The Ising Model

Brian Hanley

Introduction

The Ising Model is used to simulate phase transitions in two dimensions. The system described is composed of particles in a lattice with a spin component that can be either positive or negative. There is no external magnetic field so particles can more easily flip their spin at any given point in the simulation. At the critical temperature of our system we will see a drop off in net magnetization as particles are allowed to change the direction of their spin more freely than they would at lower temperatures. At this temperature we will also start to see a decrease in the slope of the Energy vs. Temperature curves and a spike in both the specific heat and the magnetic susceptibility which all corresponds to the phase change of our system.

Methods

The Ising Model itself seems simple, but can get complicated very fast. The energy of the system in the Ising Model can be expressed as $E = -J \sum_{<k>}^{N} s_k s_l$ where s_k=±1, <kl> means we sum over the nearest neighbors, N is the number of particles in our system, and J is a coupling constant expressing the strength of interactions between neighboring particles. In the two-dimensional lattice Ising model this means that for each spin we sum over each particles four closest neighbors to get the energy for that particle. For values of J greater than 0 it becomes energetically favorable for neighboring particles to be aligned, resulting in a greater net magnetization of the system at low temperatures. At higher temperatures and thus higher energies, particles have enough energy to overcome this coupling and we get more varied systems. As time goes on we will eventually reach a state of zero net magnetization as the number of spin up and spin down particles reaches equilibrium. To simulate this we use Monte Carlo Methods and the Metropolis Algorithm

Monte Carlo Methods refers to a category of methods used for the statistical analysis of models that use probability distribution functions (PDFs) as a main component. In a Monte Carlo method we can skip over any differential equations governing the system and instead use a high number of random samplings to analyze the system. Each random sampling event is called a Monte Carlo cycle. In this case our Monte Carlo method is going to be the Metropolis Algorithm. In the Metropolis algorithm we use random chance to update our system at a given turn. First we start by selecting our initial value **x** and using the probability that that value **x** will change its value we get a new value **y**. In our system this is a pretty easy step because there are only 2 states **x** can be in, so any change in **x** can only result in 1 **y**, but it is easy to see how for a system with other values the probability function would play a bigger role and we'd have to use an element of randomness to determine which state was chosen. In the second step we have to take in the probability that the system will accept our new value **y**. Here we generate a random

number and compare it to the Acceptance probability of the system. If the new value is accepted then we change **x** to that new value and update the values of our system dependent on **x**.

For our simulation of the Ising model at each Monte Carlo cycle we will choose N number of points randomly where N is the total number of points in the lattice. Repeats are allowed in this sampling so in a given cycle one point can change up to N number of times. Every time we select a particle we calculate what the change in energy resulting from flipping the spin of that particle would be by looking at its four nearest neighbors. We are using periodic boundary conditions so for particles on the edges of our lattice we look at the spin of the particle on the opposite side of the lattice when summing up nearest neighbors. Because we only look at the nearest neighbors there is a finite number of states the 5 particles can be in at the beginning and ending of our spin flipping, and therefore there is also a limited number of possible changes in energy. They are -8J, -4J, 0J, 4J, and 8J where J is the coupling constant of our system. The change in energy can be found as follows:

$$\Delta E = E_2 - E_1 = -J \sum_{<k>}^{4} s_{k2} s_{l2} + J \sum_{<k>}^{4} s_{k1} s_{l1}$$

where s_l is the spin of our central particle. As such it is the only one that actually changes so we can rewrite this as

$$\Delta E = -J \sum_{<k>}^{4} s_{k2} (s_{l2} - s_{l1})$$

Because s_{l2} and s_{l1} will have the opposite value there are only two possible values for $(s_{l2} - s_{l1})$: -2 and 2. This allows us to make one more rewrite:

$$\Delta E = 2J s_{l1} \sum_{<k>}^{4} s_k$$

The magnetic moment of the system is just the sum over all the spins and $\Delta M = 2s_{l2}$. For the acceptance part of our Metropolis algorithm we will compare the initial and final energy states of the system (i.e. ΔE). If ΔE is negative or 0 we automatically accept the new value of the system. If it is positive we put it into our acceptance probability function *w*. Here $w = e^{-\beta \Delta E}$ where β is one over the Boltzmann constant and we can make it 1 by dimensional arguments. With this we can instead say $w = e^{\Delta E/T}$ where T is the temperature. We compare this to a randomly generated number between 0 and 1. If r ≤w we accept the new value and update the system, otherwise we discard the change and move on. Once we have performed our Monte Carlo cycle we update our expectation values by adding in the current value. We store E, E^2, M, M^2, and |M|. Once we have completed all of our Monte Carlo cycles we find our expected values for that temperature. First we normalize our expectation values by dividing by the number of Monte Carlo cycles. Next we find the variance in each of the values by checking the difference of the final value of the squared terms and the final value of the singular terms squared. We divide this by the total number of particles in the system. Finally we want to output our energy per particle, E/N, our

heat capacity, E variance / T^2, our magnetic susceptibility, M variance / T, and our magnetization per particle, |M|/N. Graphing these values as a function of the temperature will show us the state of the system as it approaches net zero magnetization. The complete program that I wrote is "ising.py" and it shows step by step how I executed the above methods.

Simulations and Results

For my simulations I looked at several different aspects of the model. First I wanted to determine what the effect the number of Monte Carlo cycles had on the final expectation values. To that end I created a 2x2 board of spin 1 particles and ran a simulation at T=1.0. I went for 100000 cycles and every 1000 cycles I had it read out the desired expectation values. I then plotted these values versus the cycle number.

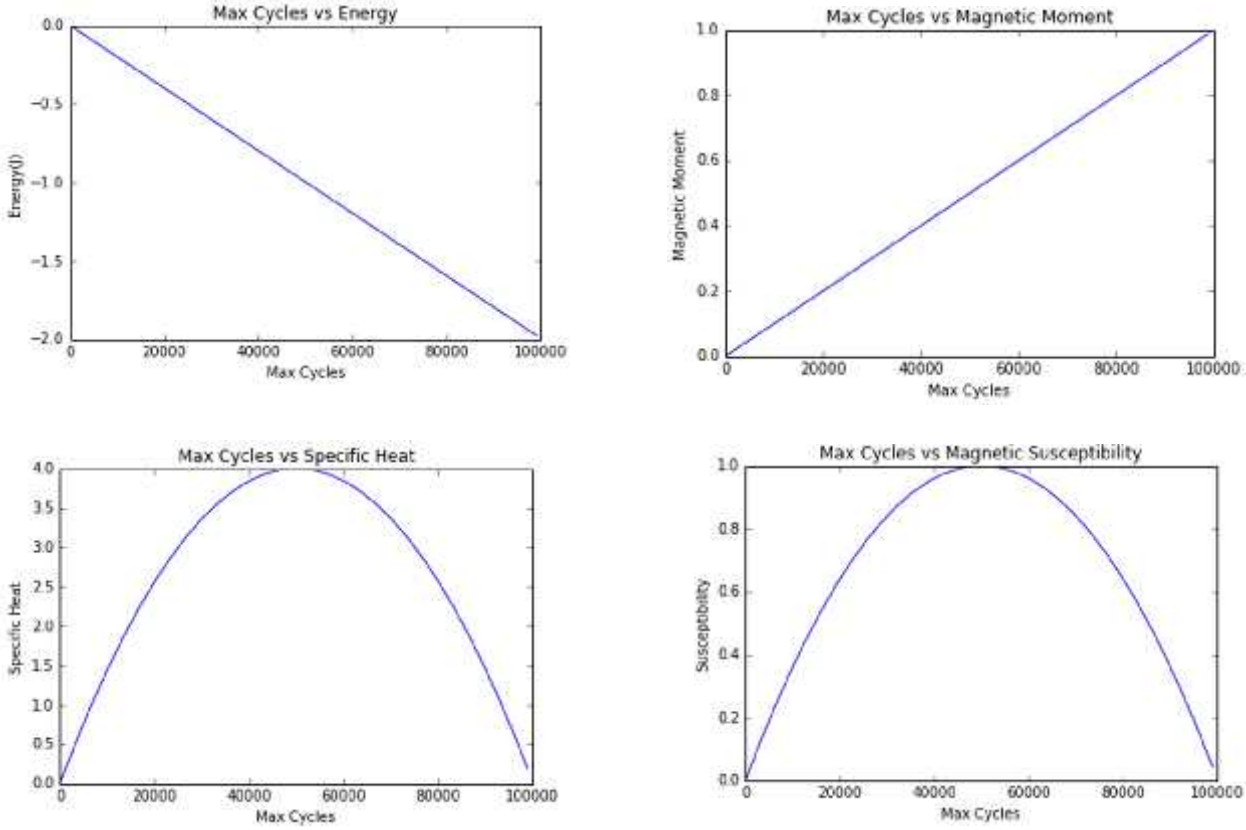

Figure 1 2x2 Matrix with Expectation values as a function of step size

For the 2x2 case we want to have a final equilibrium energy of about -16J. We do not get close to this value in our trial runs, but due to computational limits we can only reasonably do 100000 Monte Carlo cycles.

Next we want to see how the temperature, size, and starting matrix affect the end result. To that end I did 4 runs using the same method as above to show the end values of a 20x20 matrix, both from a starting position of all 1's and a randomly determined matrix, and then both at temperatures 1.0 and 2.4. We plot the end expectation values versus the number of cycles to show our end results.

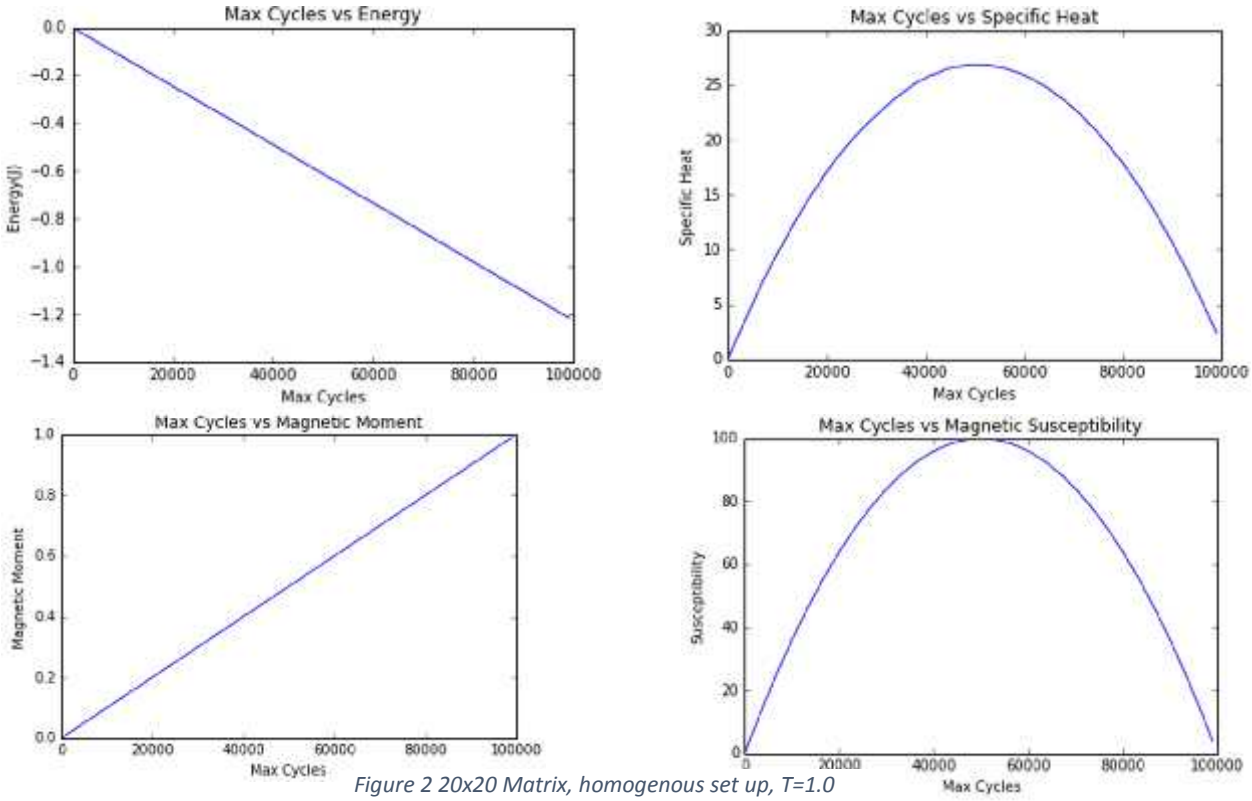

Figure 2 20x20 Matrix, homogenous set up, T=1.0

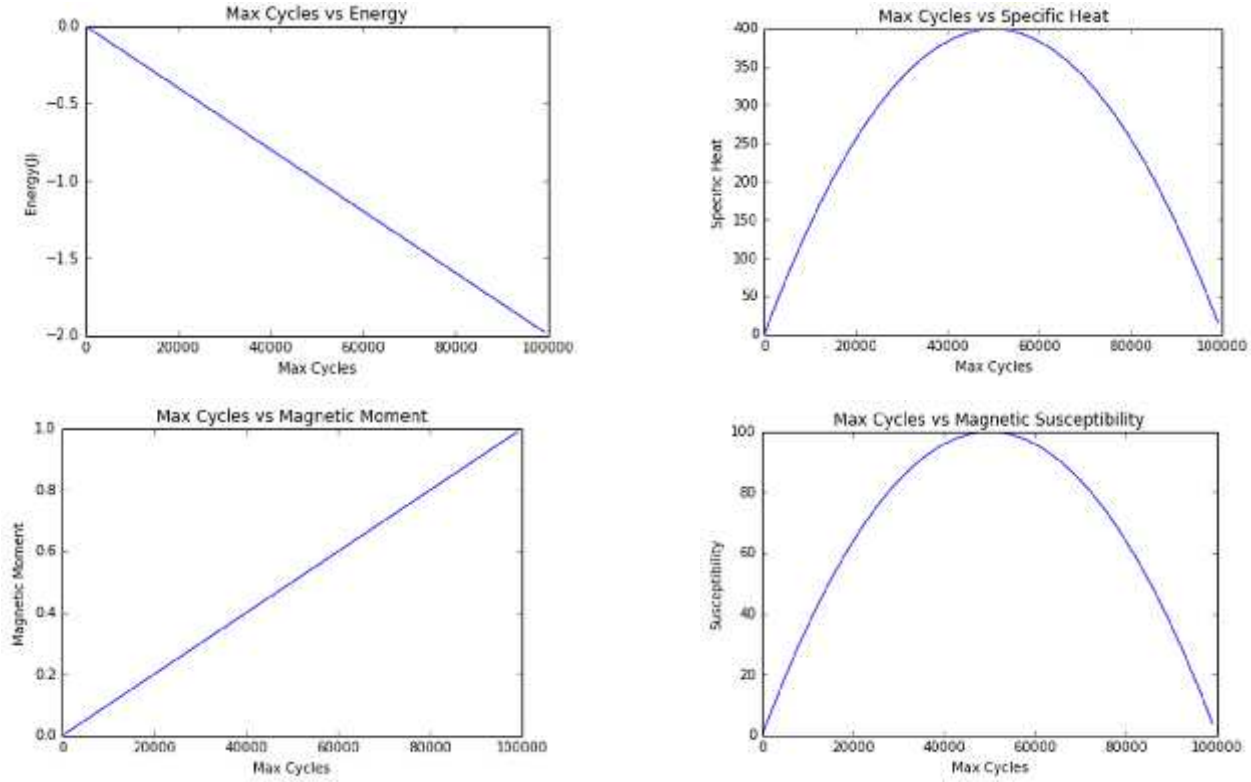

Figure 3 20x20 Matrix, Random set up, T=1.0

As you can see from Figures 2 and 3 there is no significant difference between a randomly

initialized matrix and a homogenous matrix. By comparing figures 2 and 4 we can see that

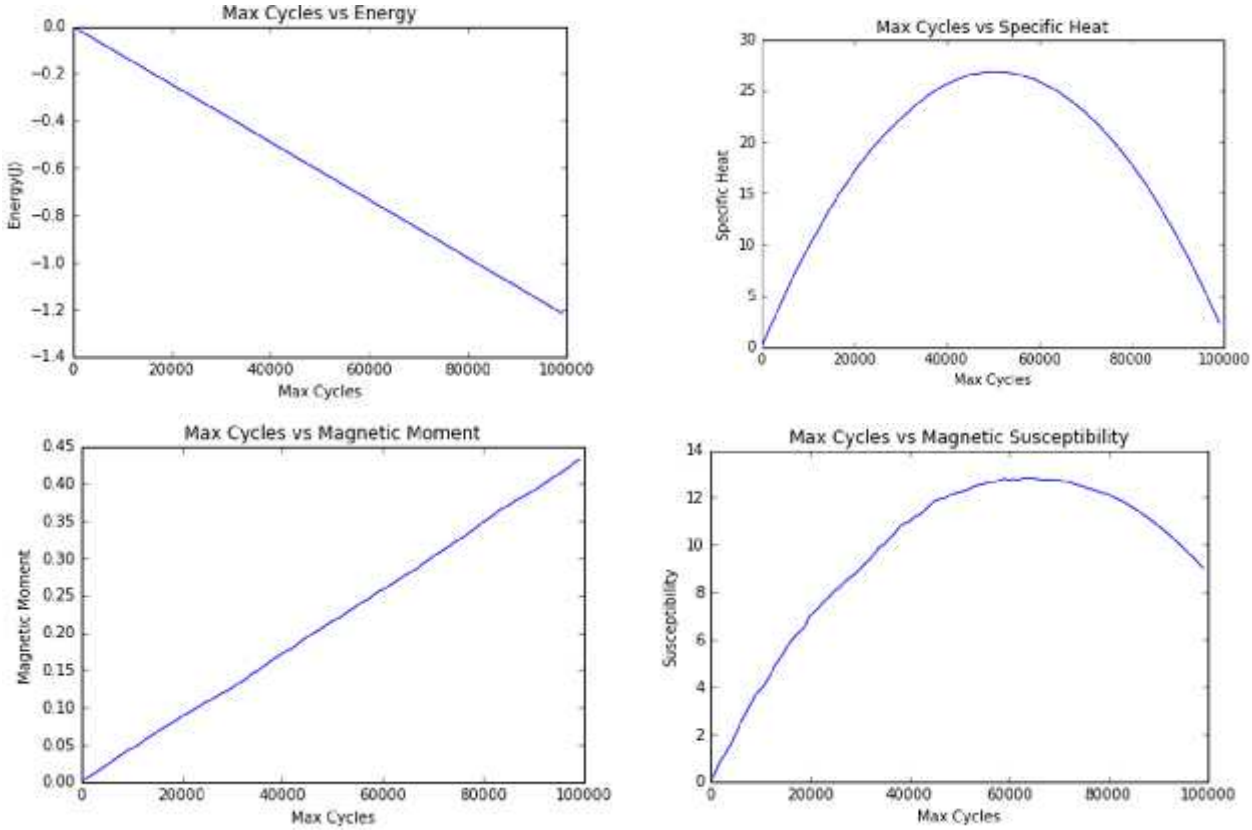

Figure 4 20x20 matrix, homogenous set up, T=2.4

changing the temperature has a significant effect on the expectation values. This makes sense as all the values are dependent on temperature. As we can see, the magnitudes of all values are drastically lowered.

I now want to see how the expectation values change as a function of temperature and how those relations change as we increase the size of the matrix. To that end I built my test to run a Monte Carlo simulation for the matrix at each time step. I only checked values for the temperature between 2.0 and 2.4 with a step size of 0.05. I ran this simulation for matrices of size 20, 40, 60, and 80. I ran each simulation with a maximum cycle number of 100000. For this set

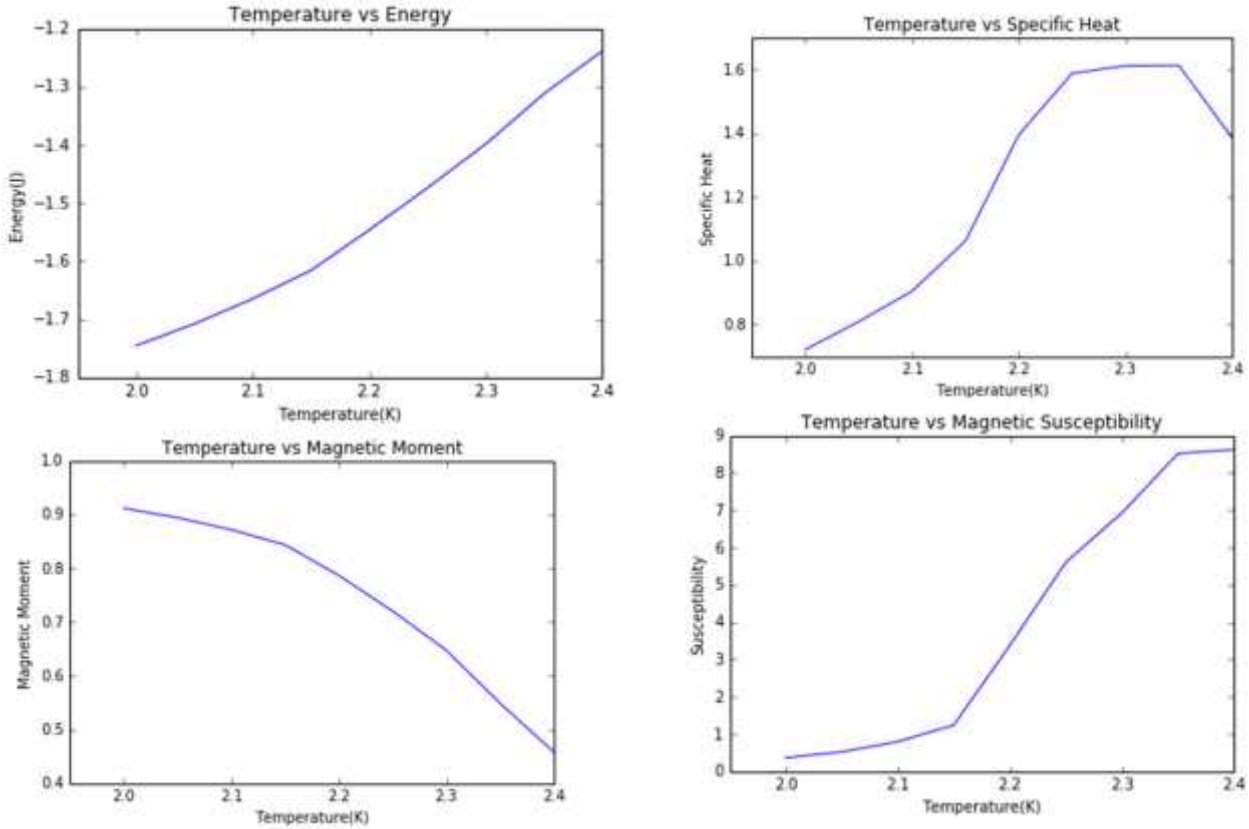

Figure 5 20x20 Matrix, T=2.0-2.4

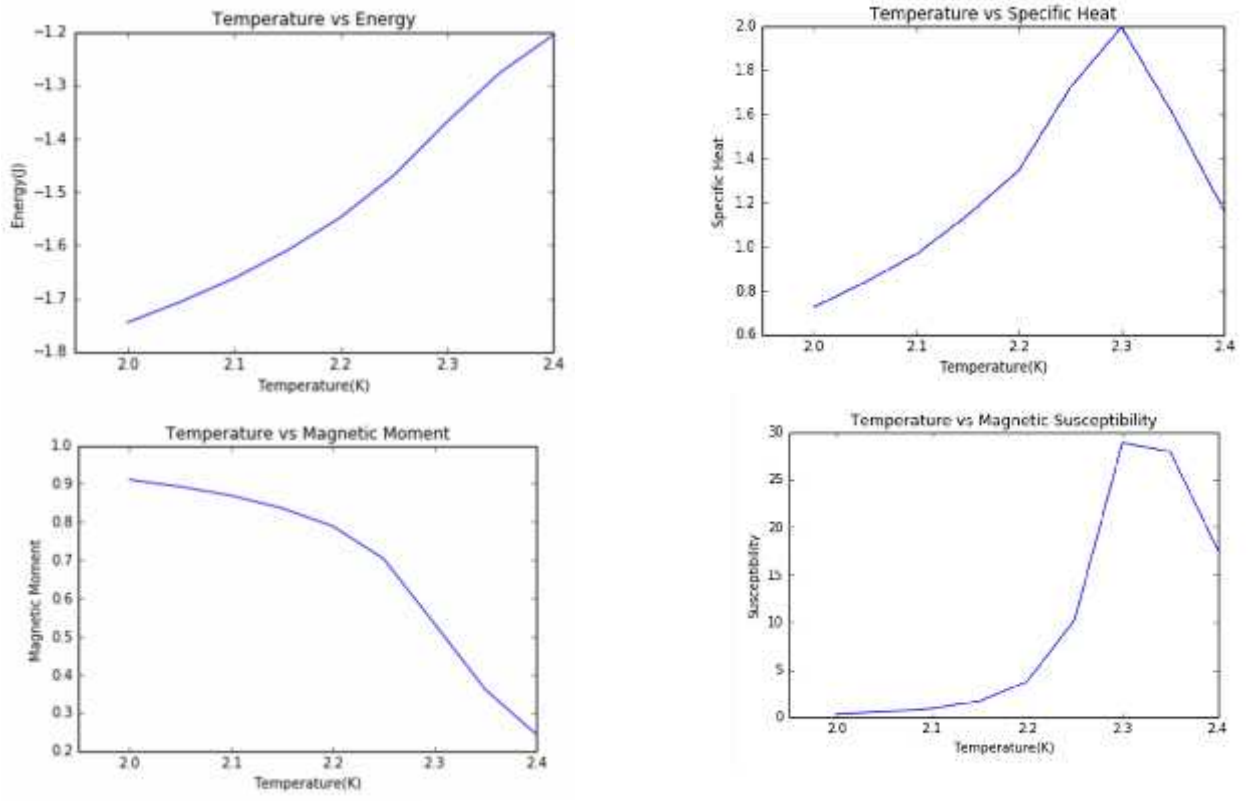

Figure 6 40x40 matrix, T=2.0-2.4

up I could potentially have over 10 billion spin switches with the associated updates to

expectation values on my 60 and 80 runs so I employed several different servers that I had access to through my MSU engineering account. The size 80 run ended up taking close to 11 hours to run on this server. Figures 5-8 show these simulations.

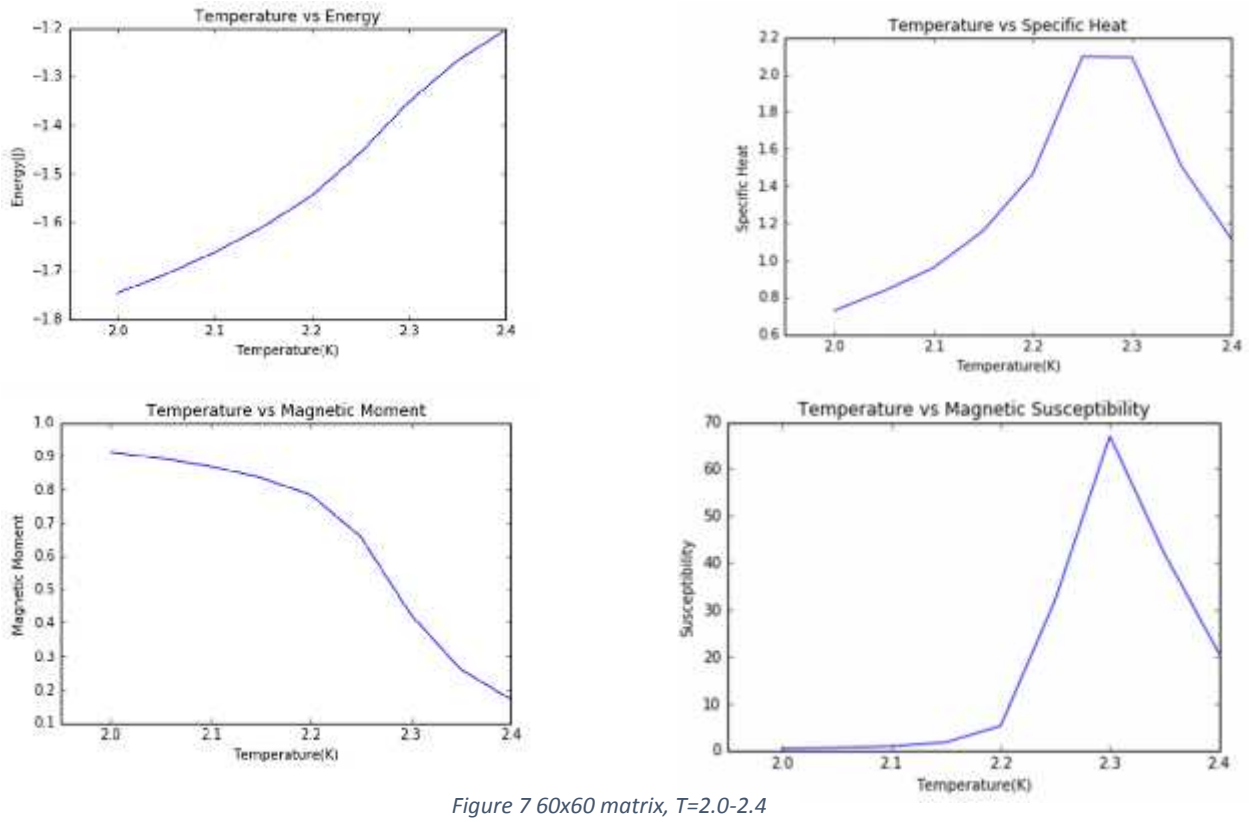

Figure 7 60x60 matrix, T=2.0-2.4

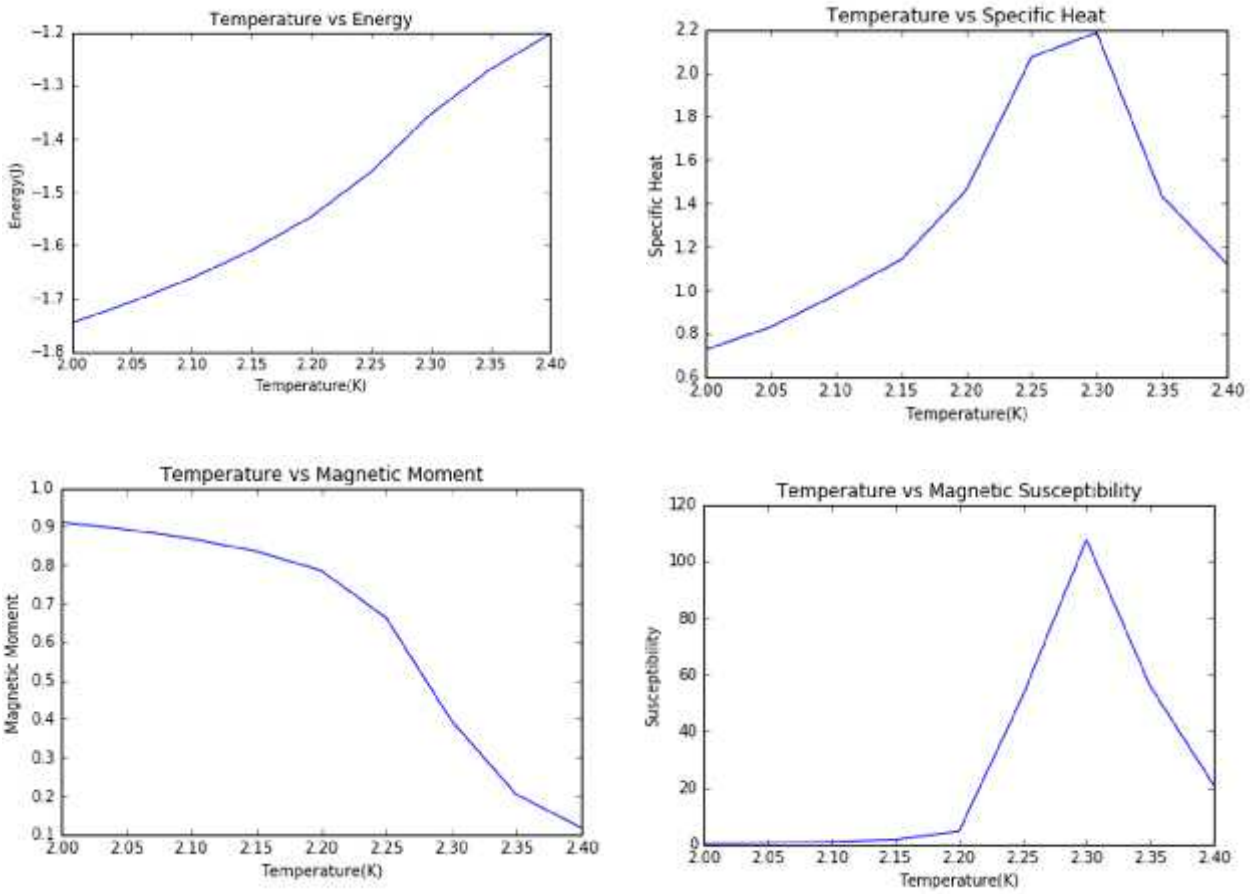

Figure 8 80x80 matrix, T=2.0-2.4

The peaks on the specific heat and magnetic susceptibility graphs correspond to the approximate critical temperature. On the energy and magnetic moment graphs these temperatures correspond to a noticeable change in the second derivative of the graph. As we can see, these temperatures seem to be close to 2.3 in all cases, but are shifted slightly in the positive direction as we increase the matrix size. Furthermore, the larger we make our matrix the clearer a picture we get.

Using the value 2.3 as the most accurate value of Tc we will get from this model, we can find the limit of Tc as L goes to infinity from the equation $T_c(L) - T_c(L = \infty)) = aL^{-1/v}$. Using this I have approximated the critical temperature to be $\sim$2.28. The current accepted value is 2.269. This is a 0.4% difference which is pretty insignificant.

Conclusions and Final Thoughts

The results of my simulations appear to be accurate to the current accepted values of the Ising model. Given more time and better computational resources I believe that my simulation could eventually calculate the exact value for the critical temperature. As it stands I do not have enough time to run the number of trials with the number of Monte Carlo cycles I believe would give an accurate picture. If I ever was given these resources I would probably try running closer to 1 million cycles per temperature step. I would also look at a range of temperatures from 2.25

to 2.35 with a much finer step size. I know the critical temperature has to be in that range. Lastly I'd probably try running my simulation with much larger matrices. I have shown that all three of these can affect the accuracy of simulations as well as the final outcome. Another interesting simulation would be looking at the model when particles interact with neighbors diagonal to them as well, or if they were on a non-square grid. I believe these would have a drastic effect on the final outcome of the simulation.