Ising Model

Nicholas Dagher

April 1, 2015

1 Introduction

In this project we create code to demonstrate a model of ferromagnetism called the Ising model. We initialize a 100 by 100 lattice of dipoles, with each dipole having spin up (+1). Since this lattice is supposed to represent just a fraction of the magnet, we must apply periodic boundary conditions. We then apply the wolf algorithm in order to create a cluster of dipoles that we want to flip "all at once".

2 Method

2.1 The Wolf Algorithm

The wolf algorithm consisted of first picking a random point in the matrix(all elements contain magnetization +1). Then we flip the sign of that random point ("adding it to the cluster"). We need compare this point to its nearest neighbors directly above, below, to the left, and to the right. If the signs of spins are different then we use the probability shown bellow to in order to determine if we add each neighbors to the cluster.

$$Po = 1 - e^{-}J(1 + m_1 m_2)/kT(1)$$

If a random number chosen is less than the probability above then we flipped its sign and looked at its neighbors. The neighbors that had an opposite sign and followed the probability rule above get their sign flipped and If this happens then we repeat these steps until we cant consider any more dipoles.

2.2 Boundary conditions

In order to take into account our periodic boundary conditions we will have the code loop around to the other side of the lattice if a point to be considered is on the edge.

2.3 Calculations

The things that are to be calculated are average Magnetization m which is just the sum of the spins Specific heat C, Susceptibility X, Internal Energy U,γ which comes from the relation bellow and T_c which is the critical temperature, the temperature at which magnetization begins to become chaotic.

$$X = dm/dT (2)$$

$$C = dU/dT (3)$$

$$U = -\sum_{i,j}^{N} -m_i m_j (4)$$

$$m = |T - T_c|^{\gamma} (5)$$

3 The Code

3.1 Outline

For each temperature step the code runs the simulation 10 times, each simulation consists of re-initializing the lattice and running the wolf algorithm a number of times in order to build the cluster. The number of times the wolf algorithm is run depends on temperature. As the temperature rises the probability of equation (1) allowing a neighbor to join the cluster greatly decreases, so we used 10 clusters for a temperature less than 2, 2N clusters for a a temperature between 2 and 3.1, and $2N^2$ for anything higher. This is repeated in time steps of .07 kT, starting at 1.5 kT and ending at 3.5 kT. The temperature, magnetization, susceptibility, specific heat, and internal energy are all calculated and put into lists and graphed.

3.2 functions

In order to make the code easier to debug the code is broken up into functions.

The neighbor() function looks at a given points nearest neighbors and adds them to the cluster and a list entitled que if the spins are not equal and the probability condition is meet. The que is a list of points who have been added to the cluster and whos neighbors we need to consider.

I also have functions written to calcualte the values mentioned earlier. The calcmag() function iterates through the matrix and calculates the magnetization of the matrix. The calcsus() function calculates the susceptibility. The internalenergy(matrix,N) function calculates the internal energy. Calcspecificheat() calculates the specific heat.

To make data analysis easier I created functions to create the plots shown bellow and functions to write the data to a separate file in order to manipulate them without re running the experiment.

The main() function is where all the functions are called and the main outline discussed above is implemented.

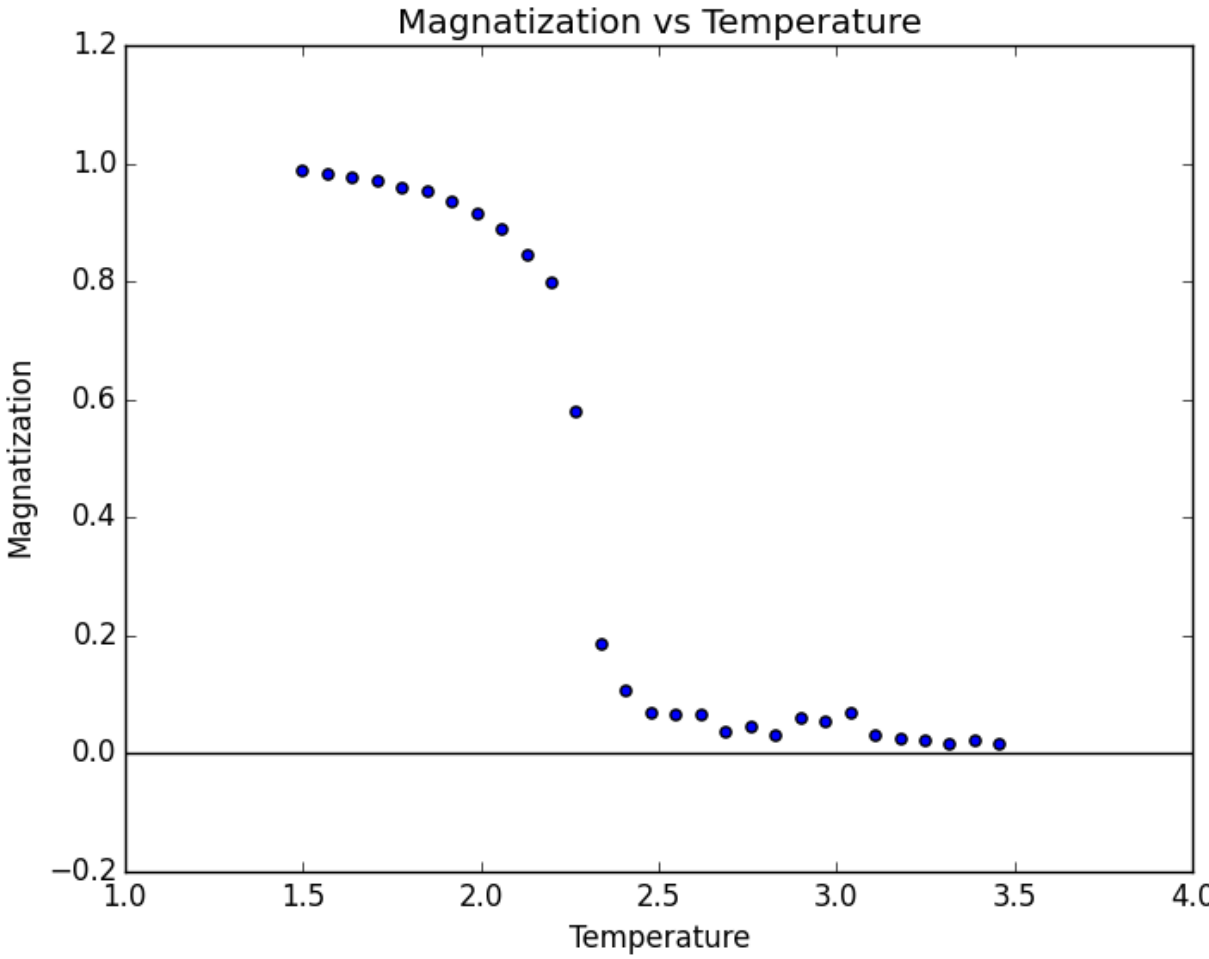

Figure 1: This is a graph of average magnetization vs temperature

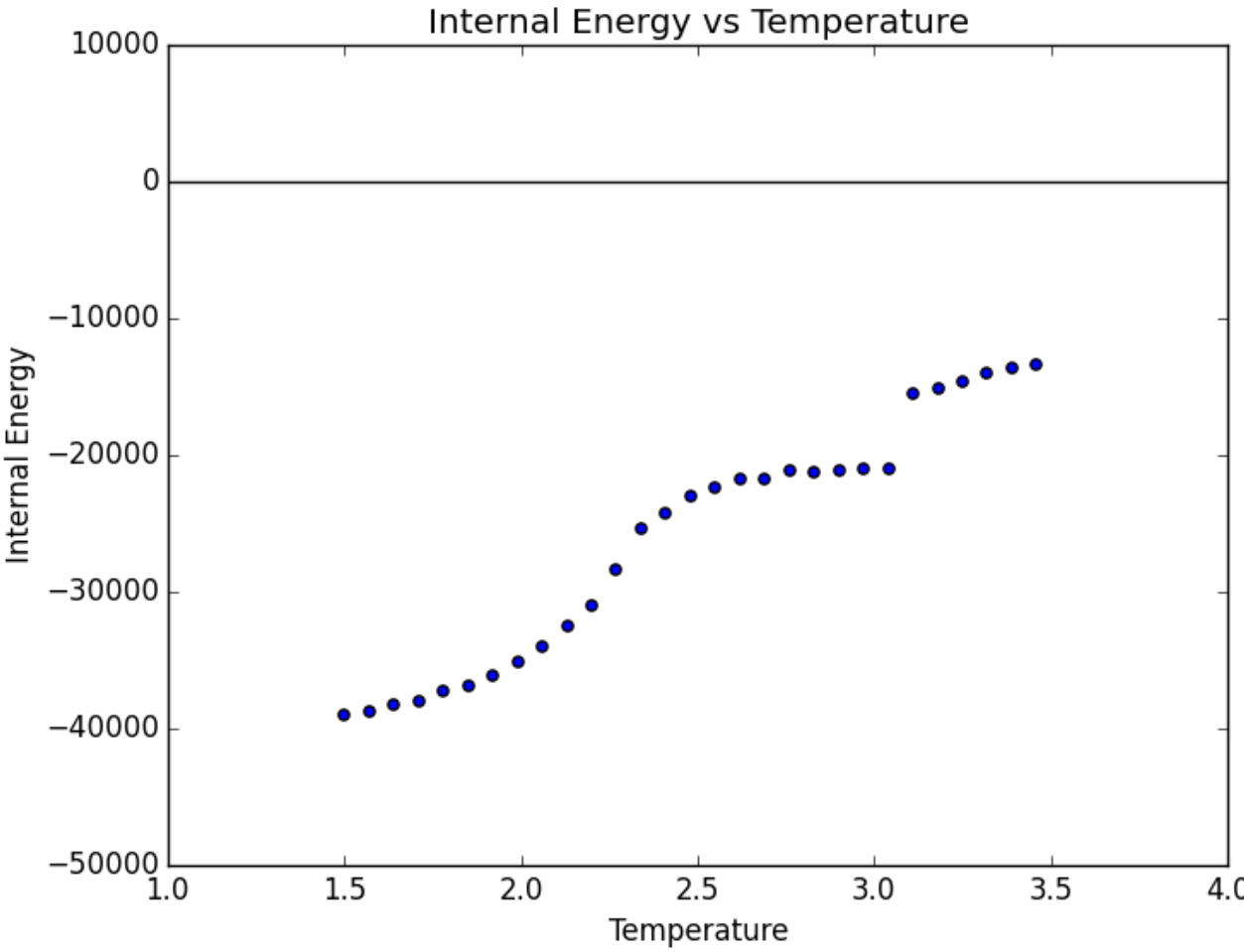

Figure 2: This is a graph of internal energy vs temperature

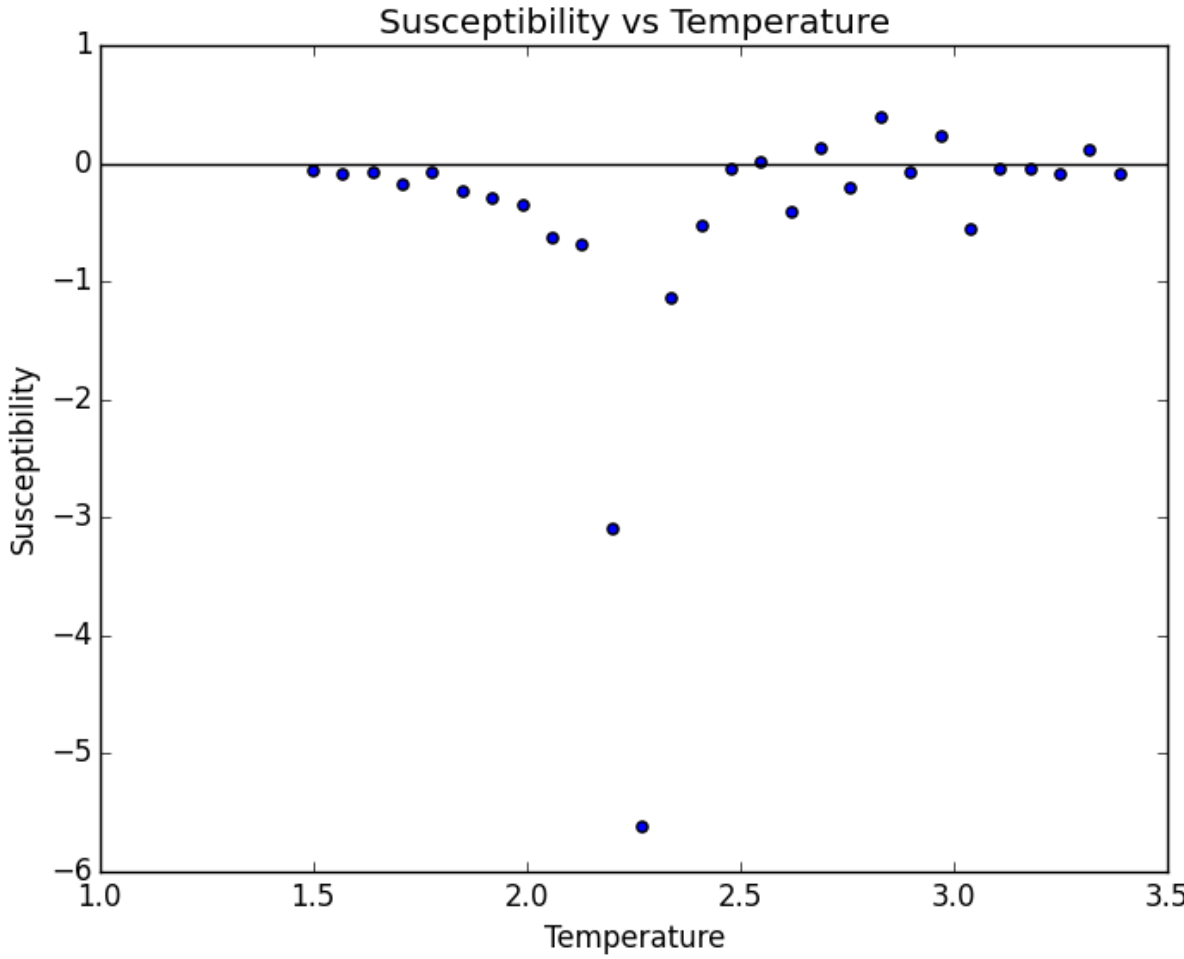

Figure 3: This is a graph of susceptibility vs temperature

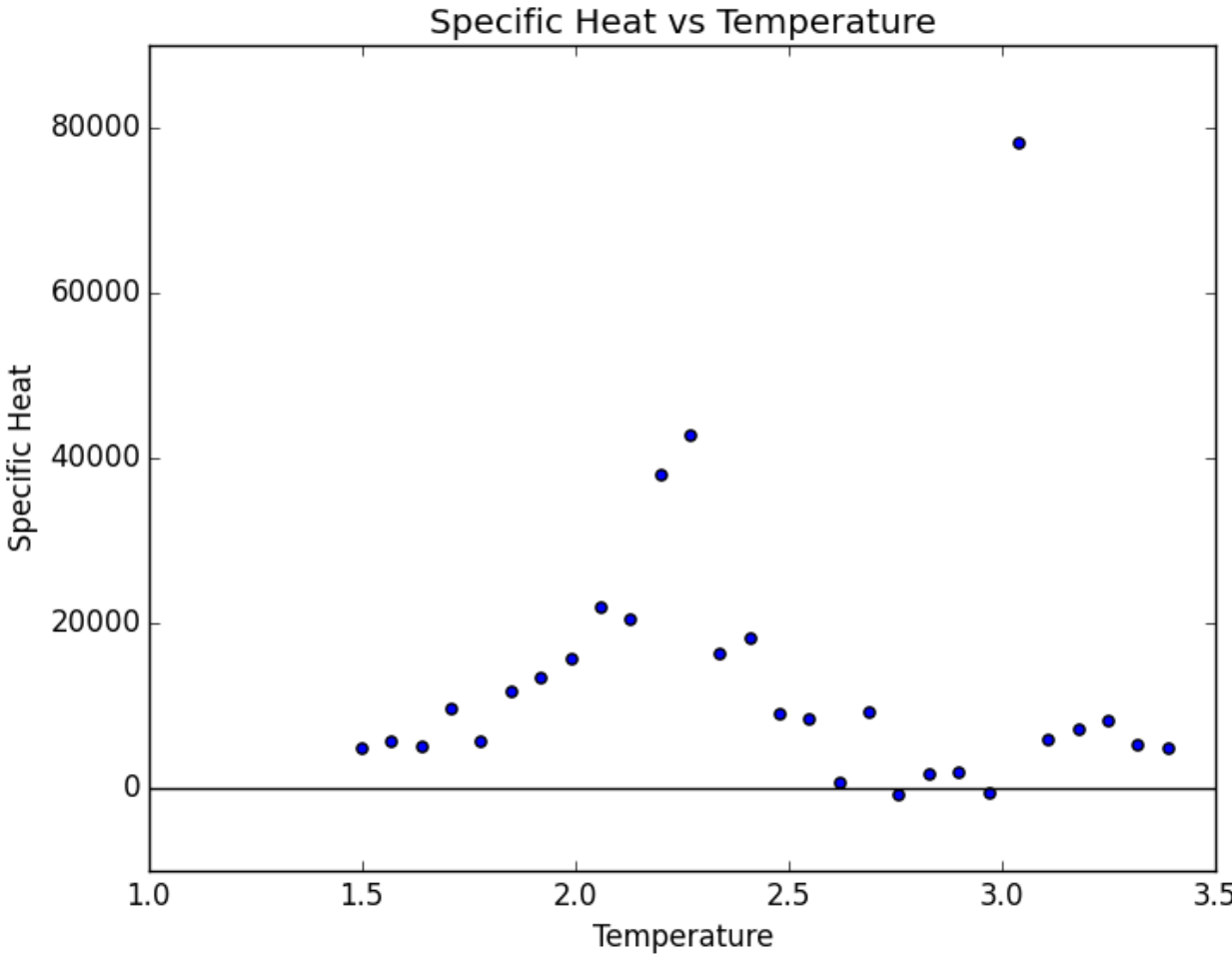

Figure 4: This is a graph of specific heat vs temperature.

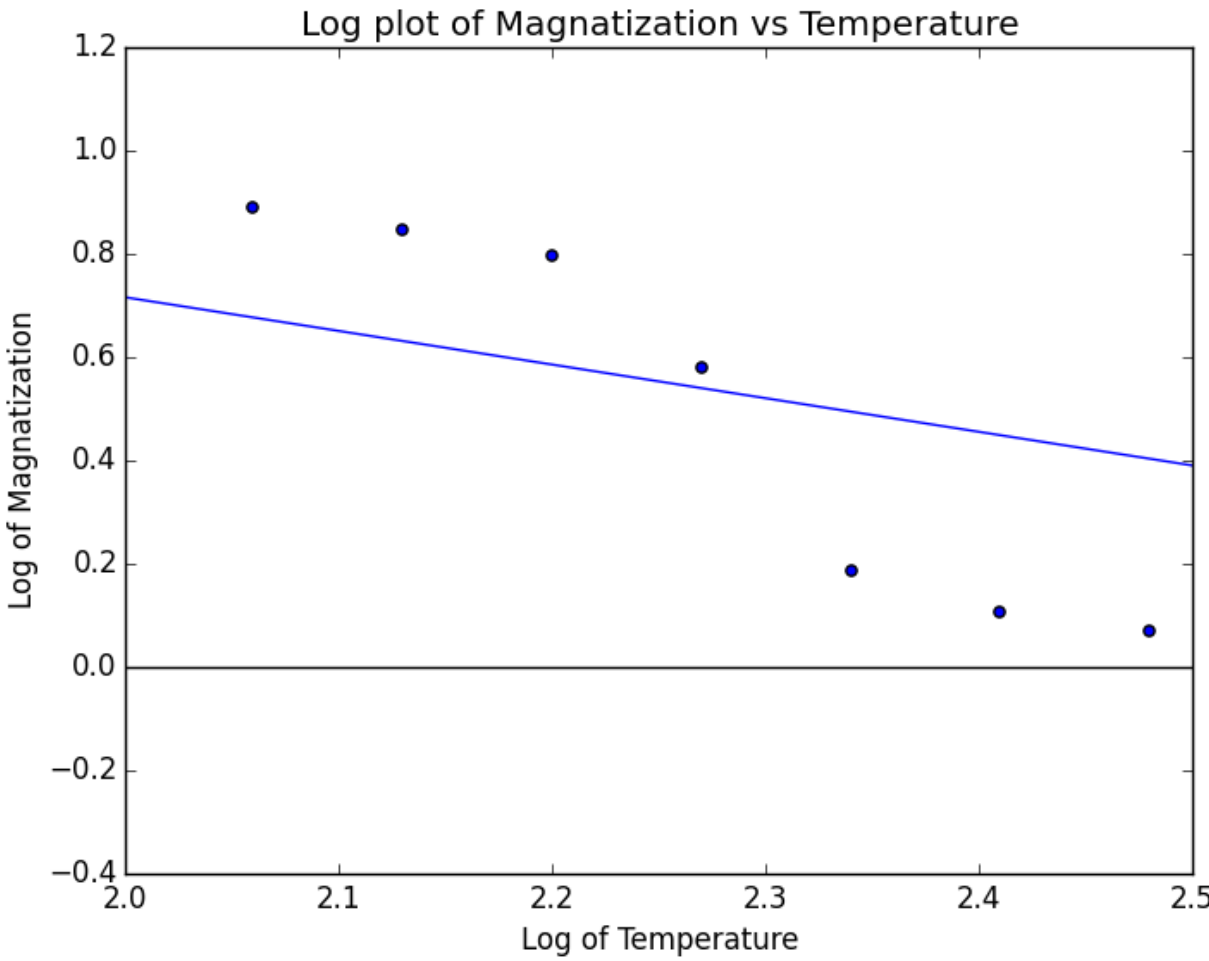

Figure 5: This is a log plot of $ln(m)$ vs $ln(|T - T_c|)$ where the slope of the line is γ

4 Figures

5 Results

The graph of Magnetization vs temperature as shown in figure (1) is exactly what we wanted to see. The average magnetization stays at about 1 until the temperature reaches the the critical temperature of about 2.1 Kt, then the magnetization drastically decreases to near 0 where it remains at high temperatures. This happens because as temperature rises the probability of adding a neighbor to the cluster exponentially decreases which can be seen from equation (1).It should also be noted that the error bars increase as temperature increases.

The graph of internal energy vs temperature as shown in figure (2) also looks like what we expected. It follows the same trend as the Magnetization. Which makes sense because as the temperature increases the average magnetization of each element is so chaotic that the internal energy is less likely to sum up to a large number instead of at low temperatures where almost all the cluster is oriented in the same direction which makes the internal energy of each interaction the same sign so the sum of the energy is my much higher. This can be seen by the high error bars in figure (1) at high temperatures. This high error is the cause of the noise. So taking these error bars into account, figure (2) seems correct.

The graph of susceptibility vs temperature shown in figure(3) is close to what we expected. This is the first derivative of figure (1) so we expect a peak to occur at the critical temperature. The critical temperature is where the point where the spins of each element begin to get the most chaotic, or in other words the temperature at which the rate of change of the magnetization with respect to temperature is the greatest. That point is the point where the slope is the steepest in figure (1) or the peak of figure (3). Therefore our critical temperature from this graph is about 2.26 kt. Past T_c the graph should look like a mirror of the left of T_c this is due to our increase of error as time goes on. Since our magnetization averages to 0 at high temperatures because the spin of the elements is so chaotic, our spread of spin is large!This large spread causes a large standard deviation, so if you take the error of the points to be large to the right of T_c figure (3) remains correct.

Figure (4) shows the graph of specific heat vs temperature. The graph looks similar to figure (3) as expected because since the trends of magnetization and internal energy are so similar then we would expect their 1st derivatives to be similar as well. Like figure (3) the graph peaks at T_c and looks pretty noisy to the right of it. This can be explained just like the noise in figure(3), due to the large spread of positive and negative internal energies.

Figure (5) is a plot showing log of magnatization vs log of the temperature minus the critical temp ($ln(m)$ vs $ln(T - T_c)$). We graphed this plot and varied the T_c we found from figure (3) or (4) until we got a straight line. This T_c is the actual T_c we wanted to find which has a value of 2.29 kT. This occurred at $\gamma = -.6512$, which is the slope of the line in figure (5). This is not what we expected. After trying to make the graph fit the line this was the best we could come up with. We expected to see a value of .125. The reason we are so off could be due to noise in the data and not getting enough data, the code already took hours too run so this could not of been easily taken care of.

6 Conclusion

Overall the code ran well, the only problems that occurred were the noise in the figures and the poor calculation of γ. Although this was largely due to the large error, if we would of ran more samples and a larger matrix we could of decreased the noise. This would of been very time consuming though seeing as for a 100 by 100 lattice with 10 samples the code took many hours to run. Going back, making the lists into numpy arrays would of saved some time. We would say that our code and experiment simulated the Ising model quite accurately.