Example: 1D Ising Model

Exercise 10.9 on page 487-489 discusses the Ising model of magnetization, well known in the field of statistical mechanics. In Wednesday's lab you will do this exercise which involves a 2D lattice of spins. In lecture, we will start solving the 1D version below.

1. Consider N dipoles each with a spin state s_i that can equal either +1 or -1. Create an array to hold the spin state for all N dipoles. Initialize the array so that all spins are initially set to +1. Write a function to calculate the total energy of the system which is given by:

   $$E = -J \sum_{<ij>} s_i s_j \tag{1}$$

   where J is an exchange energy constant which you can set = 1.0 and $< ij >$ means that the sum is over pairs i, j that are adjacent on the lattice. For example, if N=5, then:

   $$E = -J(s_0 s_1 + s_1 s_2 + s_2 s_3 + s_3 s_4) \tag{2}$$

   Notice that you don't double-count pairs (i.e. if you have $s_1 s_2$ then you don't need $s_2 s_1$). You will want to use array math for the sum rather than a for loop over all the possible states, so think about how to implement that (hint: the functions "dot" or "sum" from numpy will work well for this).

   Also write a function to calculate the total magnetization of the system which is given by:

   $$M = \sum_{i=1}^{N} s_i \tag{3}$$

   Set N=100 and call your function. Print out the total energy and magnetization.

2. Write a function that implements the Metropolis algorithm to flip a spin state randomly, calculate the new total energy and decide whether to accept the flip. Work in units such that k_B and T=1.0. Here is what this function needs to do:

   - Randomly select an element of the spin array and flip its spin.
   - Call your energy function from Q1 to calculate the energy of this new spin state (E_{new}).
   - Calculate the Boltzmann factor for this change in energy:

     $$p = \exp[-(E_{new} - E_{old})/k_B T] \tag{4}$$

     where E_{old} is the energy before the flip.
   - Keep this new state if one of the following conditions is met:
     (a) If $E_{new} - E_{old} <= 0$
     (b) If $E_{new} - E_{old} > 0$ AND $p > random()$
   - If neither of these is true, keep the old state.

Run your program to calculate the new energy and new magnetization for one flip. Print out the values.

3. Now implement a loop in your program to run your Metropolis algorithm for 1000 flips, calculating the energy and magnetization after each flip. Plot the energy and magnetization as a function of the number of flips. You have now implemented the Markov Chain process.