

Lab 10 - The Ising Model

Due: Thursday 1 November

In this lab, you will study a simple model of ferromagnetism, called the Ising model.

1 Simulating the Ising model

1. Download the code `ising.py` demonstrated in lecture and paste it into an IDLE window. This code does a Monte Carlo simulation of the two-dimensional Ising Model at some fixed temperature T . Notice how we create a “list of lists” to hold the 2D lattice of elementary Ising magnets (usually just called “spins”). A given spin object is just a `sphere()` object augmented with the attribute `state` which describes whether the spin is “up” or “down” represented by ± 1 . The use of a base `sphere` object allows us to visualize whether the spin is up or down by simply coloring it either red or blue.

Notice also that we have employed *periodic boundary conditions* in both directions which imply that the lattice site with x-coordinate $L - 1$ has righthand neighbor with x-coordinate 0 and a similar statement is true for the y-direction – the lattice is wrapped around a circle on both x and y directions. This helps decrease finite size effects. Notice how this is accomplished in the code using the modulo function `%`.

The spin object at lattice coordinates i, j is just accessed with the code `spin[i][j]` like a 2D array. Run the code and capture a picture of what you see. Notice that the code does a fixed (`MAXITS`) number of updates of every lattice spin at this temperature.

2. First we will modify the code so as to loop over a range of temperatures T and compute the mean energy and magnetization for each temperature. After the definition of `MAXITS` and before the `its` loop starts add the following code

```
for T in [0.5,1.0,1.5,2.0,2.125,2.5,2.75,3.0,3.5,4.0]:
    magnetization=0.0
    energy=0.0
```

Important – make sure that you indent all the existing code within the `its` loop once you add this loop over the temperature. One simple way to do this is to select all the code from `for its in range...` and below and hit the tab key.

3. To measure the magnetization and energy we simply average them over all the configurations generated by the Monte Carlo. To do this we add some measurement code after the simulation code used to generate a new configuration. The new code is shown below which also indicates schematically how it should fit into your existing program.

```

for T in [0.5,1.0,1.5,2.0,2.125,2.25,2.5,2.75,3.0,3.5,4.0]:
    magnetization=0.0
    energy=0.0
    for its in range (0,MAXITS):
        ###GENERATE MONTE-CARLO CONFIGURATION
        ###
        m=0
        e=0
        for i in range(0,L):
            for j in range(0,L):
                m+=spin[i][j].state
                e+=spin[i][j].state*(spin[(i+1)%L][j].state+
                                     spin[(i-1+L)%L][j].state+
                                     spin[i][(j+1)%L].state+
                                     spin[i][(j-1+L)%L].state)

        if(its>WARM):
            magnetization+=abs(m)
            energy+=e

    magnetization=magnetization/(MAXITS-WARM)
    energy=energy/(MAXITS-WARM)

```

Notice that we only start averaging the results *after* an initial number of updates `WARM` which allows the system to equilibrate to a new temperature. Add the line `WARM=100` right after the definition of `MAXITS`.

4. Finally add the following line to print out the energy and magnetization at each temperature.

```
print T, energy, magnetization
```

This line should come right after the line `energy=energy/(MAXITS-WARM)` and flush with it.

5. You may also want to comment out the `rate` command to speed up the simulation. Set the lattice size to `L=4` initially.
6. Run the code and cut/paste your results for the energy and magnetization directly into your lab writeup. Also include a copy of your modified `ising.py` program.

2 Plotting results

1. Now we will add some code to plot out a picture of the magnetization as a function of temperature. Add the following code somewhere near the top of the existing program

```
pic=gdisplay(x=600,y=0,width=400,height=200,
             xmin=0.0,xmax=4.0,ymin=0.0,ymax=1.5,
             xtitle="Temp", ytitle="Magnetization",
             title="Magnetization (L=%d)" % L,
             foreground=color.black, background=color.white)
magnet=gdots(gdisplay=pic,color=color.blue)
```

After the print statement add the following python to code to plot the magnetization to the corresponding graph

```
magnet.plot(pos=(T,magnetization/(L*L)))
```

2. Add similar code to create a new plot for the energy. You may have to change the scale of the y-axis of the plot.

3 Susceptibility

1. The magnetic susceptibility is of special interest. It is defined as

$$\chi = \frac{1}{L^2} (\langle M^2 \rangle - \langle M \rangle^2)$$

where M is the magnetization. It can be computed by adding a new variable `magsq` to the code which adds up the square of the magnetization i.e after the line `magnetization+=fabs(m)` add the line

```
magsq+=m*m
```

2. Remember to set `magsq` to zero at the beginning of the `its` loop as for `energy` and `magnetization`. Also remember to divide it by `(MAXITS-WARM)` after the `its` loops is ended.
3. Finally the susceptibility is given by

```
sus=(1.0/(L*L))*(magsq-magnetization*magnetization)
```

Add code to compute this and print its value out to the screen.

4. Also, add code to make a 3rd plot to graph the suseceptibility as a function of temperature.

5. You should see that the susceptibility curve possesses a peak at some temperature $T_c \sim 2.4$. This corresponds to the phase transition between magnetized and unmagnetized phases.
6. Increase L. You should see the value of this maximum peak height increases with lattice size. You may need to increase **MAXITS** with L. Complete the table below for the various

| L | $\log(L)$ | χ_{max} | $\log(\chi_{max})$ |
|-----|-----------|--------------|--------------------|
| 8 | | | |
| 10 | | | |
| 12 | | | |
| 14 | | | |
| 16 | | | |

values of the linear size L

7. Plot (using Excel for example) a graph of $\log(L)$ vs. $\log(\chi_{max})$ and determine the slope of the best-fitting line. Use that slope to determine the critical exponent γ .