Exercise #8 - Monte Carlo Simulation

Computational Physics Lab

Instructor: Prof. Sean Dobbs

March 23, 2020

Due by 11 PM, April 24th

Submission

For each problem, you will write one or more python programs. These programs should follow the Python coding and formatting conventions outlined for the course. Many problems will have additional questions to answer, or will ask for a record of the terminal showing the output when the program is run with various arguements. These answers can be given either in block comments in the prologue of the corresponding assignment, or in clearly labeled text files. The terminal output should contain some information on the username and machine you are running on (this will often be in the command prompt).

You will be evaluated based on the files contained in your remote GitHub repository at the due date. You should and commit all the files you created to your local repository on a regular basis. You should do this by adding any new or modified files using "git add", and then finalizing changes using "git commit". Remember that "git status" will give you information on which files in your repository. have been changed. Remember to add short, but useful comments when performing a commit. When you are finished with the exercise, push the current status of your local repository to the remote repository on GitHub using the command "git push -u". You are encouraged to push your local files to the remote repository periodically before you are done, and certainly well before the deadline, if possible.

1. The Ising Model

The Ising model is a theoretical model of a magnet. The magnetization of a magnetic material is made up of the combination of many small magnetic dipoles spread throughout the material. If these dipoles point in random directions then the overall magnetization of the system will be close to zero, but if they line up so that all or most of them point in the same direction then the system can acquire a macroscopic magnetic moment - it becomes magnetized. The Ising model is a model of this process in which the individual moments are represented by dipoles or "spins" arranged on a grid or lattice. In this case we are using a square lattice in two dimensions, although the model can be defined in principle for any lattice in any number of dimensions.

The spins themselves, in this simple model, are restricted to point in only two directions, up and down. Mathematically the spins are represented by variables $s_i = \pm 1$ on the points of the lattice, $+1$ for up-pointing spins and -1 for down-pointing ones. Dipoles in real magnets can typically point in any spatial direction, not just up or down, but the Ising model, with its restriction to just the two directions, captures a lot of the important physics while being significantly simpler to understand.

Another important feature of many magnetic materials is that the individual dipoles in the material may interact magnetically in such a way that it is energetically favorable for them to line up in the same direction. The magnetic potential energy due to the interaction of two dipoles is proportional to their dot product, but in the Ising model this simplifies to just the product $s_i s_j$ for spins on sites i and j of the lattice, since the spins are one-dimensional scalars, not vectors. Then the actual energy of interaction is $-J s_i s_j$, where J is a positive interaction constant. The minus sign ensures that the interactions are \textit{ferromagnetic}, meaning the energy is lower when dipoles are lined up. A ferromagnetic interaction implies that the material will magnetize if given the chance. (In some materials the interaction has the opposite sign so that the dipoles prefer to be antialigned. Such a material is said to be \textit{antiferromagnetic}, but we will not look at the antiferromagnetic case here.)

Normally it is assumed that spins interact only with those that are immediately adjacent to them on the lattice, which gives a total energy for the entire system equal to

$$E = -J \sum_{\langle ij \rangle} s_i s_j \,,$$

where the notation $\langle ij \rangle$ indicates a sum over pairs i, j that are adjacent on the lattice. On the square lattice we use in this exercise each spin has four adjacent neighbors with which it interacts.

The starter code in your repository (`ising.py`) contains a function to calculate the total energy of the system, as given by the equation above. That is, for a given array of values of the spins, it goes through every pair of adjacent spins and adds up the contributions $s_i s_j$ from all of them, then multiplies this result by $-J$. Note that this program uses the numpy ability to calculate matrices implicitly. If we did the calculation step by step, then such a calculation would be significantly slower.

1) Use the energy function as the basis for a Metropolis-style simulation of the Ising model with $J = 1$ and temperature $T = 1$ in units where the Boltzmann constant k_B is also 1. Initially set the spin variables randomly to ± 1, so that on average about a half of them are up and a half down, giving a total magnetization of roughly zero. Then choose a spin at random, flip it, and calculate the new energy after it is flipped, and hence also the change in energy as a result of the flip. Then decide whether to accept the flip using the Metropolis acceptance formula (Eq. (10.60) from the book, or see lecture notes). If the move is rejected you will have to flip the spin back to where it was. Otherwise you keep the flipped spin. Now repeat this process for many moves.

2) Make a plot of the total magnetization $M = \sum_i s_i$ of the system as a function of time for a million Monte Carlo steps. You should see that the system develops a "spontaneous magnetization,"a nonzero value of the overall magnetization. Hint: While you are working on your program, do shorter runs, of maybe ten thousand steps at a time. Once you have it working properly, do a longer run of a million steps to get the final results.

3) Run your program 10 times and observe the sign of the magnetization that develops, positive or negative. Describe what you find and give a brief explanation of what is happening.

4) Run your program several times with $T = 1$, $T = 2$ and $T = 3$. Explain briefly what you see for the magnetization in the runs with these three different temperatures. How and why does the behavior of the system change as temperature is increased?

5) Using this method, we can also study other properties of the system. One other such property is the specific heat C, which describes how the internal energy of the system changes as the temperature changes. It turns out that in this model, the specific heat is related to the variance of the energy across the lattice, $C = k_b \beta^2 Var(E)$. Modify your program to also plot the specific heat for each step of the calculation. Run your program several times for each of the following values of T: 1, 1.5, 2, 2.5, 3. In this calculation, you should assume $k_B = 1$. Explain briefly how the resulting specific heat varies as a function of T, and what this might mean about the behavior of the system. Note: you will find the function `numpy.var()` useful.

For full credit turn in your programs, your plot for step (2), and your answers for (3), (4), and (5).