1. (Ising model, based on Newman). The Ising model is a model for ferromagnetism. Here we will consider a 2-d model a periodic lattice of atoms with magnetic dipoles (spins of either +1 or −1) interact with their 4 neighbors (up, down, left, and right). Depending on the temperature of the system, they can either all align (becoming magnetic) or take on a random configuration (no magnetism). We will model this using Markov chain Monte Carlo.

The magnetic potential energy between two spins is proportional to their dot product as $-J s_i s_j$, where $J > 0$ is the interaction strength. The total energy of the system is:

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

(1)

where the $\langle ij \rangle$ indicates the the sum is done over pairs that are adjacent on the lattice (the 4 neighbors).

Model this using the MCMC method by creating a lattice of $20 \times 20$ atoms, each initially given a random spin. When computing the total energy of the system, be careful not to double count the pairs.

This energy can be used with the Metropolis algorithm we saw in class. Take $J = 1$, and work in units where $k_B = 1$. Your move set will consist of picking an atom at random and flipping its spin. Evaluate this new energy and accept the new state if it meets the Metropolis acceptance criteria:

$$P_a = \begin{cases} 
1 & E_j \leq E_i \\
\frac{e^{-\beta(E_j-E_i)}}{E_j > E_i} & E_j > E_i
\end{cases}$$

(2)

Where $P_a$ is the acceptance probability. If the change is rejected, then revert the spin of the atom you changed.

You’ll need a lot of iterations (like $10^6$) to get a system that goes to equilibrium.

Start with a temperature $T = 1$, and increase it in small steps (like $\Delta T = 0.25$) to $T = 3$. Somewhere between $T = 2$ and $T = 2.5$, you’ll notice that the system goes from ordered (all the spins want to be aligned) to random (no clear alignment).

Make a plot of the energy of the system and the $|M|$, the absolute value of the magnetization,

$$M = \sum_i s_i$$

(3)

as a function of $T$.

Note: you’ll want to make sure that your code is reasonably optimized—for a $20 \times 20$ lattice with $10^6$ iterations, it can take ~ 3 minutes per temperature value. If using an interpretive language (like python), avoid loops in the expressions where possible.