

Computational Physics

Lecture 3

Organisation of the course

TA hour:

Thursday - 10am - Room 258 & 223

Organisation of the course - Program

Feb 09 – Introduction to MD, interactions, EoMs, boundary conditions

Feb 16 – Choice of units, energy of the system

Feb 23 – Verlet integration algorithm, minimum image convention

Mar 01 – Setting up initial conditions

Mar 08 – Observing the simulation: correlations & pressure

Mar 15 – Individual discussions

First project deadline: Thursday March 21 (midnight)

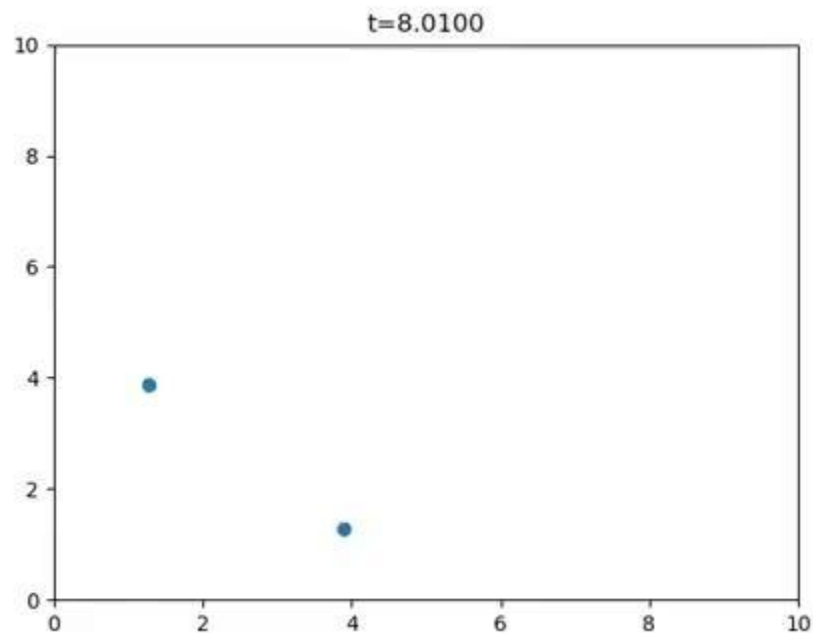
Questions?

Project 1: Time Integration

Last weeks' summary

- Considered the equations of motion for **a system of particles interacting via the Lennard-Jones potential**; and specifically the case of some Argon atoms.
- Use a simple **1st order Taylor expansion of the EoMs** to construct a way of moving the particles in small steps by way of a pair of *discretized* equations.
- Looked at a basic version of the **periodic boundary conditions**.
- Introduced **dimensionless units** to capture the **essence of the physics** at play. Used this to argue **good values for the time-step length**.

A test simulation



Adapted from Eloic Vallee's setup

General physics considerations

Ignoring the numerics, let's consider **the physics of the system**.

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

$$\mathbf{F}(\mathbf{x}_i) = - \sum_j \nabla U(\mathbf{x}_i - \mathbf{x}_j)$$

What can we learn from these?

Momentum

$$\mathbf{F}(\mathbf{x}_i) = - \sum_j \nabla U(\mathbf{x}_i - \mathbf{x}_j)$$

The equation has a natural symmetry.

Consider a pair of particles/atoms. At any time, they will each get the exact and opposite force.

→ **Momentum is conserved**

That should also be the case in our simulations.

Angular Momentum

Another way to think about momentum conservation is that the system is invariant under translation so momentum is conserved (think Noether theorem).

The system is also invariant under rotation

→ **Angular momentum is conserved**

That should also be the case in our simulations.

Total energy

We are looking at a Hamiltonian system.

→ **Total energy is conserved**

That should also be the case in our simulations.

Conservation rules

Question for you: Convince yourself by looking at the EoMs that indeed momentum, angular momentum, and energy should be conserved.

A few lessons

What are we learning from this?

- **Momentum conservation** is the “easiest” to achieve as it **only requires to get the forces (and their signs!) to be symmetric**. This does not depend on the time-step length or anything else.

A few lessons

What are we learning from this?

- **Momentum conservation** is the “easiest” to achieve as it **only requires to get the forces (and their signs!) to be symmetric**. This does not depend on the time-step length or anything else.
- **Energy conservation** is more tricky. To get it right, we must make sure **we transfer energy from the kinetic budget to the potential budget and vice-versa**. This relates the positions (for U) and velocities (for K). These are linked via our (now discretized) EoMs.

A few lessons

What are we learning from this?

- **Momentum conservation** is the “easiest” to achieve as it only requires to get the force and the time-step correct.
- **Energy conservation** is more difficult to achieve as we need to get the force and the time-step correct. This means we need to get the force and the time-step correct via our (n)umerical solution.

These are necessary* conditions to demonstrate that the simulation code is correct!

*but not sufficient!

Euler integration

We used these equations in the first lecture:

$$\mathbf{x}_{n+1} = \mathbf{x}_n + \mathbf{v}_n h + \mathcal{O}(h^2)$$

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

We truncated the series after one term \rightarrow this is 1st order accurate in time.

Unfortunately, this method does **not** guarantee that the total energy of the system is conserved.

The time symmetry of the EoM is NOT preserved!

Verlet algorithm (paper 1961 - already used in 18th cent.)

Instead of one Taylor expansion, we do two. One forward, one backward:

$$\begin{aligned}\mathbf{x}(t+h) &= \mathbf{x}(t) + h\dot{\mathbf{x}}(t) + \frac{h^2}{2}\ddot{\mathbf{x}}(t) + \frac{h^3}{6}\frac{d^3}{dt^3}\mathbf{x}(t) + \mathcal{O}(h^4) \\ \mathbf{x}(t-h) &= \mathbf{x}(t) - h\dot{\mathbf{x}}(t) + \frac{h^2}{2}\ddot{\mathbf{x}}(t) - \frac{h^3}{6}\frac{d^3}{dt^3}\mathbf{x}(t) + \mathcal{O}(h^4)\end{aligned}$$

“Fun” break: Higher-order derivatives of positions

$\frac{d^3}{dt^3}\mathbf{x}$ (aka. the first time derivative of the acceleration) is called the “jerk”.

What are the higher order ones (4th, 5th and 6th) called?

“Fun” break: Higher-order derivatives of positions

$$\frac{d^3}{dt^3}\mathbf{x} \text{ (aka. the first time deriv)}$$

What are the higher order ones?

- 4th: “snap”
- 5th: “crackle”
- 6th: “pop”



Verlet algorithm

Instead of one Taylor expansion, we do two. One forward, one backward:

$$\begin{aligned}\mathbf{x}(t+h) &= \mathbf{x}(t) + h\dot{\mathbf{x}}(t) + \frac{h^2}{2}\ddot{\mathbf{x}}(t) + \frac{h^3}{6}\frac{d^3}{dt^3}\mathbf{x}(t) + \mathcal{O}(h^4) \\ \mathbf{x}(t-h) &= \mathbf{x}(t) - h\dot{\mathbf{x}}(t) + \frac{h^2}{2}\ddot{\mathbf{x}}(t) - \frac{h^3}{6}\frac{d^3}{dt^3}\mathbf{x}(t) + \mathcal{O}(h^4)\end{aligned}$$

Can be
related to
the forces

Verlet algorithm

Adding both equations and replacing the acceleration with forces:

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

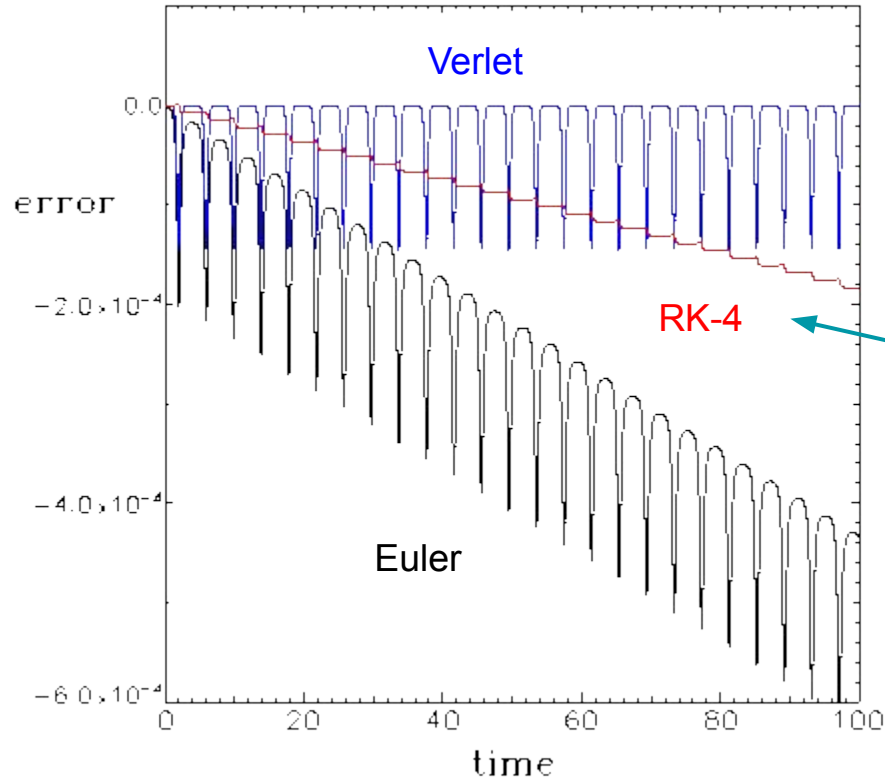
There is now an obvious symmetry in time!

And it's order 3 accurate.

There is a drawback: We need to store an extra set of old positions.

Visually...

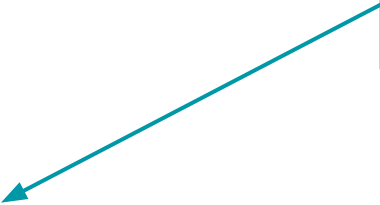
Sun-Earth system
using gravitational
law on a more
elliptical orbit than
reality.



Runge-Kutta of order 4:
High-order
non-symmetric method

Verlet algorithm - “tricks”

Notice the time symmetry !



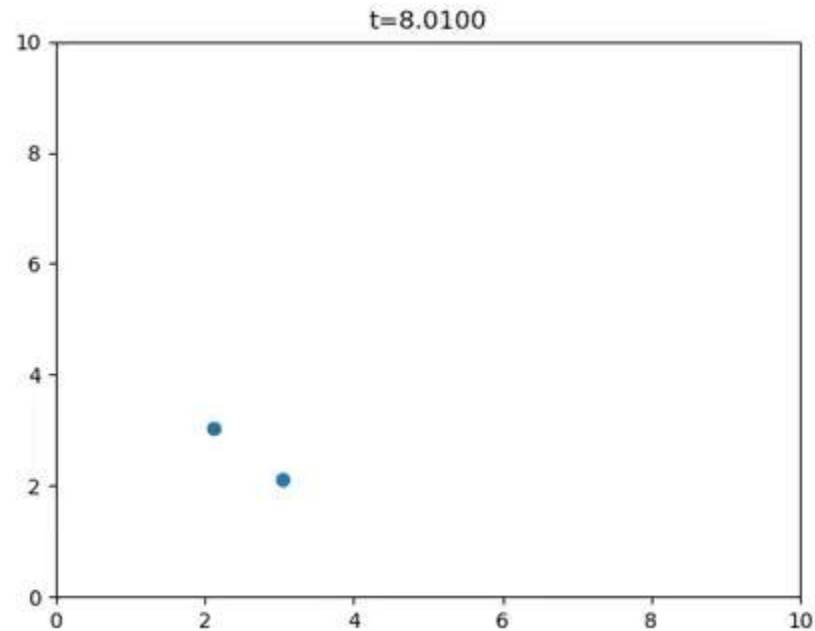
We may also want to have velocities:

$$\mathbf{v}(t) = \frac{\mathbf{x}(t+h) - \mathbf{x}(t-h)}{2h} + \mathcal{O}(h^2)$$

There is an additional complexity. At the start, we don't have a set of older positions to use. So we need to use one set of the *Euler* method backwards:

$$\mathbf{x}(-h) = \mathbf{x}(0) - h\mathbf{v}(0) + \mathcal{O}(h^2)$$

Back to the test simulation



Velocity-Verlet algorithm

The Verlet algorithm is often altered a bit to make the velocity appear explicitly.

Going back to the forward-only expressions but now for x and v :

$$\begin{aligned}\mathbf{x}(t+h) &= \mathbf{x}(t) + h\mathbf{v}(t) + \frac{h^2}{2m}\mathbf{F}(\mathbf{x}(t)) + \mathcal{O}(h^3) \\ \mathbf{v}(t+h) &= \mathbf{v}(t) + h\dot{\mathbf{v}}(t) + \frac{h^2}{2}\ddot{\mathbf{v}}(t) + \mathcal{O}(h^3)\end{aligned}$$

Now, there are some derivatives of v that we need to express.

Velocity-Verlet algorithm

We know the time derivative of \mathbf{v} is the acceleration, which is the force.

Using a Taylor expansion once more, we have:

$$\dot{\mathbf{v}}(t+h) = \dot{\mathbf{v}}(t) + h\ddot{\mathbf{v}}(t) + \mathcal{O}(h^2)$$

And hence:

$$\ddot{\mathbf{v}}(t) = \frac{1}{h} (\mathbf{F}(\mathbf{x}(t+h)) - \mathbf{F}(\mathbf{x}(t))) / m$$

Velocity-Verlet algorithm

Plugging that expression into the previous one, we get:

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

Note again the **symmetry in time** of this expression.

The full (velocity Verlet) algorithm

1. Calculate $\mathbf{x}(t + h)$

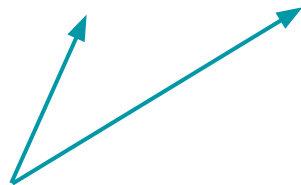
$$\mathbf{x}(t + h) = \mathbf{x}(t) + h\mathbf{v}(t) + \frac{h^2}{2m}\mathbf{F}(\mathbf{x}(t)) + \mathcal{O}(h^3)$$

2. Calculate the forces based on the positions at $t + h$.

3. Compute $\mathbf{v}(t + h)$

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

4. Back to 1.

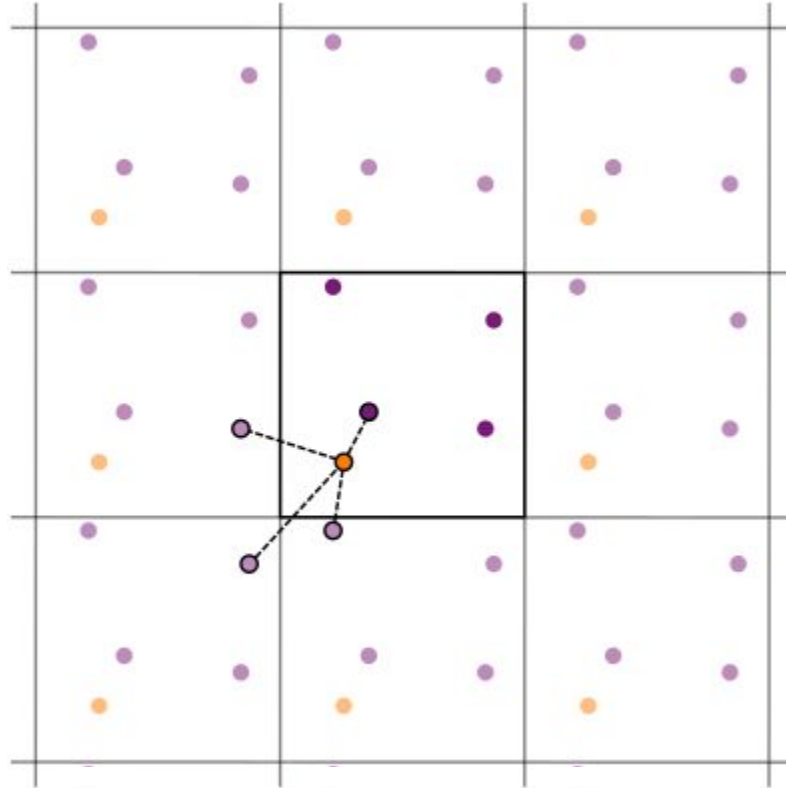


Instead of storing two sets of positions (in basic Verlet), we store two sets of forces.

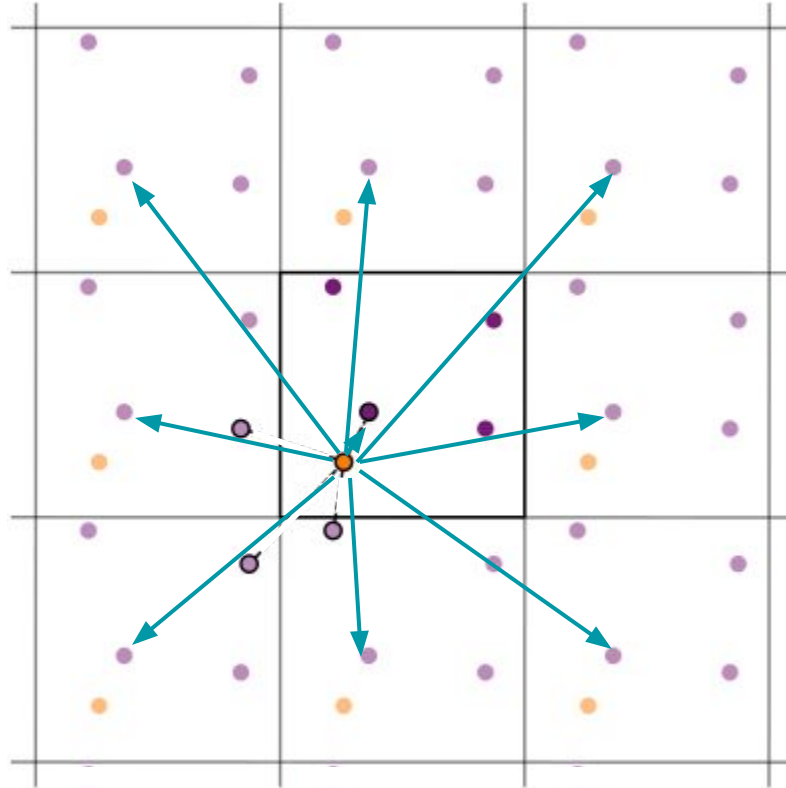
Questions?

Project 1: Boundary Conditions Trick

Back to the minimum image convention



Back to the minimum image convention



The shortest arrow is the one we are going to use for this specific interaction.

A trick to speed things up

We want to compute the distance between two particles i and j

$$r_{ij}^2 = (x_i - x'_j)^2 + (y_i - y'_j)^2 + (z_i - z'_j)^2$$

Box size


taking into account **all** possible copy j' of j in all the infinitely many virtual boxes.

The minimal image convention means we use

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

Need to test 27 r_{ij} for every pair of particles (in 3D).

A trick to speed things up

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


We can get a minimum simply by **taking the minimum** along each of x, y, and z **independently**.

→ 3 tests along the x-axis, 3 tests along the y-axis, 3 tests along the z-axis.

That is a total of **9 tests**, or **3x fewer** than the 27 we had before!

A trick to speed things up

Now, we are effectively solving **three independent 1D problems**.

We can make the following decisions:

$$-L/2 < x_i - x_j < L/2 \rightarrow x'_j = x_j \quad \text{“Same box”}$$

$$x_i - x_j > L/2 \rightarrow x'_j = x_j + L \quad \text{“Right box”}$$

$$x_i - x_j < -L/2 \rightarrow x'_j = x_j - L \quad \text{“Left box”}$$

(and then same for y and z)

The magic of python's modulo operator

In python, instead of a set of if-else-tests, we can condense the operations of the previous page in a single “math” line:

```
rij = (xi - xj + L/2) % L - L/2
```

numpy N x 3 array of positions



Remainder of division.



*Question for you: **Contemplate** this and convince yourself on why it works by looking at the different possible cases.*

In most languages the modulo operator only works on integers!

Third Milestone

- Implement the more efficient minimal image convention “trick”.
- Modify your MD code to make use of the *velocity-Verlet* method.
- Redo the simulations you ran in the previous weeks. Verify that the changes in kinetic energy are exactly compensated by changes in potential energy and that the **total energy is hence conserved**.
- Try to run **more atoms (particles)**. See how many you can handle in a reasonable amount of simulation time.