Parallelising Spin Models on Different Geometries

Filip Sosnowski

July 28, 2021

1 Abstract

2 Introduction

The Ising model is mathematical model used to study the properties of a thermodynamic system. Named after Ernst Ising, it is one of the most important models of statistical mechanics as it helps us to study phase transitions. These phase transitions describe the transition of a system into a different state of matter. Classically, the Ising model deals with the ferromagnetic properties of a system, showing how a system's magnetic properties change with temperature.

In the Ising model the system is comprised of "particles" arranged in a lattice. These particles are then represented by their "spins", which can be either +1 or -1.

3 Procedure

3.1 Serial Ising Square Grid Model

In developing my code I began by writing a simple simulation of the two dimensional square lattice Ising model. For this I created a 50x50 statically allocated square array and randomly assigned each site with either +1 or -1. I initially decided to use statically allocated arrays as the array sizes are relatively small so storing them on the stack should make the code run faster, because it is very likely to always sit in the cache.

Next, I had to modify the metropolis algorithm so that it is more suited to computation. I removed the Boltzmann constant from the step which involves generating a random number, as it was causing my results to be wrong. Another issue I encountered was that the start of the grid was not communicating properly with the end of the grid during the sweeps. This issue appears to have been caused by the % operator, as it operator does not function like the usual mathematical modulo operator, in the sense that $-1\%(m)$ does not return m-1 but rather -1. To fix this issue I wrote my own modulo function, which adds +m to the % operation if the result is negative. Of course this wouldn't work

for any numbers smaller than -m, but in this case it is of no concern as I only needed the function to deal with $-1 mod(m)$.

I ran the simulation 100 times to obtain an average for each observable at each temperature, with the temperature ranging from 0 to 5 in increments of 0.05.

To measure the average magnetisation per site I decided to make my results be give the absolute value of the average magnetisation, as this made the graph of the results easier to read. I also computed the average energy per site, as well as the magnetic susceptibility and specific heat.

3.2 Parallelising the Ising Model

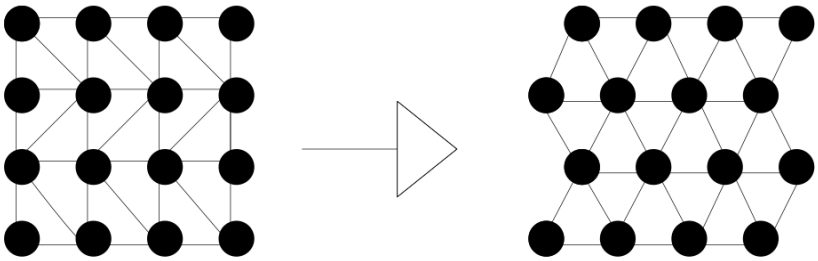
In order to parrallelise the code I decided to employ ghost row exchange. The first thing I did was write a function which ensures that the number of rows is divided between the processors as evenly as possible, ie. so that the number differs by at most one between each processor. This allowed me to redistribute the grid between the processors, so that every process has the most efficient portion of the grid. I chose to divide the lattice along the rows rather than the columns as row-wise reading of matrices is faster.

Next I reworked my metropolis algorithm so that each processor only performs the sweep on its assigned part of the grid. Clearly, as each consecutive iteration of the metropolis algorithm requires information of the state of the closest neighbours, I had to devise a way so that neighbour processes share information about the rows that are adjacent to each process. To do this I used non-blocking MPI calls (Irecv and Isend) to send copies of neighbouring rows to adjacent processes between each Metropolis sweep. This gave each process all the information that was required for the sweeps.

To calculate the observables, I calculated the average energy and magnetisation locally on each process, then used MPI_Reduce to sum the local values up on the root process, which was then divided by the number of processors to give the global average.

Like in the serial case, I ran the simulation 100 times for each temperature so that I can calculate the average value for each observable. The temperature range was once again from 0 to 5 and went up in increments of 0.05. An issue that I encountered was resetting the grid after every simulation so that every process starts with an identical grid. My initial solution was to save the grid from the first simulation, and reset it to the same grid between each simulation. Although not ideal, the results were comparable to the serial case so I left it for the timebeing.

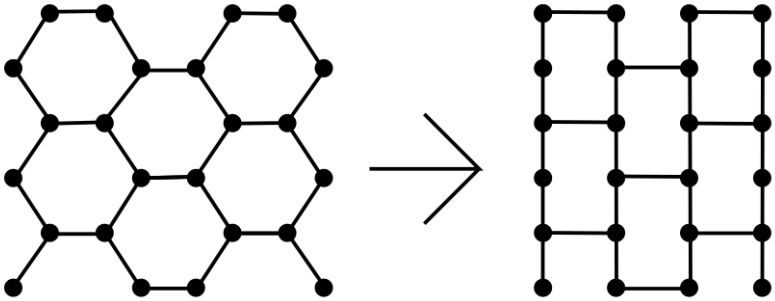
3.3 Triangular Ising Model



To find the nearest neighbours in the triangular Ising model I drew the triangular lattice first then mapped it to a normal square grid. From this I was able to deduce the nearest neigbours. As can be seen, each particle on the lattice has six neighbours, the four same ones as on the 2-D square grid and two additional ones. These two additional neighbours differ for even and odd rows, going forwards for even rows and backwards for odd rows. This gives the following neighbours:

g[i-1][j]
g[i+1][j]
g[i][j-1]
g[i][j+1]
g[i+1][j+(-1)^i]
g[i-1][j+(-1)^i]

3.4 Hexagonal Ising Model



To find the nearest neighbours for each site in the hexagonal lattice I once again drew it and mapped it to a square lattice manually. This allowed me to easily find the nearest neighbours for this kind of lattice, especially seeing as each site has only three nearest neighbours:

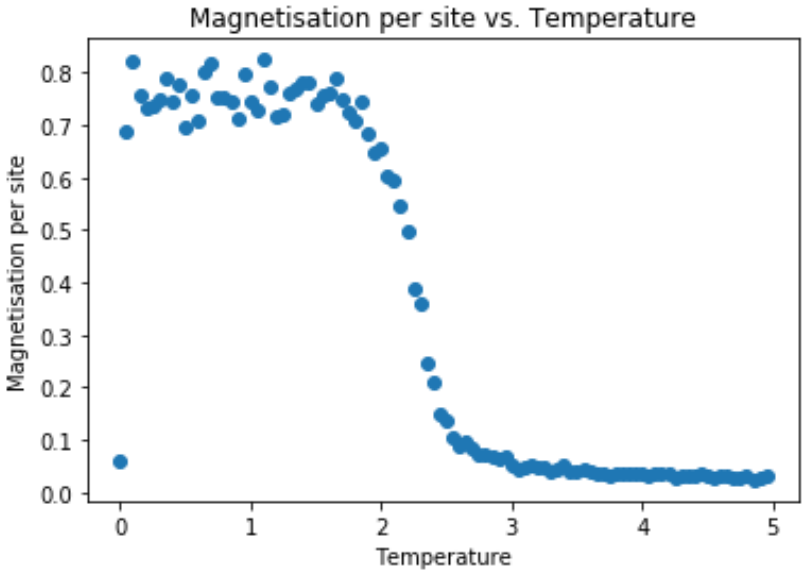
g[i+1][j]
g[i-1][j]
g[i][j+(-1)$^{i+j}$]

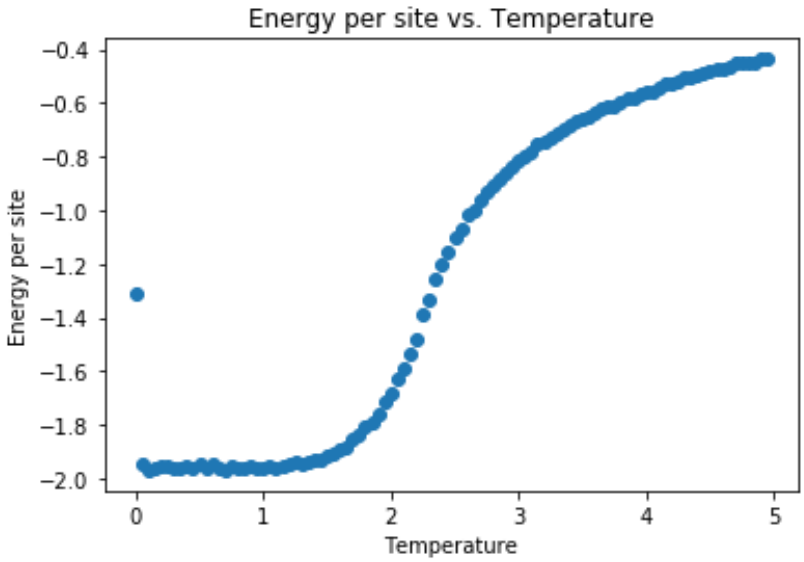
The third nearest neighbour can be deduced from the fact that it alternates between left and right not only for every second element in each row, but also along each column.

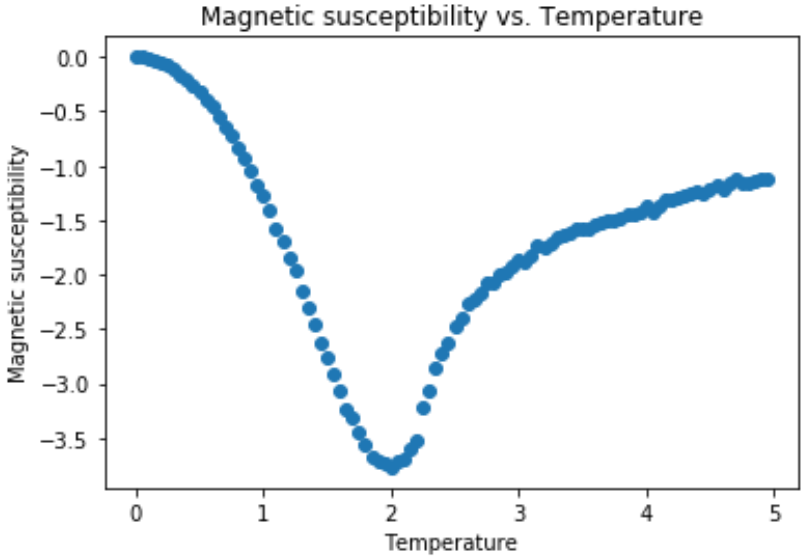
A challenge that this system proposes is having the right number of elements such that the sites along the edge of the grid communicate with the other ends properly, as to mimmick the behaviour of an infinite lattice properly. If the grid was to have its nearest neighbours assigned as above, then I believe that the number of columns would have to be a multiple of 4 and the number of rows would have to be a multiple of 2. As can be seend from the graph above, this allows the edges of the lattice to sync up perfectly with the opposite edge.

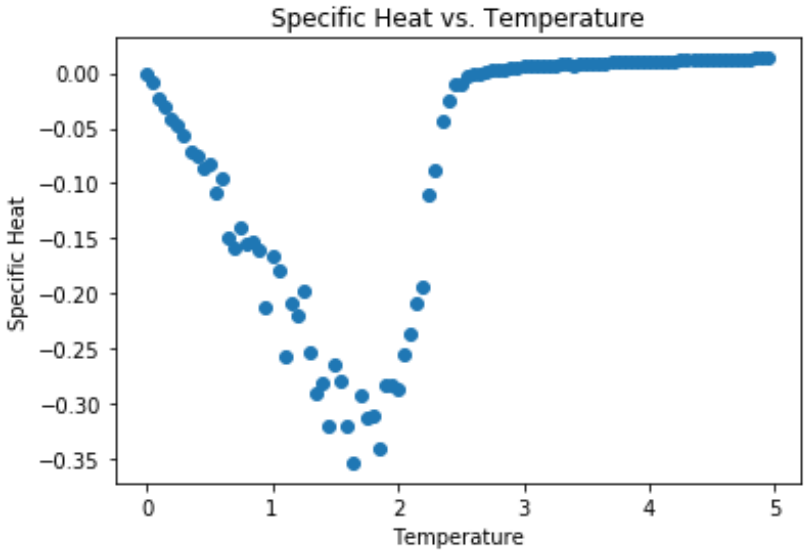
4 Results

4.1 2D Square Grid Sequential Ising Model

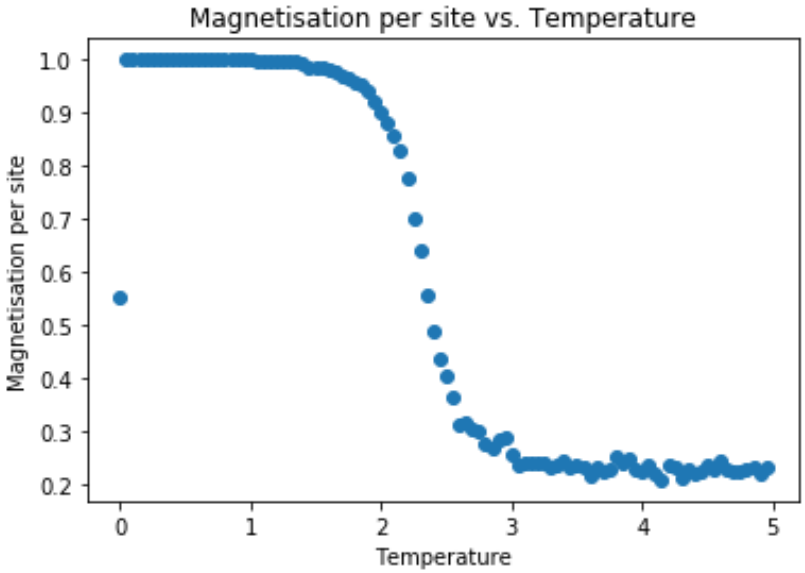


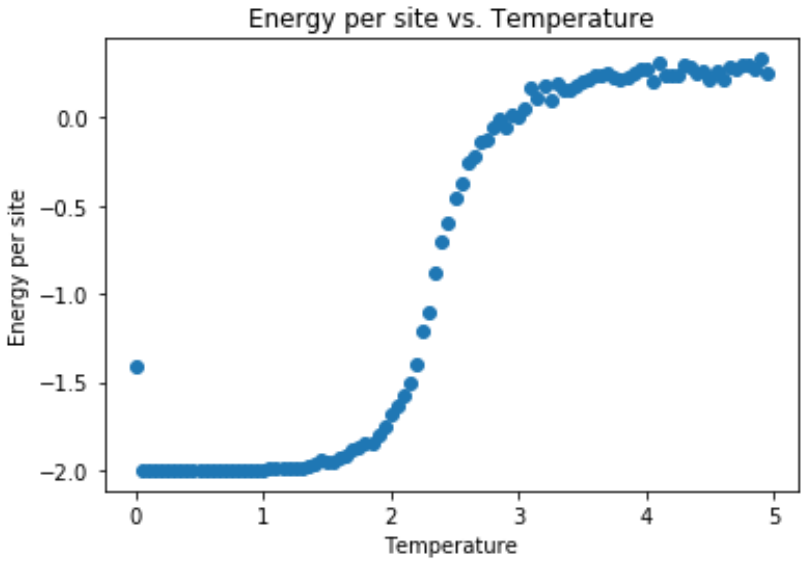

Energy per site vs. Temperature
Energy per site
−0.4
−0.6
−0.8
−1.0
−1.2
−1.4
−1.6
−1.8
−2.0
0
1
2
3
4
5
Temperature

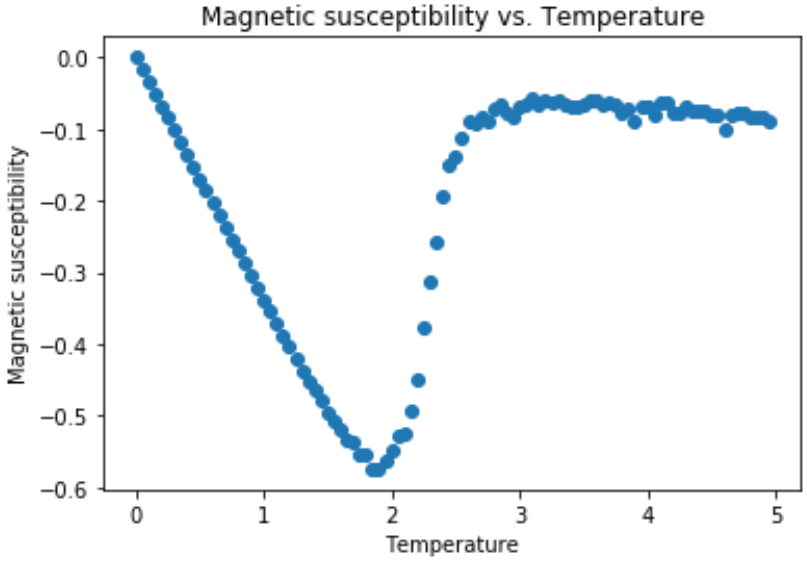

Magnetic susceptibility vs. Temperature
Magnetic susceptibility
0.0
−0.5
−1.0
−1.5
−2.0
−2.5
−3.0
−3.5
0
1
2
3
4
5
Temperature

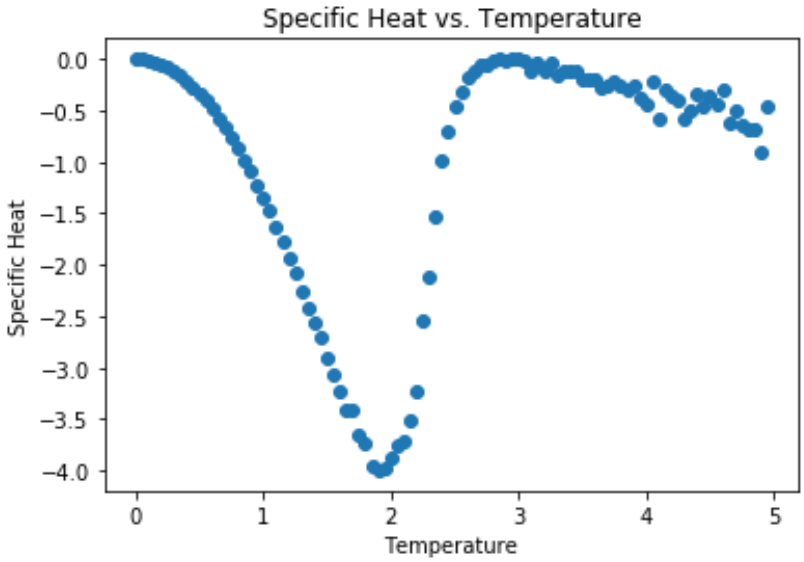


4.2 2D Square Grid Parallel Ising Model

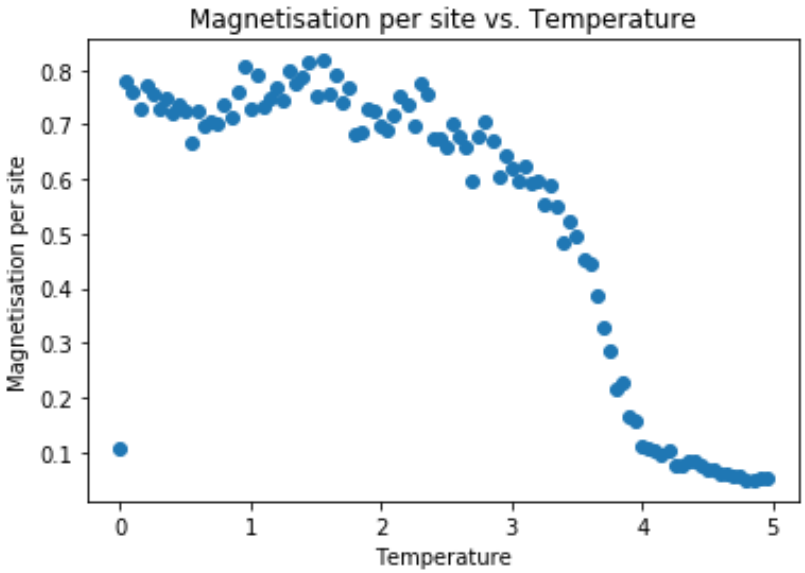


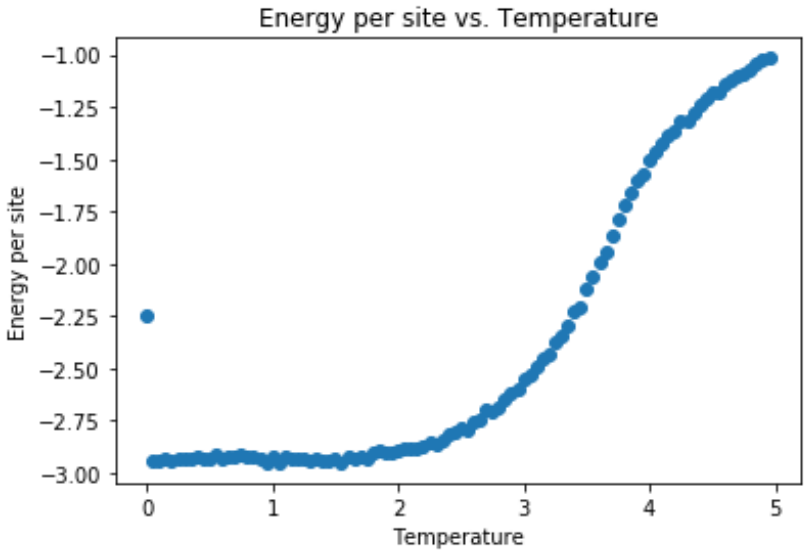

Energy per site vs. Temperature
Energy per site
0.0
−0.5
−1.0
−1.5
−2.0
0
1
2
3
4
5
Temperature

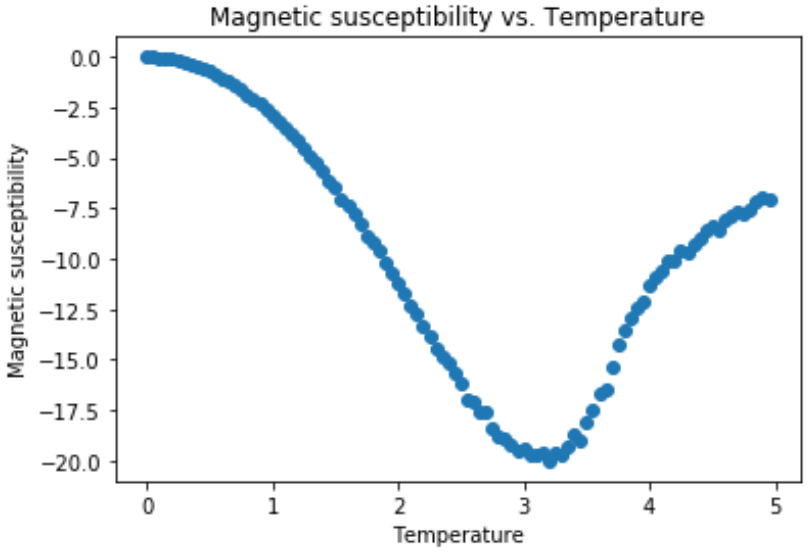

Magnetic susceptibility vs. Temperature
Magnetic susceptibility
0.0
−0.1
−0.2
−0.3
−0.4
−0.5
−0.6
0
1
2
3
4
5
Temperature

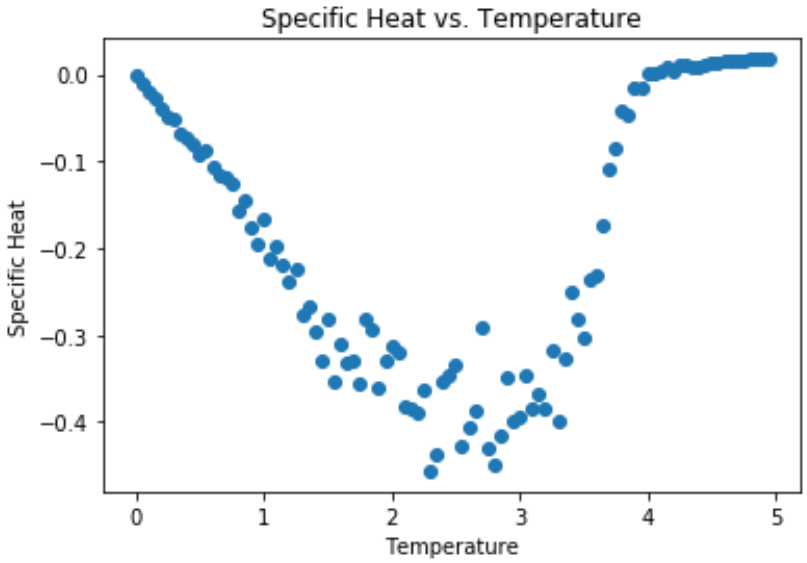


4.3 2D Triangular Grid Sequential Ising Model

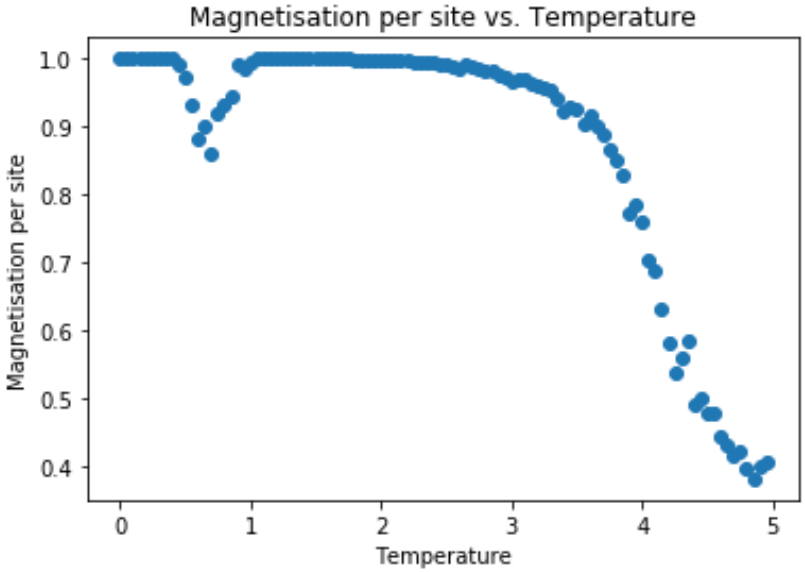


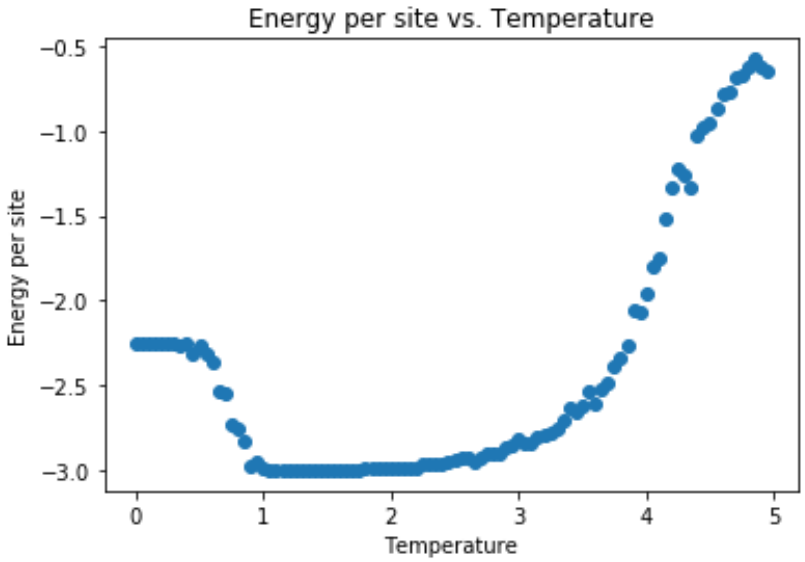

Energy per site vs. Temperature
Energy per site
−1.00
−1.25
−1.50
−1.75
−2.00
−2.25
−2.50
−2.75
−3.00
0
1
2
3
4
5
Temperature

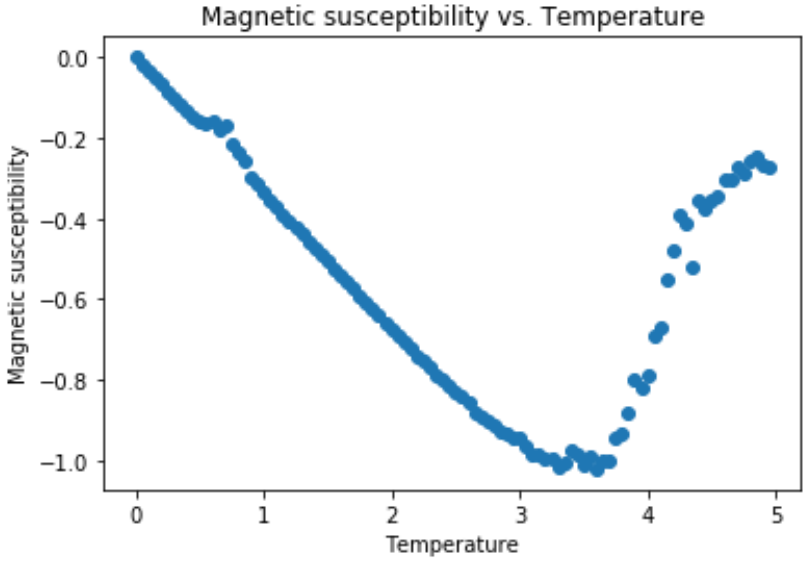

Magnetic susceptibility vs. Temperature
Magnetic susceptibility
0.0
−2.5
−5.0
−7.5
−10.0
−12.5
−15.0
−17.5
−20.0
0
1
2
3
4
5
Temperature

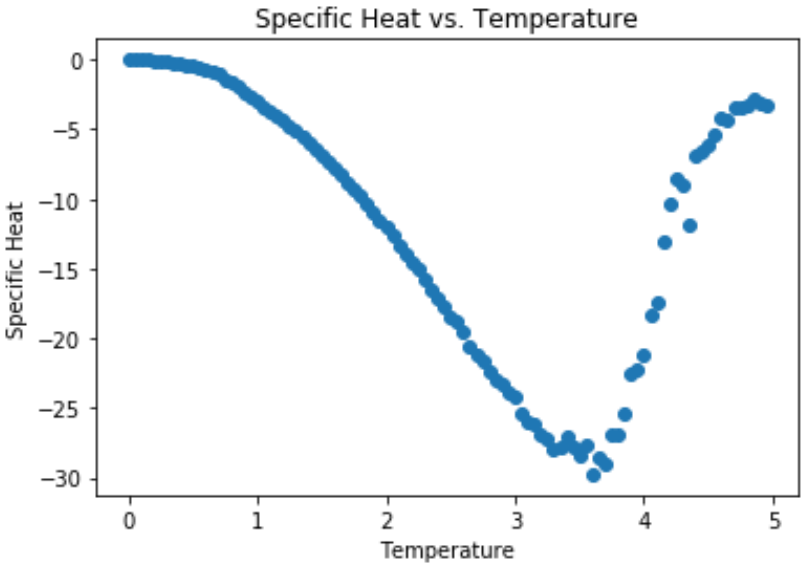


4.4 2D Triangular Grid Parallel Ising Model




Energy per site vs. Temperature
Energy per site
Temperature
−0.5
−1.0
−1.5
−2.0
−2.5
−3.0
0
1
2
3
4
5


Magnetic susceptibility vs. Temperature
Magnetic susceptibility
Temperature
0.0
−0.2
−0.4
−0.6
−0.8
−1.0
0
1
2
3
4
5


Specific Heat vs. Temperature
Specific Heat
Temperature
0
−5
−10
−15
−20
−25
−30
0
1
2
3
4
5