Physics 212 – Problem Set # 1

(due Friday, October 7)

1. The purpose of this problem is to allow you to play with a simulation of the 2-dimensional Ising model. With this tool (or, better, toy), you can adjust the temperature and the external field of the model to values in a wide range and generate typical states in the thermal ensemble. I hope that this will give you some good intuition about the ideas that we will study in this course.

   To operate the simulation, first install python according the instructions in the General Course Information. If you already have python, note that you will need at least python3 and the libraries `numpy` and `matplotlib`. Then download the files `Ising.py` and `IsingGUI.py` and place them in the same directory. From the command line in this directory, type: `python IsingGUI.py`. The simulation should start running. (If this does not happen, please ask me for help.) You can control the temperature and the magnetic field by clicking in the slide bars below the figure.

   You should be able to easily understand the code in `Ising.py`. The code in `IsingGUI.py` is not very transparent. The only important line, near the top, is

   ```
   sweeps = 5
   ```

   I will explain this statement below.

   The Hamiltonian of the 2-dimensional Ising model is

   $$\mathcal{H} = -J \sum_{i,\nu} S_i S_{i+\nu} - H \sum_i S_i \tag{1}$$

   In the code, $J = 1$ and $H = h$ is the external magnetic field. The index i runs over points of 2-dimensional lattice of size N, indexed in each dimension from 0 to (N-1), and ν is summed over $\{(1,0),(0,1)\}$ The lattice is implemented by the class `lattice` in `Ising.py`, including periodic boundary conditions in both directions. The Ising model Hamiltonian is implemented in the the class `Isingmodel` in this file.

   It will be useful for you to know that the critical temperature T_c of the 2-d Ising model is

   $$T_c = \frac{2}{\log(\sqrt{2}+1)} J = 2.26919\ J\ . \tag{2}$$

   I will derive this result later in the course.

   (a) We would like to create a kinetic model that drives a system to thermal equilibrium represented by the canonical ensemble. Consider two states A and B which appear with some weight in the ensemble. Let $r(A \to B)$ and $r(B \to A)$ be the transition

rates (jumps per second) between the two states. Show that the relative weights of states in the canonical ensemble is preserved if, for all A, B,

$$\frac{r(A \to B)}{r(B \to A)} = \frac{e^{-\beta \mathcal{H}[B]}}{e^{-\beta \mathcal{H}[A]}} \tag{3}$$

This relation is called *detailed balance*.

(b) A simple way to implement detailed balance is by using the *Metropolis algorithm*. Given a state A, consider a transition from this state to a neighboring state B. Compute $\Delta\mathcal{H} = \mathcal{H}[B] - \mathcal{H}[A]$. If $\Delta\mathcal{H} < 0$ (downhill), make the transition. If $\Delta\mathcal{H} > 0$ (uphill), make the transition with probability $\exp[-\beta\Delta\mathcal{H}]$. This is an example of a more general kinetic scheme called *Markov Chain Monte Carlo*.

Show that the method `update_point` in the class `Isingmodel` implements this algorithm and thus insures detailed balance. Then in principle the set of states generated by the program should approach the canonical ensemble.

The code proceeds by visiting each site of the lattice and deciding whether or not to flip the spin at that site. When every site has been visited, this constitutes what I call a "sweep". One sweep is implemented by the method `update_lattice`. The "sweep" is a useful unit of time to discuss this simulation.

The code as written carries out 5 sweeps, then displays the new configuration of spins and the value of $\langle\mathcal{H}\rangle$ and $\langle M\rangle$ per site averaged over the 5 sweeps. To change this number, modify the statement `sweeps = 5` noted above. Raising this number will cause the simulation to appear to run more slowly, but the numerical values shown will have less fluctuation. Adjust the number of sweeps per display according your patience and the speed of your computer.

(c) Starting from a high temperature, say, $T = 3.5$, with $h = 0$, watch the evolution of the configurations as a function of time. The program is initialized with (almost) all spins up. How long (in sweeps) does it take to reach thermal equilibrium, as a function of T and h? Notice that you an get a uniform array of spins without restarting the simulation by setting the temperature temporarily to a low value.

(d) Notice that the spins form correlated clusters which are small at high temperature and grow as T is lowered toward T_c. In what region of T is the size of the clusters smaller than 5 lattice sites?

(e) Now set the temperature to a value below T_c, for example, $T = 2.0$, with $h = 0$. Create a uniform configuration of spins, and let it come to equilibrium at this temperature. Describe the resulting equilibrium configuration. Notice that the system is not perfectly magnetized. In what region of temperature for $T < T_c$ are the correlated clusters of spins smaller than 5 lattice sites?

(f) Consider the effect of the external field, by repeating the above experiment at $h = 0.1$ and $h = 0.4$. Where is the size of the clusters maximal as T is lowered from above to below T_c, and what is the maximum value?

(g) Setting $h = 0$, make a plot of the equilibrium magnetization as a function of temperature, starting at $T = 1.8$ and increasing in steps. You might have to collect and average a number of values of $\langle M \rangle$. Note that this exercise gets more difficult as T approaches T_c.

(h) Because the system here is finite, the ensembles called "superselection sectors" in the lecture are not completely distinct. Near $T = T_c$, $h = 0$, you should be able to observe an almost uniform state of up spins turn itself into an almost uniform state of down spins. See if you can observe this, and take a few screen shots.

(i) Go into the lattice code and rewrite it to impose fixed boundary conditions, all spins up $(+1)$ on the left-hand boundary and all spins down (-1) on the right-hand boundary. This has an interesting effect on the spin configurations for $T < T_c$, $h = 0$. Take some screen shots. How does the thickness of the boundary depend on the temperature? Why is it that the boundary does not straighten itself out to minimize its size?

There are more sophisticated things to do with this simulator if you are experienced with python programming. If you are interested, ask me.