Computational Physics

Lecture 1

Organisation of the course - 2 versions

3 ECTS course

6 weeks - 1 project:

- Molecular Dynamics simulation of different phases of matter

6 ECTS course

12 weeks - 3 projects:

- Molecular Dynamics simulation of different phases of matter (6 weeks)
- Monte-Carlo simulation of the the Ising model (3 weeks)
- Individual projects (3 weeks)

Organisation of the course - Lectures

- Fridays 11:00 in HL111.
- Detailed schedule of topics on *Brightspace*.
- Lecture ~20-40 min then time for coding the weekly milestones, running simulations and analysing them.
- Teaching team will be there too to answer questions and help with the coding aspects.

Organisation of the course - Marking

- Project-based assessment.
- Projects done in **teams of 2 people**. 1 report per team.
  → Please register your teams on *Brightspace*.
- Grading based on the simulation code (30%) and report (70%).
- The final 10% of a project are obtained by completing unguided extensions.
- Specific guidelines for each project will be put on *Brightspace* in due course.

Organisation of the course - Team

Two teaching assistants here to help:

- Eloic Vallee (vallee@lorentz.leidenuniv.nl)
- Orestis Karapiperis (karapiperis@lorentz.leidenuniv.nl)

There will be a TA hour organised. Check your *Brightspace* announcements.

Organisation of the course - Books

If you want to go beyond the course, this textbook is recommended:

J.M. Thijssen, *Computational physics*, Cambridge University Press, 2007

Or the classic:

W. Press, *Numerical Recipes in C++, The Art of Scientific Computing*, 2nd Edition, Cambridge University Press, 2002

Organisation of the course - Programming

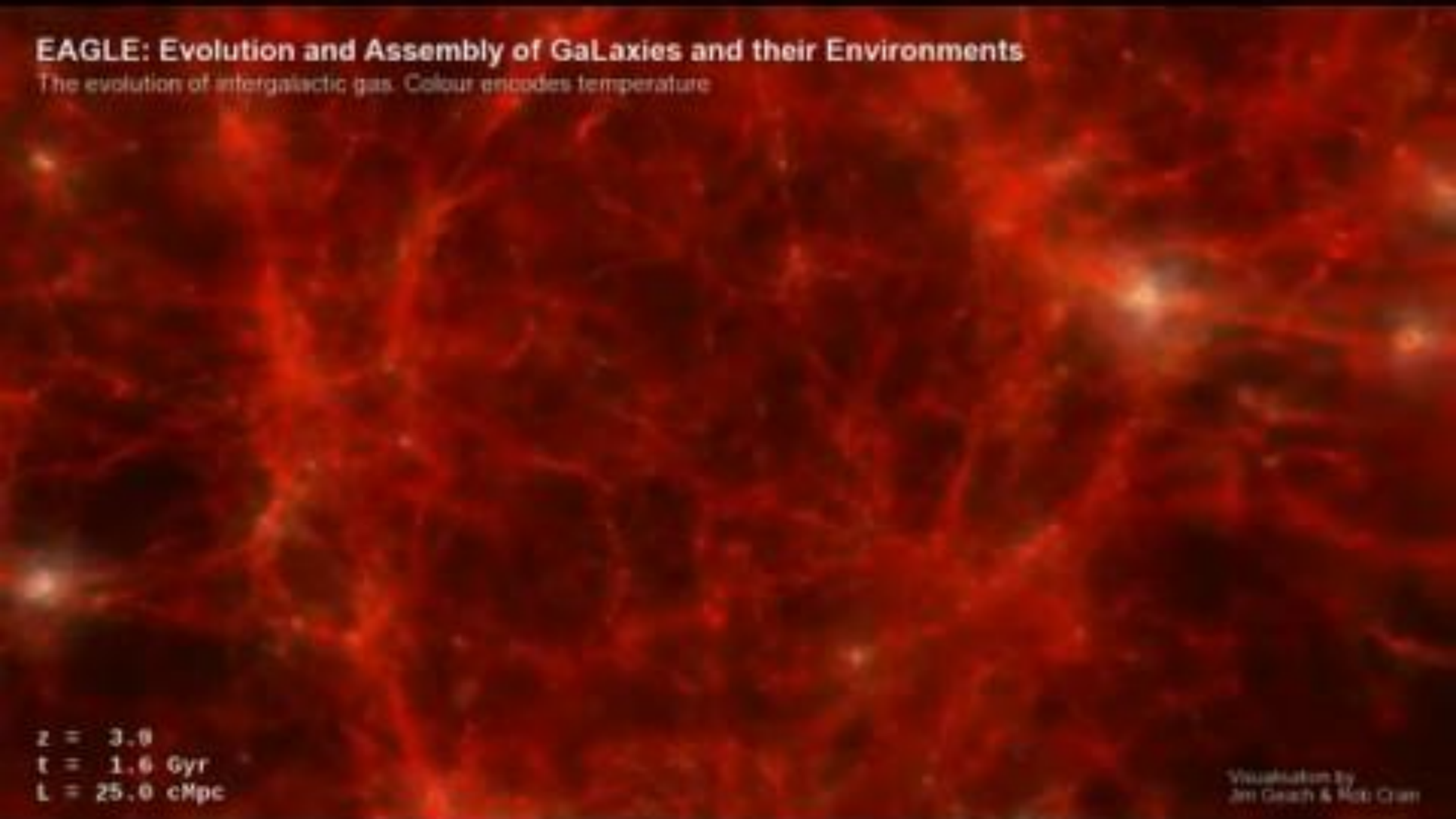
- Language of choice: `python3`

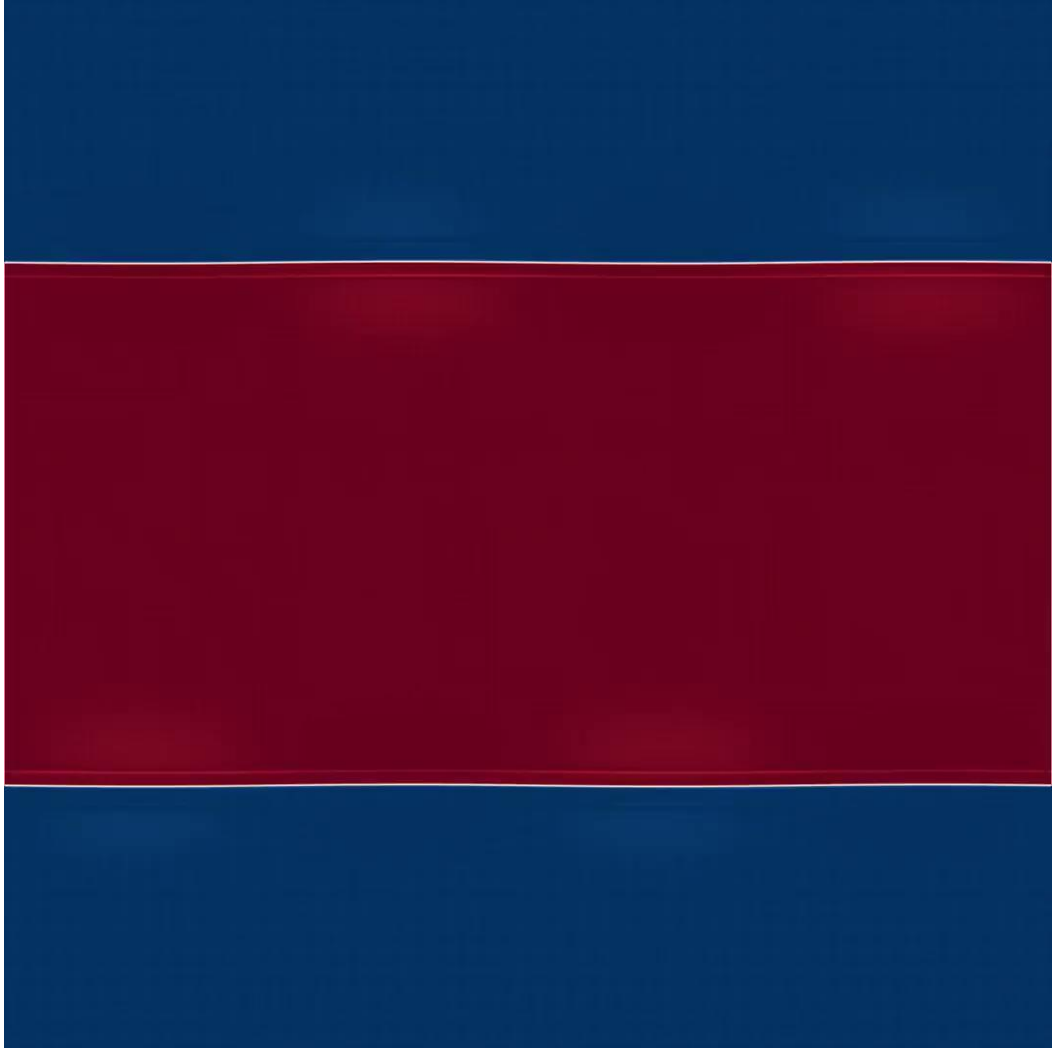
- Only basic knowledge required.
  Nothing beyond what was covered in your 1st year course.
  Links to `python` material on *Brightspace* if need be.

- Coding guidelines and best-practice “rules” are on *Brightspace*.

Feel free to ask for coding help.
This is a physics class, not a programming class.

Questions?


EAGLE: Evolution and Assembly of GaLaxies and their Environments
The evolution of intergalactic gas. Colour encodes temperature
z = 3.0
t = 1.6 Gyr
L = 25.0 cMpc
Visualisation by
Jim Geach & Rob Crain



Project 1: Molecular Dynamics

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)

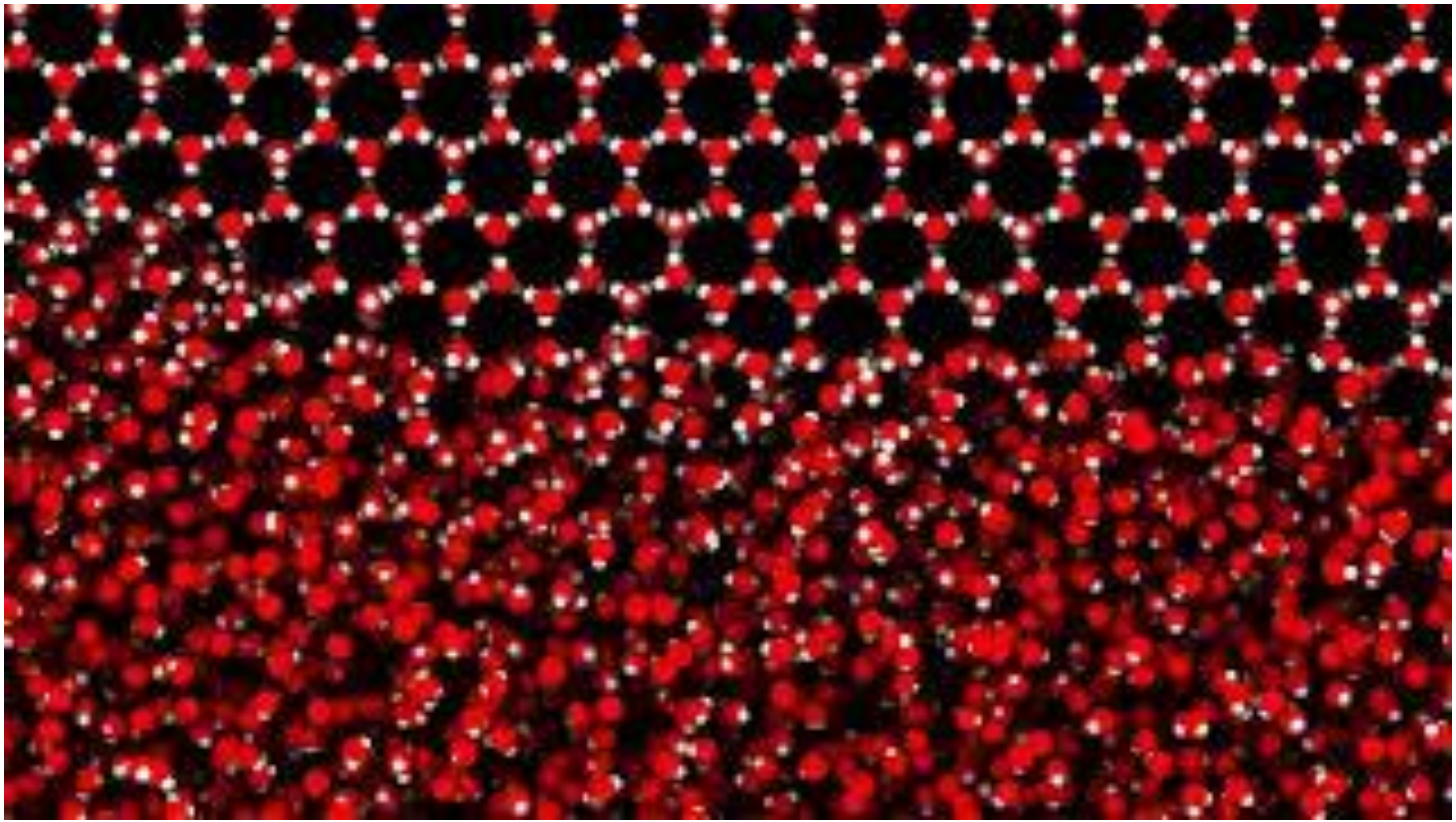
Molecular Dynamics (MD)

Main technique used to analyse the movement of **individual molecules (or atoms)** in systems too complex to be solved analytically.

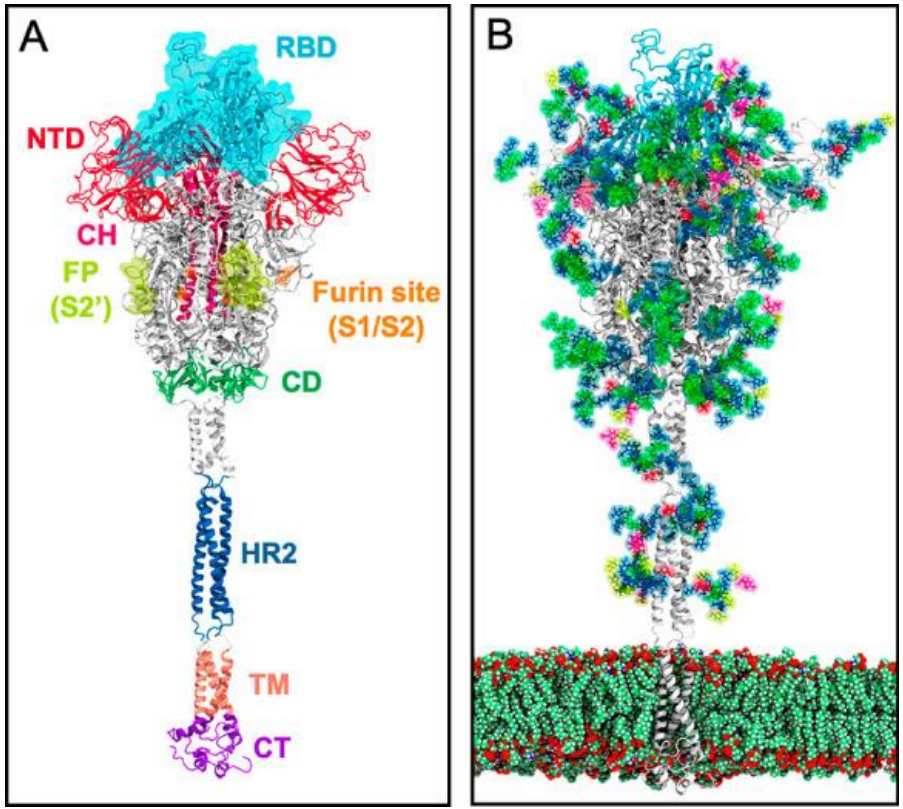
It is very popular in the fields of biophysics, material physics, soft matter, protein studies, etc.

Newtonian physics is assumed to study the trajectory of particles (molecules, atoms) that interact using potentials.

Molecule-by-molecule approach



Recent example

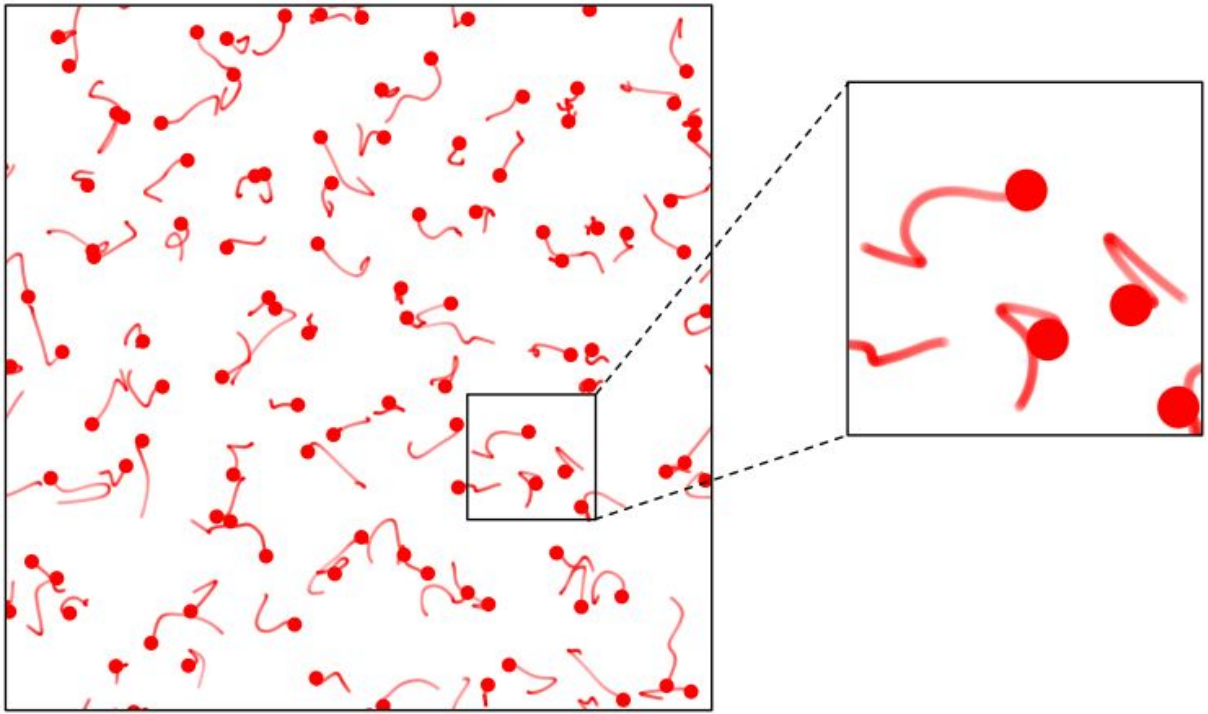


Arantes et al. 2020

Our target: Different phases of Argon

- Study the macroscopic properties (e.g. pressure) of Argon in its different phases (solid, liquid, gas) by varying the density and pressure of the medium.
- This will be done using an **atom-by-atom molecular dynamics simulation** of a ***small*** number of atoms.
- Atoms will interact via a simple **potential-term**.
- In the language of the simulation, each atom will be a **particle**.

Our target: Different phases of Argon



Particle motion

We assume **Newtonian dynamics with a potential** U:

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

Particle motion

We assume **Newtonian dynamics with a potential** U:

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

For a system with many particles, the RHS becomes (for particle i):

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

Numerical integration - Taylor expansion in time

We can't solve these equations analytically.
We will **discretize time** with a step in time of length h.

To move the particles from step n to step $n+1$, we will update the positions x and velocities v as follow:

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

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

If h is ***small enough***, then this will approximate the correct answer.

The potential term

We choose to use the *Lennard-Jones* potential:

$$U(r) = 4\epsilon \left(\left(\frac{\sigma}{r} \right)^{12} - \left(\frac{\sigma}{r} \right)^{6} \right)$$

For Argon:

$$\epsilon / k_B = 119.8\,\mathrm{K}$$
$$\sigma = 3.405\,\text{Å}$$

It depends only on the distance between particles. It contains an attractive term scaling as $1/r^6$ (van der Waals interaction) and a repulsive term scaling as $1/r^{12}$ (Pauli repulsion).

Simulation steps

1. Have a set of particles (atoms) with masses m, positions x, and velocities v.

2. Compute **the force acting on each particle due to every other particle** in the system.

3. **Update the positions and velocities** using our time-step length h.

4. Go back to (1) and **repeat** until the end time is reached.

Time-step length

How do we decide what the time-step length h should be?

We got the expression for the integration of the positions and velocities from a Taylor expansion of the differential equations. We kept only the first term in that expansion. So we need to choose h small enough that the other higher-order terms do not matter.

→ For now, we can just make it small and see when it goes wrong.

System boundaries

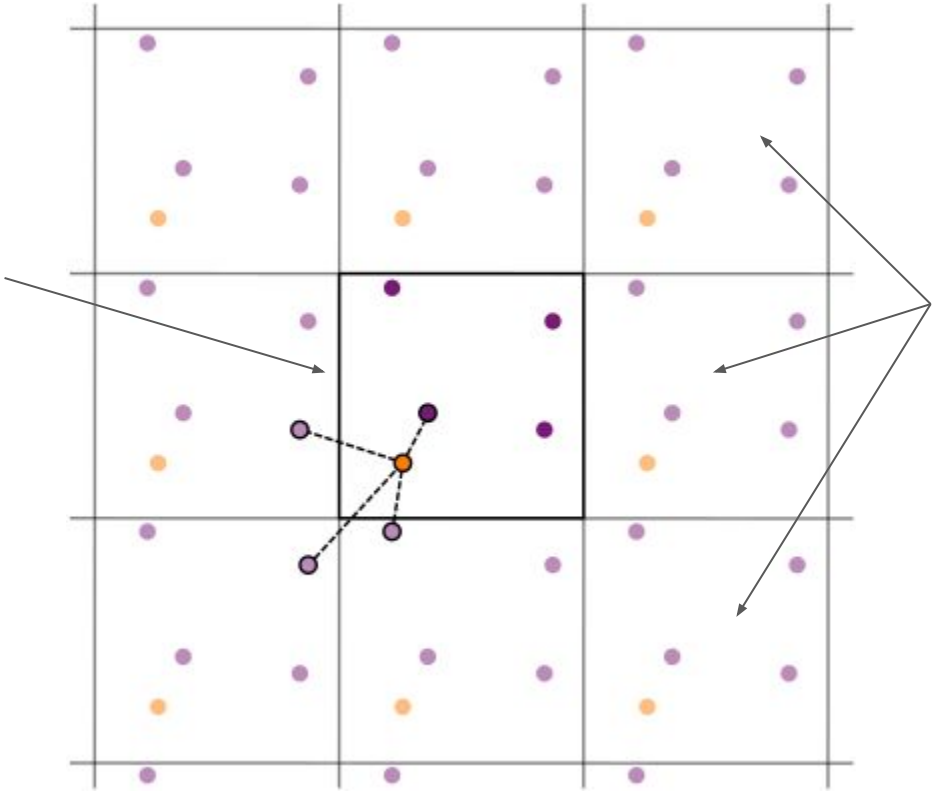
For obvious practical reasons, we can’t simulate an infinite system.

We will hence use a box of size L and introduce so-called *periodic boundary conditions* such that the system has no effective edge.

→ When a particle leaves the box on one side, it re-enters on the other side.

Minimum image convention

Simulation volume



Periodic copies
(for illustration only, not part of the simulation)

Questions?

Coding aspects

A few things to think about before typing:

What is the best way to represent particles in the code?
- `numpy` arrays are probably quite convenient for positions and velocities.

How can you decompose the problem into functions?

What are the constants of the problem and how to store them?

First Milestone

- Code up and play around with a simple system of ***a few*** atoms obeying the equations above and using a Lennard-Jones interaction potential.
- Use random positions and velocities in a periodic box using the minimum image convention.
- Store the particle trajectories in a file and plot them.

Note that 2D might be easier first!