10 February 2025 – PHYS 6260
Prof. John Wise
Newman Chapter 10: Random Processes and Monte Carlo Methods

10.3: Monte Carlo Simulation

All branches of physics use Monte Carlo (MC) simulations, which model physical processes with random numbers. However, the largest user of MC simulations is statistical mechanics because it is fundamentally the description of apparently random processes. So we will focus on Monte Carlo simulations with statistical mechanics applications.

Importance sampling and statistical mechanics

Suppose that we have a system at temperature T. The fundamental problem of statistical mechanics is calculating the average value of some quantity. Statistical mechanics describes the system to have some probability of occupying some state i with energy E_i, which is given by the Boltzmann formula:

$$P(E_i) = \frac{e^{-\beta E_i}}{Z}, \qquad \text{with} \qquad Z = \sum_i e^{-\beta E_i}, \tag{1}$$

where $\beta = 1/k_{\rm B}T$. Then the average value of some quantity X has a value X_i in the $i^{\rm th}$ state is

$$\langle X \rangle = \sum_i X_i P(E_i) \tag{2}$$

We cannot simply sum over all of the states because there are too many of them – remember Avogadro's number and then think about all of the combinations of the different atomic states.

We can use the same idea of **importance sampling** that we covered when integrating infinite domains when solving computational statistical mechanics problems. We will choose N states at random and then take the average value over them,

$$\langle X \rangle \simeq \frac{\sum_{k=1}^N X_k P(E_k)}{\sum_{k=1}^N P(E_k)} \tag{3}$$

Notice that there will be many states with $E_i \gg k_{\rm B}T$ that will contribute very little to the average quantity $\langle X \rangle$. However there will be a range of states that will contribute the most to the average. We want to use importance sampling to favor choosing those states from some non-uniform probability function.

To do so, let's inspect a weighted average of a quantity g,

$$\langle g \rangle_w = \frac{\sum_i w_i g_i}{\sum_i w_i} \tag{4}$$

where the weighting function is arbitrary at this point. Let's choose to evaluate the function $g_i = X_i P(E_i)/w_i$, giving

$$\left\langle \frac{X_i P(E_i)}{w_i} \right\rangle_w = \frac{\sum_i w_i X_i P(E_i)/w_i}{\sum_i w_i} = \frac{\sum_i X_i P(E_i)}{\sum_i w_i} = \frac{\langle X \rangle}{\sum_i w_i} \tag{5}$$

Solving for $\langle X \rangle$, we find

$$\langle X \rangle = \left\langle \frac{X_i P(E_i)}{w_i} \right\rangle_w \sum_i w_i \tag{6}$$

To choose a random state that favors the most probable states, we can simply use the $w_i = P(E_i)$ and recall that $P(E_i)$ is normalized so that $\sum_i w_i = 1$. The weighted average is now

$$\langle X \rangle \simeq \frac{1}{N} \sum_{k=1}^{N} X_k, \tag{7}$$

where the k states are chosen from the Boltzmann formula's probability, $e^{-\beta E_i}$.

The Markov chain method

However choosing a random state isn't enough to fully describe the system. In the probability function (Equation 1), the **partition function** Z was in the denominator, and we never addressed the value of this number. It's the sum over all states, which we can't address on a one-to-one basis. However, we can randomly choose states *without knowing* Z using a **Markov chain**.

In a Markov chain, we start with a system in some state i. Then the next state j that we sample is not randomly chosen, but connected to the current state i. We randomly select a **move set** that changes the system by one degree of freedom. For example, suppose that we are simulating a set of N molecules each with its own quantum state. A move set would change the quantum state of one molecule but leave all other molecules alone. The change to a new state is determined through a set of *transition probabilities* T_{ij} that give a probability of changing from a state i to j. If T_{ij} is chosen right, then the Boltzmann probability $P(E_i)$ will be recovered given enough changes (see Appendix D in the book for the complete derivation).

The transition probabilities are inherently normalized

$$\sum_j T_{ij} = 1 \tag{8}$$

because on every step in the Markov chain, we will end up in some state j. The secret in this method is to choose the T_{ij} values so that the partition function vanishes,

$$\frac{T_{ij}}{T_{ji}} = \frac{P(E_j)}{P(E_i)} = \frac{e^{-\beta E_j}/Z}{e^{-\beta E_i}/Z} = e^{-\beta(E_j - E_i)} \tag{9}$$

Suppose during the Markov chain, we happen to achieve some state i that has a Boltzmann probability $P(E_i)$ for all i. What happens if make another change in the chain? Will we throw away this perfect solution? Let's use Equation (9) to find the probability of being in state j on the next step. It will be the sum of the probabilities that we got there from every other possible state i:

$$\sum_i T_{ij} P(E_i) = \sum_i T_{ji} P(E_j) = P(E_j) \sum_i T_{ji} = P(E_j), \tag{10}$$

which proves that once a Markov chain obtains a Boltzmann probability it will remain one *forever*. Furthermore, starting from any random state, the Markov chain will converge to the Boltzmann probability over many changes.

However, we still haven't addressed how we choose the values of T_{ij} that controls the move set. One method is called the *Metropolis algorithm*. The move set can be chosen at random (uniformly) and does not necessarily satisfy Equation (10). But the key in the Metropolis algorithm is the **acceptance or rejection** of the change, which has the probability

$$P_a = \begin{cases} 1 & \text{if } E_j \leq E_i \\ e^{-\beta(E_j - E_i)} & \text{if } E_j > E_i \end{cases} \tag{11}$$

Another way to think about this equation is that if the total energy of the system decreases, we always accept the change. If the energy increases, then there's some non-zero probability that we accept the change, otherwise, we reject it.

We can show that the Metropolis algorithm agrees with Equation (10) by considering some state i changing into one of M other states j. The probability of morphing into one of those states is $1/M$, so the total probability if $E_j > E_i$ is

$$T_{ij} = \frac{1}{M} \times e^{-\beta(E_j - E_i)} \quad \text{and} \quad T_{ji} = \frac{1}{M} \times 1. \tag{12}$$

For $E_i \geq E_j$, we have

$$T_{ij} = \frac{1}{M} \times 1. \quad \text{and} \quad T_{ji} = \frac{1}{M} \times e^{-\beta(E_i - E_j)} \tag{13}$$

both of whose ratios agree with Equation (10).

The complete Markov chain algorithm is as follows:

1. Choose a random initial state
2. Choose a move uniformly at random from an allowed set of moves
3. Calculate the value of the acceptance probability P_a
4. Accept or reject the proposed change
5. Measure the value of the quantity X in the current state and add it to a running sum of such measurements
6. Repeat from step 2

However, there are some subtleties about the Metropolis algorithm and Markov chains.

1. When a move is rejected, the quantity X must still be measured and added to the sum.
2. When choosing a new state, *every possible state* must be accessible through some path. This is known as an *ergodic* move set.
3. Although we know that a Markov chain will converge toward the answer, it's unknown exactly how long it will take to *equilibrate*.

Example 10.3: Monte Carlo Simulation of an Ideal Gas

The quantum states of a particle or atom of mass m in a cubic box of length L have three integer quantum numbers $(n_x, n_y, n_z) = 1 \ldots \infty$ and energies given by

$$E(n_x, n_y, n_z) = \frac{\pi^2 \hbar^2}{2mL^2}(n_x^2 + n_y^2 + n_z^2) \tag{14}$$

Particles in an ideal gas are non-interacting, and the total energy is just the sum of individual particle energies, given by Equation (14).

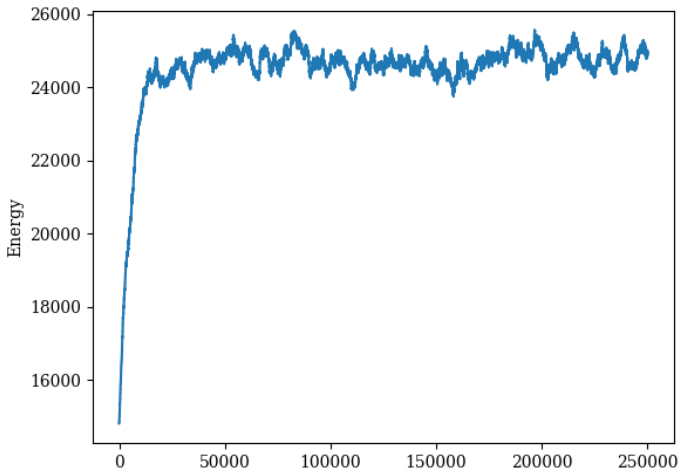
Our move set will be the combination of these random choices: some random particle will have one of its quantum numbers randomly chosen changed by either $+1$ or -1 (random, too)! It can be shown that the change in energy for a $n_i \pm 1$ change in state will be, respectively,

$$\Delta E = \frac{\pi^2 \hbar^2}{2mL^2}(\pm 2n_x + 1). \tag{15}$$

The following program simulates $N = 1000$ particles, starting from a "ground-state" with $n_x = n_y = n_z = 1$ for all particles. The system has a temperature of $k_B T = 10$. For simplicity, we set $\hbar = m = L = 1$. Note that we don't allow the quantum numbers to change below 1. The Markov chain is executed 250,000 times here and the resulting figure is shown.

```
from random import random,randrange
from math import exp,pi
import numpy as np
import matplotlib.pyplot as plt

T = 10.0
N = 1000
steps = 250000
```




```
# Create a 2D array to store the quantum numbers
n = np.ones([N,3],int)

# Main loop
eplot = []
E = 3*N*pi*pi/2
for k in range(steps):

    # Complete in-class
    #
    # Determine a random move set and associate change in energy
    #

    # Complete in-class
    #
    # Decide whether to accept the move. If so, change the state and total energy
    #

    eplot.append(E)

# Make the graph
plt.plot(eplot)
plt.xlabel("Iteration")
plt.ylabel("Energy")
plt.show()

plt.hist(n, 3, label=['x','y','z'])
plt.legend(loc='best')
plt.show()
```