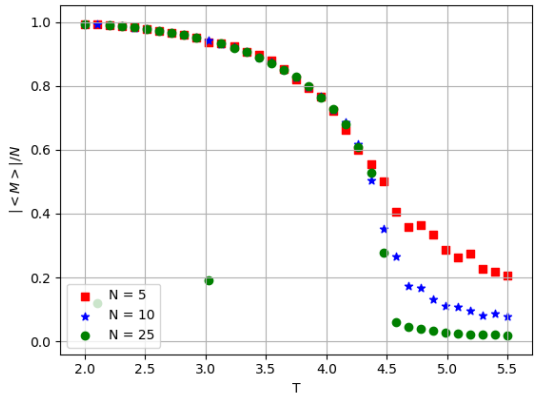


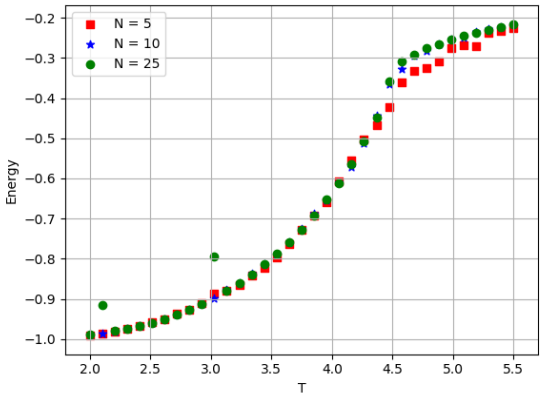
Fig. 2: Spin fields at $T = 1.1$, 2.1, and 5.0 respectively

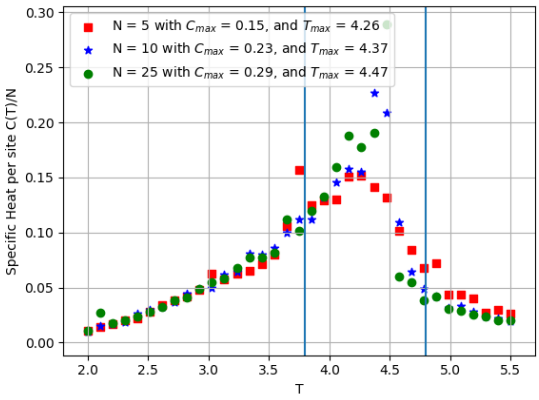
To find the critical exponent produced from the Monte Carlo method we want to plot $\log(M)$ versus $\log(T_c - T)$ for $T < T_c$ and find the value of T_c such that the graph is linear, because $\langle s \rangle \sim (T_c - T)^\beta$ the slope of this line will be β. This graph is shown at the end of the paper and we can see from the first two graphs that $\beta \approx .11$and$.14$ respectively but the third has too many errors for us to get an accurate measurement of β. The actual value for β is .125.

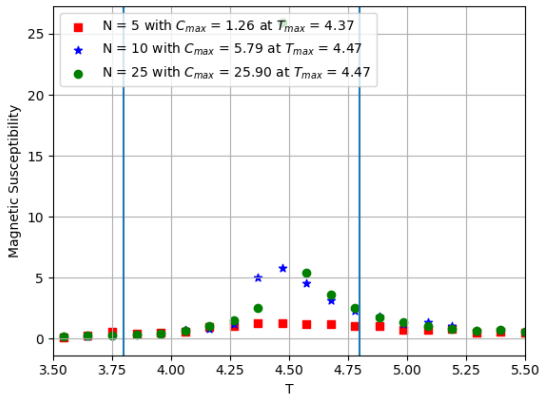
3D Ising Model)

We will now do the same procedure for a 3D square lattice in the absence of a magnetic field. The code to produce these graphs is almost identical to that shown above so I will not list it for brevity's sake. This produced the desired results with a phase transition taking place at $T_c \approx 4.37 - 4.47$. The graph of $\log(M)$ versus $\log(T_c - T)$ gave us β of .13 for all three lattice sizes. For the N = 25 graph the value of β was not accurate until we removed the outliers. This was done with a function that rejects values that are off of the mean by more than .5 standard deviations. Again to visualize the spin field I have included graphs of the field at three different temperatures after equilibrium was reached.









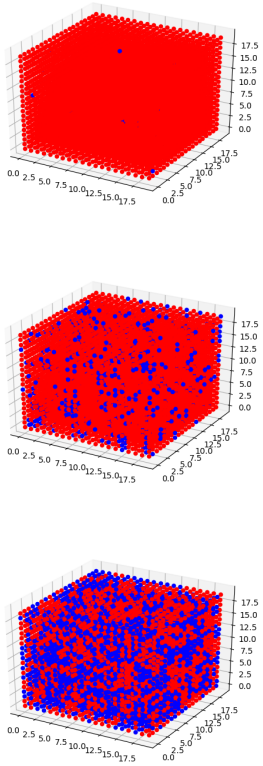
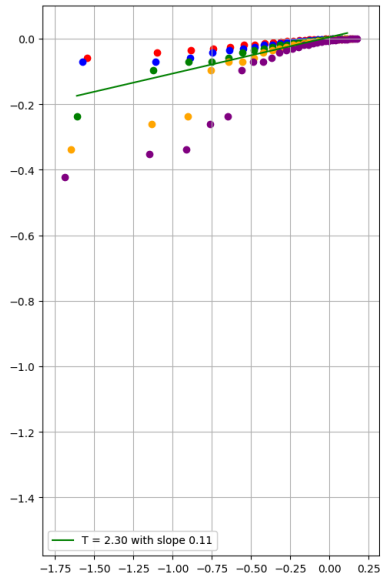
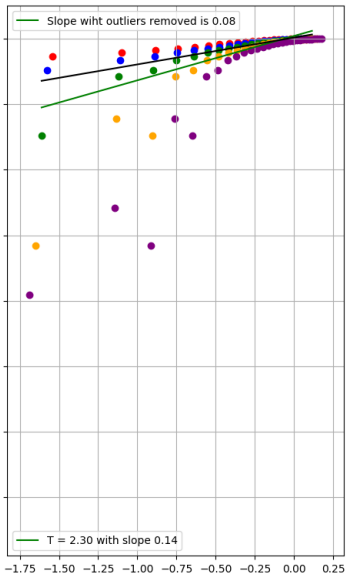
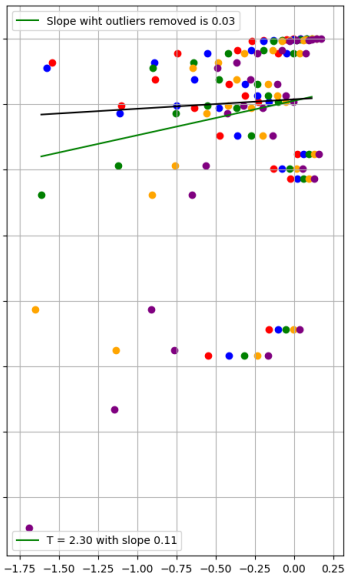
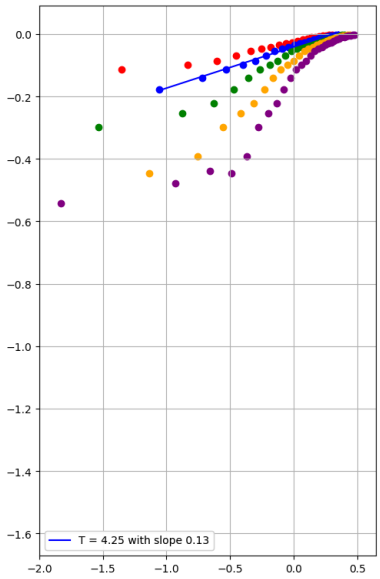


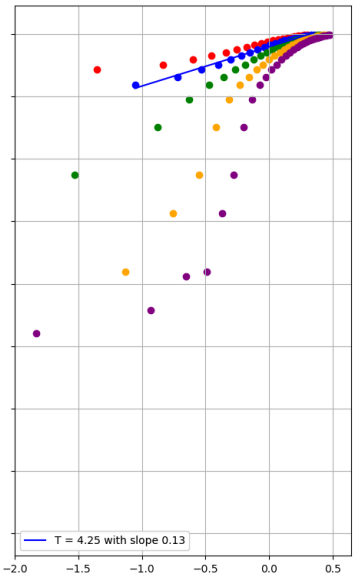
Fig. 3: Spin fields at $T = 2.1$, 4.1, and 8.0 respectively

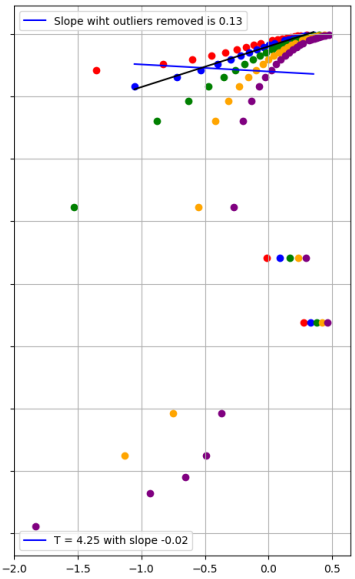

T = 2.30 with slope 0.11


Slope wiht outliers removed is 0.08
T = 2.30 with slope 0.14


Slope wiht outliers removed is 0.03
T = 2.30 with slope 0.11


T = 4.25 with slope 0.13


T = 4.25 with slope 0.13


Slope wiht outliers removed is 0.13
T = 4.25 with slope -0.02

REFERENCES

[1] Giordano, Nicholas J., and Hisao Nakanishi. Computational Physics. 2nd ed., Pearson Prentice Hall, 2006.

[2] Prtkm. "Prtkm/Ising-Monte-Carlo." GitHub, github.com/prtkm/ising-monte-carlo/blob/master/ising-monte-carlo.org.

[3] "Blog Home¶." IsingModel, rajeshrinet.github.io/blog/2014/ising-model/.

[4] Nikos Drakos, and Ross Moore. The Ising Model, 1 Feb. 2002, farside.ph.utexas.edu/teaching/329/lectures/node110.html.