Introduction to molecular simulation

Petr Šulc[*], Jonathan P. K. Doye[†], Ard A. Louis[‡]

1 Introduction

Computer simulations have become an indispensable tool in many areas of science. They can be used to study systems that cannot be solved analytically. They can be employed as "computer experiments" to test theories or to generate new theoretical concepts. They also permit access to levels of detail that are often not accessible to experiments. For example, one has complete control over the initial conditions and can track movements of all particles at all times. A domain where computer simulations have historically enjoyed much success is the study of the properties of materials. Indeed, simulations of crystals and liquids were among the first applications of computer simulation techniques. Molecular dynamics simulations of biomolecular systems – including applications to a wide range of protein properties, such as details of molecular conformations, transmembrane transport, and protein-ligand binding – are rapidly growing in importance [5].

Here, we will provide an introduction to the basics techniques and theory behind computer simulations. More complete descriptions of background theory and key algorithms can be found for instance in [6, 1, 9, 11, 14]. Statistical mechanics is crucial for understanding the theory and techniques behind molecular simulations, and some background knowledge is assumed in this chapter. Many textbooks cover this topic, including [4, 15].

Rapid increases in computer processing power, the emergence of new hardware architectures and simulation algorithms that allow massive parallelization, as well as the development of novel simulation algorithms, together with software packages that facilitate their use have greatly increased the complexity of systems that can be simulated. However, the more difficult the studied system, the more dangerous it becomes to treat simulation software just as a black box. Without deeper understanding of the algorithms and methods employed, it is easy to fool oneself.

Our aim is to provide a bit a better appreciation of what is going under the hood when such a simulation package is employed. What are the approximations used? How do these impose limits on validity and applicability? How do I know whether I can extract real physical insight, or have created nothing but a pretty, but potentially misleading, picture or movie? Such questions must never be left aside if one wants to properly distinguish bon fide predictions of real system behavior from simulation artifacts.

The most common techniques employed to study thermodynamic properties of systems of particles are Molecular dynamics and Monte Carlo algorithms, which we briefly review. We then discuss some common techniques for speeding-up simulations and overcoming free-energy barriers, and finally provide a list of some of the popular simulation packages for biomolecular systems.

2 Molecular dynamics

Molecular dynamics (MD) simulations fundamentally just solve Newton's laws of motion for a set of classical interacting particles. This process generates time trajectories of particles in the simulated system that can be visualized and used to measure non-equilibrium properties such as transport coefficients or equilibrium thermodynamic ensemble averages. For example, to measure the mean value of a quantity A of interest (such as pressure, distance or binding energy between two molecules), one first records its value A_k at different times t_k, from which an average can be extracted:

$$\overline{A} = \frac{1}{M} \sum_{k=1}^{M} A_k \tag{1}$$

where M is the total number of measurements. $\overline{A}$ is then the simulation estimate of $\langle A \rangle$, the mean value of quantity A in the thermodynamic equilibrium.

[*]Center for Studies in Physics and Biology, Rockefeller University, New York, NY 10065, USA
[†]Department of Chemistry, University of Oxford, Oxford, OX1 3QZ, UK
[‡]Rudolf Peierls Centre for Theoretical Physics, University of Oxford, Oxford, OX1 3NP, UK

To generate time trajectories for use in averages such as Eq. (1), we first introduce a common integration scheme for Newton's equations in section 2.1. Newton's equations conserve energy, so that temperature fluctuates, while many experimental systems are at constant temperature. We therefore show how to couple the system to a heat-bath for constant temperature T in section 2.2.

2.1 Integrating the equations of motion

The particles in an MD system may represent for example a molecule of gas, connected individual atoms in a protein or some larger-scale coarse-grained entity. The interactions between particles are specified by the interaction potential function $V(\mathbf{r}_1, \mathbf{r}_2, \ldots, \mathbf{r}_N)$, which is uniquely determined by their positions. We will use the symbol $\mathbf{r}^N$ to refer to a set of positions of all N particles. Often, we deal with systems that only contain pairwise interaction between particles and the interaction potential can be simplified as a sum between interacting pairs:

$$V\left(\mathbf{r}^N\right) = \sum_{i<j}^{N} v_{ij}\left(\mathbf{r}_{ij}\right), \tag{2}$$

where v_{ij} is the pairwise potential function and $\mathbf{r}_{ij}$ is the distance between particles i and j.

The dynamics of the system is determined by Newton's equations of motion:

$$m\dot{\mathbf{v}}_i = \mathbf{f}_i = -\frac{\partial V}{\partial \mathbf{r}_i} \tag{3}$$

where $\mathbf{v}_i$, $\mathbf{r}_i$ and $\mathbf{f}_i$ are velocity, position and force acting on the i-th particle respectively. To obtain the trajectory of the particles we need to integrate Eq. (3). A simple integration scheme that is widely used for molecular dynamics simulation is called the Verlet algorithm. The simulation time is discretized into time steps of equal length Δt. Taylor expansion of the positions at times $t + \Delta t$ and $t - \Delta t$ gives

$$\mathbf{r}_i(t+\Delta t) = \mathbf{r}_i(t) + \mathbf{v}_i(t)\Delta t + \frac{\mathbf{f}_i(t)}{2m}\Delta t^2 + \frac{\Delta t^3}{6}\dddot{\mathbf{r}} + O(\Delta t^4) \tag{4}$$

$$\mathbf{r}_i(t-\Delta t) = \mathbf{r}_i(t) - \mathbf{v}_i(t)\Delta t + \frac{\mathbf{f}_i(t)}{2m}\Delta t^2 - \frac{\Delta t^3}{6}\dddot{\mathbf{r}} + O(\Delta t^4). \tag{5}$$

Adding (4) and (5), we obtain

$$\mathbf{r}(t+\Delta t) = 2\mathbf{r}_i(t) - \mathbf{r}_i(t-\Delta t) + \frac{\mathbf{f}_i(t)}{m}\Delta t^2 + O(\Delta t^4) \tag{6}$$

for the position update. Note that the expression for positions of particles are precise up to the $O(\Delta t^4)$, because the terms proportional to Δt^3 canceled out. We further obtain velocity by subtracting (4) from (5) and rearranging the terms:

$$\mathbf{v}_i(t) = \frac{1}{2\Delta t}\left(\mathbf{r}_i(t+\Delta t) - \mathbf{r}_i(t-\Delta t)\right) + O(\Delta t^3). \tag{7}$$

The integration scheme in Eqs. (6), (7) can be also equivalently expressed [6] as so-called velocity Verlet scheme

$$\mathbf{r}_i(t+\Delta t) = \mathbf{r}_i(t) + \mathbf{v}_i(t)\Delta t + \frac{\mathbf{f}_i(t)}{2m}\Delta t^2 \tag{8}$$

$$\mathbf{v}_i(t+\Delta t) = \mathbf{v}_i(t) + \frac{\mathbf{f}_i(t) + \mathbf{f}_i(t+\Delta t)}{2m}\Delta t, \tag{9}$$

which provides the values for position and velocity for each time step, whereas in previous formulation in Eq. (7) it was necessary to evaluate positions both at times t and $t + \Delta t$ to obtain the velocity at t. Other integrators can be found in the literature, but the principles are similar [6].

The appropriate choice of step size Δt will depend on the particular form of interaction potential between particles. Choosing Δt too large will lead to large differences in the forces at adjacent time-steps, and generate artifacts because the integrator does not converge. Typically, the steeper the potential or the higher the temperature (and mean square velocities) the smaller the maximal time-step. On the other hand, picking too small a time-step will unnecessarily slow down the simulation, which could mean that the time trajectories are too short to generate meaningful time averages.

Despite its relative simplicity, the Verlet scheme remains a popular choice for MD simulations. Higher-order integration schemes are useful for reproducing a single trajectory as accurately as possible for given initial condition, like when navigating a space probe through asteroid belt. But for sampling thermodynamic properties of a large molecular system, the Verlet integration has several important properties that make it

a good algorithm. It only requires evaluation of the first derivative of the potential function V, which makes it very efficient, as the force calculation is typically the most time-consuming part of simulation. Algorithms that use second and higher derivatives of the potential allow for greater precision, offering the possibility to use larger time-step size. However, the increase in a step size is relatively minor and does not compensate for the computational cost of evaluating higher order derivatives of potential.

Molecular simulations are chaotic systems, where the trajectories quickly diverge if they start from slightly different initial conditions. But even though the trajectories generated by the Verlet algorithm diverge from the exact solution of the equations of motion, it conserves energy over much longer time-scales. As opposed to several higher-order, more precise integration schemes, Verlet integration can be shown to rigorously preserve the volume of the phase space, a necessary (but not sufficient) condition to avoid long-term energy drift during the course of the simulation [6].

2.2 Controlling the temperature

The numerical integrator described in section 2.1 allows for simulation at fixed number of particles (N), volume (V) and energy (E), *i.e.* the NVE ensemble. However, we are usually interested in studying the properties of a system at a constant temperature T, where energy fluctuates. Such a system is called an NVT ensemble (also called the canonical ensemble in statistical mechanics). Algorithms designed to couple MD simulations to heat bath at temperature T are called thermostats. Some of the popular ones include:

- **The Andersen thermostat** is relatively simple. It couples the simulated system with a heat bath via random "collisions." In each MD simulation step, a particle undergoes a "collision" with a probability ν, in which case its velocity vector components are set to randomly sampled values from a Maxwell-Boltzmann distribution at temperature T. Their dynamics is strongly perturbed on time-scales shorter than the mean time between collisions $1/\nu$, but on longer time-scales this scheme provides a good approximation of diffusive dynamics at a fixed temperature T, and so approximates the canonical ensemble in some cases. It also has the virtue of being straightforward to implement.

- **Berendsen thermostat** uses the kinetic energy at a given simulation time t to define an instantaneous temperature $T'(t)$ as

$$\sum_{i=1}^{N} m_i v_i^2 = N_f k_{\mathrm{B}} T'(t) \tag{10}$$

  where N_f represents the number of degrees of freedom ($N_f = 3N - 3$ for N free particles with fixed total momentum). If the instantaneous temperature is not equal to the temperature T at which we want to simulate our ensemble, the Berendsen thermostat rescales at each time step the velocity of each particle i as $\mathbf{v}_i \to \lambda \mathbf{v}_i$. One could choose λ so that $T'(t) = T$ in each step, but such a simulation would not be representative of a canonical ensemble, where kinetic energy is not constant and fluctuates around its mean value. Therefore, the Berendsen thermostat sets λ so that the change in instantaneous temperature satisfies

$$\Delta T'(t) = \frac{\Delta t}{\tau} \left(T - T'(t)\right) \tag{11}$$

  where τ determines the coupling to the heat bath. Hence we need to rescale the velocities in each step with a factor

$$\lambda = \sqrt{1 + \frac{\Delta t \left(T - T'(t)\right)}{\tau T'(t)}}, \tag{12}$$

  where $T'(t)$ is updated according to Eq. (11). Although Berendsen thermostat does not exactly reproduce the canonical ensemble, it is suitable for initialization and preparation of a system in a desired temperature T.

- **The Nosé-Hoover thermostat** introduces an extra coordinate into the system that has the function of coupling the system to a thermostat. The equations of motion change to

$$m\dot{\mathbf{v}}_i = \mathbf{f}_i - \xi m \mathbf{v}_i \tag{13}$$

$$Q\dot{\xi} = 2\sum_{i}^{N} m\mathbf{v}_i^2 - k_{\mathrm{B}} T N_f, \tag{14}$$

  where N_f is the number of degrees of freedom, ξ is the newly introduced friction thermodynamic coefficient and Q is a parameter that sets the strength of the coupling to the thermostat. It can be shown that the microcanonical partition function of the extended ensemble with variable ξ is proportional to the canonical partition function of a system of N particles at temperature T [6]. Integrating Eqs. (13), (14) can hence be

used to obtain correct thermodynamic averages of chosen observables. Note that the dynamics generated by Nosé-Hoover scheme is deterministic, as the formulation does not include any stochastic random terms. For certain systems, this thermostat may fail to be ergodic and correctly sample the canonical function. This can be resolved by introducing further auxiliary coordinates and constructing so-called Nosé-Hoover Chains [6].

- **Langevin dynamics** evolves the system according to the equation

$$m\dot{\mathbf{v}}_i = \mathbf{f}_i - \gamma \mathbf{v}_i + \mathbf{F}_R^i(t). \tag{15}$$

Two new terms, the friction term $\gamma \mathbf{v}_i$ and a random force $\mathbf{F}_R^i(t)$ respectively, are introduced on the right hand side with respect to Eq. (3). These terms mimic the effect of the collisions with the molecules of a solvent that are not simulated explicitly. Thus this method can be viewed not just as a thermostat, but also as the approximate equations of motion for an object in a background solution. The random force is represented as white noise:

$$\langle \mathbf{F}(t)_{R\alpha}^i \rangle = 0 \tag{16}$$

$$\langle \mathbf{F}_{R\alpha}^i(t) \mathbf{F}_{R\beta}^j(t') \rangle = 2\gamma k_\mathrm{B} T \delta_{ij} \delta_{\alpha\beta} \delta(t - t'), \tag{17}$$

where α and β are components of the vector $\mathbf{F}(t)_R^i$. The damping coefficient γ depends on the shape and size of the simulated particle, which is usually assumed to be spherical. Integration schemes, similar to Verlet algorithm, can be derived for Eq. (15), which also include the damping (velocity-dependent) force $\gamma \mathbf{v}_i$ and the random force $\mathbf{F}_R^i(t)$ generated at every step. The random force means that Langevin dynamics is fundamentally stochastic (as is for example the Andersen thermostat).

The list of thermostats above is certainly not exhaustive, but covers some of the popular choices offered in simulation software. The stochastic Andersen and Langevin thermostats are often used for coarse-grained simulations where the solvent is not simulated explicitly, but its effects are reproduced by the thermostat. The dynamics is however perturbed due to the stochastic nature of the thermostats. Because relative momentum is not conserved, these thermostats all destroy hydrodynamic correlations (e.g. on time-scales longer than $1/\nu$ in the Andersen thermostat). Other techniques must be used to ensure correct hydrodynamic interactions [13].

Finally, we note that it is often of interest to simulate a system with constant number of particles (N), temperature (T) and pressure (P), the so called NPT ensemble. In such a simulation, the volume is dynamically adjusted during the course of the simulation to ensure the desired pressure. Algorithms that perform such scaling are called barostats. Some of the examples include Nosé-Hoover and Berendsen barostat, which are based on similar principles to the corresponding thermostats, except that they are dynamically rescaling the unit cell volume in the simulation to ensure a desired pressure.

3 Monte Carlo Sampling

Another popular method for sampling the equilibrium properties of a molecular system is the Monte Carlo (MC) algorithm. While the molecular dynamics simulations solve equations of motion, MC simulations of molecular systems propose random moves (translation or rotation) of a particle, which are either rejected or accepted with a given probability p, set in a way that ensures correct sampling from a desired ensemble (typically NPT or NVT).

3.1 Metropolis Monte Carlo

Arguably the most widely used MC algorithms for materials is the Metropolis Monte Carlo algorithm:

1. Select a particle at random and apply to it either a random translation vector or a random rotation.
2. Evaluate the interaction energy $U(\mathbf{r}_\mathrm{new}^N)$ after the particle has been displaced, described as a change in position/orientation to $\mathbf{r}_\mathrm{new}^N$ from $\mathbf{r}_\mathrm{old}^N$, the original configuration of all the particles.
3. Accept the transition from the old to the new configuration with probability

$$p(o \to n) = \min \left[1, \exp \left(\frac{U(\mathbf{r}_\mathrm{old}^N) - U(\mathbf{r}_\mathrm{new}^N)}{k_\mathrm{B} T} \right) \right]. \tag{18}$$

Note that moves are always accepted if they lower the energy U, while steps that increase the energy are accepted with a Boltzmann weight of the difference in energy. These "uphill" steps allow the system to exit local minima and efficiently sample the phase space. If a proposed move is rejected and the system remains in the same configuration, it will still be counted for ensemble averages.

A sufficient condition that guarantees that the MC algorithm samples from a canonical ensemble is that it satisfies the detailed balance condition:

$$P(o)\pi(o \to n) = P(n)\pi(n \to o) \tag{19}$$

where $P(o)$ and $P(n)$ are the probabilities of the configuration $o = \mathbf{r}^N_{\text{old}}$ and $n = \mathbf{r}^N_{\text{new}}$ respectively appearing in the ensemble in the first place, while $\pi(o \to n)$ denotes the probability that if the system is in configuration o, it moves to n in the MC simulation (and analogously for $\pi(n \to o)$). It can be split into two terms

$$\pi(o \to n) = \alpha(o \to n)p(o \to n),$$

where $\alpha(o \to n)$ is the probability of suggesting a trial move from o to n, and $p(o \to n)$ is the probability of accepting such a proposed move. In the Metropolis scheme outlined above, we have

$$\alpha(o \to n) = \alpha(n \to o), \tag{20}$$

the probability of proposing a move is symmetric in the direction. However, for the canonical ensemble we require that the respective probabilities of configurations are given by

$$P(o) = \exp\left(-\frac{U(\mathbf{r}^N_{\text{old}})}{k_{\text{B}}T}\right) \quad P(n) = \exp\left(-\frac{U(\mathbf{r}^N_{\text{new}})}{k_{\text{B}}T}\right) \tag{21}$$

Inserting (21) and (20) into the detailed balance condition (19), we obtain criterion

$$\frac{p(o \to n)}{p(n \to o)} = \exp\left(\frac{U(\mathbf{r}^N_{\text{old}}) - U(\mathbf{r}^N_{\text{new}})}{k_{\text{B}}T}\right), \tag{22}$$

which is satisfied by the Metropolis acceptance criterion in Eq. (18).

Since we only move one particle in each step, it is convenient to denote N moves in an N-particle system as one "sweep." By reporting the number of sweeps rather than number of individual steps, data can be more easily compared between different system sizes.

The maximum size of the generated random translation or rotation are parameters whose optimal value depends on particular system of interest. Setting the maximum move size too large will likely result in the majority of moves being rejected, as a random trial move is likely to break too many particle interactions. Too small a maximum move size will slow down the time necessary to equilibrate system and will not sample efficiently. As a rule of thumb, one typically aims for about $\approx 20 - 50\%$ acceptance rate [6].

3.2 Using Monte Carlo for Molecular Simulation

MC algorithms can be used to study equilibrium properties of the system which do not depend on the velocity. Similarly as in the case of MD (Eq. (1)), we can estimate the average of a quantity of interest by recording their values for the configurations sampled by the MC algorithm.

For large biomolecular systems (such as a fully atomistic simulation of a molecule with solvent), the MD simulations described in section 2 remain the preferred choice. This is in part because they are more suitable for acceleration by parallelization on multiple processors and in part because MD is better for studying dynamical properties. MC is furthermore not very efficient for tightly bound molecules (such as strongly interacting particles in an off-lattice polymer), where large moves cannot be proposed, because they would likely break the strong interaction and be rejected. MC can be however used for the relaxation of the initial configuration, which can then subsequently be simulated with MD.

However, more advanced MC schemes are available for bio-molecules [9]. In particular, cluster MC methods can greatly increase efficiency and have been successfully used both in fully atomistic and coarse-grained models. For efficient sampling of smaller systems they can even outperform MD. A recently developed cluster algorithm Virtual Move Monte Carlo [21] has been also used to study dynamics of non-equilibrium systems. Finally, MC methods are often preferred for more coarse-grained systems, such as polymers on a lattice [9].

4 Practical aspects of numerical simulations

4.1 Ergodicity and error estimation

When sampling a studied system with MD or MC simulations, we assume an ergodic hypothesis, which states that the average $\overline{A}$ of a measured quantity A obtained from sampling in Eq. (1) is equal to an ensemble average over configurations $\langle A \rangle$ as defined in statistical physics, *i.e.* for canonical ensemble at temperature T:

$$\langle A \rangle = \frac{\sum_j A_j \exp\left(\frac{-E_j}{k_{\text{B}}T}\right)}{Z} = \frac{1}{M}\sum_{k=1}^{M} A_k = \overline{A} \tag{23}$$

where the ensemble average is taken over all possible states j of the system, while the average on the right is taken at different times as in Eq. (1). Z is the partion function of the system, $Z = \sum_j \exp\left(-E_j/k_{\mathrm{B}}T\right)$. If simulations could run for infinite amounts of time then this equivalence would always hold, but in practice time and resources are limited. One of the things that makes MD and MC possible is that often the number of microstates or length of time traces don't have to be that long, they just have to sample a "representative number of states" [6]. Part of the art of simulation is knowing when such criteria are met. Typically molecular simulations only run for very short times compared to experiment, often not more than a few ns. So if there are two states separated by a barrier (as illustrated in Fig. 2) that in experiment interconvert on a time-scale of seconds, this may never be observed in simulation without clever tricks. Depending on our initial configurations, you may get different results because you are only locally sampling the potential well you start in. For example if you start from different initial conditions, and get clearly different answers, then you probably have an ergodicity problem.

4.1.1 Statistical errors

Statistical errors arise due to random perturbations (such as thermal fluctuations) in a measured quantity and they can be estimated by taking large number of measurements. To evaluate the statistical error of the measured $\overline{A}$ in Eq. (1), one can calculate the variance σ_M^2 from M measurements of A:

$$\sigma_M^2(A) = \frac{1}{M}\sum_{k=1}^{M}\left(A_k - \overline{A}\right)^2. \tag{24}$$

However, care should be taken because the respective values of A_k whose values are taken k' steps apart can be correlated so that you are effectively not taking statistically independent measures and will (possibly strongly) underestimate your errors. The time-scale for statistically independent measurements can be measured with a correlation function which typically decays exponentially:

$$C(k') = \frac{1}{M-k'}\sum_{k=1}^{M-k'}\left(A_{k+k'} - \overline{A}\right)\left(A_k - \overline{A}\right) \propto \exp\left(-\frac{k'}{\tau}\right), \tag{25}$$

where τ is the decay time. If, for example, you take measurements separated by a time (or in MC number of steps) $k' \approx 2\tau$ then it is reasonable to say that they are uncorrelated. The estimate of the variance of average $\overline{A}$ is then

$$\sigma^2\left(\overline{A}\right) \approx \frac{1}{M_u - 1}\sigma_M^2 \tag{26}$$

where M_u is the number of uncorrelated measurements taken. The standard deviation in our averages therefore decays with M_u. If an observable has long correlation time τ, then very long MD time-trajectories or MC particle trajectories need to be used to obtain statistically accurate measurements.

Another way of estimating the errors is by the block average method. Divide up the M measurements into L consecutive blocks of length M/L each and calculate the average value of A within each block, denoted as $\overline{A}_l$ for l-th block. If the measurements are truly uncorrelated, then the block variance

$$\sigma_L^2(\overline{A}_l) = \frac{1}{L}\sum_{l=1}^{L}\left(\overline{A}_l - \overline{A}\right)^2 \tag{27}$$

will be independent of L. So by plotting the block variance as a function of L, a value L^* can be identified above which this holds, and block averages will no longer be correlated. The variance of the average $\overline{A}$ estimated from the block method is then

$$\sigma^2\left(\overline{A}\right) \approx \frac{\sigma_{L^*}^2(\overline{A}_l)}{L^* - 1}. \tag{28}$$

which can be assumed to be a measure of the true statistical errors in the simulations. It is amazing how rarely these methods are employed in the literature – most published simulation error bars are strong underestimates!

4.1.2 Systematic errors

Systematic errors can arise from many sources. Your model could be wrong, your simulation method flawed (e.g. too large time-steps), you could not be properly equilibrating your system before trying to measure properties, your initial conditions could bias your outcomes (if there is, say a large barrier as in Figure 2), or your simulated system could be too small so that finite size effects overwhelm the signal you are trying to extract. Opportunities for error are boundless!

There is no reliable procedure to detect systematic errors. For instance, when simulating a molecule that transitions between multiple conformation states (such as opening and closing of a DNA hairpin close to its melting temperature), you need to check that your simulation has visited all of those states. It is further necessary to ensure that the simulation has made frequent transitions between these states. It can be helpful to find several order parameters that characterize transitions in our system (for instance the number of bonds between strands, distance between interacting molecules, or value of dihedral angle) and monitor their value over the course of the simulation. Visualization (either by converting a trajectory to a movie or by taking snapshots at separated time intervals) of simulated system can also provide valuable insight. Another useful check is to run multiple independent simulations, each starting from a different initial state, and compare the estimates of quantity of interest from each of them.

Finite-size effects are another possible source of systematic error: our simulated system of N particles might be too small to correctly capture bulk behavior. Finite size errors typically scale as $1/\sqrt{N}$ and by examining how measured properties change as a function of simulation size N, one can sometime reliably extrapolate to the bulk limit.

4.2 Reduced units

In order to reduce the likelihood of errors, it is often convenient to formulate a simulation in reduced units. For example, units for energy, mass and length are chosen and then all other quantities are expressed in terms of these simulation units. When simulating a system with interaction potential with its minimum value $-\epsilon$ at distance σ, a natural choice of unit length is σ, ϵ for unit energy and $k_{\mathrm{B}}T/\epsilon$ for unit temperature.

Using reduced units also helps highlight relationships between different quantities. It can also make code more efficient, for example in a system of identical particles, we can use units where their mass is equal to 1 , thus saving the operation of multiplying or dividing by mass when calculating velocity and momentum. Furthermore, all real values are represented with a finite precision in computer memory. Working with reduced units in which the magnitude of the simulated quantities such as velocity or energy are of order ~ 1 prevents the possibility of the numbers to be too small or too large to be stored in memory. Finally, it is much easier to monitor and log simulation runs when measured quantities are ~ 1. If very large values suddenly appear, it is a strong indication of a bug or another problem in the simulation.

4.3 Interaction potentials in molecular simulations

The interactions between particles in the molecular simulations depend on the particular system that we are studying. However, the generic potentials in biomolecular simulations typically consist of two different classes of interaction terms, bonded (V_{b}) and non-bonded (V_{nb}) interactions.

V_{b} represents for example interactions between covalently bonded atoms, or neighboring nucleotides/aminoacids in a coarse grained polymer model. V_{nb} usually consists of the following terms

$$V_{\mathrm{b}} = V_{\mathrm{st}} + V_{\mathrm{angle}} + V_{\mathrm{dihedral}}. \tag{29}$$

V_{st} is the stretching potential between adjacent bonded particles, which can take for example a quadratic form. In that case, for particles at distance r_{ij}, $V_{\mathrm{st}} = k\left(r_{ij} - r_0\right)^2$. V_{angle} depends on the mutual orientations of the particles. For atomistic-level representation simulation, quadratic form of angular potential is often used, i.e. $V_{\mathrm{angle}}\left(\theta_{ijk}\right) = k\left(\theta_{ijk} - \theta_0\right)^2$, where θ_{ijk} is the the angle between the vectors $\mathbf{r}_{ij}$ and $\mathbf{r}_{jk}$ connecting particles i, j and j, k respectively. Finally, V_{dihedral} typically takes the form $V_{\mathrm{dihedral}}\left(\phi_{ijkl}\right) = k\left(1 - \cos\left(n\phi_{ijkl} - \phi_0\right)\right)$, where ϕ_{ijkl} is the dihedral angle defined as then angle between planes that contain particles i,j,k and (j,k,l) respectively, and n is an integer defining the periodicity. Other forms of V_{dihedral} are also used, for example a quadratic function of $\cos\left(\phi_{ijkl}\right)$.

V_{nb} contains non-bonded interactions. This term typically includes

$$V_{\mathrm{non-bonded}} = V_{\mathrm{electrostatic}} + V_{\mathrm{vdw}}, \tag{30}$$

where $V_{\mathrm{electrostatic}}(r) = -k_e q_i q_k / r$ is the Coulomb law for particles at distance r with charges q_i, q_k, and V_{vdw} is the van der Waals term, which can be modeled for instance with Lennard-Jones potential

$$V_{\mathrm{vdw}}(r) = 4\epsilon\left(\left(\frac{\sigma}{r}\right)^{12} - \left(\frac{\sigma}{r}\right)^{6}\right). \tag{31}$$

The parameters of the potential functions can depend on the type of interacting particles. The type of interactions and their concrete interactions form presented here is by no means exhaustive, and it will depend on the particular model used.

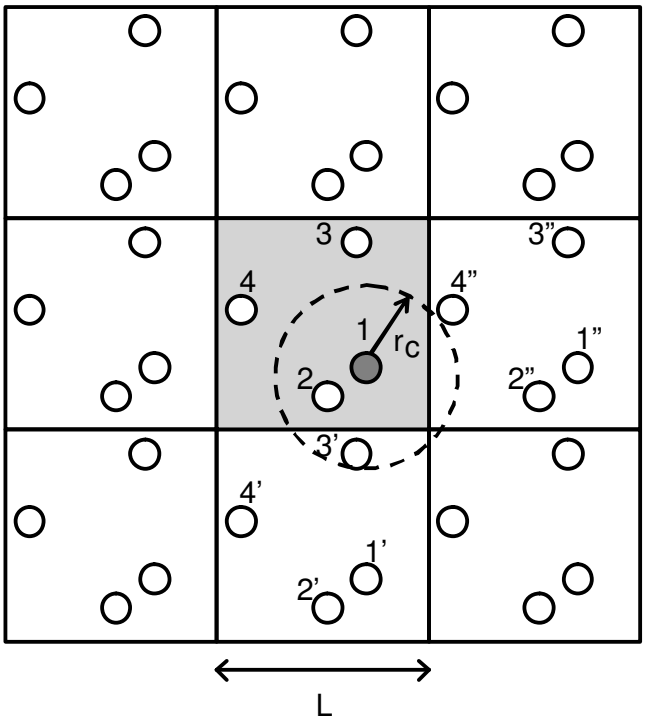

Figure 1: A schematic periodic square lattice. The gray square in the middle corresponds to the simulated system. When calculating forces acting on particle 1, all particles and their images that are within its interaction radius r_c are considered.

4.4 Truncation of interactions

The most expensive part of the simulation is typically the evaluation of the potential. For simulations of N particles, if all pairs of particles need to be considered, the time required for evaluation is $O(N^2)$. However, for short-range potentials, interactions are very small for a distant particle. One can therefore reasonably choose a cutoff distance r_c beyond which potentials are set to zero. The great advantage of such a scheme is scaling as $O(N)$ instead of $O(N^2)$ (we will describe how to do this in more detail in section 4.6).

A naive truncation can lead to discontinuous forces, which is problematic for MD. So instead a truncated and shifted potential v'_{ij} is typically used to ensure that calculated forces are finite:

$$v'_{ij}(r) = \begin{cases} v_{ij}(r) - v_{ij}(r_c) & \text{for } r \leq r_c \\ 0 & \text{for } r > r_c \end{cases} \tag{32}$$

Note that the derivative of Eq. (32) can have a discontinuity at $r = r_c$, and sometimes the potential is further modified to have continuous derivatives as well.

For a system at density ρ with a pairwise potential v_{ij} that depends on the distance between a pair of particles r_{ij}, the systematic error of not counting contributions from the potential above r_c can be corrected by adding a constant to the total interaction potential:

$$V'(\mathbf{r}^N) = \sum_{r_{ij}<r_c} v_{ij}(r_{ij}) + \int_{r_c}^{\infty} \frac{N\rho}{2} 4\pi r^2 v(r) dr. \tag{33}$$

We note that for interactions which depend on interparticle distance as $\propto r^n$ where $n \leq 3$ (such as for example Coulombic interactions), the correction above will diverge. More sophisticated methods such as Ewald sums or fast multipole method [6] need to be introduced to efficiently handle such long-range interactions.

4.5 Periodic boundary conditions

Even with rapid increases in computer power, the system sizes that can be handled by MC or MD simulations typically remain orders of magnitudes below the number of molecules present in an experimental setting. This can cause problems. For example, a cubic box of 1000 closely packed particles would have nearly half of the particles on the surface, and these surface particles would behave differently from the bulk particles. One way to get around such surface effects is to use periodic boundary conditions (PBC). The simulation is still run with

the same number of particles, but the simulation box is repeated periodically, and particles near the edge can interact with their closest neighbours in adjoining boxes, as shown in Figure 1.

During the course of the simulation the positions of each particle is determined modulo L, where L is the chosen length of the side of the simulation box. Effectively, this means that if particle leaves the simulation box through one side, it enters at the opposite side of the box.

The PBC lattice is usually chosen to be cubic (or square in 2D), but other options are also possible. Particles inside the simulation box can interact with other particles in the box as well as their images in the neighboring boxes, as illustrated in Figure 1. In practice choosing box size so that the cutoff $r_c < L/2$ will assure that each particle will only interact with the closest image of other simulated particles, and not with itself.

PBC can however cause some artifacts. For example the largest wavelength of fluctuations in the simulation cannot be larger than L and, for example crystallization into geometries that are not commensurate with the PBC will be artificially suppressed.

4.6 Verlet lists and cell lists

If the interaction cutoff is r_c, it would be wasteful to calculate distances between all pairs of particles, as only a fraction of them will be within interaction radius. The most widely used techniques of keeping track of particles which are close enough to possibly interact are Verlet lists and cell lists.

A Verlet lists stores for each particle a list of particles which are at a distance smaller than $r_c + r_s$, where r_s is called the "skin distance". When calculating forces acting on a particle, only interactions with particles stored in its Verlet list are considered. All lists need to be updated when any particle moves a distance larger than r_s from its original position when lists were last updated. The value of r_s has to be carefully optimized: too small r_s will result in very frequent updates, while large values will result in too many particles kept in each list.

Cell lists partition the simulation box into smaller boxes, called cells, with side length set to r_c. Each cell stores a list of particles inside it. For a given particle, we only consider interactions with particles inside its own cell and its neighboring cells (8 in 2D and 26 in 3D). When a particle moves, we check whether it has moved into a different cell, in which case we remove it from its old cell's list and add it to the particle list in the new cell.

Verlet lists and cell lists can be combined: When regenerating Verlet lists, we can only consider inclusion of particles in neighboring cells. The evaluation time for calculating interactions between N particles with cell lists scales as $O(N)$, while Verlet lists method scales as $O(N^2)$. Combining both removes the N^2 dependence of the Verlet list scheme. The optimal choice, however, depends on the particular system and simulation method. Verlet lists are well suited for MD simulations, where particle displacement per time step is small. In MC simulations, which can allow large translation moves, it may be preferable to use cell lists alone.

4.7 Parallelization

While the speed of a single CPU core is no longer increasing according to Moore's law, the law (a doubling of processing power every 18 months) still holds because the number of cores that can be put on a single chip keeps growing. In addition to multi-core CPUs, graphical processing units (GPUs), originally designed for accelerating computer graphics operations, have successfully made their way into high performance computing community. They consist of up to thousands of simple computing units, connected to high-speed memory, capable of running hundreds of threads concurrently. In order to take full advantage of latest hardware developments, it is often necessary to design a parallelized version of simulation algorithms and employ programming interfaces appropriate for chosen architecture.

A trivial way to take advantage of multiple CPU cores is to simply run multiple copies of the same simulation, each with a different initial condition. We can change the seed of the random number generator for stochastic algorithms or the initial configuration (such as initial velocities of particles in MD simulations). If the simulations are run concurrently we can include points sampled by all simulations in our estimations in Eq. (1). This method is suitable both for MC and MD, as long as the system is not too large to be properly equilibrated in an individual single-core simulation.

Advanced parallelization methods require cooperation between CPU cores. A popular programming library, MPI, offers a set of routines for communication between multiple processes that can be running on a single multicore computer or multiple computers connected in a network. An MD simulation box is typically split up into cubic domains, where interactions between all particles in one domain are handled by a single CPU core. Communication between cores is necessary to properly synchronize iterations and handle particles crossing between domains. While simulation time ideally would grow linearly with the number of cores uses, the increased communication overhead for more cores means that the speedup will eventually saturate.

For multiple cores on a single machine, it is also possible to use OpenMP programming interface for multithreaded programs with shared memory. For instance, updates of particle positions or calculations of forces

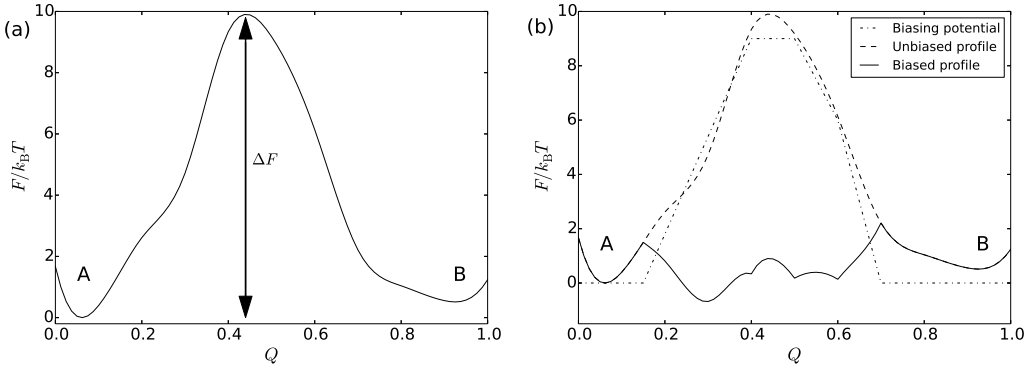

Figure 2: (a) A schematic free-energy profile as a function of order parameter Q. The probability density for a given value of Q is $p(Q) \propto \exp(-F(Q)/k_\mathrm{B}T)$. Crossing from state A to state B is slowed down by the barrier. (b) To accelerate transitions between A and B, an artificial biasing potential $-k_\mathrm{B}T\log(w(Q))$ is added to the system, which flattens the free-energy landscape, making the transitions much easier. One must subsequently correct for the biasing potential when extracting results from the simulation.

can be split up into multiple threads, each handling particles assigned to it.

GPU molecular simulations are written as multithreaded codes with shared memory executed on graphical processors, which are specifically designed to perform vector operations in parallel. The codes are usually written in CUDA or OpenCL, which are C-like programming languages extended for GPU architectures. While using GPUs is still not as well developed as CPUs, and so is typically harder to program and implement, the speedups can be large, e.g. one or two orders of magnitude for MD simulations [19]. It is also possible to split a simulation between multiple GPU cards in a network, with communication between them handled via MPI.

The parallelization methods outlined above are typically better suited for MD simulations, with the exception of the trivial parallelization that is equally suitable for MC simulations as well.

5 Acceleration of equilibration and simulating rare events

If free-energy barriers are much larger than $k_\mathrm{B}T$ (the typical energy scale in molecular simulations) then only a very rare fluctuation will allow the simulated system to cross the barrier, leading to sampling problems. But if you have some knowledge of the barrier, then progress can be made to accelerate the simulation. Several such techniques are described below

But first, to characterize a barrier, choose an order parameter Q and plot the free energy $F(Q)$, as illustrated in Figure 2(a). The (meta)stable minima separated by a barrier can represent for example different conformational states of a molecule, or a system with unbound and bound ligand respectively. The order parameter (reaction coordinate) Q characterizing the transition can be for example torsional angle or distance between a receptor and a ligand. It is very important to pick a suitable order parameter (possibly multidimensional) to describe the desired transition.

5.1 Umbrella sampling and free-energy calculation

Perhaps the most straightforward method to understand is called umbrella sampling. It introduces an additional biasing potential $-k_\mathrm{B}T\log(w(Q(\mathbf{r}^N)))$, which depends on the order parameter Q and is designed to flatten the barrier to allow the simulation to more efficiently sample the relevant state space (see Figure 2(b)). During the simulation, states are sampled according to

$$p_w(Q) \propto w\left(Q\right) \exp\left(\frac{-F(Q)}{k_\mathrm{B}T}\right) \tag{34}$$

The probability $p_w(Q)$ that a simulated biased system is in a state given by the value of order parameter Q is estimated by simply counting the time the simulated system spends in configurations with a given value of an order parameter and dividing it by the total simulation time. Of course this biasing procedure does not reproduce the right statistical sampling of the ensemble. But the probability $p(Q)$ of having order parameter Q in the canonical ensemble can be estimated by correcting for the bias:

$$p(Q) \propto \frac{p_w(Q)}{w(Q)}. \tag{35}$$

For umbrella sampling to work efficiently, it is important to choose the appropriate order parameter that characterizes the transition over the barrier. Furthermore, it is usually not obvious how to choose the biasing functions $w(Q)$, as we normally do not know beforehand what $F(Q)$ looks like. A typical procedure is to start with an (educated) initial guess for the biasing potential, run simulations that sample some of the values of order parameter, and iteratively adapt the values of the biasing potential, each time setting weights $w(Q)$ inversely proportional to the estimated $p_w(Q)$ of sampled values of Q, until one finds that $p_w(Q)$ is more or less flat.

Umbrella sampling is often combined with weighted histogram analysis methods (WHAM) [8], which splits values of Q into multiple independent overlapping intervals ("windows"). For each window, we run simulations that sample values of Q within given interval. The intervals overlap between neighboring windows and the estimates of probability $p(Q)$ from different windows can be combined to obtain probabilities for all allowed values of Q. The sampling simulations can be run concurrently to reduce the time required to obtain the desired probabilities, from which the free-energy landscape of studied system can be derived as

$$F(Q) = -k_{\mathrm{B}}T \log(p(Q)) + C. \tag{36}$$

Free energy is defined up to a constant C, since shifting the profile by an arbitrary constant independent of Q will not change the thermodynamics.

5.2 Parallel tempering

Since the time required to cross the barrier is $\propto \exp(\Delta F/k_{\mathrm{B}}T)$, it will be easier to cross the barrier at higher temperature. Parallel tempering exploits this principle by coupling simulations at multiple temperatures. The higher temperature runs have the advantage that they allow the simulation system to escape local free-energy minima, while simulations at the lower temperature T of interest generate the correct ensemble averages.

In this scheme n simulations (MD or MC) are run in parallel at respective temperatures $T_1 < T_2 < T_3 < \ldots T_n$. After a given number of steps, two neighboring temperatures $T_a < T_{a+1}$ are chosen and configurations are swapped with acceptance probability that satisfies detailed balance [6]

$$p = \min\left(1, \exp\left[\left(\frac{1}{k_{\mathrm{B}}T_a} - \frac{1}{k_{\mathrm{B}}T_{a+1}}\right)\left(V(\mathbf{r}_a^N) - V(\mathbf{r}_{a+1}^N)\right)\right]\right), \tag{37}$$

where $V(\mathbf{r}_a^N)$ and $V(\mathbf{r}_{a+1}^N)$ denote the potential energy evaluated for the configuration of particles in a-th and $a+1$-th replicas respectively. In the variant of parallel tempering for MD, called replica exchange molecular dynamics (REMD) [16], it is further necessary to rescale velocities of particles by a factor $\sqrt{T_a/T_{a+1}}$ for configuration which was at T_{a+1} prior to accepting the exchange, and by an inverse factor for velocities of configuration originally at T_a.

Since the probability of accepting a configuration swap will rapidly drop to zero if T_a and T_{a+1} are too far apart, multiple intermediate temperatures may be more efficient than running just two replicas at T_1 and T_n. More replicas are more expensive of course, but closer temperatures lead to larger acceptance rates for temperature swaps. So the number of replicas needs to be optimized to achieve the most efficient simulation. If you are interested in what happens at different temperatures, this scheme has the bonus of speeding up the lower temperature simulations at the same time. The advantage of parallel tempering is that there is no need to specify an order parameter, as increasing the temperatures generally helps to overcome barriers. However it does not allow the control in terms of pushing the system in a desired direction that can be achieved with umbrella sampling.

5.3 Forward flux sampling

While umbrella sampling and parallel tempering help to efficiently sample the configuration space, they perturb the dynamics of the system by adding an additional non-physical potential or by randomly switching configurations between different temperatures. To study the dynamics of barrier crossing, for instance to estimate the rate of a process or to study in detail how the transition happens, we need to use rare event methods [6, 2, 3, 18]. We briefly outline one example of rare event methods, forward flux sampling (FFS) [3, 2], which can be employed for studying of transitions in systems in equilibrium as well as non-equilibrium.

FFS is best suited for a transition between two states (A and B) separated by a single barrier with no metastable intermediates. The transition from A to B is partitioned by dividing interfaces $\lambda_0, \ldots, \lambda_n$, corresponding to values of order parameter(s) Q, as schematically indicated in Figure 3(a). For $Q < \lambda_0$, system is in state A, and for $Q > \lambda_n$, it is in state B. FFS method starts by estimating the flux ϕ_A^0 of trajectories leaving from state A that cross interface λ_0: We run a simulation and record coordinates of our system in phase space whenever we reach interface λ_0 while coming from state A. The simulation is stopped after a desired number of crossings was generated. We divide the number of recorded states by the simulation time to obtain estimate

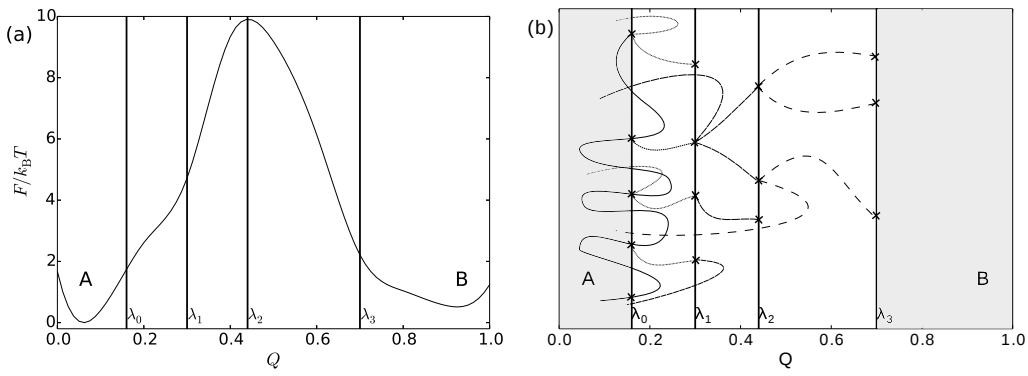

Figure 3: (a) Interfaces for forward flux sampling are specified for a given value of order parameter Q. (b) A schematic illustration of FFS method. Flux is generated by recording configurations (crosses) from the trajectory that crosses λ_0 coming from A. The stored configurations are used to launch simulations that are stopped until they reach next interface (indicated with crosses), or go back to A. The configurations at the interface λ_i are used to launch simulations to sample transition to λ_{i+1}. Process is repeated iteratively for each interface.

of ϕ_A^0. The rate of transition from A to B is then obtained as

$$k_{AB} = \phi_A^0 \prod_{i=0}^{n-1} P\left(\lambda_{i+1}|\lambda_i\right) \tag{38}$$

where $P\left(\lambda_{i+1}|\lambda_i\right)$ is the probability that trajectory coming from A and crossing λ_i will also cross λ_{i+1} before returning to state A. In practice, we can estimate $P\left(\lambda_{i+1}|\lambda_i\right)$ by launching multiple simulations from randomly selected saved configurations at interface λ_i and recording what fraction reaches λ_{i+1} and what fraction goes back to A, as schematically shown in Figure 3(b). We record the states that reached λ_{i+1} and use them to start simulations to estimate $P\left(\lambda_{i+2}|\lambda_{i+1}\right)$. FFS method requires simulation with stochastic dynamics (such as Langevin thermostat) to ensure that two simulations started from the same saved configuration will not follow an identical path. FFS hence allows us to partition the transition from A to B into several intermediate steps, where the probability estimation can be trivially parallelized by launching multiple simulations for a given interface concurrently.

6 Simulation tools

6.1 Choice of a model

One popular choice for biomolecular simulations is to use fully atomistic representations. Several parameterized force-fields are available for fully-atomistic simulations, with parameters based on quantum chemistry calculations or on empirical studies that fine-tune the potentials to reproduce experimental data (or both). Some force-fields are optimized for certain molecules such as DNA or RNA duplexes, or folded proteins. Simulations of biomolecules with fully atomistic resolution often requires explicit simulation of solvent molecules (water and ions) as well, which makes the simulations particularly demanding for computational resources (although there are methods to take water and ions into account implicitly). The maximum simulated time that can be achieved by massively parallel MD simulations with fully atomistic resolution has only recently began to approach ms timescale at which some of the biologically relevant processes happen [5]. There are ongoing efforts to refine and improve these atomistic force-fields, in order to obtain more accurate descriptions of physical reality.

Another option is to use coarse-grained models, where multiple degrees of freedom are integrated out and groups of atoms are replaced by a simplified coarse-grained representation (such as a single bead representing a nucleotide in DNA or a protein residue). These simplifications mean it is possible to simulate processes and systems that are out of reach for fully-atomistic models. Coarse-grained models usually do not explicitly simulate the solvent, which is another reason they are often much faster than fully-atomistic models. However, the reduced degrees of freedom come at a cost. Interactions in coarse-grained models need to be parameterized to reproduce a desired property of the studied system. Such procedures are not straightforward. For example, fitting too closely to one set of data (for example structure) will usually result in larger errors in other properties of the model (say thermodynamics), a consequence of a general phenomenon of that has been called "representability problems" [10]. Coarse-grained systems should always be used with care and physical insight is paramount to avoid getting fooled by results that look good, but do not track physical reality.

6.2 Choice of simulation software

Writing your own molecular simulation software allows you to tailor the code to the needs of your specific system of interest, and often helps you understand more deeply what the strengths and weaknesses of your simulation method are. But it can be a lot of work where you re-invent the wheel. Luckily there are now many good software packages available that allow you to choose interaction potentials, the type of ensemble, the integration method, the thermostat and more. These packages are typically highly optimized and support various techniques for acceleration of the sampling.

Simulation software typically does not include visualization software, but allows simulation trajectories to be stored in file formats which can be viewed in molecular visualization tools such as VMD (Visual Molecular Dynamics, available at `www.ks.uiuc.edu/Research/vmd/`) or Chimera (available at `www.cgl.ucsf.edu/chimera/`). We can't stress enough the importance of regularly visualizing configurations throughout a simulation and thinking carefully about the results to make sure they are physically plausible.

We provide a list of some available molecular simulation software packages below. Neither the list nor description of their properties is by any means exhaustive. All of the packages listed below are under active development, with new functionalities continuously being added, and it is hence advised to check the documentation available on the website of the package for the full list of capabilities.

- **Amber**
  Website: `ambermd.org`
  Licence: Commercial, with separate pricing option for academic and industrial use
  The Amber package contains simulation tools for MD simulations, as well as its own parameterized force-fields. It is primarily aimed at simulating biomolecular systems such as proteins, lipids, nucleic acids, as well as solvents and ions with fully-atomistic resolution. The simulation code is highly scalable and supports parallelization on multiple CPUs as well as on GPUs. It implements several methods for acceleration equilibration and for free-energy estimation. Various other tools for preparation of the initial configuration of molecules and for the trajectory analysis are also provided.

- **CHARMM**
  Website: `www.charmm.org`
  License: Available for academic use for a nominal fee
  CHARMM, like Amber, is aimed at fully-atomistic MD simulations. It comes with its own CHARMM force-field parametrization. It supports parallelization and implements standard algorithms for simulations of various thermodynamic ensembles, along with advanced sampling techniques.

- **Gromacs**
  Website: `www.gromacs.org`
  Licence: GNU Public License
  Gromacs is a general purpose MD simulation package, which is a popular choice for MD simulations with fully-atomistic resolution, often used for biomolecular systems. It can be used for simulations with user-specified interaction, including imported force-fields from Amber or CHARMM. It supports parallelization on multiple CPUs, as well as carrying out simulations that split up the calculations between CPUs and GPUs. It implements a vast number of methods for acceleration of equilibration (including REMD and Umbrella sampling), relaxation and preparation of the initial configuration, and tools for trajectory analysis.

- **NAMD**
  Website: `www.ks.uiuc.edu/Research/namd`
  Licence: Free for non-commercial use
  NAMD is a software package for biomolecular simulation, especially aimed at large-scale systems with fully atomistic resolution. It is optimized for a high scalability of parallelized MD simulations on CPUs and GPUs.

- **HOOMD-blue**
  Website: `codeblue.umich.edu/hoomd-blue`
  Licence: Free
  HOOMD-blue is a general purpose MD simulation package. It supports common integration methods for several thermodynamics ensembles and also supports MD with rigid bodies. The simulation is setup and analyzed via a script in Python programming language, and also produces trajectory information in common file formats. It supports parallelization on multiple CPUs and on GPU. It allows to choose from a large variety of interaction potentials commonly used in coarse-grained models or to specify a custom one in a table.

- **OxDNA**
  Website: `dna.physics.ox.ac.uk`
  Licence: GNU Public License
  OxDNA was originally developed as a simulation package for a coarse-grained model for DNA and it has since been reworked as a general simulation package for both MC and MD simulations. It implements Lennard-Jones particles, patchy particles, and coarse-grained models of DNA and RNA. New systems can be added by implementing a user-defined interaction in a C++ interface. It supports parallelization on GPUs.

- **LAMMPS**
  Website: `lammps.sandia.gov`
  Licence: GNU Public License
  LAMMPS is a general-purpose simulation package, applicable both to a wide range of coarse-grained systems as well as simulations with fully atomistic resolution. It implements MD integrators for several thermodynamic ensembles and supports parallelization on CPUs and GPU.

- **ESPResSo**
  Website: `espressomd.org`
  Licence: GNU Public License
  ESPResSo is a software tool aimed at MD simulations of coarse-grained models, and supports potentials and simulations constraints often used for coarse-grained systems. The simulations are setup using Tcl scripting language, and software allows for parallelization on CPUs and GPU.

7 Summary

- Molecular dynamics and Monte Carlo algorithms allow us to simulate a system on a computer in order to sample its properties at equilibrium and also study its dynamics. Computer simulations can be used to study properties of a molecular system at the level of detail not accessible to experiments, but also be used to test a theory.
- The most popular algorithm for integrating equations of motion in MD is Verlet algorithm and its variants. For studying thermodynamic ensembles at constant temperature and/or pressure, one needs to use thermostats/barrostats coupled to the MD simulation.
- Using reduced units helps monitor errors and clarify physical principles in simulations.
- Runing simulations in a box with periodic boundary conditions helps ensure better sampling of bulk properties.
- Make sure to consider both statistical and systematic errors that arise in the simulations. Simulation software tools allow to regularly log information such as kinetic and potential energy, so remember to check their values are always reasonable. Periodically saving system configuration and visualizing the snapshots/movies is also advised.
- Simulated systems often involve large barriers and local minima which can greatly increase the simulation time necessary to obtain enough statistics to accurately estimate quantities of interest, often making it impossible to study these processes by a simple MD/MC simulation. Advanced sampling and rare event methods (such as umbrella sampling, parallel tempering or forward flux sampling) need to be employed to overcome such barriers.
- For biomolecular simulations, one usually chooses between simulations with fully-atomistic resolution or coarse-grained models, which are more efficient, but describe the system at a lower level of detail. It is important to carefully consider which properties of the system we are interested in and if they can be accurately captured by the model used.

8 Exercise

8.1 Lennard-Jones fluid

Write a simulation code (or use one of the available packages) to perform MD and MC simulation of particles interacting with a pairwise Lennard-Jones potential:

$$v_{ij}(r) = 4\epsilon \left(\left(\frac{\sigma}{r}\right)^{12} - \left(\frac{\sigma}{r}\right)^{6} \right). \tag{39}$$

Use a system of reduced units (measure energy in ϵ, distance in σ, and set mass $m = 1$), periodic boundary conditions and truncated and shifted version of potential in Eq. (39) with tail corrections, with r_c set to half of the simulated box size.

1. Simulate an NVE ensemble in MD and check how step size affects conservation of energy. Then simulate NVT ensemble with MC and check how selection of maximum move size affects acceptance probability.

2. To obtain the equation of state of Lennard-Jones fluid, carry out simulations of an NVT ensemble (using MD or MC) with 108 particles at temperature $T = 2\epsilon/k_{\mathrm{B}}$ at densities ρ ranging from 0.1 to 1.0 (in reduced units). Use virial equation [6] to calculate pressure for each simulated density, and compare the resulting equation of state with the values in Ref. [7].

8.2 Melting of an RNA duplex

Use a coarse-grained model of RNA, oxRNA [17] (implemented as a part of oxDNA package), to simulate melting of an RNA 8-mer. Setup a duplex in a simulation box with box side length 16.8 nm (convert appropriately to model's reduced units). Use the averaged-strength version of the model (which treats A-U and G-C base pairing as equally strong) and setup an NVT simulation at $T = 62\,^{\circ}\mathrm{C}$, which corresponds to the bulk melting temperature for the strand concentration used.

1. Run MC, MD, and VMMC simulations, starting from fully formed duplex. Monitor the number of base-pairs between the strands over the course of simulation. How often do you observe transition between single stranded and duplex state with various methods?

2. Use VMMC combined with umbrella sampling to accelerate the transitions between the single-stranded and duplex states. Compare the number of transitions observed in the biased simulation with the previously run unbiased ones. From simulation data, estimate the probability of the system having n_b base-pairs, where n_b ranges from 0 to 8. Plot the free-energy profile as a function of n_b. How would the free-energy difference between $n_b = 0$ and $n_b = 1$ change if you ran the simulation in 8-times larger simulation box? Is it necessary to run a new simulation to find out, or can it be calculated from data generated in a smaller box?

3. Calculate the melting temperature of the duplex. The melting temperature T_m is defined as a temperature at which half of the duplexes in the bulk dissociate into single strands. You can run multiple simulations at temperatures in the neighborhood of the expected melting temperature (around 62 °C), and calculate the bulk duplex yield for each one of them:

$$f = 1 + \frac{1}{2\Phi} - \sqrt{\left(1 + \frac{1}{2\Phi}\right)^2 - 1} \tag{40}$$

where Φ is the number (unbiased) number of duplex states divided by the number of single-stranded states as observed in the simulation. Note that f differs from the simulation box yield $\Phi/(1 + \Phi)$ due to the correction for finite size effects [12]. Plot f as a function of temperature, the melting temperature T_m corresponds to $f = 0.5$. It is also possible to run simulation just at a single temperature T and extrapolate Φ to temperatures close to T using histogram reweighting [20].

References

[1] M. P. Allen and D. J. Tildesley. *Computer simulation of liquids.* Oxford University Press, 1989.

[2] R. J. Allen, D. Frenkel, and P. R. ten Wolde. Forward flux sampling-type schemes for simulating rare events: Efficiency analysis. *J. Chem. Phys.*, 124(19):194111, 2006.

[3] R. J. Allen, C. Valeriani, and P. R. ten Wolde. Forward flux sampling for rare event simulations. *Journal of Physics: Condensed Matter*, 21(46):463102, 2009.

[4] D. Chandler. *Introduction to modern statistical mechanics.* Oxford University Press, 1987.

[5] R. O. Dror, R. M. Dirks, J. Grossman, H. Xu, and D. E. Shaw. Biomolecular simulation: a computational microscope for molecular biology. *Annual review of biophysics*, 41:429–452, 2012.

[6] D. Frenkel and B. Smit. *Understanding Molecular Simulation: From Algorithms to Applications.* Academic Press, Inc., 2006.

[7] J. K. Johnson, J. A. Zollweg, and K. E. Gubbins. The lennard-jones equation of state revisited. *Molecular Physics*, 78(3):591–618, 1993.

[8] S. Kumar, J. M. Rosenberg, D. Bouzida, R. H. Swendsen, and P. A. Kollman. The weighted histogram analysis method for free-energy calculations on biomolecules. i. the method. *J. Comput. Chem.*, 13(8):1011–1021, 1992.

[9] D. P. Landau and K. Binder. *A guide to Monte Carlo simulations in statistical physics.* Cambridge University Press, 2014.

[10] A. A. Louis. Beware of density dependent pair potentials. *J. Phys.: Condens. Matter*, 14:9187, 2002.

[11] M. E. J. Newman and G. T. Barkema. *Monte Carlo Methods in Statistical Physics.* Oxford University Press, 1999.

[12] T. E. Ouldridge, A. A. Louis, and J. P. K. Doye. Extracting bulk properties of self-assembling systems from small simulations. *J. Phys.: Condens. Matter*, 22:104102, 2010.

[13] J. T. Padding and A. A. Louis. Hydrodynamic interactions and brownian forces in colloidal suspensions: Coarse-graining over time and length scales. *Phys. Rev. E*, 74(3):031402, 2006.

[14] T. Schlick. *Molecular modeling and simulation: an interdisciplinary guide.* Springer Science & Business Media, 2010.

[15] J. Sethna. *Statistical mechanics: entropy, order parameters, and complexity.* Oxford University Press, 2006.

[16] Y. Sugita and Y. Okamoto. Replica-exchange molecular dynamics method for protein folding. *Chemical Physics Letters*, 314:141–151, Nov. 1999.

[17] P. Šulc, F. Romano, T. E. Ouldridge, J. P. K. Doye, and A. A. Louis. A nucleotide-level coarse-grained model of RNA. *J. Chem. Phys.*, 140(23):235102, 2014.

[18] T. S. Van Erp. *Dynamical Rare Event Simulation Techniques for Equilibrium and Nonequilibrium Systems*, pages 27–60. John Wiley & Sons, Inc., 2012.

[19] J. A. Van Meel, A. Arnold, D. Frenkel, S. Portegies Zwart, and R. G. Belleman. Harvesting graphics power for MD simulations. *Molecular Simulation*, 34(3):259–266, 2008.

[20] P. Šulc, F. Romano, T. Ouldridge, L. Rovigatti, J. Doye, and A. Louis. Sequence-dependent thermodynamics of a coarse-grained DNA model. *J. Chem. Phys.*, 137(13):135101, 2012.

[21] S. Whitelam, E. H. Feng, M. F. Hagan, and P. L. Geissler. The role of collective motion in examples of coarsening and self-assembly. *Soft Matter*, 5(6):1521–1262, 2009.