Molecular Dynamics

The Molecular Sciences Software Institute Eliseo Marin-Rimoldi and John D. Chodera

I. INTRODUCTION

Your first assignment consisted on implementing a Monte Carlo (MC) simulation for the Lennard-Jones (LJ) particle in the canonical ensemble. As part of this task, you developed code that computed pair-wise energies and forces for this model, as well as functions that computed interaction energies between molecules, the total system potential energy, and the atomic virial for the estimation of pressure.

The objective of the new task is to extend your code by including the necessary functionality to simulate the NVT ensemble of configurations of the LJ model using the molecular dynamics (MD). You will be assigned to work in teams to develop this project. The team will decide how is best to design the new code: What functionality should be shared between the two methods? What should the library should look like? How can we improve reusability and extensibility? What data structures should be used? Which unit tests should be implemented? The primary learning goal of this activity is to think about interoperability and teamwork.

In the following sections, we illustrate how to implement a Langevin integrator to sample from the NVT ensemble, where our primary focus will be on expectations of configurational properties.

A. Molecular dynamics (MD) and Langevin integrators

Molecular dynamics (MD) is a highly popular technique to simulate both equilibrium and nonequilibrium properties of molecular systems. Here, we focus on the estimation of equilibrium configurational properties $\langle Q \rangle$ of isothermal (NVT) systems at fixed volume. While the MC scheme we implemented earlier is a simple and practical way to estimate configurational properties in simple systems, in more complex systems, it becomes difficult to construct moves that have high acceptance rates because couplings between various degrees of freedom become more complex and nonlinear. Molecular dynamics can circumvent these problems by allowing the instantaneous *forces* to help guide sampling of configuration space, which can follow highly nonlinear regions of high probability—at the expense of requiring potentially many timesteps in order to generate effectively uncorrelated samples from the target equilibrium probability density. If a high-quality integrator that preserves dynamical properties is also used, we can also get additional dynamical properties inaccessible in Monte Carlo simulation, such time-correlation functions, transition rates, and viscosities.

Langevin integrators are widely employed to numerically simulate the equilibrium thermodynamics and kinetics of microscopic molecular systems. The origin of these integrators lies in the *memoryless Langevin equation*? , which describes the behavior of condensed phase systems subject to random weak collisions with fictitious bath particles at thermal equilibrium,

$$d\begin{bmatrix}\mathbf{x}\\ \mathbf{v}\end{bmatrix} = \underbrace{\begin{bmatrix}\mathbf{v}\\ 0\end{bmatrix}}_{\mathsf{R}}\mathrm{d}t + \underbrace{\begin{bmatrix}0\\ -M^{-1}\nabla U(\mathbf{x})\end{bmatrix}}_{\mathsf{V}}\mathrm{d}t + \underbrace{\begin{bmatrix}0\\ -\gamma\mathbf{v}\mathrm{d}t + \sigma M^{1/2}dW\end{bmatrix}}_{\mathsf{O}} \tag{1}$$

Here $\mathbf{x}$ and $\mathbf{v}$ denote the positions and velocities of all particles in the system, t is time, and M the diagonal mass matrix. The constant $\sigma^2 = (2k_BT\gamma)$ quantifies the rate of heat exchange with the bath, with k_BT denoting the thermal energy and γ the collision rate (with dimensions of inverse time), and $W(t)$ is a standard multidimensional Wiener process[1,2].

While this equation in principle has an exact solution to update the positions $\mathbf{x}$ and velocities $\mathbf{v}$ by a timestep Δt,

$$\rho(\mathbf{x},\mathbf{v};t+\Delta t) = e^{-\mathcal{L}\Delta t}\rho(\mathbf{x},\mathbf{v};t) = e^{-(\mathcal{L}_{\mathsf{R}}+\mathcal{L}_{\mathsf{V}}+\mathcal{L}_{\mathsf{O}})\Delta t}\rho(\mathbf{x},\mathbf{v};t) \tag{2}$$

we cannot integrate the whole equation analytically. Instead, we can take the time-evolution operator $\mathcal{L}$ that corresponds to the action of Eq. 1 applied to a probability density and factor it into an *approximate* symmetric product of operators corresponding to parts of Eq. 1 that can be analytically integrated, forming a variety of integrators:

$$e^{-\mathcal{L}\Delta t} \approx \tag{3}$$

$$\approx e^{-\mathcal{L}_{\mathsf{OVRVO}}\Delta t} = e^{-\mathcal{L}_{\mathsf{O}}\frac{\Delta t}{2}}e^{-\mathcal{L}_{\mathsf{V}}\frac{\Delta t}{2}}e^{-\mathcal{L}_{\mathsf{R}}\Delta t}e^{-\mathcal{L}_{\mathsf{V}}\frac{\Delta t}{2}}e^{-\mathcal{L}_{\mathsf{O}}\frac{\Delta t}{2}} \tag{4}$$

$$\approx e^{-\mathcal{L}_{\mathsf{VRORV}}\Delta t} = e^{-\mathcal{L}_{\mathsf{V}}\frac{\Delta t}{2}}e^{-\mathcal{L}_{\mathsf{R}}\frac{\Delta t}{2}}e^{-\mathcal{L}_{\mathsf{O}}\Delta t}e^{-\mathcal{L}_{\mathsf{R}}\frac{\Delta t}{2}}e^{-\mathcal{L}_{\mathsf{V}}\frac{\Delta t}{2}} \tag{5}$$

$$\tag{6}$$

These symmetric splittings are called Trotter splittings, and they possess good numerical properties that have been analyzed in detail[1,2].

The action of these individual operators can be analytically expressed:

$$\mathsf{R}\ :\ \begin{bmatrix}\Delta\mathbf{x}\\ \Delta\mathbf{v}\end{bmatrix} = \underbrace{\begin{bmatrix}\mathbf{v}\\ 0\end{bmatrix}}_{\mathsf{R}} h \tag{7}$$

$$\mathsf{V}\ :\ \begin{bmatrix}\Delta\mathbf{x}\\ \Delta\mathbf{v}\end{bmatrix} = \qquad + \underbrace{\begin{bmatrix}0\\ -m^{-1}\nabla U(\mathbf{x})\end{bmatrix}}_{\mathsf{V}} h \tag{8}$$

$$\mathsf{O}\ :\ \begin{bmatrix}\Delta\mathbf{x}\\ \Delta\mathbf{v}\end{bmatrix} = \qquad\qquad + \underbrace{\begin{bmatrix}0\\ (a_h - 1)\mathbf{v} + b_h\sqrt{\frac{k_B T}{m}}\xi\end{bmatrix}}_{\mathsf{O}} \tag{9}$$

where h is the time interval over which the integrator is applied (either Δt or $\Delta t/2$) for this substep, $a_h \equiv e^{-\gamma h}$, and $b_h = \sqrt{1 - a_h^2}$, and $\xi \sim \mathcal{N}(0, 1)$ is a standard normal variate. You will recognize $-m^{-1}\nabla U$ as the instantaneous acceleration.

Putting this together, we can construct two useful concrete Langevin integrators from these components which possess very different properties (Figure 1):

B. OVRVO

The OVRVO integrator (also known as VVVR[3] or the integrator of Bussi and Parrinello[4]) is likely to be most familiar as the core of the integrator is equivalent to the incredibly popular velocity Verlet integrator of Swope et. al[5]:

$$\begin{aligned}
\mathbf{v}_{t+1/4} &= e^{-\gamma\Delta t/2}\mathbf{v}_t + \sqrt{1 - e^{-\gamma\Delta t}}\sqrt{\frac{k_B T}{m}}\xi_{t+1/4}\\
\mathbf{v}_{t+1/2} &= \mathbf{v}_{t+1/4} + \frac{\Delta t}{2} m^{-1} f(\mathbf{x}_t)\\
\mathbf{x}_{t+1} &= \mathbf{x}_t + \Delta t\, \mathbf{v}_{t+1/2}\\
\mathbf{v}_{t+3/4} &= \mathbf{v}_{t+1/2} + \frac{\Delta t}{2} m^{-1} f(\mathbf{x}_{t+1})\\
\mathbf{v}_{t+1} &= e^{-\gamma\Delta t/2}\mathbf{v}_{t+3/4} + \sqrt{1 - e^{-\gamma\Delta t}}\sqrt{\frac{k_B T}{m}}\xi_{t+3/4}
\end{aligned} \tag{10}$$

The fractional sub-timesteps are simply intermediate values that are convenient for both writing and implementing the integrator. The stochastic variables $\xi_{t+1/4}, \xi_{t+3/4} \sim \mathcal{N}(0, 1)$ are standard normal random variates (of the sort generated by `numpy.random.randn`) and a separate random variable must be used for each degree of freedom for each particle and for each instance of ξ.

C. VRORV

The VRORV integrator (also known as the BAOAB integrator by Leimkuhler and Matthews[6,7]) will be less familiar, but it possesses superior accuracy for configurational properties, so we will make use of it here:

$$\begin{aligned}
\mathbf{v}_{t+1/4} &= \mathbf{v}_t + \frac{\Delta t}{2} m^{-1} f(\mathbf{x}_t) \\
\mathbf{x}_{t+1/2} &= \mathbf{x}_t + \frac{\Delta t}{2} \mathbf{v}_{t+1/4} \\
\mathbf{v}_{t+3/4} &= e^{-\gamma \Delta t} \mathbf{v}_{t+1/4} + \sqrt{1 - e^{-2\gamma \Delta t}} \sqrt{\frac{k_B T}{m}} \xi_{t+1/2} \\
\mathbf{x}_{t+1} &= \mathbf{x}_{t+1/2} + \frac{\Delta t}{2} \mathbf{v}_{t+3/4} \\
\mathbf{v}_{t+1} &= \mathbf{v}_{t+3/4} + \frac{\Delta t}{2} m^{-1} f(\mathbf{x}_{t+1})
\end{aligned} \tag{11}$$

D. Forces and switching functions

Unlike Monte Carlo methods, molecular dynamics integrators require both the the potential energies and forces to be everywhere continuous in order to produce well-behaved dynamics and sampling. Because the potential function you implemented for Monte Carlo simply used a cutoff, this requires we add an additional option to use a switching function to ensure force continuity:

$$U_{\mathrm{sw}}(r) = U_{LJ}(r)\, S(r) \tag{12}$$

where $U_{LJ}(r)$ is the Lennard-Jones potential you implemented earlier,

$$U(r) = 4\epsilon \left[\left(\frac{\sigma}{r} \right)^{12} - \left(\frac{\sigma}{r} \right)^{6} \right] \tag{13}$$

and the switching function $S(r)$ is selected to have a value of unity for $r \in [0, r_{\mathrm{sw}}]$, smoothly interpolates between 1 and 0 for $r \in (r_{\mathrm{sw}}, r_{\mathrm{cut}})$, and is zero for $r \in [r_{\mathrm{cut}}, \infty)$, with continuous first and second derivatives everywhere. One such switching function come from Steinbach and Brooks[8,9]:

$$S(r) = \begin{cases} 1 & r \leq r_{\mathrm{sw}} \\ \frac{(r_{\mathrm{cut}}^2 - r^2)^2 (r_{\mathrm{cut}}^2 + 2r^2 - 3r_{\mathrm{sw}}^2)}{(r_{\mathrm{cut}}^2 - r_{\mathrm{sw}}^2)^3} & r_{\mathrm{sw}} < r < r_{\mathrm{cut}} \\ 0 & r \geq r_{\mathrm{cut}} \end{cases} \tag{14}$$

To compute the force on atom i, we use the chain rule:

$$\begin{aligned} f_k(\mathbf{r}^N) &= -\nabla_{\mathbf{r}_k} \sum_{i<j} U(r_{ij}) = - \sum_{i<j} \nabla_{\mathbf{r}_k} \left[U_{LJ}(r_{ij})\, S(r_{ij}) \right] \\ &= - \sum_{i \neq k} \left\{ \left[\nabla_{\mathbf{r}_k} U_{LJ}(r_{ik}) \right] S(r_{ik}) + U_{LJ}(r_{ik})\, \nabla_{\mathbf{r}_k} S(r_{ik}) \right\} \qquad (15) \end{aligned}$$

The component derivatives are simply given by

$$\begin{aligned} \nabla_{\mathbf{r}_k} U_{LJ}(r_{ik}) &= \left. \frac{\partial U_{LJ}(r)}{\partial r} \right|_{r_{ik}} \nabla_{\mathbf{r}_k} r_{ik} && (16) \\ &= 4\epsilon \left[-12 \sigma^{12} r_{ik}^{-13} + 6 \sigma^6 r_{ik}^{-7} \right] \nabla_{\mathbf{r}_k} ||\mathbf{r}_k - \mathbf{r}_i||_2 && (17) \\ &= - \frac{48\epsilon}{r_{ik}} \left[\left(\frac{\sigma}{r_{ij}} \right)^{12} - \frac{1}{2} \left(\frac{\sigma}{r_{ik}} \right)^6 \right] \left[\frac{\mathbf{r}_k - \mathbf{r}_i}{r_{ik}} \right] && (18) \\ &= -48\epsilon \left[\left(\frac{\sigma}{r_{ik}} \right)^{12} - \frac{1}{2} \left(\frac{\sigma}{r_{ik}} \right)^6 \right] \left[\frac{\mathbf{r}_k - \mathbf{r}_i}{r_{ik}^2} \right] && (19) \end{aligned}$$

and

$$\begin{aligned} \nabla_{\mathbf{r}_k} S(r_{ik}) &= \left. \frac{\partial S(r)}{\partial r} \right|_{r_{ik}} \nabla_{\mathbf{r}_k} r_{ik} && (20) \\ &= \frac{(12 r_{ik})(r_{ik}^2 - r_{\mathrm{cut}}^2)(r_{ik}^2 - r_{\mathrm{sw}}^2)}{(r_{\mathrm{cut}}^2 - r_{\mathrm{sw}}^2)^3} \left[\frac{\mathbf{r}_k - \mathbf{r}_i}{r_{ik}} \right] && (21) \\ &= \frac{12(r_{ik}^2 - r_{\mathrm{cut}}^2)(r_{ik}^2 - r_{\mathrm{sw}}^2)}{(r_{\mathrm{cut}}^2 - r_{\mathrm{sw}}^2)^3} \left[\mathbf{r}_k - \mathbf{r}_i \right] && (22) \end{aligned}$$

E. Technical considerations

Choice of timestep. Unlike Metropolis Monte Carlo, molecular dynamics methods do not sample exactly from the target equilibrium density $\pi(\mathbf{x}, \mathbf{v})$, but instead sample from a density $\rho_{\Delta t}(\mathbf{x}, \mathbf{v})$ that increasingly deviates from the target equilibrium density as the timestep Δt increases (Figure 1). Using a timestep that is too large will result in large errors in the sampled density $\rho_{\Delta t}(\mathbf{x})$ or, worse yet, unstable dynamics that might cause the simulation to "explode" and rapidly accumulate NaNs.

Traditional guidelines suggest restricting your timestep to be no bigger than $\frac{1}{10}$ the fastest vibrational frequency in the system. For a pair of Lennard-Jones particles separated by $r_{\min} = 2^{1/6}\sigma$ where their interaction is minimized, the vibrational period for small oscillations about the minimum is given by

$$T \approx 2\pi \sqrt{\frac{\mu}{K}} \qquad (23)$$

Here, the effective spring constant $K = \left.\frac{\partial^2 U_{LJ}}{\partial r^2}\right|_{r_{\min}} = 72\epsilon/r_{\min}^2$ and the effective mass $\mu = (m^{-1} + m^{-1})^{-1} = m/2$ give a period of

$$T \approx 2\pi\sqrt{\frac{2^{1/3}m\sigma^2}{144\epsilon}} \approx 0.09354\sqrt{\frac{m\sigma^2}{\epsilon}} \tag{24}$$

suggesting we should use a maximum timestep of $\Delta t \leq T/10$. More detailed discussion of the timestep-induced error can be found in Leimkuhler and Matthews[1] and work that uses nonequilibrium approaches to quantify this error[3].

Choice of collision rate γ. The Langevin integrator collision rate γ has three different regimes:

- If $\gamma = 0$, standard Newtonian dynamics is recovered, and no thermal control is applied. This is sometimes a good way to check that the code is working correctly, though energy is never conserved exactly due to the use of a finite-timestep.
- If γ is small ($\gamma \ll 1/\Delta t$), the dynamics is *underdamped.* This is good for ensuring some thermal control while not hindering diffusion through phase space.
- If γ is large ($\gamma \gtrapprox 1/\Delta t$), the dynamics is *overdamped.* Velocities will decorrelate very rapidly, and the system will behave in a very Brownian fashion. This is useful for initial equilibration if large initial forces would initially cause rapid heating in the underdamped limit.

For this exercise, we suggest using a collision in the underdamped limit, such as $\gamma \sim (500\Delta t)^{-1}$.

Initial positions. For molecular dynamics simulations, the main consideration is that the initial forces must not be so large that the system rapidly heats up and becomes numerically unstable. Standard biomolecular systems must generally be subjected to energy minimization to produce forces small enough to run molecular dynamics, but several options are available for the simple Lennard-Jones system considered here. While placing atoms randomly is more dangerous as core overlaps might exist, leading to huge repulsion forces, as in the Monte Carlo example, it is possible to use a crystalline lattice as an initial configuration, since this configuration will generally have relatively small forces. You could even first relax the system using the translation moves of your MC code, which only use energy differences forces for sampling and hence will not become numerically unstable due to large forces; a sampled configuration generated from your MC code could be used to initialize dynamics.

Initial velocities. Initial velocities can be generated from the Maxwell-Boltzmann distribution at the desired temperature

$$\rho\left(\mathbf{v}_i\right) = \left(\frac{m_i}{2\pi k_B T}\right)^{1/2} \exp\left[-\frac{m_i \mathbf{v}_i^2}{2k_B T}\right] \tag{25}$$

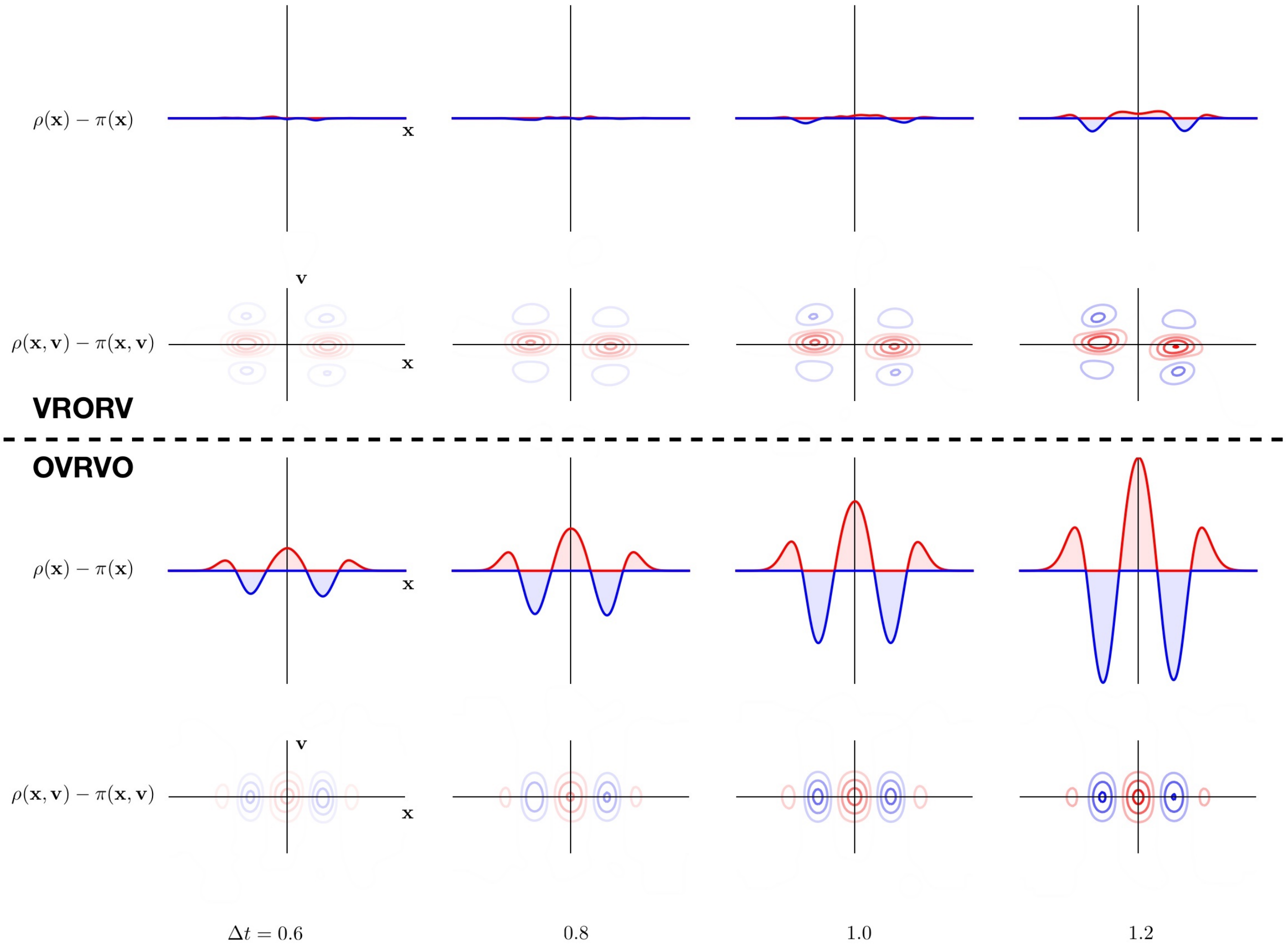

FIG. 1. **Different Langevin integrator splittings lead to different timestep-dependent errors in the sampled configuration and phase space densities.** The quantity $\rho(\mathbf{x},\mathbf{v}) - \pi(\mathbf{x},\mathbf{v})$ measures the deviation of the sampled phase space density $\rho_{\Delta t}(\mathbf{x},\mathbf{v})$ from the target equilibrium density $\pi(\mathbf{x},\mathbf{v})$ as a function of timestep for a quartic potential $U(x) = x^4$. The marginal error in configuration space only, $\rho(\mathbf{x}) - \pi(\mathbf{x})$, is the one-dimensional projection of this error. Two different Langevin integrators derived from different Trotter splittings are shown: VRORV (also known as the BAOAB integrator by Leimkuhler and Matthews[6,7]) and OVRVO (also known as VVVR[3] or the integrator of Bussi and Parrinello[4]). Integrators based on different splittings can lead to very different error properties in positions and velocities. We recommend using VRORV if accurate sampling configurations is necessary (e.g. for computing configurational properties) and OVRVO if accurate sampling of velocities is necessary (e.g. for velocity correlation functions or kinetic properties).

This can be done by sampling the velocities for each particle from a Gaussian distribution

$$\mathbf{v}_i \sim \mathcal{N}(0, \sigma_i^2) \tag{26}$$

where the variance σ_i^2 is given by

$$\sigma_v^2 = \frac{k_B T}{m_i} \tag{27}$$

Long-range dispersion correction. Because a switch function is now employed, we must use a modified form of the long-range dispersion correction that accounts for the switching off of the potential over $r \in [r_{\text{sw}}, r_{\text{cut}}]$. The simplest way to implement this is to numerically compute the long-range correction via

$$\begin{aligned} U_{\text{tail,sw}} &= N\frac{N}{V}\int_{r_{\text{sw}}}^{\infty} 4\pi r^2 \, dr \, U_{LJ}(r)[1 - S(r)] && (28) \\ &= U_{\text{tail,cut}} + \frac{N^2}{V}\int_{r_{\text{sw}}}^{r_{\text{cut}}} 4\pi r^2 \, dr \, U_{LJ}(r)[1 - S(r)] && (29) \end{aligned}$$

where the integral is performed using numerical quadrature, such as the `scipy.integrate.quad`, and $U_{\text{tail,cut}}$ is the cutoff-based long-range correction implemented for the MC code.

Periodic boundary conditions and minimum image distance. As in the MC case, your MD code should also implement these features.

Temperature. The instantaneous temperature in a simulation can be computed as

$$T = \frac{2K}{N_{DOF} k_B} \tag{30}$$

F. Concrete tasks

1. **Extend potential and force computation to support the switching function.** You will want to implement appropriate tests, but the choice of which tests are appropriate is up to you. One important feature you may want to test is the consistency between the potential and force, since this is essential to obtaining correct results for molecular dynamics simulations; this may utilize numerical tests of derivatives obtained via finite-difference or automatic differentiation schemes such as `autograd`.

2. **Implement the integrator and a simulation driver harness.** You will want to implement the integrator and appropriate tests, as well as a small example program that lets you run a simple system and compute averages.

3. **Compute expectations of the reduced potential energy and pressure of a Lennard-Jones fluid at $N = 500$, $T^* = 0.90$, $\rho^* = 0.001$ and 0.9, $r_c = 3\sigma$.** You can compare these results to your Monte Carlo code.

4. **Stretch goal: Implement a Monte Carlo barostat.** A Monte Carlo barostat is a simple way to implement pressure control and can be used with both your MC and MD codes. The move proceeds in a similar fashion to the Metropolis Monte Carlo algorithm for perturbing positions, and proceeds in the following way:

   - Compute the current box volume V_m and potential energy U_m
   - Propose a new box volume $V_n \sim \mathcal{N}(V_m, \sigma_V^2)$, where σ_V is a proposed box volume perturbation size. You can initially set this to $0.01V$, but you will need to tune this during equilibration to achieve good acceptance rates.
   - Immediately reject the move if the proposed box size $V_n \leq 0$; otherwise compute a scaling factor $s = (V_m/V_n)^{1/3}$ that will be used to scale all molecular centers in the box—which in this case is simply equal to N, the number of atoms.
   - Compute proposed box edge lengths $l_n = s l_m$ and proposed particle positions $\mathbf{r}_n^N = s\mathbf{r}_m^N$, and compute the new potential energy $U_n = U(\mathbf{r}_m^N; l_m)$ with the updated positions and box edges.
   - Accept or reject according to the Metropolis-Hastings criteria:

$$P_{acc}(m \to n) = \min\left[1, \frac{\alpha\,(n \to m)\,\rho_n\left(\mathbf{r}^N\right)}{\alpha\,(m \to n)\,\rho_m\left(\mathbf{r}^N\right)}\right] \tag{31}$$

$$= \min\left[1, \frac{(1/V_m)^N}{(1/V_n)^N} e^{-\beta(U_m - U_n)}\right] \tag{32}$$

$$= \min\left[1, \left(\frac{V_n}{V_m}\right)^N e^{-\beta\Delta U}\right] \tag{33}$$

   where the ratio of volumes is the ratio of conditional probability densities of the old vs new states because the configuration space probability density per particle has been compressed by scaling the molecular centers. Note that there are small but important differences in atomic- vs molecular-scaling Monte Carlo barostats and their implementations[10]. It is generally applied only every $n \sim$ 25–100 steps since it requires two energy calculations per move. You can then compute the average *box volume* over a simulation in the isothermal-isobaric (NPT) ensemble and compare this to the average *pressure* of a simulation in the isothermal (NVT) ensemble at the same fixed volume; for example, a simulation at reduced temperature $T^* = 0.9$ with $\rho^* = 0.9$ in the NVT ensemble should give an average reduced pressure $\langle p^* \rangle = 2.585(9)$ while an NPT simulation at $p^* = 2.585$ should give an average reduced density of $\langle \rho^* \rangle \approx 0.9$.

REFERENCES

[1] B. Leimkuhler and C. Matthews, *Molecular Dynamics: With Deterministic and Stochastic Numerical Methods*, Interdisciplinary Applied Mathematics, Springer International Publishing, 2015.

[2] T. Lelivre, G. Stoltz, and M. Rousset, *Free Energy Computations: A Mathematical Perspective*, Imperial College Press, 2010.

[3] D. A. Sivak, J. D. Chodera, and G. E. Crooks, Phys. Rev. X **3**, 011007 (2013).

[4] G. Bussi and M. Parrinello, Phys. Rev. E **75**, 056707 (2007).

[5] W. C. Swope, H. C. Andersen, P. H. Berens, and K. R. Wilson, The Journal of Chemical Physics **76**, 637 (1982).

[6] B. Leimkuhler and C. Matthews, Proceedings of the Royal Society of London A: Mathematical, Physical and Engineering Sciences **472** (2016).

[7] B. Leimkuhler and C. Matthews, The Journal of Chemical Physics **138**, 174102 (2013).

[8] P. J. Steinbach and B. R. Brooks, Journal of Computational Chemistry **15**, 667 (1994).

[9] M. R. Shirts, J. W. Pitera, W. C. Swope, and V. S. Pande, The Journal of Chemical Physics **119**, 5740 (2003).

[10] K.-H. Chow and D. M. Ferguson, Computer Physics Communications **91**, 283 (1995).