Verlet algorithm

In the first lesson, we looked at the most basic method to integrate the particles forward in time: the *Euler* method. For a system where the motion of each particle is governed by Newton's second law

$$m\frac{d^2\mathbf{x}}{dt^2} = \mathbf{F}(\mathbf{x}) = -\nabla U(\mathbf{x}), \tag{1}$$

we could write

$$\mathbf{x}_{n+1} = \mathbf{x}_n + \mathbf{v}_n h \tag{2}$$

and

$$\mathbf{v}_{n+1} = \mathbf{v}_n + \frac{1}{m}\mathbf{F}(\mathbf{x}_n)\, h. \tag{3}$$

to get the positions and velocities at time $t_{n+1} = t_n + h$ from the state of the system at time t_n. We discussed in the previous lesson what arguments can be made to pick a sensible value of the time-step length h.

One of the goals given in the last milestones was to compute the total energy of the system and monitor its evolution as the simulation evolves. You may have noticed that the energy is not conserved. This is expected as there is a flaw in the method. Our goal is to simulate a system with conserved energy (the microcanonical ensemble), so we should look for a better algorithm.

The first of such algorithms with this astonishing property was already introduced in the very beginning of molecular dynamics (see the classic paper of Verlet[1]; the method was used even earlier, before the arrival of computers, e.g. in 1909 to calculate the orbit of Halley's comet and Delambre used it in the 18th century). The Verlet algorithm makes directly use of the fact that the equation of motion 1 is a second order differential equation. For this we can derive a finite time step algorithm by, once again, using a Taylor expansion but this time of $\mathbf{x}(t \pm h)$:

$$\mathbf{x}(t+h) = \mathbf{x}(t) + h\dot{\mathbf{x}}(t) + \frac{h^2}{2}\ddot{\mathbf{x}}(t) + \frac{h^3}{6}\frac{d^3}{dt^3}\mathbf{x}(t) + \mathcal{O}\left(h^4\right) \tag{4}$$

$$\mathbf{x}(t-h) = \mathbf{x}(t) - h\dot{\mathbf{x}}(t) + \frac{h^2}{2}\ddot{\mathbf{x}}(t) - \frac{h^3}{6}\frac{d^3}{dt^3}\mathbf{x}(t) + \mathcal{O}\left(h^4\right) \tag{5}$$

where $\mathcal{O}(h^4)$ means terms of order h^4. Adding those two equations together and using $\ddot{\mathbf{x}}(t) = \mathbf{F}(\mathbf{x}(t))/m$, from Eq. 1, we obtain

$$\mathbf{x}(t+h) = 2\mathbf{x}(t) - \mathbf{x}(t-h) + \frac{h^2}{m}\mathbf{F}(\mathbf{x}(t)) + \mathcal{O}\left(h^4\right). \tag{6}$$

[1] Actually a nice and accessible read: (Verlet 1961)

Using Eq. 6 (neglecting the $\mathcal{O}(h^4)$-term) one can propagate the system with high accuracy from time t to $t+h$ by just using the two previous position of each particle. A small technicality occurs at the first time step at, say, $t=0$. As we have no information about $\mathbf{x}(-h)$, we need to replace it with $\mathbf{x}(-h) = \mathbf{x}(0) - h\mathbf{v}(0) + \mathcal{O}(h^2)$. Finally, in this version of the Verlet algorithm, only the positions enter. If we are interested in velocities (for instance to compute the kinetic energy), we can estimate them from

$$\mathbf{v}(t) = \frac{\mathbf{x}(t+h) - \mathbf{x}(t-h)}{2h} + \mathcal{O}\left(h^2\right). \tag{7}$$

Velocity Verlet algorithm

The algorithm described above can be slightly altered to use also velocities directly, this is the so-called *velocity Verlet* algorithm. For that we take the Taylor expansion of $\mathbf{x}(t+h)$, Eq. 4, and replace $\dot{\mathbf{x}}(t)$ by $\mathbf{v}(t)$ and $\ddot{\mathbf{x}}(t)$ by $\mathbf{F}(\mathbf{x}(t))/m$ to arrive at

$$\mathbf{x}(t+h) = \mathbf{x}(t) + h\mathbf{v}(t) + \frac{h^2}{2m}\mathbf{F}(\mathbf{x}(t)) + \mathcal{O}\left(h^3\right) \tag{8}$$

Additionally, we need now an equation for $\mathbf{v}$. Again, we use a Taylor expansion:

$$\mathbf{v}(t+h) = \mathbf{v}(t) + h\dot{\mathbf{v}}(t) + \frac{h^2}{2}\ddot{\mathbf{v}}(t) + \mathcal{O}\left(h^3\right). \tag{9}$$

We know that $\dot{\mathbf{v}}(t) = \mathbf{F}(\mathbf{x}(t))/m$ but we do not know $\ddot{\mathbf{v}}(t)$. To determine the latter quantity we use again a Taylor expansion:

$$\dot{\mathbf{v}}(t+h) = \dot{\mathbf{v}}(t) + h\ddot{\mathbf{v}}(t) + \mathcal{O}\left(h^2\right), \tag{10}$$

from which we find $\ddot{\mathbf{v}}(t) = \frac{1}{h}\left(\mathbf{F}(\mathbf{x}(t+h)) - \mathbf{F}(\mathbf{x}(t))\right)/m$ (up to terms of order h). Inserting this back we obtain

$$\mathbf{v}(t+h) = \mathbf{v}(t) + \frac{h}{2m}\left(\mathbf{F}(\mathbf{x}(t+h)) + \mathbf{F}(\mathbf{x}(t))\right) + \mathcal{O}\left(h^3\right). \tag{11}$$

Equation 8 followed by Eq. 11 constitute the velocity Verlet algorithm. The *leapfrog* method is similar and has the same general properties.

Algorithmically, we would do the following:

1. Calculate $\mathbf{x}(t+h)$ from 8.
2. Calculate $\mathbf{F}(t+h)$ from the potentials using the positions at time $t+h$.

3. Calculate $\mathbf{v}(t+h)$ from 11 and the forces at the previous step $\mathbf{F}(t)$ (that we stored).

4. Go back to 1.

Compared to the Euler method, the Verlet and the velocity Verlet algorithm are not just better in the trivial sense by taking terms of higher order in h into account. They happen to be *much better* than that, as they are so-called *symplectic* integrators. For systems where the forces depend only on the positions, these integrators preserve a certain quantity that is a discretized version of the energy. This conserved quantity acts like an anchor to which the total energy is bound so that it cannot drift away, something that is not the case with the Euler method. The total energy (and other conserved quantities) will oscillate around the correct value, not completely depart from the correct answer.

Revisiting the minimal image convention

We discussed earlier that our periodic boundary conditions also influence how the forces are calculated. In particular, we chose the convention that we only take into account the forces due to the nearest images of other particles (the so-called minimum image convention). Naively one might expect that in order to compute the force exerted by particle j at $\mathbf{x}_j$ on particle i at $\mathbf{x}_i$, we have to consider *all* the images of particle j in the 26 boxes (in 3D) surrounding the original box and then to chose among the particle and its 26 images the one that is closest. However, since we are working with an orthogonal coordinate system, there is a significant simplification. We need to minimize the distance $r_{ij}^2 = \left(x_i - x_j'\right)^2 + \left(y_i - y_j'\right)^2 + \left(z_i - z_j'\right)^2$ where x_j', y_j', z_j' are the coordinates of the original particle j, $x_j' = x_j$, or of an image, for example, $x_j' = x_j + L$ or $x_j' = x_j - L$. Or, mathematically:

$$r_{ij} = \min_{k=-1,0,1} \min_{l=-1,0,1} \min_{m=-1,0,1} \sqrt{(x_i - x_j + kL)^2 + (y_i - y_j + lL)^2 + (z_i - z_j + mL)^2} \tag{12}$$

Since we can shift each of the coordinates individually, we can minimize r_{ij} by minimizing each coordinate direction separately. Instead of 27 possibilities we thus just need to check nine.

The problem simplifies even more, since now we are effectively dealing with 1D problems. In particular we know that for a box of length L:

$$-L/2 < x_i - x_j < L/2 \quad \rightarrow \quad x_j' = x_j$$

$$\begin{aligned} x_i - x_j > L/2 &\rightarrow x'_j = x_j + L \\ x_i - x_j < -L/2 &\rightarrow x'_j = x_j - L \end{aligned} \tag{13}$$

where x'_j denotes now the closest image.

A very compact formulation of this operation (in Python) makes use of the Modulo operator:

```
(xi - xj + L/2) % L - L/2
```
(14)

Here `xi` and `xj` are `numpy` arrays that contain the positions of all particles (in an $3N$-dimensional array).

Question for you: Contemplate on why it works by looking at the different possible cases.

Third milestone:

Implement the more efficient minimal image convention "trick".

Modify your time integration algorithm to use the Velocity-Verlet algorithm, Eqs. 8 and 11.

Redo the simulation of the previous week. Verify that variations in the kinetic energy with time are exactly cancelled by opposite variations in the potential energy, leading to a constant total energy.