# Lab 7 (Ising Model)

## Simulation and Modeling (CSCI 3010U)

Faisal Qureshi

**Due back Nov. 17, 11:59 pm**

---

The goal of this exercise is to implement a 2D Ising model. We would use this implementation to detect phase transition.

A 2D Ising model consists of a lattice with $N \times N$ sites. Each site $(i, j)$ can either have an up (+1) or a down (-1) spin. The energy of the system is given by:

$$E = -\frac{1}{4} \sum_{i=1}^{N} \sum_{j=1}^{N} \sum_{(i',j') \in NN(i,j)} s[i,j] s[i',j'], where$$

$NN(i,j)$ refers to the 4-neighbour of $(i,j)$, specifically, $i' \in \{i+1, i-1\}$ and $j' \in \{j-1, j+1\}$. Note that we will be using periodic boundary conditions. $s[i,j]$ refers to the spin value at location $(i,j)$, $s[i,j]$ is either $+1$ or $-1$.

We will use demon algorithm to sample configurations of this model at various temperatures. And we will compute the values of interest (energy, magnetism, heat capacity and specificity) at these temperatures. Plotting these values against temperature will provide insight into the phase transition behavior of this model.

### Demon algorithm to sample configurations of this model at temperature $T$

1. Pick a spin location $(i, j)$ at random
2. Compute the change in energy $\Delta E$ for this location:

$$\Delta E = 2s[i,j] \sum_{(i',j') \in NN(i,j)} s[i',j'].$$

3. If $\Delta E < 0$, set $s[i,j] = -s[i,j]$. Jump to step 6.
4. Draw a uniform random number $u \in [0, 1]$.
5. If $u < e^{-\Delta E/T}$, set $s[i,j] = -s[i,j]$.
6. Repeat steps 1 to 5 for $N \times N$ steps.

### Computing quantities of interest

Since we are interested in average values, we need to sample multiple configurations at a given temperature, compute instantaneous values $A$ at these configurations and then use these instantaneous values to compute the desired average values $\langle A \rangle$. Say we generate $K$ states at a particular temperature then we can computed the average values as follows:

$$\langle A \rangle = \frac{\sum_{k=1}^{K} A(s_k)}{K},$$

where $A(s_k)$ is the instantaneous value computed using configuration (state) $s_k$.

#### Mean energy $\langle E \rangle$

This can be computed using the expression provided above.

#### Mean magnetism $\langle M \rangle$

Instantatneous magnetism $M$ is

$$\sum_{i=1}^{N} \sum_{j=1}^{N} s[i, j]$$

Use this expression to compute mean magnetism.

#### Mean specific heat $C$ at temperature $T$

$$C = \frac{\langle E^2 \rangle - \langle E \rangle^2}{T^2}$$

#### Mean specificify $\chi$ at temperature $T$

$$\chi = \frac{\langle M^2 \rangle - \langle M \rangle^2}{T}$$

### Code

Use the following starter code for this lab

```
%matplotlib inline
import numpy as np
import matplotlib.pyplot as plt

def initialize(N, random='yes'):
    '''Setting initial condition by assigning random spins'''
    if random == 'yes':
        state = 2*np.random.randint(2, size=(N,N))-1
    else:
        state = np.ones([N,N])
    return state

def plot_state(state, ax, title_str):
    w, h = state.shape
```

```
    X, Y = np.meshgrid(range(w), range(h))
    ax.pcolormesh(X, Y, state, cmap=plt.cm.bwr)
    plt.title(title_str)
```

Below is the example usage of this code, which creates a $16 \times 16$ lattice shown in Figure 1.

```
state = initialize(16, random='yes')
plt.figure(figsize=(4,4))
ax = plt.subplot(111)
plot_state(state, ax, 'Initial state')
plt.show()
```

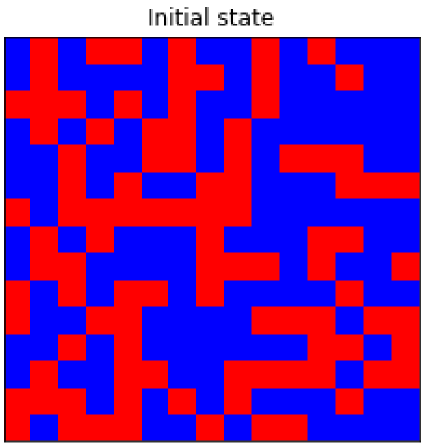


Figure 1: A 16 by 16 Lattice

### Complete the following pieces of code

```
def compute_magnetization(state):
    // TO DO

    return

def compute_energy(state):
    // TO DO

    return state

def mcstep(state, one_over_temp=1.):
    // TO DO

    return state
```

You will also need to write the simulation code that will make use of `mcstep` to estimate the quantities of interest at various temperatures and plot these quantities against temperature. For the sake of this lab, lets assume that we are intrested in the following temperatures only: `T = np.linspace(1.2,3.8,256)`.

**Complete the code below**

```
# TO DO

for m in range(len(T)):
    state = initialize(N)

    for i in range(1000):         # Run for some steps to achieve equilibrium
        mcstep(state, 1./T[i])  # before collecting statistics

    for i in range(num_steps):  # Now sample states for num_steps and compute statistics
        mcstep(state, 1./T[i])

    # Calculate the quantities of interest and
    # save for plotting
```

The following piece of code can be used to plot the quantities of interest.

```
fig = plt.figure(figsize=(20, 10));

plt.subplot(2, 2, 1 );
plt.plot(T, Energy, 'o', color="red");
plt.xlabel("Temperature (T)", fontsize=20);
plt.ylabel("Energy ", fontsize=20);

plt.subplot(2, 2, 2 );
plt.plot(T, abs(Magnetization), 'o', color="red");
plt.xlabel("Temperature (T)", fontsize=20);
plt.ylabel("Magnetization ", fontsize=20);

plt.subplot(2, 2, 3 );
plt.plot(T, SpecificHeat, 'o', color="red");
plt.xlabel("Temperature (T)", fontsize=20);
plt.ylabel("Specific Heat ", fontsize=20);

plt.subplot(2, 2, 4 );
plt.plot(T, Susceptibility, 'o', color="red");
plt.xlabel("Temperature (T)", fontsize=20);
plt.ylabel("Susceptibility", fontsize=20);
```

If all goes according to the plan you will get the plots shown in Figure 2. Notice that phase transition is visible here. The magnetism suddently dropped from 1.0 to nearly 0 at around 2.5.

## Submission

Via Blackboard.

- Python file that includes your code.
- A pdf file containing the plots.

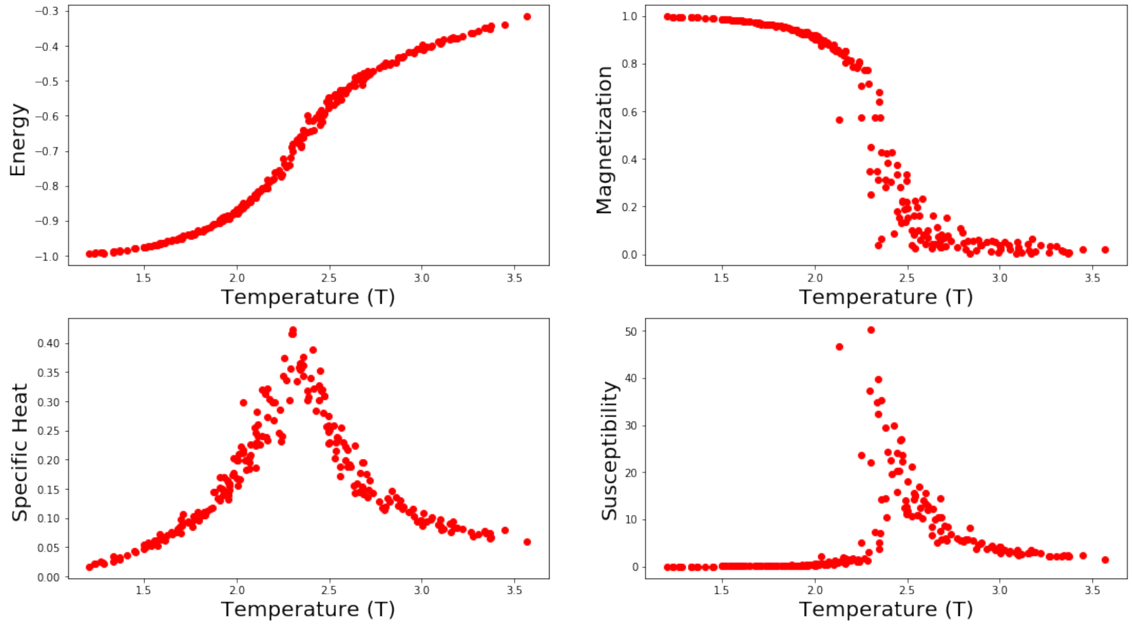


Figure 2: Plots