2.8

BASIC MOLECULAR DYNAMICS

Ju Li

Department of Materials Science and Engineering,
Ohio State University, Columbus, OH, USA

A working definition of molecular dynamics (MD) simulation is technique by which one generates the atomic trajectories of a system of N particles by numerical integration of Newton's equation of motion, for a specific interatomic potential, with certain initial condition (IC) and boundary condition (BC).

Consider, for example (see Fig. 1), a system with N atoms in a volume Ω. We can define its internal energy: $E \equiv K + U$, where K is the kinetic energy,

$$K \equiv \sum_{i=1}^{N} \frac{1}{2} m_i |\dot{\mathbf{x}}_i(t)|^2, \tag{1}$$

and U is the potential energy,

$$U = U(\mathbf{x}^{3N}(t)). \tag{2}$$

$\mathbf{x}^{3N}(t)$ denotes the collective of 3 D coordinates $\mathbf{x}_1(t), \mathbf{x}_2(t), \ldots, \mathbf{x}_N(t)$. Note that E should be a conserved quantity, i.e., a constant of time, if the system is truly isolated.

One can often treat a MD simulation like an experiment. Below is a common flowchart of an ordinary MD run:

[system setup]		[equilibration]		[simulation run]		[output]
sample selection	→	sample preparation	→	property average	→	data analysis
(pot., N, IC, BC)		(achieve T, P)		(run L steps)		(property calc.)

in which we fine-tune the system until it reaches the desired condition (here, temperature T and pressure P), and then perform property averages, for instance calculating the radial distribution function $g(r)$ [1] or thermal conductivity [2]. One may also perform a non-equilibrium MD calculation, during which the system is subjected to perturbational or large external driving forces,

S. Yip (ed.),
Handbook of Materials Modeling, 565–588.

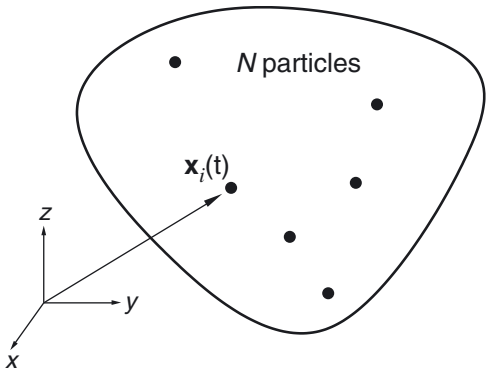

Figure 1. Illustration of the MD simulation system.

and we analyze its non-equilibrium response, such as in many mechanical deformation simulations.

There are five key ingredients to a MD simulation, which are boundary condition, initial condition, force calculation, integrator/ensemble, and property calculation. A brief overview of them is given below, followed by more specific discussions.

Boundary condition. There are two major types of boundary conditions: isolated boundary condition (IBC) and periodic boundary condition (PBC). IBC is ideally suited for studying clusters and molecules, while PBC is suited for studying bulk liquids and solids. There could also be mixed boundary conditions such as slab or wire configurations for which the system is assumed to be periodic in some directions but not in the others.

In IBC, the N-particle system is surrounded by vacuum; these particles interact among themselves, but are presumed to be so far away from everything else in the universe that no interactions with the outside occur except perhaps responding to some well-defined "external forcing." In PBC, one explicitly keeps track of the motion of N particles in the so-called supercell, but the supercell is surrounded by infinitely replicated, periodic images of itself. Therefore a particle may interact not only with particles in the same supercell but also with particles in adjacent image supercells (Fig. 2).

While several polyhedron shapes (such as hexagonal prism and rhombic dodecahedron from Wigner–Seitz construction) can be used as the space-filling unit and thus can serve as the PBC supercell, the simplest and most often used supercell shape is a parallelepiped, specified by its three edge vectors $\mathbf{h}_1$, $\mathbf{h}_2$ and $\mathbf{h}_3$. It should be noted that IBC can often be well mimicked by a large enough PBC supercell so the images do not interact.

Initial condition. Since Newton's equations of motion are second-order ordinary differential equations (ODE), IC basically means $\mathbf{x}^{3N}(t = 0)$ and

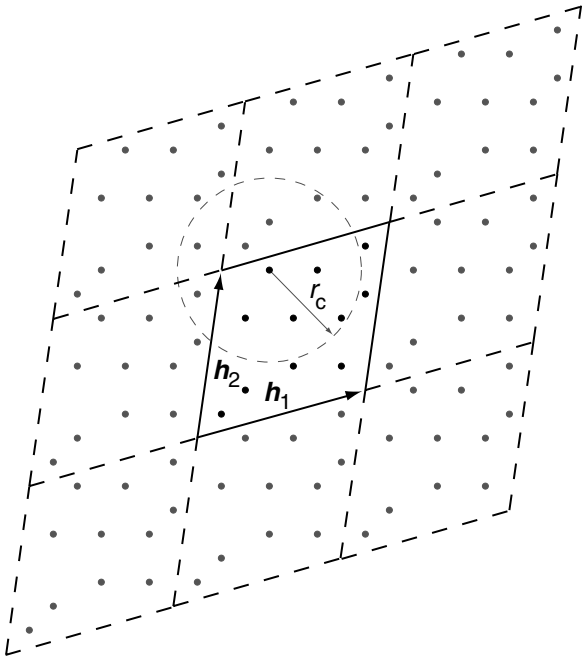

Figure 2. Illustration of periodic boundary condition (PBC). We explicitly keep track of trajectories of only the atoms in the center cell called the supercell (defined by edge vectors $\mathbf{h}_1$, $\mathbf{h}_2$ and $\mathbf{h}_3$), which is infinitely replicated in all three directions (image supercells). An atom in the supercell may interact with other atoms in the supercell as well as atoms in the surrounding image supercells. r_c is a cut-off distance of the interatomic potential, beyond which interaction may be safely ignored.

$\dot{\mathbf{x}}^{3N}(t = 0)$, the initial particle positions and velocities. Generating the IC for crystalline solids is usually quite easy, but IC for liquids needs some work, and even more so for amorphous solids. A common strategy to create a proper liquid configuration is to melt a crystalline solid. And if one wants to obtain an amorphous configuration, a strategy is to quench the liquid during a MD run.

Let us focus on IC for crystalline solids. For instance, $\mathbf{x}^{3N}(t = 0)$ can be a fcc perfect crystal (assuming PBC), or an interface between two crystalline phases. For most MD simulations, one needs to write an initial structure generation subroutine. Before feeding the initial configuration thus created into a MD run, it is a good idea to visualize it first, checking bond lengths and coordination numbers, etc. [3]. A frequent cause of MD simulation breakdown is pathological initial condition, as the atoms are too close to each other initially, leading to huge forces.

According to the equipartition theorem [4], each independent degree of freedom should possess $k_B T/2$ kinetic energy. So, one should draw each

component of the $3N$-dimensional $\dot{\mathbf{x}}^{3N}(t{=}0)$ vector from a Gaussian–Maxwell normal distribution $N(0, k_\mathrm{B}T/m_i)$. After that, it is a good idea to eliminate the center of mass velocity, and for clusters, the net angular momentum as well.

Force calculation. Before moving into the details of force calculation, it should be mentioned that two approximations underly the use of the classical equation of motion

$$m_i \frac{\mathrm{d}^2\mathbf{x}_i(t)}{\mathrm{d}t^2} = \mathbf{f}_i \equiv -\frac{\partial U}{\partial \mathbf{x}_i}, \quad i = 1, \ldots, N \tag{3}$$

to describe the atoms. The first is the Born–Oppenheimer approximation [5] which assumes the electronic state couples adiabatically to nuclei motion. The second is that the nucleus motion is far removed from the Heisenberg uncertainty lower bound: $\Delta E \Delta t \gg \hbar/2$. If we plug in $\Delta E = k_\mathrm{B}T/2$, the kinetic energy, and $\Delta t = 1/\omega$, where ω is a characteristic vibrational frequency, we obtain $k_\mathrm{B}T/\hbar\omega \gg 1$. In solids, this means the temperature should be significantly greater than the Debye temperature, which is actually quite a stringent requirement. Indeed, large deviations from experimental heat capacities are seen in classical MD simulations of crystalline solids [2]. A variety of schemes exist to correct this error [1], for instance the Wigner–Kirkwood expansion [6] and path integral molecular dynamics [7].

The evaluation of the right-hand side of Eq. (3) is the key step that usually consumes most of the computational time in a MD simulation, so its efficiency is crucial. For long-range Coulomb interactions, special algorithms exist to break them up into two contributions: a short-ranged interaction, plus a smooth, field-like interaction, both of which can be computed efficiently in separate ways [8]. In this article we focus on issues concerning short-range interactions only. There is a section about the Lennard–Jones potential and its trunction schemes, followed by a section about how to construct and maintain an atom–atom neighborlist with $\mathcal{O}(N)$ computational effort per timestep. Finally, see Chap. 2.2–2.6 for the development of interatomic potential functions.

Integrator/ensemble. Equation (3) is a set of second-order ODEs, which can be strongly nonlinear. By converting them to first-order ODEs in the $6N$-dimensional space of $\{\mathbf{x}_N, \dot{\mathbf{x}}_N\}$, general numerical algorithms for solving ODEs such as the Runge–Kutta method [9] can be applied. However, these general methods are rarely used in MD, because the existence of a Hamiltonian allows for more accurate integration algorithms, prominent among which are the family of predictor-corrector integrators [10] and the family of symplectic integrators [8, 11]. A section in this article gives a brief overview of integrators.

Ensembles such as the micro-canonical, canonical, and grand-canonical are concepts in statistical physics that refer to the distribution of *initial conditions*. A system, once drawn from a certain ensemble, is supposed to follow strictly

the Hamiltonian equation of motion Eq. (3), with E conserved. However, ensemble and integrator are often grouped together because there exists a class of methods that generates the desired ensemble *distribution* via time integration [12, 13]. Equation (3) is modified in these methods to create a special dynamics whose trajectory over time forms a cloud in phase space that has the desired distribution density. Thus, the time-average of a single-point operator in one such trajectory approaches the thermodynamic average. However, one should be careful in using it to calculate two-point correlation function averages. See Chap. 2.9 for detailed description of these methods.

Property calculation. A great value of MD simulation is that it is "omnipotent" at the level of classical atoms. All properties that are well-posed in classical mechanics and statistical mechanics can in principle be computed. The issues remaining are accuracy (the error comes from the interatomic potential) and computational efficiency. The properties can be roughly grouped into four categories:

1. Structural characterization. Examples include radial distribution function, dynamic structure factor, etc.
2. Equation of state. Examples include free-energy functions, phase diagrams, static response functions like thermal expansion coefficient, etc.
3. Transport. Examples include viscosity, thermal conductivity (electronic contribution excluded), correlation functions, diffusivity, etc.
4. Non-equilibrium response. Examples include plastic deformation, pattern formation, etc.

1. The Lennard–Jones Potential

The solid and liquid states of rare-gas elements Ne, Ar, Kr, Xe are better understood than other elements because their closed-shell electron configurations do not allow them to participate in covalent or metallic bonding with neighbors, which are strong and complex, but only to interact via weak van der Waals bonds, which are perturbational in nature in these elements and therefore mostly additive, leading to the pair-potential model:

$$U(\mathbf{x}^{3N}) = \sum_{j>i}^{N} V(|\mathbf{x}_{ji}|), \quad \mathbf{x}_{ji} \equiv \mathbf{x}_j - \mathbf{x}_i, \tag{4}$$

where we assert that the total potential energy can be decomposed into the direct sum of individual "pair-interactions." If there is to be rotational invariance in $U(\mathbf{x}^{3N})$, V can only depend on $r_{ji} \equiv |\mathbf{x}_{ji}|$. In particular, the Lennard–Jones potential

$$V(r) = 4\epsilon \left[\left(\frac{\sigma}{r} \right)^{12} - \left(\frac{\sigma}{r} \right)^{6} \right], \tag{5}$$

is a widely used form for $V(r)$, that depends on just two parameters: a basic energy-scale parameter ϵ, and a basic length-scale parameter σ. The potential is plotted in Fig. 3.

There are a few noteworthy features about the Lennard–Jones potential:

- $V(r = \sigma) = 0$, at which point the potential is still *repulsive*, meaning $V'(r = \sigma) < 0$ and two atoms would repel each other if separated at this distance.
- The potential minimum occurs at $r_{\min} = 2^{1/6}\sigma$, and $V_{\min} = -\epsilon$. When $r > r_{\min}$ the potential switches from being repulsive to being attractive.
- As $r \to \infty$, $V(r)$ is attractive and decays as r^{-6}, which is the correct asymptote for dispersion (London) forces between closed-shell atoms. To get a feel for how fast $V(r)$ decays, note that $V(r=2.5\sigma) = -0.0163\epsilon$, $V(r = 3\sigma) = -0.00548\epsilon$, and $V(r = 3.5\sigma) = -0.00217\epsilon$.
- As $r \to 0$, $V(r)$ is repulsive as r^{-12}. In fact, r^{-12} blows up so quickly that an atom seldom is able to penetrate $r < 0.9\sigma$, so the Lennard–Jones potential can be considered as having a "hard core." There is no conceptual basis for the r^{-12} form, and it may be unsuitable as a model for certain materials, so it is sometimes replaced by a "soft core" of the form $\exp(-kr)$, which combined with the r^{-6} attractive part is called the Buckingham exponential-6 potential. If the attractive part is also of an exponential form $\exp(-kr/2)$, then it is called a Morse potential.

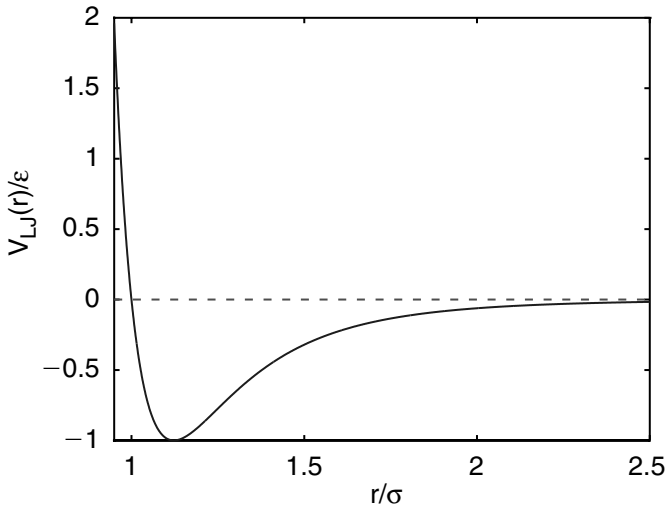

Figure 3. The Lennard–Jones potential.

For definiteness, $\sigma = 3.405$ Å and $\epsilon = 119.8\,k_B = 0.01032$ eV for Ar. The mass can be taken to be the isotopic average, 39.948 a.m.u.

1.1. Reduced Units

Unit systems are constructed to make physical laws look simple and numerical calculations easy. Take Newton's law: $f = ma$. In the SI unit system, it means that if an object of mass x (kg) is undergoing an acceleration of y (m/s^2), the force on the object must be xy (N).

However, there is nothing intrinsically special about the SI unit system. One (kg) is simply the mass of a platinum–iridium prototype in a vacuum chamber in Paris. If one wishes, one can define his or her own mass unit – ($\tilde{\text{kg}}$), which say is $1/7$ of the mass of the Paris prototype: 1 (kg) = 7 ($\tilde{\text{kg}}$).

If ($\tilde{\text{kg}}$) is one's choice of the mass unit, how about the unit *system*? One really has to make a decision here, which is either keeping all the other units unchanged and only making the (kg) $\to$ ($\tilde{\text{kg}}$) transition, or, changing some *other* units along with the (kg) $\to$ ($\tilde{\text{kg}}$) transition.

Imagine making the first choice, that is, keeping all the other units of the SI system unchanged, including the force unit (N), and only changes the mass unit from (kg) to ($\tilde{\text{kg}}$). That is all right, except in the new unit system the Newton's law must be rewritten as $F = ma/7$, because if an object of mass $7x$ ($\tilde{\text{kg}}$) is undergoing an acceleration of y (m/s^2), the force on the object is xy (N).

There is nothing wrong with the $F = ma/7$ formula, which is just a recipe for computation – a correct one for the newly chosen unit system. Fundamentally, $F = ma/7$ and $F = ma$ describe the same physical law.

But it is true that $F = ma/7$ is less elegant than $F = ma$. No one likes to memorize extra constants if they can be reduced to unity by a *sensible* choice of units. The SI unit system is sensible, because (N) is picked to work with other SI units to satisfy $F = ma$.

How can we have a sensible unit system but with ($\tilde{\text{kg}}$) as the mass unit? Simple, just define ($\tilde{\text{N}}$) = (N)/7 as the new force unit. The (m)–(s)–($\tilde{\text{kg}}$)–($\tilde{\text{N}}$)– unit system is sensible because the simplest form of $F = ma$ is preserved. Thus we see that when a certain unit in a sensible unit system is altered, other units must also be altered correspondingly in order to constitute a new sensible unit system, which *keeps the algebraic forms of all fundamental physical laws unaltered.* (A notable exception is the conversion between SI and Gaussian unit systems in electrodynamics, during which a non-trivial factor of 4π comes up.)

In science people have formed deep-rooted conventions about the *simplest* algebraic forms of physical laws, such as $F = ma$, $K = mv^2/2$, $E = K + U$, $P\Omega = \rho RT$, etc. Although nothing forbids one from modifying the constant coefficients in front of each expression, one is better off not to. Fortunately, as long as one uses a sensible unit system, these algebraic expressions stay invariant.

Now, imagine we derive a certain composite law from a set of simple laws. On one side, we start with and consistently use a sensible unit system A. On the other side, we start with and consistently use another sensible unit system B. Since the two sides use exactly the same algebraic forms, the resultant algebraic expression must also be the same, even though for a given physical instance, *a variable takes on two different numerical values* on the two sides as different unit systems are adopted. This means that the final algebraic expression describing the physical phenomena must satisfy certain concerted scaling invariance with respect to its dependent variables, corresponding to *any* feasible transformation between sensible unit systems. This strongly limits the form of possible algebraic expressions describing physical phenomena, which is the basis of *dimensional analysis*.

As mentioned, once certain units are altered, other units must be altered correspondingly to make the algebraic expressions of physical laws look invariant. For example, for a single-element Lennard–Jones system, one can define new energy unit $(\tilde{\text{J}}) = \epsilon$ (J), new length unit $(\tilde{\text{m}}) = \sigma$ (m), and new mass unit $(\tilde{\text{kg}}) = m_{\text{a}}$ (kg) which is the atomic mass, where ϵ, σ and m_{a} are pure numbers. In the $(\tilde{\text{J}})$–$(\tilde{\text{m}})$–$(\tilde{\text{kg}})$ unit system, the potential energy function is,

$$V(r) = 4(r^{-12} - r^{-6}), \tag{6}$$

and the mass of an atom is $m = 1$. Additionally, the forms of all physical laws should be preserved. For example, $K = mv^2/2$ in the SI system, and it should still hold in the $(\tilde{\text{J}})$–$(\tilde{\text{m}})$–$(\tilde{\text{kg}})$ unit system. This can only be achieved if the derived time unit (also called reduced time unit), $(\tilde{\text{s}}) = \tau\,(\text{s})$, satisfies,

$$\epsilon = m_{\text{a}}\sigma^2/\tau^2, \quad \text{or} \quad \tau = \sqrt{\frac{m_{\text{a}}\sigma^2}{\epsilon}}. \tag{7}$$

To see this, note that $m = 1$ $(\tilde{\text{kg}})$, $v = 1$ $(\tilde{\text{m}})/(\tilde{\text{s}})$, and $K = 1/2$ $(\tilde{\text{J}})$ is a solution to $K = mv^2/2$ in the $(\tilde{\text{J}})$–$(\tilde{\text{m}})$–$(\tilde{\text{kg}})$ unit system, but must also be a solution to $K = mv^2/2$ in the SI system.

For Ar, τ turns out to be 2.156×10^{-12}, thus the reduced time unit $(\tilde{\text{s}}) = 2.156$ (ps). This is roughly the timescale of one atomic vibration period in Ar.

1.2. Force Calculation

For pair potential of the form (4), there is,

$$\begin{aligned}\mathbf{f}_i &= -\sum_{j\neq i} \frac{\partial V(r_{ij})}{\partial \mathbf{x}_i} = \sum_{j\neq i} \left(-\frac{\partial V(r)}{\partial r}\bigg|_{r=r_{ij}} \right) \hat{\mathbf{x}}_{ij} \\ &= \sum_{j\neq i} \left(-\frac{1}{r}\frac{\partial V(r)}{\partial r}\bigg|_{r=r_{ij}} \right) \mathbf{x}_{ij},\end{aligned} \tag{8}$$

where $\hat{\mathbf{x}}_{ij}$ is the unit vector,

$$\hat{\mathbf{x}}_{ij} \equiv \frac{\mathbf{x}_{ij}}{r_{ij}}, \quad \mathbf{x}_{ij} \equiv \mathbf{x}_i - \mathbf{x}_j. \tag{9}$$

One can define *force on i due to atom j*,

$$\mathbf{f}_{ij} \equiv \left(-\frac{1}{r} \frac{\partial V(r)}{\partial r}\bigg|_{r=r_{ij}} \right) \mathbf{x}_{ij}, \tag{10}$$

and so there is,

$$\mathbf{f}_i = \sum_{j \neq i} \mathbf{f}_{ij}. \tag{11}$$

It is easy to see that,

$$\mathbf{f}_{ij} = -\mathbf{f}_{ji}. \tag{12}$$

MD programs tend to take advantage of symmetries like the above to save computations.

1.3. Truncation Schemes

Consider the single-element Lennard–Jones potential in (5). Practically we can only carry out the potential summation up to a certain cutoff radius. There are many ways to truncate, the simplest of which is to modify the interaction as

$$V_0(r) = \begin{cases} V(r) - V(r_c), & r < r_c \\ 0, & r \geq r_c \end{cases}. \tag{13}$$

However, $V_0(r)$ is discontinuous in the first derivative at $r = r_c$, which causes large numerical error in time integration (especially with high-order algorithms and large time steps) if an atom crosses r_c, and is detrimental to calculating correlation functions over long time. Another commonly used scheme

$$V_1(r) = \begin{cases} V(r) - V(r_c) - V'(r_c)(r - r_c), & r < r_c \\ 0, & r \geq r_c \end{cases} \tag{14}$$

makes the force continuous at $r = r_c$, but also makes the potential well too shallow (see Fig. 4). It is also slightly more expensive because we have to compute the square root of $|\mathbf{x}_{ij}|^2$ in order to get r.

An alternative is to define

$$\tilde{V}(r) = \begin{cases} V(r) \exp(r_s/(r - r_c)), & r < r_c \\ 0, & r \geq r_c \end{cases}$$

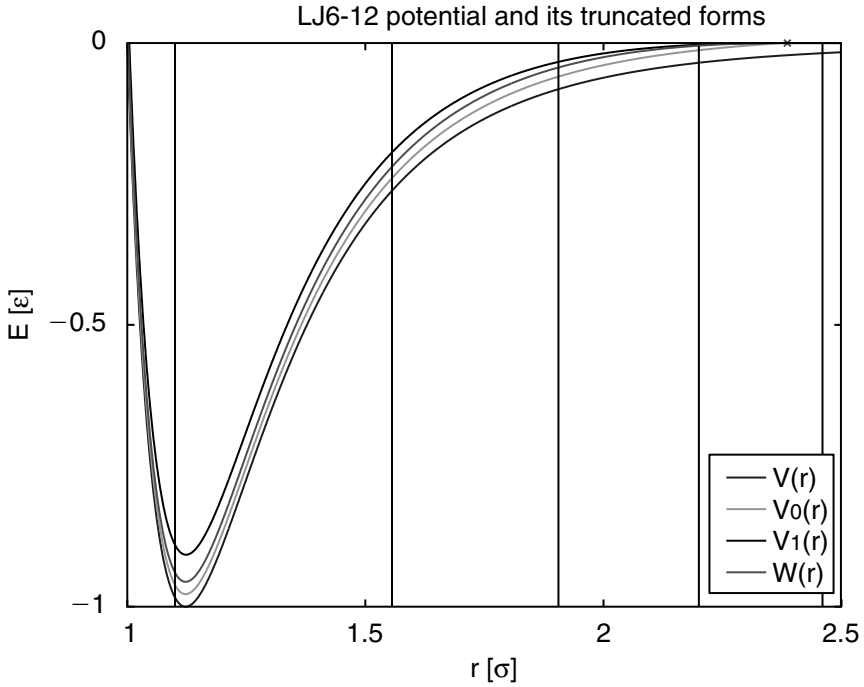

Figure 4. Lennard–Jones potential and its modified forms with cutoff $r_c = 2.37343\,\sigma$. Vertical lines indicate positions of neighbors in a single-element fcc crystal at 0 K.

which has all orders of derivative continuous at $r = r_c$. However, this truncation scheme requires another tunable parameter r_s.

The following truncation scheme,

$$W(r) = \begin{cases} 4\varepsilon \left[\left(\dfrac{\sigma}{r}\right)^{12} - \left(\dfrac{\sigma}{r}\right)^{6} + \left(2\left(\dfrac{\sigma}{r_c}\right)^{18} - \left(\dfrac{\sigma}{r_c}\right)^{12}\right) \right. \\ \quad \left. \times \left(\dfrac{r}{\sigma}\right)^{6} - 3\left(\dfrac{\sigma}{r_c}\right)^{12} + 2\left(\dfrac{\sigma}{r_c}\right)^{6} \right], & r < r_c \\ 0, & r \geq r_c \end{cases} \tag{15}$$

is a better alternative. $W(r)$, $V(r)$, $V_0(r)$ and $V_1(r)$ are plotted in Fig. 4 for comparison. r_c is chosen to be 2.37343σ, which falls exactly at the $2/3$ interval between the fourth and fifth neighbors at equilibrated fcc lattice of 0 K.

There is clearly a tradeoff in picking r_c. If r_c is large, the effect of the artificial truncation is small. On the other hand, maintaining and summing over a large neighbor list (size $\propto r_c^3$) costs more. For a properly written $\mathcal{O}(N)$ MD code, the cost versus neighbor number relation is almost linear.

Let us see what is the minimal r_c for a fcc solid. Figure 5 shows the neighboring atom shells and their multiplicity. Also drawn are the three glide planes.

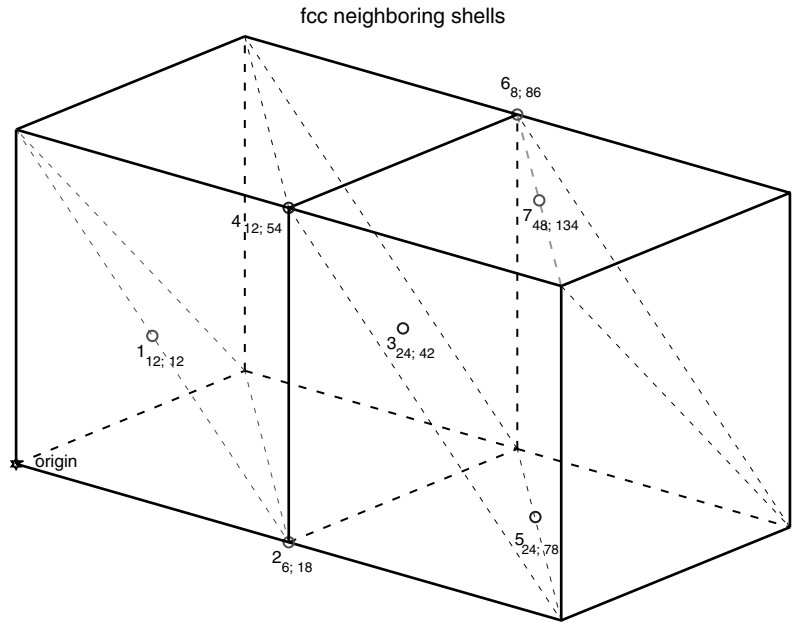

Figure 5. FCC neighboring shells. For example, label "$6_{8;\,86}$" means there are eight sixth nearest neighbors of the type shown in figure, which adds up to 86 neighbors with equal or less distance.

With (15), once the number of interacting neighbor shells are determined, we can evaluate the equilibrium volume and bulk modulus of the crystal in closed form. The total potential energy of each atom is

$$e_i = \frac{1}{2} \sum_{j \neq i}^{r_{ji} < r_c} W(r_{ji}). \tag{16}$$

For fcc crystal, we can extract scale-independent coefficients from the above summation and differentiate with respect to the lattice constant a – the minima of which yields the equilibrium lattice constant a_0. If we demand r_c to fall into an exact position between the highest included shell and the lowest excluded shell, we can iterate the process until mutual consistency is achieved. We then plug a_0 into (16) to calculate the binding energy per atom, e_0; the atomic volume

$$\Omega_0 = \frac{a_0^3}{4}, \tag{17}$$

and the bulk modulus

$$B \equiv -\frac{dP}{d\log\Omega}\bigg|_{\Omega_0} = \frac{a_0^2}{9\Omega_0}\frac{d^2e}{da^2}\bigg|_{a_0} = \frac{4}{9a_0}\frac{d^2e}{da^2}\bigg|_{a_0} \quad \text{(for fcc)}. \tag{18}$$

Table 1. FCC neighboring shells included in Eq. (15) vs. properties

n	N	$r_c[\sigma]$	$a_0[\sigma]$	$\Omega_0[\sigma^3]$	$e_0[\varepsilon]$	$B[\varepsilon\sigma^{-3}]$
1	12	1.44262944953	1.59871357076	1.02153204121	−2.03039845846	39.39360127902
2	18	1.81318453769	1.57691543349	0.98031403353	−4.95151157088	52.02448553061
3	42	2.11067974132	1.56224291246	0.95320365252	−6.12016548816	58.94148705580
4	54	2.37343077641	1.55584092331	0.94153307381	−6.84316556834	64.19738627468
5	78	2.61027143673	1.55211914976	0.93479241591	−7.27254778301	66.65093979162
6	86	2.82850677530	1.55023249772	0.93138774467	−7.55413237921	68.53093399765
7	134	3.03017270367	1.54842162594	0.92812761235	−7.74344974981	69.33961787572
8	140	3.21969263257	1.54727436382	0.92606612556	−7.88758411490	70.63452119577
9	176	3.39877500485	1.54643096926	0.92455259927	−7.99488847415	71.18713376234
10	200	3.56892997792	1.54577565469	0.92337773387	−8.07848627384	71.76659559499

The self-consistent results for r_c ratio $2/3$ are shown in Table 1. That is, r_c is exactly at $2/3$ the distance between the nth interacting shell and the $(n+1)$th non-interacting shell. The reason for $2/3(> 1/2)$ is that we expect thermal expansion at finite temperature.

If one is after converged Lennard–Jones potential results, then $r_c = 4\sigma$ is recommended. However, it is about five times more expensive per atom than the minimum-cutoff calculation with $r_c = 2.37343\sigma$.

2. Integrators

An integrator advances the trajectory over small time increments Δt:

$$\mathbf{x}^{3N}(t_0) \to \mathbf{x}^{3N}(t_0 + \Delta t) \to \mathbf{x}^{3N}(t_0 + 2\Delta t) \to \cdots \to \mathbf{x}^{3N}(t_0 + L\Delta t)$$

where L is usually $\sim 10^4 - 10^7$. Here we give a brief overview of some popular algorithms: central difference (Verlet, leap-frog, velocity Verlet, Beeman algorithm) [1, 14], predictor-corrector [10], and symplectic integrators [8, 11].

2.1. Verlet Algorithm

Assuming $\mathbf{x}^{3N}(t)$ trajectory is smooth, perform Taylor expansion

$$\mathbf{x}_i(t_0 + \Delta t) + \mathbf{x}_i(t_0 - \Delta t) = 2\mathbf{x}_i(t_0) + \ddot{\mathbf{x}}_i(t_0)(\Delta t)^2 + \mathcal{O}((\Delta t)^4). \tag{19}$$

Since $\ddot{\mathbf{x}}_i(t_0) = \mathbf{f}_i(t_0)/m_i$ can be evaluated given the atomic positions $\mathbf{x}^{3N}(t_0)$ at $t = t_0$, $\mathbf{x}^{3N}(t_0 + \Delta t)$ in turn may be approximated by,

$$\mathbf{x}_i(t_0 + \Delta t) = -\mathbf{x}_i(t_0 - \Delta t) + 2\mathbf{x}_i(t_0) + \left(\frac{\mathbf{f}_i(t_0)}{m_i}\right)(\Delta t)^2 + \mathcal{O}((\Delta t)^4). \tag{20}$$

By throwing out the $\mathcal{O}((\Delta t)^4)$ term, we obtain a recursion formula to compute $\mathbf{x}^{3N}(t_0 + \Delta t)$, $\mathbf{x}^{3N}(t_0 + 2\Delta t)$, ... successively, which is the Verlet [15] algorithm. The velocities do not participate in the recursion but are needed for property calculations. They can be approximated by

$$\mathbf{v}_i(t_0) \equiv \dot{\mathbf{x}}_i(t_0) = \frac{1}{2\Delta t}\left[\mathbf{x}_i(t_0 + \Delta t) - \mathbf{x}_i(t_0 - \Delta t)\right] + \mathcal{O}((\Delta t)^2). \tag{21}$$

To what degree does the outcome of the above recursion mimic the real trajectory $\mathbf{x}^{3N}(t)$? Notice that in (20), assuming $\mathbf{x}_i(t_0)$ and $\mathbf{x}_i(t_0 - \Delta t)$ are exact, and assuming we have a perfect computer with no machine error storing the numbers or carrying out floating-point operations, the computed $\mathbf{x}_i(t_0+\Delta t)$ would still be off from the real $\mathbf{x}_i(t_0 + \Delta t)$ by $\mathcal{O}((\Delta t)^4)$, which is defined as the *local truncation error* (LTE). LTE is an intrinsic error of the algorithm. Clearly, as $\Delta t \to 0$, LTE $\to 0$, but that does not guarantee the algorithm works, because what we want is $\mathbf{x}^{3N}(t_0 + t')$ for a given t', not $\mathbf{x}_i(t_0 + \Delta t)$. To obtain $\mathbf{x}^{3N}(t_0 + t')$, we must integrate $L = t'/\Delta t$ steps, and the difference between the computed $\mathbf{x}^{3N}(t_0 + t')$ and the real $\mathbf{x}^{3N}(t_0 + t')$ is called the *global error*. An algorithm can be useful only if when $\Delta t \to 0$, the global error $\to 0$. Usually (but with exceptions), if LTE in position is $\sim(\Delta t)^{k+1}$, the global error in position should be $\sim(\Delta t)^k$, in which case we call the algorithm a k-th order method.

This is only half the story because the order of an algorithm only characterizes its performance when $\Delta t \to 0$. To save computational cost, most often one must adopt a quite large Δt. Higher-order algorithms do not necessarily perform better than lower-order algorithms at practical Δt's. In fact, they could be much worse by diverging spuriously (causing `overflow` and `NaN`), while a more robust method would just give a finite but manageable error for the same Δt. This is the concept of the *stability* of a numerical algorithm. In linear ODEs, the global error e of a certain normal mode k can always be written as $e(\omega_k \Delta t, T/\Delta t)$ by dimensional analysis, where ω_k is the mode's angular frequency. One then can define the *stability domain* of an algorithm in the $\omega\Delta t$ complex plane as the border where $e(\omega_k \Delta t, T/\Delta t)$ starts to grow exponentially as a function of $T/\Delta t$. To rephrase, a higher-order algorithm may have a much smaller stability domain than the lower-order algorithm even though its e decays faster near the origin. Since e is usually larger for larger $|\omega_k \Delta t|$, the overall quality of an integration should be characterized by $e(\omega_{\max} \Delta t, T/\Delta t)$ where $\omega_{\max}$ is the maximum intrinsic frequency of the molecular dynamics system that we explicitly integrate. The main reason behind developing constraint MD [1, 8] for some molecules is so that we do not have to integrate their stiff intra-molecular vibrational modes, allowing one to take a larger Δt, so one can follow longer the "softer modes" that we are more interested in. This is also the rationale behind developing multiple time step integrators like r-RESPA [11].

In addition to LTE, there is *round-off error* due to the computer's finite precision. The effect of round-off error can be better understood in the stability domain: (1) In most applications, the round-off error $\ll$ LTE, but it behaves like white noise which has a very wide frequency spectrum, and so for the algorithm to be stable at all, its stability domain must include the entire real $\omega\Delta t$ axis. However, as long as we ensure non-positive gain for all real $\omega\Delta t$ modes, the overall error should still be characterized by $e(\omega_k \Delta t, T/\Delta t)$, since the white noise has negligible amplitude. (2) Some applications, especially those involving high-order algorithms, do push the machine precision limit. In those cases, equating LTE $\sim \epsilon$ where ϵ is the machine's relative accuracy, provides a practical lower bound to Δt, since by reducing Δt one can no longer reduce (and indeed would increase) the global error. For single-precision arithmetics (4 bytes to store one real number), $\epsilon \sim 10^{-8}$; for double-precision arithmetics (8 bytes to store one real number), $\epsilon \approx 2.2 \times 10^{-16}$; for quadruple-precision arithmetics (16 bytes to store one real number), $\epsilon \sim 10^{-32}$.

2.2. Leap-frog Algorithm

Here we start out with $\mathbf{v}^{3N}(t_0 - \Delta t/2)$ and $\mathbf{x}^{3N}(t_0)$, then,

$$\mathbf{v}_i\left(t_0 + \frac{\Delta t}{2}\right) = \mathbf{v}_i\left(t_0 - \frac{\Delta t}{2}\right) + \left(\frac{\mathbf{f}_i(t_0)}{m_i}\right)\Delta t, \tag{22}$$

followed by,

$$\mathbf{x}_i(t_0 + \Delta t) = \mathbf{x}_i(t_0) + \mathbf{v}_i\left(t_0 + \frac{\Delta t}{2}\right)\Delta t, \tag{23}$$

and we have advanced by one step.

The velocity at time t_0 can be approximated by,

$$\mathbf{v}_i(t_0) = \frac{1}{2}\left[\mathbf{v}_i\left(t_0 - \frac{\Delta t}{2}\right) + \mathbf{v}_i\left(t_0 + \frac{\Delta t}{2}\right)\right] + \mathcal{O}((\Delta t)^2). \tag{24}$$

2.3. Velocity Verlet Algorithm

We start out with $\mathbf{x}^{3N}(t_0)$ and $\mathbf{v}^{3N}(t_0)$, then,

$$\mathbf{x}_i(t_0 + \Delta t) = \mathbf{x}_i(t_0) + \mathbf{v}_i(t_0)\Delta t + \frac{1}{2}\left(\frac{\mathbf{f}_i(t_0)}{m_i}\right)(\Delta t)^2, \tag{25}$$

evaluate $\mathbf{f}^{3N}(t_0 + \Delta t)$, and then,

$$\mathbf{v}_i\,(t_0 + \Delta t) = \mathbf{v}_i(t_0) + \frac{1}{2}\left[\frac{\mathbf{f}_i(t_0)}{m_i} + \frac{\mathbf{f}_i(t_0 + \Delta t)}{m_i}\right]\Delta t, \tag{26}$$

and we have advanced by one step. Since we can have $\mathbf{x}^{3N}(t_0)$ and $\mathbf{v}^{3N}(t_0)$ simultaneously, it is popular.

2.4. Beeman Algorithm

It is similar to the velocity Verlet algorithm. We start out with $\mathbf{x}^{3N}(t_0)$, $\mathbf{f}^{3N}(t_0 - \Delta t)$, $\mathbf{f}^{3N}(t_0)$ and $\mathbf{v}^{3N}(t_0)$, then,

$$\mathbf{x}_i(t_0 + \Delta t) = \mathbf{x}_i(t_0) + \mathbf{v}_i(t_0)\Delta t + \left[\frac{4\mathbf{f}_i(t_0) - \mathbf{f}_i(t_0 - \Delta t)}{m_i}\right]\frac{(\Delta t)^2}{6}, \quad (27)$$

evaluate $\mathbf{f}^{3N}(t_0 + \Delta t)$, and then,

$$\mathbf{v}_i(t_0 + \Delta t) = \mathbf{v}_i(t_0) + \left[\frac{2\mathbf{f}_i(t_0 + \Delta t) + 5\mathbf{f}_i(t_0) - \mathbf{f}_i(t_0 - \Delta t)}{m_i}\right]\frac{\Delta t}{6}, \quad (28)$$

and we have advanced by one step. Note that just like the leap-frog and velocity Verlet algorithms, the Beeman algorithm gives identical trajectory as the Verlet algorithm [1,14] in the absence of machine error, with 4th-order LTE in position. However, it gives better velocity estimate (3rd-order LTE) than the leap-frog or velocity Verlet (2nd-order LTE). The best velocity estimate (4th-order LTE) can be achieved by the so-called velocity-corrected Verlet algorithm [14], but it requires knowing the next two steps' positions.

2.5. Predictor-corrector Algorithm

Let us take the often used 6-value predictor-corrector algorithm [10] as an example. We start out with $6 \times 3N$ storage: $\mathbf{x}^{3N(0)}(t_0)$, $\mathbf{x}^{3N(1)}(t_0)$, $\mathbf{x}^{3N(2)}(t_0)$, $\ldots$, $\mathbf{x}^{3N(5)}(t_0)$, where $\mathbf{x}^{3N(k)}(t)$ is defined by,

$$\mathbf{x}_i^{(k)}(t) \equiv \left(\frac{\mathrm{d}^k \mathbf{x}_i^{(t)}}{\mathrm{d}t^k}\right)\left(\frac{(\Delta t)^k}{k!}\right). \quad (29)$$

The iteration consists of prediction, evaluation, and correction steps:

2.5.1. Prediction step

$$\begin{aligned}
\mathbf{x}_i^{(0)} &= \mathbf{x}_i^{(0)} + \mathbf{x}_i^{(1)} + \mathbf{x}_i^{(2)} + \mathbf{x}_i^{(3)} + \mathbf{x}_i^{(4)} + \mathbf{x}_i^{(5)},\\
\mathbf{x}_i^{(1)} &= \mathbf{x}_i^{(1)} + 2\mathbf{x}_i^{(2)} + 3\mathbf{x}_i^{(3)} + 4\mathbf{x}_i^{(4)} + 5\mathbf{x}_i^{(5)},\\
\mathbf{x}_i^{(2)} &= \mathbf{x}_i^{(2)} + 3\mathbf{x}_i^{(3)} + 6\mathbf{x}_i^{(4)} + 10\mathbf{x}_i^{(5)},\\
\mathbf{x}_i^{(3)} &= \mathbf{x}_i^{(3)} + 4\mathbf{x}_i^{(4)} + 10\mathbf{x}_i^{(5)},\\
\mathbf{x}_i^{(4)} &= \mathbf{x}_i^{(4)} + 5\mathbf{x}_i^{(5)}. \quad (30)
\end{aligned}$$

The general formula for the above is

$$\mathbf{x}_i^{(k)} = \sum_{k'=k}^{M-1} \left[\frac{k'!}{(k'-k)!k!} \right] \mathbf{x}_i^{(k')}, \quad k = 0, \ldots, M-2, \tag{31}$$

with $M = 6$ here. The evaluation must proceed from 0 to $M - 2$ sequentially.

2.5.2. *Evaluation step*

Evaluate force $\mathbf{f}^{3N}$ using the newly obtained $\mathbf{x}^{3N(0)}$.

2.5.3. *Correction step*

Define the error $\mathbf{e}^{3N}$ as,

$$\mathbf{e}_i \equiv \mathbf{x}_i^{(2)} - \left(\frac{\mathbf{f}_i}{m_i} \right) \left(\frac{(\Delta t)^2}{2!} \right). \tag{32}$$

Then apply corrections,

$$\mathbf{x}_i^{(k)} = \mathbf{x}_i^{(k)} - C_{Mk}\mathbf{e}_i, \quad k = 0, \ldots, M-1, \tag{33}$$

where C_{Mk} are constants listed in Table 2.

It is clear that the LTE for $\mathbf{x}^{3N}$ is $\mathcal{O}((\Delta t)^M)$ after the prediction step. But one can show that the LTE is reduced to $\mathcal{O}((\Delta t)^{M+1})$ after the correction step if $\mathbf{f}^{3N}$ depends on $\mathbf{x}^{3N}$ only, and not on the velocity. And so the global error would be $\mathcal{O}((\Delta t)^M)$.

2.6. Symplectic Integrators

In the absence of round-off error, certain numerical integrators *rigorously* maintain the phase-space volume conservation property (Liouville's theorem) of Hamiltonian dynamics, which are then called symplectic integrators. This

Table 2. Gear predictor-corrector coefficients

C_{Mk}	$k=0$	$k=1$	$k=2$	$k=3$	$k=4$	$k=5$	$k=6$	$k=7$
$M=4$	1/6	5/6	1	1/3				
$M=5$	19/120	3/4	1	1/2	1/12			
$M=6$	3/20	251/360	1	11/18	1/6	1/60		
$M=7$	863/6048	665/1008	1	25/36	35/144	1/24	1/360	
$M=8$	1925/14112	19087/30240	1	137/180	5/16	17/240	1/120	1/2520

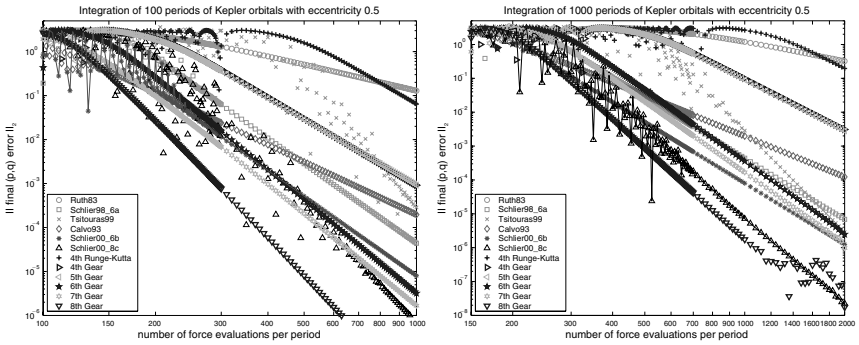

Figure 6. (a) Phase error after integrating 100 periods of Kepler orbitals. (b) Phase error after integrating 1000 periods of Kepler orbitals.

property severely limits the possibilities of mapping from initial to final states, and for this reason symplectic integrators tend to have much better total energy conservation in the long run. The velocity Verlet algorithm is in fact symplectic, followed by higher-order extensions [16, 17].

We have benchmarked the two families of integrators (Fig. 6) by numerically solving the two-body Kepler's problem (eccentricity 0.5) which is nonlinear and periodic, and comparing with the exact analytical solution. The two families have different global error versus time characteristics: non-symplectic integrators tend to have linear energy error ($\Delta E \propto t$) and quadratic phase error ($|\Delta\Gamma| \propto t^2$), while symplectic integrators tend to have constant (fluctuating) energy error ($\Delta E \propto t^0$) and linear phase error ($|\Delta\Gamma| \propto t$), with respect to time. Therefore the long-term performance of a symplectic tends to be asymptotically superior to that of a non-symplectic integrator. But, it is found that for a reasonable integration duration, say 100 Kepler periods, high-order predictor-corrector integrators can have a better performance than the best of the current symplectic integrators at *large* integration timestep (small number of force evaluations per period). This is important, because it means that in a real condensed-matter system if one does not care about the correlation of a mode beyond 100 oscillation periods, then high-order predictor-corrector algorithms can achieve the desired accuracy at a lower computational cost.

3. Order-N MD Simulation With Short-Range Potential

We outline here a linked-bin algorithm that allows one to perform MD simulation in a PBC supercell with $\mathcal{O}(N)$ computational effort per timestep, where N is the number of atoms in the supercell (Fig. 7). Such approach

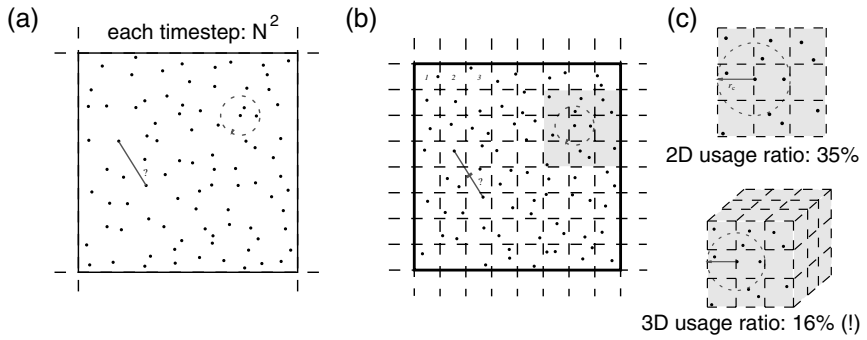

Figure 7. There are N atoms in the supercell. (a) The circle around a particular atom with radius r_c indicates the range of its interaction with other atoms. (b) The supercell is divided into a number of bins, which have dimensions such that an atom can only possibly interact with atoms in adjacent 27 bins in 3D (9 in 2D). (c) This shows that an atom–atom list is still necessary because on average there are only 16% of the atoms in 3D in adjacent bins that interact with the particular atom.

is found to outperform the brute-force Verlet neighbor-list update algorithm, which is $\mathcal{O}(N^2)$, when N exceeds a few thousand atoms. The algorithm to be introduced here allows for arbitrary supercell deformation during a simulation, and is implemented in large-scale MD and conjugate gradient relaxation programs as well as a visualization program [3].

Denote the three edges of a supercell in Cartesian frame by row vectors $\mathbf{h}_1$, $\mathbf{h}_2$, $\mathbf{h}_3$, which stack together to form a 3×3 matrix $\mathbf{H}$. The inverse of the $\mathbf{H}$ matrix $\mathbf{B} \equiv \mathbf{H}^{-1}$ satisfies

$$\mathbf{I} = \mathbf{HB} = \mathbf{BH}. \tag{34}$$

If we define row vectors

$$\mathbf{b}_1 \equiv (B_{11}, B_{21}, B_{31}), \quad \mathbf{b}_2 \equiv (B_{12}, B_{22}, B_{32}), \quad \mathbf{b}_3 \equiv (B_{13}, B_{23}, B_{33}), \tag{35}$$

then (34) is equivalent to

$$\mathbf{h}_i \cdot \mathbf{b}_j \equiv \mathbf{h}_i \mathbf{b}_j^T = \delta_{ij}. \tag{36}$$

Since $\mathbf{b}_1$ is perpendicular to both $\mathbf{h}_2$ and $\mathbf{h}_3$, it must be collinear with the normal direction $\mathbf{n}$ of the plane spanned by $\mathbf{h}_2$ and $\mathbf{h}_3$: $\mathbf{b}_1 \equiv |\mathbf{b}_1|\mathbf{n}$. And so by (36),

$$1 = \mathbf{h}_1 \cdot \mathbf{b}_1 = \mathbf{h}_1 \cdot (|\mathbf{b}_1|\mathbf{n}) = |\mathbf{b}_1|(\mathbf{h}_1 \cdot \mathbf{n}). \tag{37}$$

But $|\mathbf{h}_1 \cdot \mathbf{n}|$ is nothing other than the *thickness* of the supercell along the $\mathbf{h}_1$ edge. Therefore, the thicknesses (distances between two parallel surfaces) of the supercell are,

$$d_1 = \frac{1}{|\mathbf{b}_1|},\ d_2 = \frac{1}{|\mathbf{b}_2|},\ d_3 = \frac{1}{|\mathbf{b}_3|}. \tag{38}$$

The position of atom i is specified by a row vector, $\mathbf{s}_i = (s_{i1}, s_{i2}, s_{i3})$, with $s_{i\mu}$ satisfying

$$0 \leq s_{i\mu} < 1,\ \mu = 1, \ldots, 3, \tag{39}$$

and the Cartesian coordinate of this atom, $\mathbf{x}_i$, also a row vector, is

$$\mathbf{x}_i = s_{i1}\mathbf{h}_1 + s_{i2}\mathbf{h}_2 + s_{i3}\mathbf{h}_3 = \mathbf{s}_i\mathbf{H}, \tag{40}$$

where $s_{i\mu}$ has the geometrical interpretation of the fraction of the μth edge in order to construct $\mathbf{x}_i$. We will simulate particle systems that interact via short-range potentials of cutoff radius r_c (see previous section for potential truncation schemes). In the case of multi-component system, r_c is generalized to a matrix $r_c^{\alpha\beta}$, where $\alpha \equiv c(i)$, $\beta \equiv c(j)$ are the chemical types of atom i and j, respectively. We then define

$$\mathbf{x}_{ji} \equiv \mathbf{x}_j - \mathbf{x}_i, \quad r_{ji} \equiv |\mathbf{x}_{ji}|, \quad \hat{\mathbf{x}}_{ji} \equiv \frac{\mathbf{x}_{ji}}{r_{ji}}. \tag{41}$$

The design of the program should allow for arbitrary changes in $\mathbf{H}$ that include strain and rotational components (see Chap. 2.19). One should use the Lagrangian strain $\boldsymbol{\eta}$, a true rank-2 tensor under coordinate frame transformation, to measure the deformation of a supercell. To define $\boldsymbol{\eta}$, one needs a reference $\mathbf{H}_0$ of a previous time, with $\mathbf{x}_0 = \mathbf{s}\mathbf{H}_0$ and $\mathrm{d}\mathbf{x}_0 = (\mathrm{d}\mathbf{s})\mathbf{H}_0$, and imagine that with $\mathbf{s}$ fixed, $\mathrm{d}\mathbf{x}_0$ is transformed to $\mathrm{d}\mathbf{x} = (\mathrm{d}\mathbf{s})\mathbf{H}$, under $\mathbf{H}_0 \to \mathbf{H} \equiv \mathbf{H}_0\mathbf{J}$.

The Lagrangian strain is defined by the change in the differential line length,

$$\mathrm{d}l^2 = \mathrm{d}\mathbf{x}\,\mathrm{d}\mathbf{x}^{\mathrm{T}} \equiv \mathrm{d}\mathbf{x}_0(\mathbf{I} + 2\boldsymbol{\eta})\mathrm{d}\mathbf{x}_0^{\mathrm{T}}, \tag{42}$$

where by plugging in $\mathrm{d}\mathbf{x} = (\mathrm{d}\mathbf{s})\mathbf{H} = (\mathrm{d}\mathbf{x}_0)\mathbf{H}_0^{-1}\mathbf{H} = (\mathrm{d}\mathbf{x}_0)\mathbf{J}$, $\boldsymbol{\eta}$ is seen to be

$$\boldsymbol{\eta} = \tfrac{1}{2}\left(\mathbf{H}_0^{-1}\mathbf{H}\mathbf{H}^T\mathbf{H}_0^{-T} - \mathbf{I}\right) = \tfrac{1}{2}\left(\mathbf{J}\mathbf{J}^T - \mathbf{I}\right). \tag{43}$$

Because $\boldsymbol{\eta}$ is a symmetric matrix, it always has three mutually orthogonal eigen-directions $\mathbf{x}_1\boldsymbol{\eta} = \mathbf{x}_1\eta_1$, $\mathbf{x}_2\boldsymbol{\eta} = \mathbf{x}_2\eta_2$, $\mathbf{x}_3\boldsymbol{\eta} = \mathbf{x}_3\eta_3$. Along those directions, the line lengths are changed by factors $\sqrt{1+2\eta_1}$, $\sqrt{1+2\eta_2}$, $\sqrt{1+2\eta_3}$, which achieve extrema among all line directions. Thus, as long as η_1, η_2 and η_3 oscillate between $[-\eta_{\mathrm{bound}}, \eta_{\mathrm{bound}}]$ for some chosen η_{bound}, any line segment at $\mathbf{H}_0$ can be scaled by no more than $\sqrt{1+2\eta_{\mathrm{bound}}}$ and no less than $\sqrt{1-2\eta_{\mathrm{bound}}}$. That is, if we define length measure

$$L(\Delta\mathbf{s}, \mathbf{H}) \equiv \sqrt{\Delta\mathbf{s}\mathbf{H}\mathbf{H}^T\Delta\mathbf{s}^T}, \tag{44}$$

then so long as η_1, η_2, η_3 oscillate between $[\eta_{\min}, \eta_{\max}]$, there is

$$\sqrt{1+2\eta_{\min}}\, L(\Delta\mathbf{s}, \mathbf{H}_0) \le L(\Delta\mathbf{s}, \mathbf{H}) \le \sqrt{1+2\eta_{\max}}\, L(\Delta\mathbf{s}, \mathbf{H}_0). \tag{45}$$

One can use the above result to define a *strain session*, which begins with $\mathbf{H}_0 = \mathbf{H}$ and during which no line segment is allowed to shrink by less than a threshold $f_c \le 1$, compared to its length at $\mathbf{H}_0$. This is equivalent to requiring that,

$$f \equiv \sqrt{1+2\,(\min(\eta_1, \eta_2, \eta_3))} \;\le\; f_c. \tag{46}$$

Whenever the above condition is violated, the session terminates and a new session starts with the present $\mathbf{H}$ as the new $\mathbf{H}_0$, and triggers a repartitioning of the supercell into equally-sized bins, which is called a strain-induced bin repartitioning.

The purpose of bin partition (see Fig. 7) is the following: it can be a very demanding task to determine if atoms i, j are within r_c or not, for all possible ij combinations. Formally, this requires checking

$$r_{ji} \equiv L(\Delta\mathbf{s}_{ji}, \mathbf{H}) \le r_c. \tag{47}$$

Because $\mathbf{s}_i$, $\mathbf{s}_j$ and $\mathbf{H}$ are in general all moving – they differ from step to step, it appears that we have to check this at each step. This would indeed be the case but for the observation that, in most MD, MC and static minimization procedures, $\mathbf{s}_i$'s of most atoms and $\mathbf{H}$ often change only slightly from the previous step. Therefore, once we ensured that (47) held at some previous step, we can devise a *sufficient condition* to test if (47) still must hold now, at a much smaller cost. Only when this sufficient condition breaks down do we resort to a more complicated search and check in the fashion of (47).

As a side note, it is often more efficient to count interaction *pairs* if the potential function allows for easy use of such half-lists, such as pair- or EAM potentials, which achieves 50% saving in memory. In these scenarios we pick a unique "host" atom among i and j to store the information about the ij-pair, that is, a particle's list only keeps possible pairs that are under its own care. For load-balancing it is best if the responsibilities are distributed evenly among particles. We use a pseudo-random choice of: if $i + j$ is odd and $i > j$, or if $i + j$ is even and $i < j$, then i is the host; otherwise it is j. As $i > j$ is "uncorrelated" with whether $i + j$ is even or odd, significant load imbalance is unlikely to occur even if the indices correlate strongly with the atoms' positions.

The step-to-step small change is exploited as follows: one associates each $\mathbf{s}_i$ with a semi-mobile reduced coordinate $\mathbf{s}_i^a$ called atom i's *anchor* (Fig. 8). At each step, one checks if $L(\mathbf{s}_i - \mathbf{s}_i^a, \mathbf{H})$, that is, the current distance between i and its anchor, is greater than a certain $r_{\text{drift}} \ge r_{\text{drift}}^0$ or not. If it is not, then $\mathbf{s}_i^a$ does not change; if it is, then one redefines $\mathbf{s}_i^a \equiv \mathbf{s}_i$ at this step, which is called

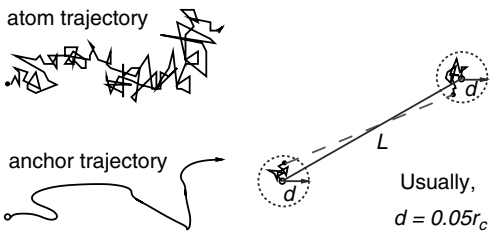

Figure 8. This illustrates the concepts of an anchor, which is the relative immbobile part of an atom's trajectory. Using an anchor–anchor list, we can derive a "flash" condition that locally updates an atom's neighbor-list when the atom drifts sufficiently far away from its anchor.

atom i's *flash* incident. At atom i's flash, it is required to update records of all atoms (part of the records may be stored in j's list, if 50%-saving is used and j happens to be the host of the ij pair) whose anchors satisfy

$$L(\mathbf{s}_j^a - \mathbf{s}_i^a, \mathbf{H}_0) \leq r_{\text{list}} \equiv \frac{r_{\text{c}} + 2r_{\text{drift}}^0}{f_{\text{c}}}. \tag{48}$$

Note that the distance is between *anchors* instead of *atoms* ($\mathbf{s}_i^a = \mathbf{s}_i$, though), and the length is measured by $\mathbf{H}_0$, not the current $\mathbf{H}$. (48) nominally takes $\mathcal{O}(N)$ work per flash, but we may reduce it to $\mathcal{O}(1)$ work per flash by partitioning the supercell into $m_1 \times m_2 \times m_3$ bins at the start of the session, whose thicknesses by $\mathbf{H}_0$ (see (38)) are required to be greater than or equal to r_{list}:

$$\frac{d_1(\mathbf{H}_0)}{m_1}, \ \frac{d_2(\mathbf{H}_0)}{m_2}, \ \frac{d_3(\mathbf{H}_0)}{m_3} \geq r_{\text{list}}. \tag{49}$$

The bins deform with $\mathbf{H}$ and remains commensurate with it, that is, its $\mathbf{s}$-width $1/m_1$, $1/m_2$, $1/m_3$ remains fixed during a strain session. Each bin keeps an updated list of all *anchors* inside. When atom i flashes, it also updates the bin-anchor list if necessary. Then, if at the time of i's flash two anchors are separated by more than one bin, there would be

$$L(\mathbf{s}_j^a - \mathbf{s}_i^a, \mathbf{H}_0) > \frac{d_1(\mathbf{H}_0)}{m_1}, \ \frac{d_2(\mathbf{H}_0)}{m_2}, \ \frac{d_3(\mathbf{H}_0)}{m_3} \geq r_{\text{list}}, \tag{50}$$

and they cannot possibly satisfy (48). Therefore we only need to test (48) for anchors within adjacent 27 bins. To synchronize, all atoms flash at the start of a strain session. From then on, atoms flash individually whenever $L(\mathbf{s}_i - \mathbf{s}_i^a, \mathbf{H}) > r_{\text{drift}}$. If two anchors flash at the same step in a loop, the first flash may get it wrong – that is, missing the second anchor, but the second flash will correct the mistake. The important thing here is not to *lose* an interaction. We see that to maintain anchor lists that captures all solutions to (48) among the latest anchors, it takes only $\mathcal{O}(N)$ work per step, and the pre-factor of which is also small because flash events happen quite infrequently for a tolerably large r_{drift}^0.

The central claim of the scheme is that if j is not in i's anchor records (suppose i's last flash is more recent than j's), which was created *some time ago* in the strain session, then $r_{ji} > r_c$. The reason is that the *current* separation between the anchor i and anchor j, $L(\mathbf{s}_j^a - \mathbf{s}_i^a, \mathbf{H})$, is greater than $r_c + 2r_{\text{drift}}^0$, since by (45), (46) and (48),

$$L(\mathbf{s}_j^a - \mathbf{s}_i^a, \mathbf{H}) \geq f \cdot L(\mathbf{s}_j^a - \mathbf{s}_i^a, \mathbf{H}_0) > f \cdot r_{\text{list}} \geq f_c \cdot r_{\text{list}} = f_c \cdot \frac{r_c + 2r_{\text{drift}}^0}{f_c}. \tag{51}$$

So we see that $r_{ji} > r_c$ maintains if neither i or j currently drifts more than

$$r_{\text{drift}} \equiv \frac{f \cdot r_{\text{list}} - r_c}{2} \geq r_{\text{drift}}^0, \tag{52}$$

from respective anchors. Put it another way, when we design r_{list} in (48), we take into consideration both atom drifts and $\mathbf{H}$ shrinkage which both may bring ij closer than r_c, but since the current $\mathbf{H}$ shrinkage has not yet reached the designed critical value, we can convert it to more leeway for the atom drifts.

For multi-component systems, we define

$$r_{\text{list}}^{\alpha\beta} \equiv \frac{r_c^{\alpha\beta} + 2r_{\text{drift}}^0}{f_c}, \tag{53}$$

where both f_c and r_{drift}^0 are species-independent constants, and r_{drift}^0 can be thought of as putting a lower bound on r_{drift}, so flash events cannot occur too frequently. At each bin repartitioning, we would require

$$\frac{d_1(\mathbf{H}_0)}{m_1}, \ \frac{d_2(\mathbf{H}_0)}{m_2}, \ \frac{d_3(\mathbf{H}_0)}{m_3} \geq \max_{\alpha,\beta} r_{\text{list}}^{\alpha\beta}. \tag{54}$$

And during the strain session, $f \geq f_c$, we have

$$r_{\text{drift}}^{\alpha} \equiv \min\left[\min_{\beta}\left(\frac{f \cdot r_{\text{list}}^{\alpha\beta} - r_c^{\alpha\beta}}{2}\right), \ \min_{\beta}\left(\frac{f \cdot r_{\text{list}}^{\beta\alpha} - r_c^{\beta\alpha}}{2}\right)\right], \tag{55}$$

a time- and species-dependent atom drift bound that controls whether an atom of species α needs to flash.

4. Molecular Dynamics Codes

At present there are several high-quality molecular dynamics programs in the public domain, such as LAMMPS [18], DL_POLY [19, 20], Moldy [21], IMD [22, 23], and some codes with biomolecular focus, such as NAMD [24, 25] and Gromacs [26, 27]. CHARMM [28] and AMBER [29] are not free but are standard and extremely powerful codes in biology.

References

[1] M. Allen and D. Tildesley, *Computer Simulation of Liquids*, Clarendon Press, New York, 1987.

[2] J. Li, L. Porter, and S. Yip, "Atomistic modeling of finite-temperature properties of crystalline beta-SiC - II. Thermal conductivity and effects of point defects," *J. Nucl. Mater.*, 255, 139–152, 1998.

[3] J. Li, "AtomEye: an efficient atomistic configuration viewer," *Model. Simul. Mater. Sci. Eng.*, 11, 173–177, 2003.

[4] D. Chandler, *Introduction to Modern Statistical Mechanics*, Oxford University Press, New York, 1987.

[5] M. Born and K. Huang, *Dynamical Theory of Crystal Lattices*, 2nd edn., Clarendon Press, Oxford, 1954.

[6] R. Parr and W. Yang, *Density-functional Theory of Atoms and Molecules*, Clarendon Press, Oxford, 1989.

[7] S.D. Ivanov, A.P. Lyubartsev, and A. Laaksonen, "Bead-Fourier path integral molecular dynamics," *Phys. Rev. E*, 67, art. no.–066710, 2003.

[8] T. Schlick, *Molecular Modeling and Simulation*, Springer, Berlin, 2002.

[9] W. Press, B. Flannery, S. Teukolsky, and W. Vetterling, *Numerical Recipes in C: the Art of Scientific Computing*, 2nd edn., Cambridge University Press, Cambridge, 1992.

[10] C. Gear, *Numerical Initial Value Problems in Ordinary Differential Equation*, Prentice-Hall, Englewood Cliffs, NJ, 1971.

[11] M.E. Tuckerman and G.J. Martyna, "Understanding modern molecular dynamics: techniques and applications," *J. Phys. Chem. B*, 104, 159–178, 2000.

[12] S. Nose, "A unified formulation of the constant temperature molecular dynamics methods," *J. Chem. Phys.*, 81, 511–519, 1984.

[13] W.G. Hoover, "Canonical dynamics – equilibrium phase-space distributions," *Phys. Rev. A*, 31, 1695–1697, 1985.

[14] D. Frenkel and B. Smit, "Understanding molecular simulation: from algorithms to applications," 2nd ed., Academic Press, San Diego, 2002.

[15] L. Verlet, "Computer "experiments" on classical fluids. I. Thermodynamical properties of Lennard–Jones molecules," *Phys. Rev.*, 159, 98–103, 1967.

[16] H. Yoshida, "Construction of higher-order symplectic integrators," *Phys. Lett. A*, 150, 262–268, 1990.

[17] J. Sanz-Serna and M. Calvo, *Numerical Hamiltonian Problems*, Chapman & Hall, London, 1994.

[18] S. Plimpton, "Fast parallel algorithms for short-range molecular-dynamics," *J. Comput. Phys.*, 117, 1–19, 1995.

[19] W. Smith and T.R. Forester, "DL_POLY_2.0: a general-purpose parallel molecular dynamics simulation package," *J. Mol. Graph.*, 14, 136–141, 1996.

[20] W. Smith, C.W. Yong, and P.M. Rodger, "DL_POLY: application to molecular simulation," *Mol. Simul.*, 28, 385–471, 2002.

[21] K. Refson, "Moldy: a portable molecular dynamics simulation program for serial and parallel computers," *Comput. Phys. Commun.*, 126, 310–329, 2000.

[22] J. Stadler, R. Mikulla, and H.-R. Trebin, "IMD: A Software Package for Molecular Dynamics Studies on Parallel Computers," *Int. J. Mod. Phys. C*, 8, 1131–1140, 1997.

[23] J. Roth, F. Gähler, and H.-R. Trebin, "A molecular dynamics run with 5,180,116,000 particles," *Int. J. Mod. Phys. C*, 11, 317–322, 2000.

[24] M.T. Nelson, W. Humphrey, A. Gursoy, A. Dalke, L.V. Kale, R.D. Skeel, and K. Schulten, "NAMD: a parallel, object oriented molecular dynamics program," *Int. J. Supercomput. Appl. High Perform. Comput.*, 10, 251–268, 1996.

[25] L. Kale, R. Skeel, M. Bhandarkar, R. Brunner, A. Gursoy, N. Krawetz, J. Phillips, A. Shinozaki, K. Varadarajan, and K. Schulten, "NAMD2: Greater scalability for parallel molecular dynamics," *J. Comput. Phys.*, 151, 283–312, 1999.

[26] H.J.C. Berendsen, D. Vanderspoel, and R. Vandrunen, "Gromacs – a message-passing parallel molecular-dynamics implementation," *Comput. Phys. Commun.*, 91, 43–56, 1995.

[27] E. Lindahl, B. Hess, and D. van der Spoel, "GROMACS 3.0: a package for molecular simulation and trajectory analysis," *J. Mol. Model.*, 7, 306–317, 2001.

[28] B.R. Brooks, R.E. Bruccoleri, B.D. Olafson, D.J. States, S. Swaminathan, and M. Karplus, "Charmm – a program for macromolecular energy, minimization, and dynamics calculations," *J. Comput. Chem.*, 4, 187–217, 1983.

[29] D.A. Pearlman, D.A. Case, J.W. Caldwell, W.S. Ross, T.E. Cheatham, S. Debolt, D. Ferguson, G. Seibel, and P. Kollman, "Amber, a package of computer-programs for applying molecular mechanics, normal-mode analysis, molecular-dynamics and freeenergy calculations to simulate the structural and energetic properties of molecules," *Comput. Phys. Commun.*, 91, 1–41, 1995.