IM/S Supplementary Material on Molecular Dynamics

May 3, 2006

Choice of timestep and non-dimensional (reduced) units

In molecular simulation, reduced units are typically defined using the mass, energy and distance characteristic scale of the problem. Working with reduced units is preferable for the following reasons:

1. Suitably no-dimensionalized (in reduced units) results are applicable to all materials modelled by the same potential.
2. Reduced units can be more physically meaningful because they relate to the fundamental physics of the problem. As an example, consider the choice of integration timestep: a number such as $10^{-14}s$ is not meaningful (it may or may not be appropriate). However $0.01\sqrt{\frac{m\sigma^2}{\epsilon}} = 10^{-14}s$ is meaningful, as a small fraction of the system characteristic timescale.

In the case of the Lennard-Jones potential

$$\phi_{ij}(r) = 4\varepsilon\left[\left(\frac{\sigma}{r}\right)^{12} - \left(\frac{\sigma}{r}\right)^{6}\right]$$

the parameters ε and σ along with the mass of the molecule serve as the fundamental units of energy, length and mass respectively, which can be used to define the following

reduced units:

$$\begin{aligned}
\text{number density } n^\star &= n\sigma^3 \\
\text{temperature } T^\star &= \frac{kT}{\varepsilon} \\
\text{energy } E^\star &= \frac{E}{\varepsilon} \\
\text{pressure } P^\star &= \frac{P\sigma^3}{\varepsilon} \\
\text{time } t^\star &= \sqrt{\frac{\varepsilon}{m\sigma^2}}\ t
\end{aligned}$$

Using these units, the characteristic timescale for motion is $t^\star$, so a reasonable timestep will be $\delta t = (0.001 - 0.01)t^\star$. This is sufficiently small in the sense that a number of timesteps (100-1000) are used to discretize the characteristic timescale.

In the presence of more degrees of freedom (other than just translation e.g. bond vibrations), the timestep needs to be much smaller than the characteristic timescale of the *fastest* process. This may lead to "stiffness," whereby the significantly faster process (e.g. bond vibration) requires a much smaller timestep than the timescale of interest (e.g. fluid flow) and causes limitations in simulation or even intractability. In some cases, to avoid excessive cost, fast degrees of freedom *that do not play an important role in the physics of interest* are frozen, e.g. bonds frozen at fixed length and molecules treated as rigid (dumbbell).

Systems with non-smooth potentials or high molecular speeds will, in general, require a smaller timestep. In the limit of "hard" collisions the interaction is not treated through the equations of motion; instead momentum principles are preferable to determine the "post-collision" state (for an example see notes on the hard-sphere gas).

Clearly a compromise exists between accuracy and minimum number of timesteps. Hydrodynamic effects, or thermally activated processes (such as diffusion in a solid), are very troublesome because their characteristic timescales are much longer than those of atomic motion. This causes trouble when simulating by MD. Monte Carlo methods are one approach used to avoid these kinetic limitations. Acceleration methods is another.

Integration algorithms

Newton's equations for all particles are typically integrated using a finite difference method. In fact, integration of the equations of motion is fairly simple; the most

challenging and time-consuming part of an MD code is the calculation of the forces. For this reason it is very important for an integration algorithm to allow the use of a large timestep (minimize the number of force evaluations). It is also unreasonable to expect the integration algorithm to follow particle trajectory as closely as possible for all times: Because of the highly nonlinear nature of the many body problem, any two trajectories that are close at time t diverge exponentially in time, so even a small error (truncation) will cause the computer trajectory to diverge from the "true" trajectory.

It takes a few hundred timesteps for these two diverging trajectories to become statistically uncorrelated. Fortunately this is not so disastrous. We need MD to calculate exact trajectories only for very short times (calculate correlation functions); otherwise, our objective is to generate states that are consistent with the macroscopic constraints of the system of interest. In this sense, conservation of energy and momentum are more important.

Integration algorithms can be divided into two categories:

1. Verlet type algorithms (cannot handle velocity dependent forces)
2. Predictor-corrector algorithms

The reason for the limited "selection" is that the integration algorithm does not make a big difference. As discussed above, the most expensive part is the force calculation.

The Verlet algorithm

$$\begin{aligned} \vec{r}(t+\delta t) &= 2\vec{r}(t) - \vec{r}(t-\delta r) + \delta t^2 \vec{a}(t) + O(\delta t^4) \\ \vec{v}(t) &= \frac{\vec{r}(t+\delta t) - \vec{r}(t-\delta t)}{2\delta t} + O(\delta t^2) \end{aligned}$$

is a very simple, time-reversible algorithm that became very popular in the early days of Molecular Dynamics. The Verlet algorithm has two main disadvantages:

1. $\vec{v}(t)$ errors $O(\delta t^2)$, $\vec{v}(t)$ only known after $\vec{r}(t+\delta t)$ is known (important in thermostating, for example).
2. Imprecision from a small term ($\delta t^2 \vec{a}(t)$) added to the difference of two large terms ($2\vec{r}(t) - \vec{r}(t-\delta t)$).

A modified version which addresses the above disadvantages is the "Leap-frog" algorithm:

$$\begin{aligned} \vec{r}(t) &= \vec{r}(t+\delta t) + \delta t \vec{v}\left(t - \frac{1}{2}\delta t\right) \\ \vec{v}(t+\frac{1}{2}\delta t) &= \vec{v}\left(t - \frac{1}{2}\delta t\right) + \delta t \vec{a}(t) \\ \vec{v}(t) &= \frac{1}{2}\left(\vec{v}\left(t + \frac{1}{2}\delta t\right) + \vec{v}\left(t - \frac{1}{2}\delta t\right)\right) \end{aligned}$$

It gets its name from the fact that velocities and then coordinates leap over to provide the next step values. It addresses the truncation error issues above and the fact that velocities appear explicitly. However, note that it is algebraically equivalent to the Verlet algorithm.

Predictor-corrector algorithms are in general expected to be more accurate and also handle velocity-dependent forces that become important when rotational degrees of freedom are considered. The general algorithm for a predictor-corrector integrator is:

1. Predict positions, velocities at $t + \delta t$.
2. Evaluate the forces and accelerations from the new positions and velocities.
3. Correct the predicted positions, velocities using the new accelerations.
4. Go to 2) and repeat to convergence.

Although this type of algorithm would be expected to be significantly more accurate, this is not necessarily the case, because molecular motion is heavily influenced by the motion of neighbors. In other words, there is no reason to iterate for extra accuracy based on the current locations of all neighbors when some of those have not been updated yet. The extra cost associated with evaluation of the forces at the predicted positions makes these types of algorithms not very popular.

An algorithm that does not require the re-evaluation of the forces from the positions, but allows for velocity-dependent forces, and calculates velocities correct to $0(\delta t^3)$ in the absence of velocity dependent forces, is the Beeman algorithm shown below in its "modified" version.

1. $$\vec{r}(t+\delta t) = \vec{r}(t) + \delta t \vec{v}(t) + \frac{\delta t^2}{6}\left[4\vec{a}(t) - \vec{a}(t-\delta t)\right]$$
2. $$\vec{v}^P(t+\delta t) = \vec{v}(t) + \frac{\delta t}{2}\left[3\vec{a}(t) - \vec{a}(t-\delta t)\right]$$
3. $$\vec{a}(t+\delta t) = \frac{1}{m}\vec{F}\left(\{\vec{r}(t+\delta t),\ \vec{v}^P(t+\delta t)\}\right)$$
4. $$\vec{v}^C(t+\delta t) = \vec{v}(t) + \frac{\delta t}{6}\left[2\vec{a}(t+\delta t) + 5\vec{a}(t) - \vec{a}(t-\delta t)\right]$$
5. Replace $\vec{v}^P$ with $\vec{v}^C$ and go to 3. Iterate to convergence. Note that this algorithm does not require re-evaluation of the coordinate-dependent part of the forces (which is the expensive one).

Boundary conditions

In MD simulations the number of particles is typically limited by computational resources, and so systems are typically very small. This raises the issue of the role of surface effects on the calculation. For this reason, a very popular boundary condition is that of a periodic simulation cell that allows the system to interact with an infinite number of replicas of itself (in all directions) thus minimizing the role of surface effects.

In periodic boundary conditions the particle is usually taken to interact with only the closest images of the other $N-1$ molecules. This is called "minimum image convention." This is not necessarily the only correct approach. In other approaches the particle may be allowed to interact with its own image. However, nowadays computers are sufficiently fast that simulation cells are significantly larger than the range of the interaction potential (see below about potential truncation) and thus the question of a particle interacting with images of itself typically does not arise.

Potential truncation

The force calculation is by far the most computationally intensive part of the simulation mainly because of the large number of pairs of interacting particles. To limit the simulation cost the potential is usually truncated at some distance $r = r_c$ at which it approaches 0,

$$\phi(r_{ij}) = \begin{cases} \phi(r_{ij}) & r_{ij} \leq r_c \\ 0 & r_{ij} > r_c \end{cases}$$

i.e. only neighbors within the sphere of radius r_c are considered. For a Lennard-Jones model a typical cut-off used is $r_c = 2.5\sigma$. At this distance $\phi(r_c) = 0.016 \min\{\phi\}$.

This truncation makes the potential discontinuous at r_c. Every time a molecule crosses the $r = r_c$ boundary the total energy of the simulation changes leading to lack of energy conservation. We can avoid this by shifting the potential function by $\phi(r_c)$, that is

$$\phi(r_{ij}) = \begin{cases} \phi(r_{ij}) - \phi(r_c) & r_{ij} \leq r_c \\ 0 & r_{ij} > r_c \end{cases}$$

Similar adjustments can be made to ensure continuity of higher order derivatives of the potential function.

Neighbor lists

By implementing a cutoff radius we ensure that the force calculation is concerned with a relatively small number of neighbors and no force is calculated for $r > r_c$. Although the computational savings are appreciable, the cost still scales as N^2 (neighbor search) and it is thus prohibitive as N becomes large.

Verlet suggested a method to reduce the computational cost associated with calculation of forces: at the start of the simulation, a list is constructed of all the neighbors of each molecule for which the pair separation is $r_l = r_c + r_s$ where $r_s =$ skin distance. Over the next few time-steps the list is used in the force/energy calculation routine. If the skin distance r_s is chosen thick enough that within those timesteps no molecule penetrates to within r_c, the interactions are calculated correctly. This list is updated periodically. Clearly, the key to an efficient and accurate simulation lies in the choice of r_s such that no frequent updates are required while at the same time no molecules are omitted from the force calculation. As N grows however, the storage requirement (each molecule has an associated list) of this method becomes prohibitive.

For large systems, cell methods are used. In this method the particles are sorted into cells (organized in a regular lattice)that have sides larger than r_c. Thus, when searching for neighbors, only the current cell and neighboring cells need to be searched. For 3D calculations $27NN_c$ interactions need to be computed instead of $\frac{1}{2}N(N-1)$. Here $N_c = \frac{N}{M^3}$ where M is the number of cells per dimension. Taking advantage of Newton's third law this number can be further reduced to $13.5NN_c (\ll \frac{1}{2}N(N-1)$ for N large) interactions requiring calculation. Also note that N_c is effectively a constant and so this is an $O(N)$ method, that is, the computational savings compared to $\frac{1}{2}N(N-1)$ increases as N increases.

Statistical error estimation

Sampling of macroscopic quantities (density, temperature, velocity) in a molecular simulation is an inherently statistical process. Recall that a macroscopic property $\langle A \rangle$ was defined as a running average over a time series of microscopic data once the simulation had reached steady state

$$\langle A \rangle = \frac{1}{\Delta t} \int_{t_o}^{t_o + \Delta t} A(t')dt' = \frac{1}{M} \sum_{i=1} A_i$$

where M is the number of samples collected. The second equality arises from the fact that data are available in discrete form, typically at the end of each timestep. One would expect that as $M \to \infty$, our estimate approaches the true value of A, A_t, i.e. $\langle A \rangle \to A_t$ as $M \to \infty$. Unfortunately, an infinite number of samples means infinite

simulation time. We are thus interested in quantifying the uncertainty in our estimate of A_t. This is best formulated as a statistical question. We can analyze statistical errors by using the central limit theorem. We expect this to be reasonable because by sampling we essentially take the sum of a large number of random quantities.

Our estimate of uncertainty in evaluating A_t, is linked to the variance of the distribution of $A, \sigma^2(A)$. In fact, a corollary of the central limit theorem is that the variance in our estimate of A_t, σ_u^2, is related to $\sigma(A)$ by

$$\sigma_u^2 = \frac{\sigma^2(A)}{M_i}$$

where M_i is the number of *independent* samples taken.

This equation illustrates one of the major drawbacks associated with molecular simulation: the standard deviation in the estimate of the mean decreases with the *square root* of the number of samples taken; this slow rate of convergence is the cause of a significant part of the large computational cost associated with classical molecular dynamics methods. One additional problem lies in the fact that subsequent samples may not be independent (especially if the time between them is small) since they are part of the dynamical trajectory of the system.

Let us define s, the statistical inefficiency, as the number of samples between two statistically uncorrelated samples. Then,

$$\sigma_u^2 = \frac{\sigma^2(A)}{\frac{M}{s}}$$

Unfortunately, s is unknown. To find s we can use the fact that the uncertainty σ_u can in fact be measured by looking at how estimates of $\langle A \rangle$ vary around the distribution average i.e. by measuring the actual variance of $\langle A \rangle$. A number of estimates of $\langle A \rangle$ for calculating the variance of $\langle A \rangle, \sigma^2 \langle A \rangle$, can be formed by averaging the data over sub-blocks of length M_b

$$\langle A \rangle_b = \frac{1}{M_b} \sum_{i=1}^{M_b} A_i$$

where $\langle A \rangle_b$ denotes that the average is only over M_b. As $M_b \to \infty$ then the $m_b = \frac{M_b}{M}$ estimates of $\langle A \rangle$ will be independent and thus have variance

$$\sigma_b^2 \left(\langle A \rangle \right) = \frac{1}{m_b} \sum_{i=1}^{m_b} \left(\langle A \rangle_{bi} - \langle A \rangle \right)^2$$

that can be related to the parent distribution by

$$\sigma_b^2 \left(\langle A \rangle \right) = \frac{\sigma^2(A)}{\frac{M_b}{s}}$$

thus s can be calculated from

$$s = \lim_{M_b \to \infty} \frac{M_b \sigma_b^2 \left(\langle A \rangle \right)}{\sigma^2(A)}$$

Note that although we take the limit $M_b \to \infty$, we also need to ensure that M_b is sufficiently small that a statistically significant number of $\langle A \rangle_b$ samples are present for a reliable estimation of $\sigma_b^2 \left(\langle A \rangle \right)$ (i.e. $m_b \ggg 1$).