Lecture Notes for a crash course in Molecular Dynamics Simulations

Jordi Faraudo

March 3, 2023

1 Numerical integration of the equations of motion

1.1 The basic approximations employed in Molecular Dynamics

Molecular Dynamics (MD) Simulation is a widely employed technique in the field of atomistic and molecular modelling, and it is employed to study both thermodynamic equilibrium and nonequilibrium properties.

In MD, what we do essentially is to solve the equations of motion of an atomic/molecular model of system or material using a computer.

The systems studied in MD can be very simple or really complex, in fact, we can find in the scientific literature MD simulations of almost anything of interest in physics, chemistry or materials sciences and nanosciences (liquids, solids, organic or inorganic materials or hybrid materials, molecular and supramolecular systems such as micelles, membranes, proteins or their assemblies,...). As an example, you can check the YouTube channel of my Research group.

Of course, the essential ingredient in MD simulations is the possibility of solving the equation of motion numerically. To this end, we assume the approximation that for the systems of interest in our studies the motion of all the atoms can be described by the Newton equations of motion. It means that for each atom i with mass m_i its motion is governed by:

$$\vec{F}_i(t) = m_i \cdot \vec{a}_i(t), \tag{1}$$

where $\vec{F}_i(t)$ is the force acting over atom i at time t due to forces exerted by all other atoms in the system as well as due to external forces. Once the instantaneous acceleration $\vec{a}_i$ is computed from the forces over the atom, the trajectory is determined by solving the differential equations:

$$\vec{a}_i(t) = \frac{d\vec{v}_i(t)}{dt}, \tag{2}$$

$$\vec{v}_i(t) = \frac{d\vec{r}_i(t)}{dt}. \tag{3}$$

This approximation for the motion of the atoms is valid in the usual conditions of interest in, for example, material sciences and nanosciences, biophysics, etc but it is of course not valid under extreme conditions (such as those in low-temperature physics, for example).

It is important to recall here that although we assume that the motion of the atom is assumed to be classical (i.e. described by Newton laws), in MD the force $\vec{F}_i$ between atoms is not assumed to be of classical origin. Interatomic and intermolecular forces in general have quantum mechanical origin. The question of the calculation of the forces will be discussed in another session. Here it is enough to recall that it is possible to either compute them from quantum mechanical calculations, for example using DFT for the description of the electronic structure or to rely on some approximate theory or semi-empirical description of the forces. In summary, the relevant point here is that the motion of the atoms is assumed to be classical and described by Eq.(1) and their interactions are assumed to be known from a suitable theory of inter-atomic forces.

Now the question is how to solve numerically Eqs.(1)-(3) for a large number of atoms (of the order of 10^4-10^6 in a typical MD simulation) with a computer.

In a computer we have only discrete magnitudes, not continuous ones, so it is essential to divide the time (a continuous variable) into discrete intervals with a finite size Δt, which is usually called the *time step* in MD. Therefore, we need formulate the equations of motion Eqs.(1)-(3) for the motion of the atoms during a time Δt, and iterate this motion during a certain number of steps N until we reach the desired final time $N\Delta t$. In principle, the numerical solution will be more precise by using smaller values of Δt but of course this increases N and therefore the computer time needed to finish the simulation, so a compromise is needed between using a Δt small enough to provide reliable trajectories but large enough to allow the calculation to finish in a reasonable time.

There are many different possible procedures to *discretize* Eqs.(1)-(3). In MD, we are interested in methods that are fast and efficient due to the large number of times that the motion of atoms has to be computed in a simulation (due both to the large number of atoms and the large number of time steps to be considered). We are not interested in very accurate but expensive methods, since we are not interested in determining the motion of individual atoms with high precision but the behaviour of the system as a whole. In MD it is also essential to preserve physical properties of the system even after large number of numerical iterations that may accumulate numerical errors (for example, conservation of energy must be ensured).

We now describe some of the most popular discretization algorithms for the equations of motion Eqs.(1)-(3). A further discussion can be found in Chapter 4 of the book by Smit and Frenkel [1].

1.2 Euler algorithm

The basic assumption in this algorithm is that during the (very short) time step Δt the acceleration does not change. Therefore, if we know the position $\vec{r}_i(t)$

and velocity $\vec{v}_i(t)$ of our atoms at instant t and the forces $\vec{F}_i(t)$ over each atom we can calculate the acceleration at this instant as:

$$\vec{a}_i(t) = \vec{F}_i(t)/m_i, \tag{4}$$

and because the acceleration is constant during Δt , we can calculate the new magnitudes at time $t + \Delta t$ as follows:

$$\vec{r}_i(t + \Delta t) = \vec{r}_i(t) + \vec{v}_i(t)\Delta t + \frac{1}{2}\vec{a}_i(t)\Delta t^2. \tag{5}$$

$$\vec{v}_i(t + \Delta t) = \vec{v}_i(t) + \vec{a}_i(t)\Delta t. \tag{6}$$

And after this motion of duration step Δt, the acceleration will change to a new value, according to the updated values of the forces:

$$\vec{a}_i(t + \Delta t) = \vec{F}_i(t + \Delta t)/m_i. \tag{7}$$

Therefore, starting form $t = 0$ we can propagate the motion of the atoms of the system until the desired final time t_f by repeating N times ($t_f = N\Delta t$) the evaluation of Eqs.(4)-(7): $t = 0$, $t = \Delta t$, $t = 2\Delta t$... $t = N\Delta t$.

The problem with this algorithm is that the original equations of motion Eqs. (1)-(3) are invariant under time reversal (i.e. they remain unaltered under the change $t \to -t$) but Eqs. (4)-(7) do not have this symmetry. This implies that the Euler algorithm does not conserve the energy of the system, irrespective of the value of the time step Δt employed (remember that according to Noether's theorem, energy conservation is due to the invariance of the equations of motion under time reversal). This implies that a MD simulation employing the Euler algorithm will have a substantial energy drift, so please **never** use this algorithm in MD.

So why are we considering it? Well, there are two reasons. First it is the simplest possible one, so it is worth considering it, and secondly, it can be corrected, as seen below.

1.3 Velocity Verlet Algorithm

The simplest way to correct the problem with Euler method (the absence of invariance under time reversal) is to symmetrize the equations of motion. Note that Eq.(5) is invariant under time reversal, and the problematic equation is Eq(6). Instead of assuming that the velocity change is only due to $\vec{a}_i(t)$, we can take into account that after this time step the acceleration changes to $\vec{a}_i(t + t\Delta t)$, so we can consider that the velocity change after Δt is due to an average acceleration taking into account both $\vec{a}_i(t)$ and $\vec{a}_i(t+t\Delta t)$. This new algorithm (called Velocity-Verlet), runs as follows. If we know the position $\vec{r}_i(t)$ and velocity $\vec{v}_i(t)$ of our atoms at instant t and the forces $\vec{F}_i(t)$ over each atom we again calculate the acceleration at this instant as:

$$\vec{a}_i(t) = \vec{F}_i(t)/m_i, \tag{8}$$

and we calculate the new position at time $t + \Delta t$ as in the Euler algorithm, assuming constant acceleration:

$$\vec{r}_i(t + \Delta t) = \vec{r}_i(t) + \vec{v}_i(t)\Delta t + \frac{1}{2}\vec{a}_i(t)\Delta t^2. \tag{9}$$

Once we know the new position, we can calculate the new atomic forces and the new acceleration at $t + \Delta t$:

$$\vec{a}_i(t + \Delta t) = \vec{F}_i(t + \Delta t)/m_i. \tag{10}$$

The essential aspect of the Velocity-Verlet algorithm is in the calculation of the velocity at $t + \Delta t$ which is given by:

$$\vec{v}_i(t + \Delta t) = \vec{v}_i(t) + \frac{\vec{a}_i(t) + \vec{a}_i(t + \Delta t)}{2}\Delta t \tag{11}$$

Therefore, starting form $t = 0$ we can propagate the motion of the atoms of the system until the desired final time t_f by repeating N times ($t_f = N\Delta t$) the evaluation of Eqs.(8)-(11): $t = 0$, $t = \Delta t$, $t = 2\Delta t$... $t = N\Delta t$.

This algorithm is implemented in many MD codes (for example NAMD) and it has a very good energy conservation (there is no energy drift even for large time steps) and it has a good speed and accuracy for MD simulations.

The derivation of these equations may sound not rigorous but in fact they are equivalent to the Verlet algorithm discussed below, which can be derived rigorously from a Taylor expansion in the equations of motion, as discussed below.

1.4 Verlet algorithm

Another possibility for deriving equations of motion that are symmetrical under time reversal is starting from a Taylor expansion for the equation of motion for atom i. Let us compare how the position of atom i changes after a short time Δt (going into the future):

$$\vec{r}_i(t + \Delta t) = \vec{r}_i(t) + \frac{d\vec{r}_i(t)}{dt}\Delta t + \frac{1}{2}\frac{d^2\vec{r}_i(t)}{dt^2}\Delta t^2 + \frac{1}{3!}\frac{d^3\vec{r}_i(t)}{dt^3}\Delta t^3 + O(\Delta t^4), \tag{12}$$

or how it changed after Δt (looking into the past):

$$\vec{r}_i(t - \Delta t) = \vec{r}_i(t) - \frac{d\vec{r}_i(t)}{dt}\Delta t + \frac{1}{2}\frac{d^2\vec{r}_i(t)}{dt^2}\Delta t^2 - \frac{1}{3!}\frac{d^3\vec{r}_i(t)}{dt^3}\Delta t^3 + O(\Delta t^4). \tag{13}$$

Summing up Eqs.(12) and (13) we have:

$$\vec{r}_i(t + \Delta t) + \vec{r}_i(t - \Delta t) = 2\vec{r}_i(t) + \frac{d^2\vec{r}_i(t)}{dt^2}\Delta t^2 + O(\Delta t^4). \tag{14}$$

Therefore, if we know the positions at times $t - \Delta t$ and t we can calculate the new position at time $t + \Delta t$ within an approximation of order $O(\Delta t^4)$:

$$\vec{r}_i(t + \Delta t) = 2\vec{r}_i(t) - \vec{r}_i(t - \Delta t) + \vec{a}_i(t)\Delta t^2. \tag{15}$$

We can also obtain the velocity by subtracting Eqs.(12) and (13):

$$\vec{v}_i(t) = \frac{2\Delta t}{\vec{r}_i(t + \Delta t) - \vec{r}_i(t - \Delta t)}. \tag{16}$$

It is easy to implement this equation as an algorithm in a MD code (for example SIESTA uses this algorithm). A drawback of this algorithm is that we have the velocities and positions evaluated at different times. It is important to remark that the Verlet and Velocity Verlet algorithms are equivalent and generate identical trajectories. In fact, it is not difficult to derive the equations for Velocity Verlet from Eq.(12)-(15).

References

[1] D. Frenkel. *Understanding molecular simulation : from algorithms to applications / Daan Frenkel, Berend Smit.* Computational Science: From Theory to Applications. Academic Press, San Diego, second edition edition, 2002.