22.53 Elements of Molecular Dynamics

Reference: [1]: sections 2, 3 ,4. [2]: 4.4, 5.1-5.3. [3]: 3.1, 3.2, 5.2, 5.3.

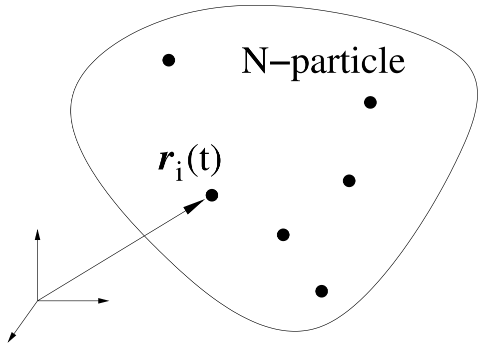

Figure 1: Illustration of the atomistic system for modeling.

Another (working) definition of MD: Techniques by which one generates the atomic trajectories of system of N particles by numerical integration of Newton's equation of motion, for a specific interatomic potential, with certain initial and boundary conditions.

Define the system: N atoms, volume Ω, temperature T (only at equilibrium).

Define **internal energy**: $E \equiv K + U$, where K is the **kinetic energy**,

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

and U is the **potential energy**,

$$U = U(\{\mathbf{r}_N(t)\}), \tag{2}$$

where we use notation $\{\mathbf{r}_N(t)\}$ to denote the collection of coordinates $\mathbf{r}_1(t), \mathbf{r}_2(t), .., \mathbf{r}_N(t)$. Note that E is a constant of motion, e.g., conserved quantity if the system is isolated.

Treat simulation like an experiment:

[system setup]	$\rightarrow$	[equilibration]	$\rightarrow$	[simulation run]	$\rightarrow$	[output]
sample selection		sample preparation		property average		data analysis
(pot., N, BC)		(achieve ρ^*, P^*, T^*)		(run N_T steps)		(property calc.)

To fully specify a simulation model in thermal equilibration (temperature T), specify a vector,

$$\Lambda \equiv (N, \Omega, T, \mathrm{BC}, \{\mathbf{r}_N, \dot{\mathbf{r}}_N(t=0)\};\ U\ ) . \tag{3}$$

In many cases, details of $\{\mathbf{r}_N, \dot{\mathbf{r}}_N(t = 0)\}$ is not critically important because simulation, which run long enough, can automatically settle to the correct phase space region. That transient process is often called **equilibration**.

For example, problem 1 (code `md.f`):

U = Lennard-Jones potential (for noble gases Ne,Ar,Kr,Xe)

$\{\mathbf{r}_N(t = 0)\}$ = fcc crystal structure

$\{\dot{\mathbf{r}}_N(t = 0)\}$ = arbitrary[1]

BC = Periodic Boundary Conditions (PBC) (no surface, bulk simulation)

$\rho = N/\Omega$, high density ([1], p. 40)

$\rho_R = \rho\sigma^3 \equiv$ reduced density (> 1.0 for a solid)

$T_R = k_{\mathrm{B}}T/\epsilon \equiv$ reduced temperature ≤ 0.5

Equation of motion (Newtonian):

$$\mathbf{F} = m_i\mathbf{a}_i, \tag{4}$$

and so,

$$m_i\frac{d^2\mathbf{r}_i(t)}{dt^2} = \mathbf{F}_i^{int} + \mathbf{F}_i^{ext} \equiv \mathbf{F}_i, \quad i = 1..N. \tag{5}$$

These are second-order ordinary differential equations (ODE's), which can be strongly non-linear. By converting them to first-order equations in the space of $\{\mathbf{r}_N, \dot{\mathbf{r}}_N(t)\}$, general numerical algorithms for solving ODE's (here on called integrators) such as the Runge-Kutta

[1] Of course, $\{\dot{\mathbf{r}}_N(t = 0)\}$ should not be too big to permanently damage the assigned solid structure by temporary melting *if we do not intend to do that*. Independently and uniformly distributed initial velocities works fine; Gaussian (Maxwell) distribution works even better. If one starts with $\{\mathbf{r}_N(t = 0)\}$ at equilibrium sites, it is usually more effective to assign initial velocities such that $\langle K\rangle/N = 3k_BT$, which is *twice* the kinetic energy an atom should have if at equilibrium, since harmonic oscillators have $\langle U\rangle = \langle K\rangle$ in the long run.

method [5] can be applied. But such general methods are rarely used in practice because the special dynamics of the phase flow due to the existence of a **Hamiltonian** allows for more efficient algorithms, prominent among which are the family of predictor-corrector integrators [6] and the family of symplectic integrators [10, 11, 12, 13].

Integrate over small time-interval Δt:

$$\{\mathbf{r}_N(t_0)\} \;\rightarrow\; \{\mathbf{r}_N(t_0+\Delta t)\} \;\rightarrow\; \{\mathbf{r}_N(t_0+2\Delta t)\} \;\rightarrow\; ... \;\rightarrow\; \{\mathbf{r}_N(t_0+N_T\Delta t)\}$$

where N_T is usually $\sim 10^3 - 10^6$.

Various algorithms: central difference (Verlet [7], leap-frog, velocity Verlet), Beeman's algorithm [8], predictor-corrector [6], symplectic integrators [10, 11, 12, 13].

Verlet algorithm

Perform Taylor expansion, assuming $\{\mathbf{r}_N(t)\}$ trajectory is smooth:

$$\mathbf{r}_i(t_0+\Delta t) + \mathbf{r}_i(t_0-\Delta t) \;=\; 2\mathbf{r}_i(t_0) + \ddot{\mathbf{r}}_i(t_0)(\Delta t)^2 + \mathcal{O}((\Delta t)^4). \tag{6}$$

Since $\ddot{\mathbf{r}}_i(t_0) = \mathbf{F}_i(t_0)/m_i$ can be evaluated given positions $\{\mathbf{r}_N(t_0)\}$ at $t = t_0$, $\{\mathbf{r}_N(t_0+\Delta t)\}$ in turn may be approximated by,

$$\mathbf{r}_i(t_0+\Delta t) \;=\; -\mathbf{r}_i(t_0-\Delta t) + 2\mathbf{r}_i(t_0) + \left(\frac{\mathbf{F}_i(t_0)}{m_i}\right)(\Delta t)^2 \;+\; \mathcal{O}((\Delta t)^4). \tag{7}$$

By throwing out the $\mathcal{O}((\Delta t)^4)$ term, we obtain a recursive formula to compute $\mathbf{r}_i(t_0+\Delta t)$, $\mathbf{r}_i(t_0+2\Delta t)$, ... successively, which is the Verlet algorithm.

The velocities do not participate in the recursive iteration but are needed for property calculations. They can be approximated by

$$\mathbf{v}_i(t_0) \equiv \dot{\mathbf{r}}_i(t_0) \;=\; \frac{1}{2\Delta t}\left[\mathbf{r}_i(t_0+\Delta t) - \mathbf{r}_i(t_0-\Delta t)\right] + \mathcal{O}((\Delta t)^2). \tag{8}$$

To what degree does the outcome of the above recursion mimic the real trajectory $\{\mathbf{r}_N(t)\}$? Notice that in (7), assuming $\mathbf{r}_i(t_0)$ and $\mathbf{r}_i(t_0 - \Delta t)$ are exact, and assuming we have a *perfect* computer with no error representing the relevant numbers or carrying out their operations, the computed $\mathbf{r}_i(t_0 + \Delta t)$ would still be off from the real $\mathbf{r}_i(t_0 + \Delta t)$ by $\mathcal{O}((\Delta t)^4)$, defined as the *local truncation error* (LTE), which is an intrinsic property 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 $\{\mathbf{r}_N(t_0 + T)\}$ for a given finite T, not $\mathbf{r}_i(t_0 + \Delta t)$. To obtain $\{\mathbf{r}_N(t_0 + T)\}$, we must integrate $N_T = T/\Delta t$ steps, and the difference between the computed $\{\mathbf{r}_N(t_0 + T)\}$ and the real $\{\mathbf{r}_N(t_0 + T)\}$ is called the *global error*. An algorithm can be useful only if when $\Delta t \to 0$, the global error $\to 0$. A careful analysis of the error propagation in (7) indicates that the global error is $\mathcal{O}((\Delta t)^2)$ as $\Delta t \to 0$, which differs from the naive expectation that if LTE $\sim (\Delta t)^{k+1}$, the global error should $\sim (\Delta t)^k$. Thus, the order of LTE can be very misleading as to deciding the quality of an algorithm. From now on when we refer to an algorithm's *order*, we mean its *global error*'s order in Δt. The Verlet algorithm is thus a *second-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`) at a certain Δt, 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 an numerical algorithm. In the context of linear ODE's, 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 arguments, where ω_k is the mode's 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 the previous observation, 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 dynamic system.

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 gains for all real $\omega\Delta t$'s, 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 [2], provides a practical lower bound to Δt, since by reducing Δt one can no longer reduce (and indeed would increase) the global error.

Leap-frog algorithm

Here we start out with $\{\mathbf{v}_N(t_0 - \Delta t/2)\}$ and $\{\mathbf{r}_N(t_0)\}$, then,

$$\mathbf{v}_i\left(t_0 + \frac{\Delta t}{2}\right) \;=\; \mathbf{v}_i\left(t_0 - \frac{\Delta t}{2}\right) + \left(\frac{\mathbf{F}_i(t_0)}{m_i}\right)\Delta t \;+\; \mathcal{O}((\Delta t)^3), \tag{9}$$

followed by,

$$\mathbf{r}_i(t_0 + \Delta t) \;=\; \mathbf{r}_i(t_0) + \mathbf{v}_i\left(t_0 + \frac{\Delta t}{2}\right)\Delta t \;+\; \mathcal{O}((\Delta t)^3), \tag{10}$$

and we have advanced by one step. This is a second-order method.

The velocity at time t_0 can be approximated by,

$$\mathbf{v}_i(t_0) \;=\; \frac{1}{2}\left[\mathbf{v}_i\left(t_0 - \frac{\Delta t}{2}\right) + \mathbf{v}_i\left(t_0 + \frac{\Delta t}{2}\right)\right] \;+\; \mathcal{O}((\Delta t)^2). \tag{11}$$

[2]For single-precision (SP) arithmetics (1 real = 4 bytes), $\epsilon \sim 10^8$; for double-precision (DP) arithmetics (1 real = 8 bytes), $\epsilon \sim 2 \times 10^{16}$; for quadruple-precision (QP) arithmetics (1 real = 16 bytes), $\epsilon \sim 10^{32}$.

Velocity Verlet algorithm

We start out with $\{\mathbf{r}_N(t_0)\}$, $\{\mathbf{F}_N(t_0)\}$ and $\{\mathbf{v}_N(t_0)\}$, then,

$$\mathbf{r}_i(t_0+\Delta t) \;=\; \mathbf{r}_i(t_0)+\mathbf{v}_i(t_0)\Delta t+\frac{1}{2}\left(\frac{\mathbf{F}_i(t_0)}{m_i}\right)(\Delta t)^2 \;+\; \mathcal{O}((\Delta t)^3), \tag{12}$$

evaluate $\{\mathbf{F}_N(t_0+\Delta t)\}$, and then,

$$\mathbf{v}_i\,(t_0+\Delta t) \;=\; \mathbf{v}_i(t_0)+\frac{1}{2}\left[\frac{\mathbf{F}_i(t_0)}{m_i}+\frac{\mathbf{F}_i(t_0+\Delta t)}{m_i}\right]\Delta t \;+\; \mathcal{O}((\Delta t)^3), \tag{13}$$

and we have advanced by one step. This is a second-order method. Since we have $\{\mathbf{r}_N(t_0)\}$ and $\{\mathbf{v}_N(t_0)\}$ simultaneously, it is very popular.

Beeman's algorithm

It is similar to the velocity Verlet algorithm. We start out with $\{\mathbf{r}_N(t_0)\}$, $\{\mathbf{F}_N(t_0-\Delta t)\}$, $\{\mathbf{F}_N(t_0)\}$ and $\{\mathbf{v}_N(t_0)\}$, then,

$$\mathbf{r}_i(t_0+\Delta t) \;=\; \mathbf{r}_i(t_0)+\mathbf{v}_i(t_0)\Delta t+\left[\frac{4\mathbf{F}_i(t_0)-\mathbf{F}_i(t_0-\Delta t)}{m_i}\right]\frac{(\Delta t)^2}{6} \;+\; \mathcal{O}((\Delta t)^4), \tag{14}$$

evaluate $\{\mathbf{F}_N(t_0+\Delta t)\}$, and then,

$$\mathbf{v}_i\,(t_0+\Delta t) \;=\; \mathbf{v}_i(t_0)+\left[\frac{5\mathbf{F}_i(t_0+\Delta t)+8\mathbf{F}_i(t_0)-\mathbf{F}_i(t_0-\Delta t)}{m_i}\right]\frac{\Delta t}{12} \;+\; \mathcal{O}((\Delta t)^4), \tag{15}$$

and we have advanced by one step. This is a third-order method.

Predictor-corrector algorithm

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

$$\mathbf{r}^k_N(t) \;\equiv\; \left(\frac{d^k\mathbf{r}_N(t)}{dt^k}\right)\left(\frac{(\Delta t)^k}{k!}\right). \tag{16}$$

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

Prediction step:

$$
\begin{aligned}
\mathbf{r}_N^0 &= \mathbf{r}_N^0 + \mathbf{r}_N^1 + \mathbf{r}_N^2 + \mathbf{r}_N^3 + \mathbf{r}_N^4 + \mathbf{r}_N^5, \\
\mathbf{r}_N^1 &= \mathbf{r}_N^1 + 2\mathbf{r}_N^2 + 3\mathbf{r}_N^3 + 4\mathbf{r}_N^4 + 5\mathbf{r}_N^5, \\
\mathbf{r}_N^2 &= \mathbf{r}_N^2 + 3\mathbf{r}_N^3 + 6\mathbf{r}_N^4 + 10\mathbf{r}_N^5, \\
\mathbf{r}_N^3 &= \mathbf{r}_N^3 + 4\mathbf{r}_N^4 + 10\mathbf{r}_N^5, \\
\mathbf{r}_N^4 &= \mathbf{r}_N^4 + 5\mathbf{r}_N^5.
\end{aligned} \tag{17}
$$

The general formula for the above is

$$
\mathbf{r}_N^k = \sum_{k'=k}^{M-1} \left[\frac{k'!}{(k'-k)!k!} \right] \mathbf{r}_N^{k'}, \qquad k = 0..M-2, \tag{18}
$$

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

Evaluation step: Evaluate forces $\{\mathbf{F}_N\}$ using the newly obtained $\{\mathbf{r}_N^0\}$.

Correction step: Define the error $\{\mathbf{e}_N\}$ as,

$$
\mathbf{e}_N \equiv \mathbf{r}_N^2 - \left(\frac{\mathbf{F}_N}{m_N} \right) \left(\frac{(\Delta t)^2}{2!} \right). \tag{19}
$$

Then apply corrections,

$$
\mathbf{r}_N^k = \mathbf{r}_N^k - C_{Mk}\mathbf{e}_N, \quad k = 0..M-1, \tag{20}
$$

where C_{Mk} are constants listed in Table 1.

It is clear that the LTE for $\{\mathbf{r}_N\}$ is $\mathcal{O}((\Delta t)^M)$ after the prediction step. But one can show that the LTE is enhanced to $\mathcal{O}((\Delta t)^{M+1})$ after the correction step if $\{\mathbf{F}_N\}$ depend on $\{\mathbf{r}_N\}$ only. And so the global error would be $\mathcal{O}((\Delta t)^M)$.

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

Table 1: Gear predictor-corrector coefficients.

Symplectic integrators

The symplectic integrators preserve the property of phase space volume conservation (Liouville's theorem) of Hamiltonian dynamics. They tend to have much better energy conservation in the long run. The velocity Verlet algorithm is in fact symplectic, followed by higher-order extensions by Yoshida [10] and Sanz-Serna [11]. As with the predictor-corrector algorithm, symplectic integrators tend to perform better at higher order, even on a per cost basis [12, 13]. The high-order predictor-corrector and high-order symplectic integrators are the real competitors for high-accuracy integrators.

We have benchmarked the two families of integrators by numerically solving the Kepler's problem (eccentricity 0.5) which is nonlinear and periodic, to compare with the exact analytical solution. The two families have different numerical error versus time characteristics: non-symplectic integrators all have linear energy error ($\Delta E \propto t$) and quadratic phase point error ($|\Delta\Gamma| \propto t^2$) with time, while symplectic integrators have constant (fluctuating) energy error ($\Delta E \propto t^0$) and linear phase point error ($|\Delta\Gamma| \propto t$) with time. Therefore the asymptotic long-term performance of a symplectic integrator is always 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 significantly better performance than the best of the published symplectic integrators at *large integration timesteps*, or small number of force evaluations per period. That is important, because it says that if one does not care about the correlation of a mode beyond 100 oscillation periods (say, the natural decay time of the mode is only 50 oscillation periods due to scattering, so there is no point ensuring numerical fidelity much beyond that), then the high-order predictor-corrector algorithm may preserve the physics at a significantly smaller computational cost.

Properties commonly calculated in MD, which are all expressible in $\{\mathbf{r}_N, \dot{\mathbf{r}}_N\}$:

Potential Energy

$$U = \langle \sum_{i<j} \phi(r_{ij}) \rangle. \tag{21}$$

Temperature

$$T = \frac{1}{3Nk_{\mathrm{B}}} \langle \sum_i m_i |\mathbf{v}_i|^2 \rangle. \tag{22}$$

Pressure

$$P = \frac{1}{3\Omega} \langle \sum_i m_i |\mathbf{v}_i|^2 + \sum_{i<j} \mathbf{r}_{ij} \cdot \mathbf{F}_{ij} \rangle. \tag{23}$$

Radial Distribution Function

$$g(r) = \frac{1}{\rho 4\pi r^2 dr} \langle \sum_{j\neq i}^{N} \theta(r \leq r_{ij} < r + dr) \rangle. \tag{24}$$

Mean Squared Displacements

$$\mathrm{MSD} = \langle |\Delta \mathbf{r}|^2 \rangle = \frac{1}{N} \sum_i^N \langle |\mathbf{r}_i(t) - \mathbf{r}_i(0)|^2 \rangle. \tag{25}$$

References

[1] J.M. Haile, *A Primer on the Computer Simulation of Atomic Fluids by Molecular Dynamics* (1980).

[2] J.M. Haile, *Molecular Dynamics Simulation: Elementary Methods* (Wiley, New York, 1997).

[3] M.P. Allen and D.J. Tildesley, *Computer Simulation of Liquids* (Clarendon, New York, 1987).

[4] D.C. Rapaport, *The Art of Molecular Dynamics Simulation* (Cambridge University Press, Cambridge, 1995).

[5] W.H. Press, S.A. Teukolsky, W.T. Vetterling, B.P. Flannery, *Numerical Recipes in C – the Art of Scientific Computing* (Cambridge University Press, Cambridge, 1992).

[6] C.W. Gear, *Numerical Initial Value Problems in Ordinary Differential Equation* (Prentice-Hall, Englewood Cliffs, N.J., 1971).

[7] L. Verlet, "Computer Experiments on Classical Fluids. I. Thermodynamical Properties of Lennard-Jones Molecules", *Phys. Rev.* **159**, 98 (1967).

[8] D. Beeman, "Some multistep methods for use in molecular dynamics calculations", *J. Comp. Phys.* **20**, 130-139 (1976).

[9] R.D. Ruth, "A Canonical Integration Technique", *IEEE Trans. Nucl. Sci.* **30**, 2669 (1983).

[10] H. Yoshida, "Construction of higher order symplectic integrators", *Phys. Lett.* A **150**, 262 (1990).

[11] J.M. Sanz-Serna, M.P. Calvo, *Numerical Hamiltonian Problems* (Chapman & Hall, London, 1994).

[12] Ch. Schlier and A. Seiter, "Symplectic Integration of Classical Trajectories: A Case Study", *J. Phys. Chem.* A **102**, 9399-9404 (1998).

[13] Ch. Schlier and A. Seiter, "High-Order Symplectic Integration: An Assessment", *Computer Physics Communications* **130**, 176-189 (2000).