Today's lecture: basic principles of stochastic Markov processes and Monte Carlo simulations.

Overview

In the previous lectures, we saw that the molecular dynamics (MD) approach generates a **deterministic** trajectory of atomic positions as a function of time, by way of solving Newton's equations of motion. In doing so, it allows us to extract both thermodynamic and kinetic property averages from the simulations.

In this lecture, we cover the other major method for generating atomic trajectories: the **Monte Carlo (MC)** approach. Unlike MD, Monte Carlo methods are **stochastic** in nature—the time progression of the atomic positions proceeds randomly and is not predictable given a set of initial conditions. The dynamic principles by which we evolve the atomic positions incorporate **random moves** or perturbations of our own design; as such, the dynamics of Monte Carlo trajectories are not representative of the true system dynamics and instead depend on the kinds of random moves that we perform. However, Monte Carlo methods rigorously generate correct thermodynamic properties as they are designed by construction to do so.

MC and MD approaches are complementary. If we are interested in kinetic properties, MD is a natural choice. On the other hand, MC methods offer several attractive features:

- MC methods naturally and easily treat **different thermodynamic ensembles**. For example, it is quite simple to perform (rigorously) a constant temperature simulation using MC methods. This contrasts with the special thermostat techniques often required with molecular dynamics simulations.
- MC methods offer great **flexibility** in choosing the random moves by which the system evolves. This can often greatly speed equilibration in complex systems, e.g., in dense polymeric systems.
- MC methods are **not subject to inaccuracies due to discrete-time approximations** of the equations of motion.
- MC methods **require only energies** in order to generate the atomic trajectories. These approaches, therefore, do not require expensive force calculation and can handle continuous and discrete intermolecular potentials in an equivalent way.

The MC approach was first developed in the 1940s and 50s by researchers at Los Alamos working on nuclear weapons projects. It was later applied extensively to model atomic systems, most notably the hard-sphere system, and has since become widely employed as a modern tool in molecular simulation and many other fields (e.g., statistics and finance). Notably, the MC method was one of the earliest algorithms to be used on the first computers.

Our discussion here will center on the use of the Monte Carlo method in computing thermodynamic properties of atomic systems in the canonical ensemble.

Simple example of canonical Monte Carlo

Before diving into the details of the method, let's consider a simple example. We will examine a Monte Carlo simulation of the Lennard-Jones liquid, whose energy function is given in dimensionless units by:

$$U(\mathbf{r}^N) = \sum_{i<j} 4\left(r_{ij}^{-12} - r_{ij}^{-6}\right)$$

The simulation will be performed at reduced temperature T. Our simulation progresses through iterations of the following basic **Monte Carlo step**:

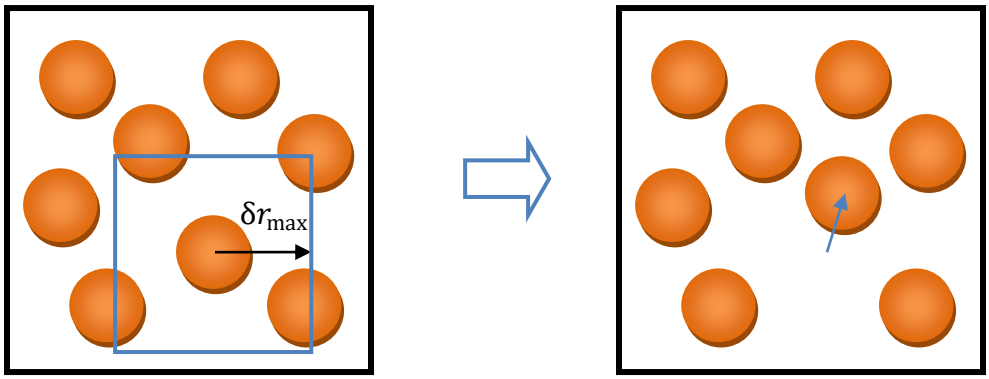
1. Randomly pick one of N particles.
2. Perturb each of the x, y, z coordinates separately by three random values taken from the uniform distribution on the interval $[-\delta r_{\max}, \delta r_{\max}]$. Here, $\delta r_{\max}$ is the **maximum displacement**.
3. Compute the change in potential energy due to the particle move, $\Delta U = U_2 - U_1$.
4. Use the following rule, called the **Metropolis criterion**, to decide whether or not to keep the move or instead revert back to the original configuration before step 2:
    - If $\Delta U < 0$, accept the move.
    - If $\Delta U > 0$, compute $P^{\text{acc}} = e^{-\Delta U/T}$. Draw a random number r on the interval $[0.0, 1.0)$ and accept the move if and only if $P^{\text{acc}} > r$.
5. If the move is accepted, keep the new configuration and update any running averages with it (e.g., the potential energy). If the move is rejected, discard the new configuration and update any running averages with the original state.

Ultimately, aggregated over many MC steps, our simulation produces configurations that obey the **canonical distribution** at T. In other words, configurations appear with probability

$$\wp(\mathbf{r}^N) \propto e^{-\frac{U(\mathbf{r}^N)}{T}}$$

Any property of interest, such as the average potential energy or pressure (via the virial), can then be computed as a "time"-average over the sequence of configurations produced during the Monte Carlo steps.

The perturbation that we apply in each step to the coordinates of an atom is called a **single-particle displacement** and it one of many possible kinds of **Monte Carlo moves**. A single-particle displacement move looks something like the following:



Notice that we have a free parameter in our approach: the maximum displacement $\delta r_{\max}$. This is a parameter that we can tune to adjust the efficiency of our moves. If it is too large, particles will be displaced far from their original positions and will likely have core overlaps with other particles in the system, resulting in a rejection of the move. If it is too small, the evolution of the system with MC steps will be very slow. Generally, maximum displacements are adjusted such that the average **acceptance rate** of proposed moves is 30-50%.

In step 4, we compute the acceptance probability and draw a random number to determine whether or not we accept the proposed move. Aside from the random move proposals, this is the stochastic element in our simulation. We can draw random numbers in the computer using pseudorandom number generators, discussed more below.

Statistical mechanics in the canonical ensemble (constant $\boldsymbol{N}, \boldsymbol{V}, \boldsymbol{T}$)

Before we proceed with a detailed treatment of the Monte Carlo method, we review some statistical mechanical concepts. If one connects a system to a very large heat bath and allows exchange of energy, the total energy of the system is no longer constant, but fluctuates. Instead, the system temperature is held constant, equal to that of the heat bath. Such conditions give rise to the **canonical ensemble,** perhaps the most common ensemble of physical interest.

In the canonical ensemble for a classical system, the **microstate distribution** is proportional to the Boltzmann factor:

$$\wp(\mathbf{p}^N, \mathbf{r}^N) \propto e^{-\beta H(\mathbf{p}^N, \mathbf{r}^N)}$$

where $\beta = 1/k_B T$. The normalizing factor for the probabilities is the **canonical partition function**:

$$Q(T, V, N) = \frac{1}{h^{3N} N!} \int e^{-\beta H(\mathbf{p}^N, \mathbf{r}^N)} d\mathbf{p}^N d\mathbf{r}^N$$

Since the Hamiltonian is additive in kinetic and potential energies, the integral over momenta can be performed analytically. For spherically-symmetric particles,

$$Q(T, V, N) = \frac{Z(T, V, N)}{\Lambda(T)^{3N} N!} \qquad \text{where } Z \equiv \int e^{-\beta U(\mathbf{r}^N)} \, d\mathbf{r}^N \,, \qquad \Lambda(T) \equiv \left(\frac{h^2}{2\pi m k_B T} \right)^{\frac{1}{2}}$$

$Z(T, V, N)$ is called the **configurational integral**, and it only depends on the potential energy function for the particles. $\Lambda(T)$ is the thermal de Broglie wavelength. For heteroatomic systems, we have to account for the different atomic masses and employ multiple de Broglie wavelengths in this expression, one for each atom type.

In the canonical ensemble, the distribution of particle positions (configurations) is separable from the momentum degrees of freedom. One can write,

$$\wp(\mathbf{p}^N, \mathbf{r}^N) = \wp(\mathbf{p}^N) \wp(\mathbf{r}^N)$$

$$\wp(\mathbf{p}^N) = \frac{e^{-\beta K(\mathbf{p}^N)}}{h^{3N} \Lambda(T)^{-3N}}$$

$$\wp(\mathbf{r}^N) = \frac{e^{-\beta U(\mathbf{r}^N)}}{Z(T, V, N)}$$

Note that the momenta distribution is known analytically, but the configurational distribution requires solution of the integral $Z(T, V, N)$, which depends on the specific form of the potential energy function. Also notice that, by construction,

$$\int \wp(\mathbf{r}^N) d\mathbf{r}^N = 1 \qquad \int \wp(\mathbf{p}^N) d\mathbf{p}^N = 1$$

The canonical partition function can also be written as a sum over energies using the microcanonical partition function or density of states,

$$Q(T,V,N) = \int \Omega(E,V,N) e^{-\beta E} dE$$

The distribution of total energies in the canonical ensemble is:

$$\wp(E) = \frac{\Omega(E,V,N) e^{-\beta E}}{Q(T,V,N)}$$

The **mean energy** is given by both

$$\begin{aligned} \langle E \rangle &= \int E \wp(E) dE \\ &= \int H(\mathbf{p}^N, \mathbf{r}^N) \wp(\mathbf{p}^N, \mathbf{r}^N) d\mathbf{p}^N d\mathbf{r}^N \end{aligned}$$

Because the partition function is separable, this can be split into kinetic and configurational parts:

$$\begin{aligned} \langle E \rangle &= \int K(\mathbf{p}^N) \wp(\mathbf{p}^N) d\mathbf{p}^N + \int U(\mathbf{r}^N) \wp(\mathbf{r}^N) d\mathbf{r}^N \\ &= \langle K \rangle + \langle U \rangle \\ &= \frac{3Nk_BT}{2} + \langle U \rangle \end{aligned}$$

At a macroscopic level, the canonical partition function relates to the **Helmholtz free energy**,

$$A(T,V,N) = -k_B T \ln Q(T,V,N)$$

whose macroscopic differential form is given by,

$$dA = -SdT - PdV + \mu dN$$

Monte Carlo simulation in the canonical ensemble

Problem formulation

Once we know the microstate distribution $\wp(\mathbf{p}^N, \mathbf{r}^N)$, we can compute virtually any property of interest. Consider a generic property $X(\mathbf{p}^N, \mathbf{r}^N)$, which could be the potential energy, kinetic energy, local density, coordination number, radius of gyration, etc. The average value of X in any ensemble is given by:

$$\langle X \rangle = \int X(\mathbf{p}^N, \mathbf{r}^N) \wp(\mathbf{p}^N, \mathbf{r}^N) d\mathbf{p}^N d\mathbf{r}^N$$

In other words, we simply average X over every microstate using the microstate distribution function. This general expression applies to any **equilibrium** property, i.e., to any property that

is time-independent. This does not include kinetic transport coefficients, which depend on integrals of time.

How might we compute averages of this sort? We need a way to determine $\wp(\mathbf{p}^N, \mathbf{r}^N)$.

Our earlier considerations show that the microstate probability in the canonical ensemble is separable, and that the kinetic part is analytic. For example, to compute the average total energy:

$$\begin{aligned}\langle E \rangle &= \langle K \rangle + \langle U \rangle \\ &= \frac{3Nk_BT}{2} + \int \wp(\mathbf{r}^N) U(\mathbf{r}^N) d\mathbf{r}^N\end{aligned}$$

Therefore, the real challenge is to develop an expression for the configurational distribution $\wp(\mathbf{r}^N)$. In any case, many bulk properties depend only on the configurational coordinates, and not on the momenta, so we will take as our baseline problem the computation an average of the form:

$$\begin{aligned}\langle X \rangle &= \int X(\mathbf{r}^N) \wp(\mathbf{r}^N) d\mathbf{r}^N \\ &= \frac{\int X(\mathbf{r}^N) e^{-\beta U(\mathbf{r}^N)} d\mathbf{r}^N}{\int e^{-\beta U(\mathbf{r}^N)} d\mathbf{r}^N}\end{aligned}$$

We might think about how we would compute this **configurational average**. We can certainly compute the exponential involving the potential energy, since this simply requires numerical evaluation of the force field for a given configuration $\mathbf{r}^N$. On the other hand, the integrals are problematic:

- There are $3N$ integrand variables, each of which varies between 0 and L (for a cubic volume).
- There are certainly numerical approaches for evaluating integrals, such as the trapezoidal or Simpson methods. However, we have far too many integrand variables for such methods to be practical. We would need to discretize each variable into a finite number of steps between 0 and L. If we only chose the three values for each, $\{0, L/2, L\}$, we would have to evaluate the potential energy 3^{3N} times. For $N = 100$, this would mean $\approx 10^{143}$ evaluations, far exceeding what we can realistically perform.
- It further turns out that these integrals are sharply peaked: there are very small regions of phase space where the energies are very low and the exponential very large. Thus, discrete approximations to these integrals are bound to fail unless we use a very large number of steps.

We seem to be at a statistical deadlock because we cannot evaluate the integrals accurately using a **numerical quadrature** approach of the form

$$\langle X \rangle = \frac{\sum_{x_1} \sum_{y_1} \dots \sum_{z_N} A(\mathbf{r}^N) e^{-\beta U(\mathbf{r}^N)}}{\sum_{x_1} \sum_{y_1} \dots \sum_{z_N} e^{-\beta U(\mathbf{r}^N)}}$$

The underlying challenge with this approach is that we must systematically **discretize configuration space**, an extremely high-dimensional space.

Monte Carlo integration and importance sampling

The basic problem with evaluating a configurational average is that we must perform **high-dimensional integrals.** For such integrals, the Monte Carlo method provides a way to compute these averages and hence is often termed **Monte Carlo integration.**

The idea is the following: rather than discretize all of the integrand coordinates and systematically iterate through different configurations, we instead generate a finite number of configurations according to the probability $\wp(\mathbf{r}^N)$:

$$\langle X \rangle = \frac{1}{n} \sum_{i=1}^{n} X(\mathbf{r}_i^N) \qquad \mathbf{r}_i^N \text{ generated according to } \wp(\mathbf{r}^N)$$

That is, we **pick n random configurations according to** $\wp(\mathbf{r}^N)$ and from this finite set of configurations take the average (unweighted) value of A. In this way, the probability weight $\wp(\mathbf{r}^N)$ is included in the average *implicitly,* by virtue of the fact that we have picked representative conformations according to that probability.

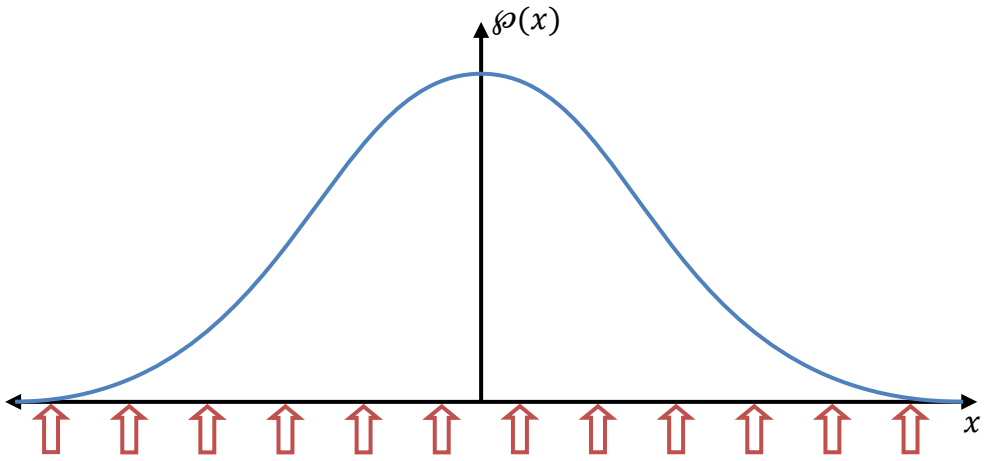
Here is an example that illustrates the difference between the importance sampling approach and systematic discretization. Imagine we have one particle in one dimension whose potential energy function is described by a harmonic potential tethering the particle to the origin:

$$U(x) = kx^2$$

The distribution of particle positions in the canonical ensemble is then Gaussian:

$$\wp(x) = \sqrt{\frac{k\beta}{\pi}} e^{-\beta k x^2}$$

We want to compute the average squared distance from the origin, $\langle x^2 \rangle$. If we were to systematically discretize x into 12 values, we would take 12 configurations similar to the following:



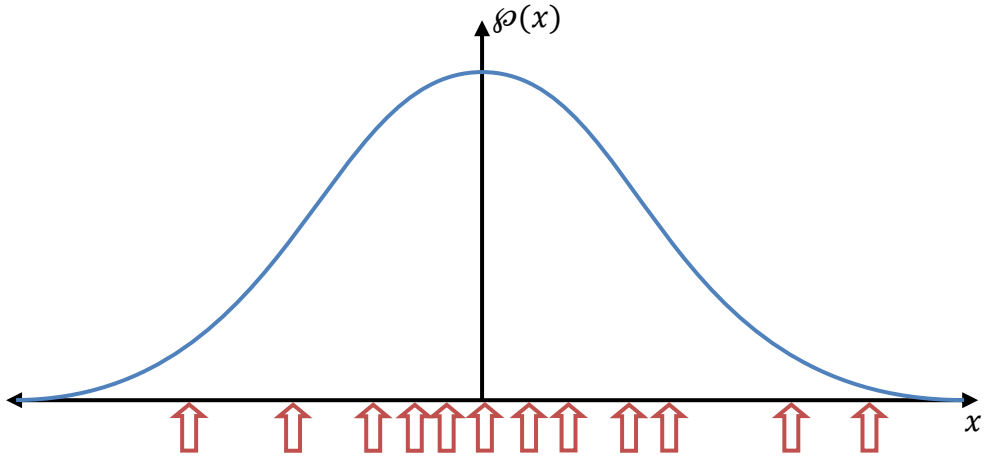
Then we would compute the average squared distance as

$$\langle x^2 \rangle = \frac{\sum_i x_i^2 e^{-\beta k x_i^2}}{\sum_i e^{-\beta k x_i^2}}$$

On the other hand, we could pick 12 configurations randomly with probabilities according to the distribution:

$$\wp(x) \propto e^{-\beta k x^2}$$

This might look something like:



Notice that the configurations are not evenly spaced for two reasons:

- Configurations where $\wp(x)$ is larger are more likely to be chosen.
- Configurations are chosen *randomly*, and not systematically. Thus, we would expect a different collection of configurations each time we performed this procedure.

In this case, our estimated value of $\langle x^2 \rangle$ is

$$\langle x^2 \rangle = \frac{1}{12} \sum_{i=1}^{12} x_i^2$$

You may be concerned that our estimate for $\langle x^2 \rangle$ now involves a stochastic element, in that we could arrive at different results each time we averaged in this way. We can be confident, however, that as we pick more representative configurations, the two methods converge to the same average. For low-dimensional integrals (including this simple example), stochastic integration typically converges slower compared to systematic discretization. However, for high-dimensional integrals of the kind we normally encounter in many-atom systems, stochastic integration typically converges much faster than systematic discretization.

Monte Carlo integration provides us with a way to compute **thermodynamic property averages** of molecular systems: we take a set of configurations generated according to $\wp(\mathbf{r}^N)$ and simply compute the averages from a simple unweighted mean over them. The generation of configurations according to a distribution is called **importance sampling**.

Markov chains

We now have a new challenge: in importance sampling, how do we generate configurations randomly according to $\wp(\mathbf{r}^N)$?

It turns out that we can importance-sample configurations using a statistical construct called a **Markov chain** of states. A Markov chain describes a stochastic process in which the state of a system (here, the instantaneous atomic configuration) changes randomly with time and has no memory of previous states. At each step in time, the system can move randomly to another state (another configuration).

In the context of a Monte Carlo simulation, a Markov chain can be generated using the following basic procedure:

1. At some step i, the system has an atomic configuration (state A).

2. The configuration is *randomly* perturbed to generate a new atomic configuration (state B). A typical perturbation might be a **single-particle displacement:** randomly pick an atom and displace it's x, y, and z location variables by small random amounts. In general, these perturbations are termed **Monte Carlo moves**.

3. The new configuration is considered a **proposed or attempted** new state of the system. Either the proposal is **accepted** (the system moves to the new state) or **rejected** (the system stays where it was before the proposed move is generated). The configuration at

step $i+1$ is then either the proposed configuration if accepted or the original configuration if rejected.

4. The acceptance or rejection of the proposed moves is performed in such a way that configurations are generated according to $\wp(\mathbf{r}^N)$ in the long-time limit.

5. The process is repeated over and over again to generate a trajectory of configurations.

6. The trajectory includes the states of the system **after each acceptance or rejection**. That is, if the proposed move was rejected at step i, then the configuration appearing at step i is also the same as the one at step $i-1$, i.e., it does *not* include the proposed configuration at step i.

In this way, Monte Carlo moves can be considered as methods for imposing artificial, random dynamics on the system that propagate it in time according to pre-specified configurational probabilities. Then, using the computed trajectory, one can perform averages according to the importance-sampling formula:

$$\langle X \rangle = \frac{1}{n_{\text{tot}}} \sum_{i=1}^{n_{\text{tot}}} X(\mathbf{r}_i^N)$$

Here, n_{tot} is the total number of MC steps in the simulation and $\mathbf{r}_i^N$ are the generated configurations.

One might compare a Monte Carlo trajectory of configurations to that generated from a molecular dynamics trajectory:

- Both simulation approaches examine discrete-time trajectories. That is, the changes from state to state in a Markov chain occur at pre-determined discrete intervals. We call the smallest basic interval at which the state can change (or stay the same) to be a **step** or **Monte Carlo step** in the simulation.

- Unlike an MD trajectory, a **Markov chain is stochastic** and is not deterministic.

- Also unlike MD, we **do not consider velocities** in the Markov chain; we only consider configurational degrees of freedom. The main reason we ignore the velocities (and momenta) is that these contributions to the total partition function are known analytically, and thus we do not need to evaluate them via simulation.

An important property of **Markov processes** is that the future state of a system after one step only depends on the current state, and no previous state. That is, it doesn't matter how our

system arrived to its current state; the probabilities that it will move at the next time step to any other state in the future only depend on where it currently is. This is called the **Markov property.**

Detailed balance

In order for this approach to work, we need a way to decide how to accept or reject proposed configurations in our simulations. We need to choose the **acceptance criterion** in such a way that our long-time trajectory correctly generates configurations according to the canonical probabilities $\wp(\mathbf{r}^N)$. We can do this by considering the evolution of state probabilities in our simulation.

For simplicity of notation, let's make the following definitions:

- $\wp_m(t)$ – probability that an instantaneous snapshot of our simulation at step t will be in state / configuration m
- π_{mn} – probability that, given the system is currently in state m, it will move to state n. This is called the **transition probability**.

Although we know that our classical system actually has access to an infinite number of molecular configurations m and n, we will assume for convenience that their number is actually finite and that we can count them. This will not affect our final conclusions and, moreover, the number of configurations *is* actually finite since computers have finite precision.

Since the above quantities are probabilities, we have at any point in time

$$\sum_m \wp_m = 1$$

$$\sum_n \pi_{mn} = 1$$

Here, the sums are performed over all states / configurations available to the system.

Imagine that our system has evolved for some time, randomly according to the Markov chain. Knowing only the initial configuration, we cannot specify the configuration to which the system has since evolved because the dynamics are not deterministic. However, we can characterize the probabilities that we would be in each configuration after many steps. This is the set $\{\wp_m\}$. Ultimately, we want

$$\lim_{t\to\infty} \wp_m(t) = \frac{e^{-\beta U_m}}{Z}$$

That is, we want these probabilities to converge to a **stationary distribution**, independent of time, that is equal to the equilibrium distribution in the canonical ensemble.

Now imagine we move one step forward in the Markov chain from t to $t+1$. How do the probabilities $\wp_m$ change? For a given state m, we have to consider both the decrease in probability associated with the system transitioning out from m to other states n and the increase in probability associated with the system transitioning from other states n into m:

$$\wp_m(t+1) = \wp_m(t) - \sum_n \pi_{mn}\wp_m(t) + \sum_n \pi_{nm}\wp_n(t)$$

If we are at equilibrium, the probabilities cannot change with time and we therefore must have:

$$\sum_n \pi_{mn}\wp_m = \sum_n \pi_{nm}\wp_n \qquad \text{for all } m$$

Here, we have omitted the step dependence to signify that these are equilibrium, time-independent probabilities. This equation is termed the **balance equation**. If our transition and state probabilities obey this equation, then we will be at equilibrium.

One way in which the balance equation can be satisfied is through the **detailed balance** condition:

$$\pi_{mn}\wp_m = \pi_{nm}\wp_n \qquad \text{for all } m, n$$

That is, detailed balance applies a constraint to the transition and state probabilities for every pair of states. It is more specific than the general balance equation. All Markov processes that obey the detailed balance condition automatically obey general balance; however, the reverse is not true.

The detailed balance equation provides us with a way to determine the acceptance criterion in our Monte Carlo simulation. We can write the transition probability as the product of two quantities:

$$\pi_{mn} = \alpha_{mn} P_{mn}^{\text{acc}}$$

Here, α_{mn} is a **move proposal probability.** It gives the probability that we will propose a random move from state m to n. It depends entirely on the kind of Monte Carlo move that we perform. Once we pick a kind of MC move, such as a single particle displacement, this probability is determined.

The quantity P_{mn}^{acc} is the **acceptance probability**. It gives the probability that we should accept a proposed move. This is the critical quantity of interest in MC simulations. We must determine

what acceptance probability to use so that we reproduce the correct importance sampling probabilities. If we impose the detailed balance condition, the acceptance probability should obey:

$$\frac{P_{mn}^{\text{acc}}}{P_{nm}^{\text{acc}}} = \frac{\alpha_{nm}\wp_n}{\alpha_{mn}\wp_m}$$

This equation now gives us a starting point for correctly performing our Monte Carlo simulation. We imposed detailed balance to arrive at it. Detailed balance is not required, but it gives a convenient way to obey general balance in order to reach a stationary distribution.

We can impose this equation at any moment in time in our Markov chain as a way to guarantee that we converge to an equilibrium distribution. Let us be at state $m = 1$ initially. A move is proposed to state $n = 2$. We can then write this expression as:

$$\frac{P_{12}^{\text{acc}}}{P_{21}^{\text{acc}}} = \frac{\alpha_{21}\wp_2}{\alpha_{12}\wp_1}$$

Keep in mind that 1 and 2 correspond to two atomic configurations $\mathbf{r}_1^N$ and $\mathbf{r}_2^N$, respectively. This equation provides us with a way to determine the acceptance criterion given the move type (and hence the α's) and the ensemble (hence the $\wp$'s).

Symmetric moves and the Metropolis criterion

So-called **symmetric Monte Carlo moves** have move proposal probabilities that are equal in the forward and reverse directions. In other words,

$$\alpha_{12} = \alpha_{21}$$

Such is the case for single-particle displacements: the probability for moving from one state to another is either a uniform constant or zero. That is, either we can get from configuration 1 to 2 and vice versa by moving one particle within $\delta r_{\max}$ in each of its position components or we cannot get there at all. There are many kinds of move types in Monte Carlo, and not all of them are symmetric.

If moves are symmetric, we have

$$\frac{P_{12}^{\text{acc}}}{P_{21}^{\text{acc}}} = \frac{\wp_2}{\wp_1}$$

We can also use the fact that the microstate probabilities in the canonical ensemble are

$$\begin{aligned}\wp_m &= \wp(\mathbf{r}^N)d\mathbf{r}^N \\ &= \frac{e^{-\beta U(\mathbf{r}^N)}d\mathbf{r}^N}{Z(T,V,N)}\end{aligned}$$

Here, we need to include the differential configurational volume element because the $\wp_m$ are dimensionless state probabilities. Using this result,

$$\frac{P_{12}^{\text{acc}}}{P_{21}^{\text{acc}}} = \frac{e^{-\beta U_2} d\mathbf{r}^N / Z(T,V,N)}{e^{-\beta U_1} d\mathbf{r}^N / Z(T,V,N)}$$
$$= e^{-\beta(U_2 - U_1)}$$

In the last line, notice that the partition functions cancel since we incur the same probability normalization constant for each configuration in the canonical ensemble. If we had wanted to simulate in another ensemble, we would have used a different expression for the microscopic probability distribution $\wp$.

We still have flexibility in choosing the acceptance criterion. We can choose any procedure that satisfies the above equation. The most frequently used criterion, and typically the most efficient in simulation, is the **Metropolis criterion:**

$$P_{12}^{\text{acc}} = \min\left[1, e^{-\beta(U_2 - U_1)}\right]$$

The min function is incorporated into this criterion. If $U_2 < U_1$, the acceptance probability is always one. Otherwise, it is less than one. Thus, this move specifies that we should always move downhill in energy if we can, an aspect which helps reach equilibration faster in Monte Carlo simulations than alternative criteria.

The rule above must be applied equally to the reverse move. We can thus show that this acceptance criterion satisfies the detailed balance equation:

$$\frac{P_{12}^{\text{acc}}}{P_{21}^{\text{acc}}} = \frac{\min\left[1, e^{-\beta(U_2 - U_1)}\right]}{\min\left[1, e^{-\beta(U_1 - U_2)}\right]}$$
$$= \begin{cases} \dfrac{e^{-\beta(U_2 - U_1)}}{1} & \text{if } U_2 > U_1 \\ \dfrac{1}{e^{-\beta(U_1 - U_2)}} & \text{if } U_1 > U_2 \end{cases}$$
$$= e^{-\beta(U_2 - U_1)}$$

The Metropolis criterion is not a unique solution for the acceptance criterion. One can also use the so-called **symmetric** criterion, although this is not frequently employed because it typically results in longer correlation times and slower equilibration:

$$P_{12}^{\text{acc}} = \frac{e^{-\beta U_2}}{e^{-\beta U_1} + e^{-\beta U_2}}$$

Practical aspects

The above considerations describe the principles underlying the simple Monte Carlo example presented at the beginning of this lecture. Practical considerations are described below.

Computing potential energy and virial changes

In computing the change in potential energy $\Delta U = U_2 - U_1$, we don't actually need to compute the total potential energy. Since this is a system of pairwise interactions and since we only moved one particle, we only need to compute the $N - 1$ interactions of the moved particle with all other particles. This is an important time savings because it means the interaction loop expense scales as N rather than N^2. The general procedure for each MC step is:

1. Pick a random particle i.
2. Compute the current energy of i with all other particles, $U_{1,i} = \sum_{j \neq i} u_{ij}$.
3. Make a random move proposal.
4. Compute the new energy of i with all other particles, $U_{2,i} = \sum_{j \neq i} u_{ij}$.
5. Compute the change in potential energy $\Delta U = U_{2,i} - U_{1,i}$.

One can maintain the total potential energy by doing updates for accepted moves of the form $U \leftarrow U + \Delta U$. Due to precision issues, however, it is a good idea to do a full update of the energy (the entire pairwise loop) once in a while, say every 10-100 or so moves per particle (e.g., every 1000-10000 moves for a 100 particle system). Moreover, it is important not to do the update $U \leftarrow U + \Delta U$ until *after* a move has been accepted; otherwise, a high-energy ΔU from a rejected move could overwhelm U and cause precision errors.

If the pressure is being calculated, the approach above can also be used to compute the change in the virial upon displacement ΔW, which can then be used to update $W \leftarrow W + \Delta W$.

At first, the scaling of the expense with N rather than N^2 may seem to make MC methods much more efficient than the MD approach. However, one must keep in mind that *every* particle moves at each time step in a MD simulation. In MC, only one particle moves with each step. If we compare the expense of one move *per atom* in a MC simulation, we find that it actually scales as N^2, just like MD methods. Therefore there is no particular gain in efficiency. The flexibility to impose MC moves of various kinds, however, may result in faster equilibration than MD techniques.

Equilibration and production periods

Like MD simulations, MC runs require separate equilibration and production periods of simulation. Initial structures can be chosen in a variety of ways, but it is important to allow the system to lose memory of them as it approaches equilibrium. This corresponds to the Markov chain approaching long-time behavior where it reaches the imposed, canonical stationary distribution of states.

What constitutes the MC trajectory

Keep in mind that *every step should be included in running averages, not just the accepted steps.* This is because there is a finite Markov chain probability that we will remain in the same state, i.e., $\pi_{mm} > 0$.

What constitutes the trajectory are the configurations remaining at each step after acceptance or rejection. Consider the following scenario. We start at state 1 and moves are proposed and rejected nine times before finally moving to state 2. For property averages over these ten MC time steps, we should include state 1 nine times and state 2 one time such that state 1 makes a 90% contribution to the average. It would be a mistake to only include state 1 once, such that it made a 50% contribution.

Property averages

As with molecular dynamics techniques, the average of any quantity from a MC trajectory is subject to statistical error according to correlation times. That is, successive configurations in an MC trajectory are not statistically independent, but related by the MC perturbations.

The general formula for the squared error in the mean of a property average A is given by:

$$\sigma_{\bar{A}}^2 = \frac{\sigma_A^2}{t_{\text{tot}}/(2\tau_A + 1)}$$

Here, t_{tot} gives the total number of MC steps used to compute the average $\bar{A}$. τ_A is a correlation time based on the discrete-time MC dynamical trajectory. It is sensitive to the kinds of MC steps performed, and is given by the formula:

$$\tau_A = \sum_{i=1}^{\infty} C_A(i)$$

Here, i is an index for the discrete Monte Carlo steps and $C_A(i)$ is the autocorrelation function for A computed at these intervals. Note that if successive steps are perfectly uncorrelated, $C_A(i \geq 1) = 0$ and $\tau_A = 0$.

Orientational moves

For rigid polyatomic molecules, one needs to perform Monte Carlo moves that make random perturbations to the molecular orientations in addition to the center-of-mass translational degrees of freedom. Typically, translational moves are separated from orientational moves, and one draws a random number to decide which to attempt. A basic procedure is:

1. Pick a random rigid molecule i.
2. Draw a random number r in $[0.0,1.0)$:
    - If $r < 0.5$, then perform a single-molecule displacement move for i.
    - If $r \geq 0.5$, perform an orientational displacement move for i.

There are a number of procedures for orientational displacement. One of the simplest is the following:

1. Pick a random unit vector. This is equivalent to picking a random point on a sphere, and there are a number of simple algorithms for doing this.
2. Pick a random angular displacement $\delta\theta$ in $[-\delta\theta_{\max}, \delta\theta_{\max}]$.
3. Rotate the molecule along the axis of the random vector and about its center of mass by the amount $\delta\theta$.

Another perhaps even simpler approach is:

1. Pick a random coordinate axis, either x, y, or z.
2. Pick a random angular displacement $\delta\theta$ in $[-\delta\theta_{\max}, \delta\theta_{\max}]$.
3. Rotate the molecule along the axis about its center of mass by the amount $\delta\theta$.

Similar to translational displacements, orientational moves also involve a maximum displacement $\delta\theta_{\max}$ that can be tuned to attain acceptance ratios in the range 30-50%.

Orientational moves of these forms are also symmetric, $\alpha_{mn} = \alpha_{nm}$, so that

$$P_{12}^{\text{acc}} = \min\left[1, e^{-\beta(U_2 - U_1)}\right]$$

Random number generators

When we pick a random number in the computer, we are actually picking a **pseudo-random number**. These numbers are not truly random—they follow a specific mathematical sequence

that is ultimately deterministic—but they have statistical properties that are reminiscent of actual random variables.

Random numbers are produced by algorithms that take in a **seed value**, an integer number that is used to begin the sequence of pseudorandom numbers. A **random number generator** will produce the same sequence of random numbers for the same initial seed. One can explicitly specify the seed at the beginning of a simulation, which can be helpful in debugging programs as it produces deterministic behavior. Alternatively, if a seed is not specified, programming languages (like Python) will often create a seed from the current time.

The basic random number generator will produce a random real number from the uniform distribution in the range $[0.0,1.0)$. Random numbers from other distributions, such as the Gaussian or Poisson distributions, can be generated by drawing one or more random numbers from this basic distribution.

Algorithms for random number generation have been the subject of much interest. Some early algorithms were found to be flawed in that they produced sequences of numbers in which subtle patterns and correlations could be detected. Currently a popular random number generator, and the one used by the numpy.random module in Python, is the **Mersenne twister.** This random number generator has quite good statistical properties (sequences of random numbers have low correlation). It also has a **period** of 2^{19937}, meaning that the same random number will not be drawn with a frequency greater than 1 in 2^{19937}.

Reduced units

When working with systems defined by dimensionless or reduced units, such as the monatomic Lennard-Jones system, the temperature is typically defined in units of the energy scale and k_B:

$$T^* = \frac{k_B T}{\epsilon}$$

Thus the acceptance criterion does not involve an explicit k_B because it is already included in the dimensionless temperature:

$$P_{12}^{\text{acc}} = \min\left[1, e^{-\frac{\Delta U^*}{T^*}}\right]$$

Summary

A general approach to any Monte Carlo simulation involves the following steps:

1. **The system potential energy function is determined.** One chooses a functional form for $U(\mathbf{r}^N)$.

2. **The statistical-mechanical ensemble of interest is chosen.** This uniquely specifies the probabilities $\wp_m$with which each microstate m should be sampled. In a classical atomic system, each microstate m corresponds to a set of particle coordinates $\mathbf{r}^N$. In the canonical ensemble, $\wp_m \propto \exp(-\beta U_m)$.

3. **The set of Monte Carlo moves is chosen**. For atomic systems, this might consist of single-particle displacements. For rigid molecules, this might consist of both molecule translational and orientational displacements (two kinds of moves). These moves uniquely specify the move proposal probabilities α_{mn}. For symmetric moves, $\alpha_{mn} = \alpha_{nm}$.

4. **One determines the appropriate acceptance criterion.** Typically we use the Metropolis criterion. The acceptance criterion then follows directly from the relation,

$$P_{12}^{\text{acc}} = \min\left[1, \frac{\wp_2 \alpha_{21}}{\wp_1 \alpha_{12}}\right]$$

5. **A simulation is performed using the determined acceptance criterion.** Equilibration must first be achieved by propagating the system for several relaxation times.

6. **Ensemble property averages are computed from trajectory averages.** The average value of any configurational property in the ensemble of interest then follows from a simple average over the "time"-progression of the production phase of the simulation:

$$\langle A \rangle = \frac{1}{n_{\text{tot}}} \sum_{i=1}^{n_{\text{tot}}} A(\mathbf{r}_i^N)$$