Our aim is to numerically solve Newton's equations of motion

$$m_i \mathbf{a}_i = F_i(\mathbf{r}_i) = -\partial V(\mathbf{r})/\partial \mathbf{r}_i. \tag{1}$$

Given the impossibility for a computer to deal with the continuum, it is necessary to discretize the time axis; we will take equal intervals of width δ, which will therefore be the *time step* of our algorithm. The time values we consider are therefore those, assuming we start at $t = 0$, corresponding to $t_i = i\delta$, $i \in \{0, 1, \ldots, n\}$. The simulated time, therefore, it will be $\tau = n\delta$. It is therefore obvious that it is in our interest to choose a δ value as large as possible compatibly with an acceptable accuracy of the integration of the equation, in order to minimize the number of steps n with τ being equal. For simplicity we will consider here a system with only one degree of freedom, and we will call the current position x_0, the one at the next step x_+, etc. Likewise F_0 will be the force at the current time and so on for all quantities.

If we assume conservative forces, we have in principle a system with constant energy (*microcanonical ensemble*). The numerical solution obviously may not respect, given the inevitable approximations and truncations, this criterion. From the point of view of calculating equilibrium properties, however, a moderate violation (within a few percentage points) of this condition can be accepted; remember that in any case the simulated system is with every probability chaotic and we cannot hope to follow its evolution perfectly. The really important condition is the absence of a secular variation (drift) of the energy, given that this would lead to a simulation not belonging to a well-defined ensemble and making it impossible to locate with certainty the position of the simulated system in the thermodynamic phase space.

As is known from Theoretical Mechanics, the conservation of energy is strictly linked to the absence of a dependence of the Lagrangian/Hamiltonian of the system from time, therefore to the property of temporal reversibility. It is also linked (Liouville's theorem) to the conservation of the volume of representative points of the system in phase space. It can therefore be thought that the algorithms for solving the equations of motion will have to belong to a category (symplectic algorithms) that respects these important properties. We can therefore imagine that an algorithm apparently adequate like Euler's

$$\begin{aligned} x_+ &= x_0 + v_0\delta + F_0\delta^2/2m \\ v_+ &= v_0 + F_0\delta/m, \end{aligned}$$

obtained by developing the solutions of the equations of motion in series at the lowest possible order in δ (in this case in Hamiltonian form) doesn't work well. Certainly the solutions tend towards the correct ones for $\delta \to 0$, with error $O(\delta^3)$, but the temporal non-reversibility of the resulting algorithm causes an energy drift that is unacceptable unless δ is chosen to be extremely small. In fact the algorithm is "unbalanced" towards the future; that is, the derivatives are not centered at the current instant. This makes the algorithm such that reversing the sign of time ($\delta \to -\delta$) does not make the system to retrace backwards exactly the same points, and therefore violates the time reversal property.

However, we can easily create an algorithm that respects the time inversion condition exactly. Writing the second derivative of x with respect to t centered, Newton's equation becomes, once discretized:

$$F_0/m = \left[(x_+ - x_0)/\delta - (x_0 - x_-)/\delta\right]/\delta = (x_+ - 2x_0 + x_-)/\delta^2,$$

that is

$$x_+ = 2x_0 - x_- + F_0\delta^2/m, \tag{2}$$

which, as it was obtained, is invariant under time inversion. To convince yourself of this, you maychange the sign of δ and invert x_+ and x_-. This algorithm is known as *Verlet algorithm*. It can also be achieved by developing in series x_+ "forward" and x_- "backward":

$$\begin{aligned} x_+ &= x_0 + v_0\delta + a_0\delta^2/2 + \partial^3 x/\partial t^3|_0\delta^3/6 + O(\delta^4) \\ x_- &= x_0 - v_0\delta + a_0\delta^2/2 - \partial^3 x/\partial t^3|_0\delta^3/6 + O(\delta^4) \end{aligned}$$

and adding them. The result will evidently be invariant under time inversion. The odd terms are all canceled and we remain with the algorithm 2. We can also see how, in addition to the time reversal property, the order of the algorithm is one higher than Euler, since the neglected terms are $O(\delta^4)$ and not $O(\delta^3)$.

This algorithm has many good properties. It is time-reversible, simple to implement, and requires neither more memory nor more computation time than the strict necessary. It will have been observed, however, that along the way the velocity has disappeared and is now implicit and does not appear among the variables. This can be inconvenient, given that many things,

starting from kinetic energy, depend on velocities, and as we will see there the ability to manipulate velocities easily is crucial for various applications. We therefore look for an algorithm that explicitly includes v. Choosing to symmetrically discretize the equations in Hamiltonian form:

$$\begin{aligned} \dot{x} &= v \\ m\dot{v} &= F \end{aligned}$$

we can obtain two equations like the following:

$$\begin{aligned} (x_+ - x_0)\delta &= v_{1/2} \\ (v_{1/2} - v_{-1/2})\delta &= F_0/m \end{aligned}$$

where the first is centered around the point $t + \delta/2$, while $1/2$ and $-1/2$ indicate points halfway between t and $t + \delta$ and $t - \delta$ respectively, and the second is centered around t. That is, the velocities are calculated at time points staggered by $\delta/2$ with respect to to the coordinates. The resulting algorithm is

$$\begin{aligned} v_{1/2} &= v_{-1/2} + F_0/m\delta \\ x_+ &= x_0 + v_{1/2}\delta \end{aligned} \tag{3}$$

and is called *leapfrog*, a term which in English indicates the "game of leapfrog", because positions and speed in a certain sense they "jump over" each other by taking turns. Note that the memory usage is the same as Verlet's algorithm, since it is necessary to remember the velocities from one step to the other but it is no longer necessary to remember the previous positions.

The fact of having two distinct temporal discretizations leads to some inconveniences. For example, potential and kinetic energies are calculated at different instants and this leads to an apparent non-conservation of energy, which, however, is not serious, and if desired can be corrected by calculating at each instant $v_0 = (v_{1/2} + v_{-1/2})/2$. However, it is possible to develop an algorithm (*velocity Verlet*) that explicitly uses the velocities at the same moments as the positions:

$$\begin{aligned} x_+ &= x_0 + v_0\delta + F_0\delta^2/2m \\ v_+ &= v_0 + (F_+ + F_0)\delta/2m. \end{aligned} \tag{4}$$

The second equation is evidently time-symmetric with respect to $t + \delta/2$. It is not immediately obvious that the entire algorithm is, since the first of the equations 4 is identical to the unsatisfactory one of Euler. However, it is possible to show that the algorithm is equivalent to the original Verlet. In fact, let's write the 2 equation for the next step and insert the second of the 4:

$$\begin{aligned} x_{++} &= 2x_+ - x_0 + F_+\delta^2/m = 2x_+ - x_0 + F_+\delta^2/2m - F_0\delta^2/2m + (v_+ - v_0)\delta \\ x_{++} &\quad -x_+ - v_+\delta - F_+\delta^2/2m = x_+ - x_0 - v_0\delta - F_0\delta^2/2m. \end{aligned}$$

But if one also satisfies the first of the equation 4 the last equation written will be identically satisfied, so the algorithms are equivalent.

Note that the second of the 4 involves the use of forces both at the current instant and at the next instant. This doesn't mean that the calculation of the forces has doubled, which would be an unacceptable computational burden given that this calculation is the most burdensome part of the entire algorithm: actually the force F_0 is nothing other than the force F_+ of the previous step and it will therefore be sufficient to keep it in memory. But we have to update the velocities "piecewise": first $F_0\delta/2m$ is calculated by assigning it to v_+, then the positions are updated, finally F_+ is calculated and the term $F_+\delta/2m$ is added to v_+, obtaining the final values.

It has been said that the time step δ is the result of a compromise between the need to accurately integrate the 1 equation, which requires in principle $\delta \to 0$, and that of limiting the computational cost, which requires long δ. Symplectic algorithms are quite robust in this respect: a drift in energy usually appears when fluctuations around an equilibrium value, which in themselves are harmless as said at the beginning, reach substantial values, sometimes 10% or even more. The optimal value must be obtained through experimentation on each specific system, but we can generally consider that the time step will have to be much shorter than the typical oscillation times of the system. One can say that, if ω is the maximum frequency found in the system, then $\delta << 1/\omega$, typically a few percentage points of the oscillation time. Note that these frequencies change depending on the state of the system, as a very hot or very compressed system will have closer approaches between atoms than a colder or less dense one, and this will lead to forces that vary rapidly. We will therefore need to check the adequacy of δ not only to every change of system, but also to every significant change of thermodynamic coordinates.