HW 7 Advanced Computational Physics

October 4, 2005 Due October 13

In this assignment you will develop a Monte Carlo simulation of the Ising model on a square lattice using the Metropolis algorithm. The model is defined by the energy

$$E = -\frac{1}{2}J\sum_{i,j=0}^{n-1} s_{ij}(s_{i+1,j} + s_{i-1,j} + s_{i,j+1} + s_{i,j-1}),$$

with $s_{ij} = \pm 1$. Use periodic boundary conditions. Then for $i = n-1$ the expression $i+1$ means 0, and for $i = 0$ the expression $i-1$ means $n-1$. This also is the case for j. Develop your code as explained on the next page in a single file with this structure:

```
from RandomArray import *
def cluster(i,j):
  """Energy of the cluster of 5 Ising spins centered on site (i,j)"""
  ...
  return ...
def energy():
  """Energy of the entire Ising spin system"""
  ...
  return ...
def metropolis():
  """Do "nsweeps" Metropolis sweeps of the entire lattice, storing
     the energy at the end of each sweep in array Energies."""
  Energies = zeros(nsweeps+1,Float)
  ...
  for sweep in range(1,nsweeps+1):
    for step in range(n**2):
      (i,j) = randint(0,n,2)
      ...
    Energies[sweep] = E
    plt.imagesc(s)  #Snapshots (don't plot every sweep)
  return ...

# Main code
n = ...
seed()     #New seed each run; or use seed(0,0) for debugging
s = ones((n,n))

# Test functions cluster and energy
# Do the Monte Carlo at various temperatures T
energies = metropolis()
```

Most of your effort in this assignment will be in debugging. I recommend that you write the functions in the order indicated below, making sure that each is correct before starting the next. While debugging look at a smaller problem (*e.g.*, n=4, nsweeps=3). It is also useful to print many intermediate results. For this entire assignment set $J = 1$ and use T to stand for $k_B T$.

PART I

1. Write a function `cluster(i,j)` which calculates the energy in the cluster of spins centered on i, j:

   $$\texttt{cluster(i,j)} = -s_{ij}(s_{i+1,j} + s_{i-1,j} + s_{i,j+1} + s_{i,j-1}).$$

   Assume that array `s` is defined in the main routine (*i.e.*, is a 'global' variable), as shown above.

2. Write a function `energy()` which calculates the total energy `E` of the spin configuration `s`. You may wish to use your function `cluster`.

3. Write a function `metropolis()` which performs a given number of Monte Carlo sweeps. Structure it as shown on the previous page.

   Here is the Metropolis algorithm in brief: Pick a random site (i, j). Calculate how much the energy would change if $s(i, j)$ were flipped; call this Δ. If $\Delta \leq 0$, do the flip $[s(i, j) \rightarrow -s(i, j)$ and $E \rightarrow E + \Delta]$. If $\Delta > 0$, generate a random number w uniformly in the range [0,1]; if $e^{-\Delta/T} \geq w$, do the flip $[s(i, j) \rightarrow -s(i, j)$ and $E \rightarrow E + \Delta]$, and otherwise do not. Repeat many, many times.

   Store the energy (the initial energy and after each sweep) in an array. Plot snapshots of the initial condition and after occasional sweeps. (The sample code uses plt. You could instead choose two symbols, one for up and one for down, and plot a scatter plot.)

4. The spins are all initially pointing up (all $s_{ij} = 1$). Use your routine to watch the evolution of a system of 32×32 spins. Do this for temperatures `T=3, 2.5, 2.25` and `1`. The snapshots produce an interesting animation of the Monte Carlo process.

   Make hard copies of (and hand in) a few of these snapshots. Also plot and hand in the energies *vs.* sweep number for each temperature, and hand in written comments explaining your results.

   You should find that during some initial time the system *equilibrates*, that is, moves away from the original spin configuration towards an equilibrium state; that after this initial period the spins continue to fluctuate about their equilibrium state; and that the equilibrium spin configurations look quite different at low and high temperatures. For most temperatures 50 sweeps should be enough, but for T=2.25 you will need many more.

5. Repeat the previous problem using a random initial configuration of spins. Again think about your results, and write comments, including comparisons with the result of the previous problem. Snippet:

   ```
   s = randint(0,2,(n,n))
   s = 2*s - 1
   ```

   In this case you may need to use many sweeps for both `T=2.25` and `T=1`.

PART II

The Ising Model has a *phase transition*: the system spontaneously magnetizes below a *critical temperature* T_c and is disordered for $T > T_c$. Now that your code is working you will study this phase transition quantitatively by modifying your it to calculate $m(T)$, the average magnetization as a function of temperature. The magnetization is simply the average spin:

$$m = n^{-2} \sum_{ij} s_{ij}.$$

The two-dimensional Ising model on a square lattice was solved analytically for $n \to \infty$ in a tour-de-force by Lars Onsager in 1944. Thus there are exact results for $m(T)$ along with other quantities. For example, $T_c = 2J/\ln(1+\sqrt{2}) \approx 2.2692J$. The exact magnetization (found by C.N. Yang, building on Onsager's solution) is:

$$m(T) = \begin{cases} 0, & T > T_c \\ \left(1 - [\sinh(2J/T)]^{-4}\right)^{1/8}, & T < T_c. \end{cases}$$

Modify your Monte Carlo code to calculate the magnetization at each sweep, storing the result in an array `mag`. Plot `mag` *vs.* sweep. Then, to get the average magnetization, skip the first `nequil` sweeps, and average over the remaining `nsteps-nequil` sweeps:

```
maverage=sum(mag[nequil:])/(nsweeps-nequil) .
```

Calculate one such average magnetization for each temperature for a system of 32×32 spins with $J = 1$ at temperatures $T = 1.75$, 2, 2.25, ..., 3.75. Plot the **absolute value** of `maverage` *vs.* `T`. Plot your numerical results (as symbols without lines) and the exact analytical results (as smooth lines – that is, calculated at many more values of T – without symbols).

In addition hand in one other plot at each temperature: the magnetization *vs.* sweep number. Put the results for each temperature on a different figure.

You may have to run a long time to get good statistics. Get everything working, all the way to your desired plots, with very short runs. Then and only then choose large enough values of `nsweeps` and `nequil` to get reasonably accurate averages. It's a good idea to save your final average values so that if you decide to change your final plots you don't need to repeat lengthy runs.