Computational Physics

Lecture 4

Organisation of the course

TA hour:

Wednesday - 10am - Room 223

Wednesday - 3pm - Room 256

Organisation of the course - Program

Feb 10 – Introduction to MD, interactions, EoMs, boundary conditions

Feb 17 – Choice of units, energy of the system

Feb 24 – 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)

Organisation of the course

Any questions?

Project 1: Initial Conditions

Last weeks' summary

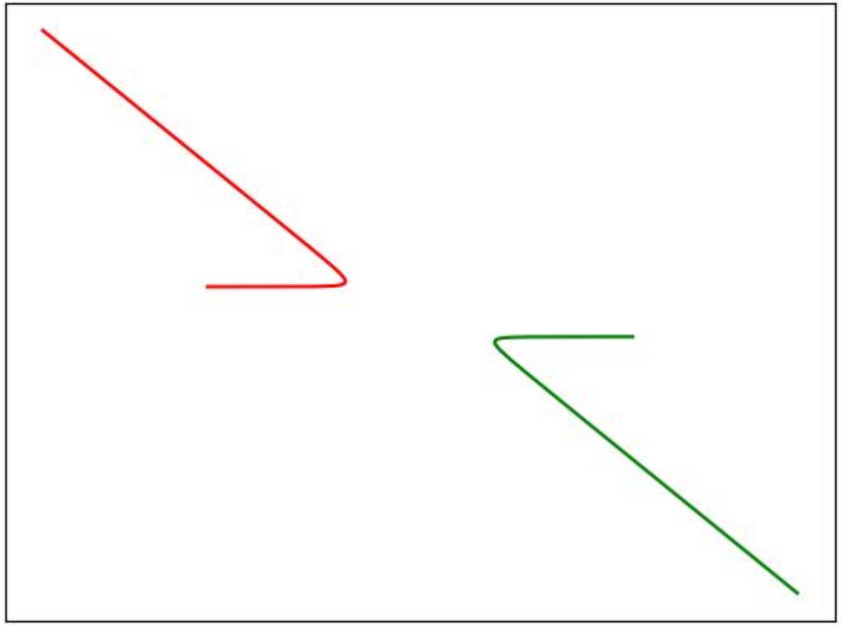
- Considered the equations of motion for a system of particles interacting via the Lennard-Jones potential; and specifically the case of some Argon atoms.

- Looked at a way to efficiently deal with periodic boundary conditions.

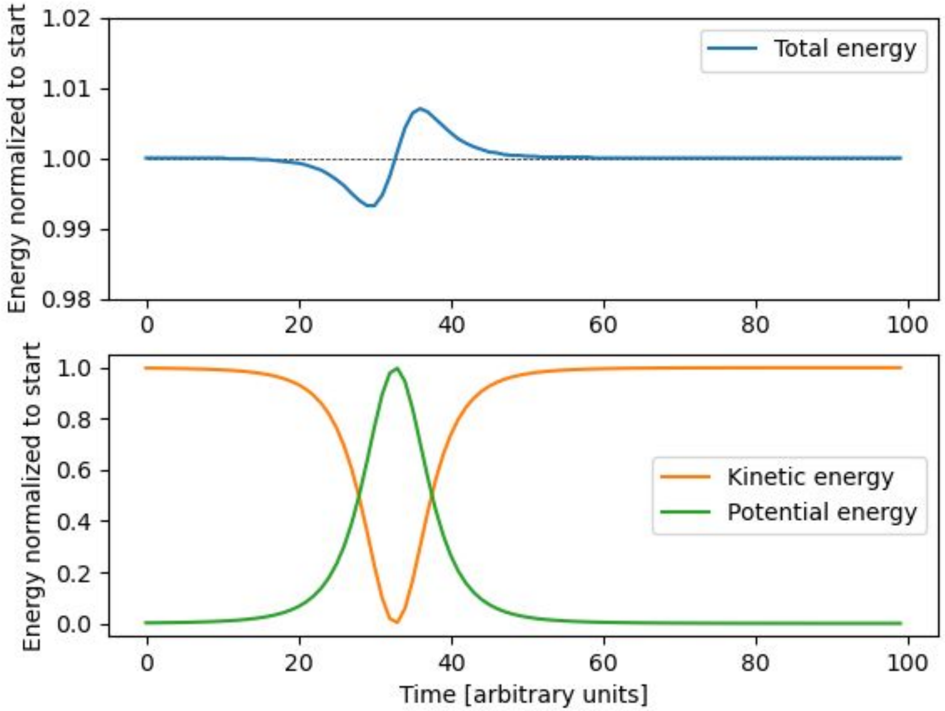
- Use a set of discretized equations to move particles forward in time and develop a method to conserve global quantities.

- Introduced dimensionless units to capture the essence of the physics at play. Used this to argue good values for the time-step length.

Energy conservation



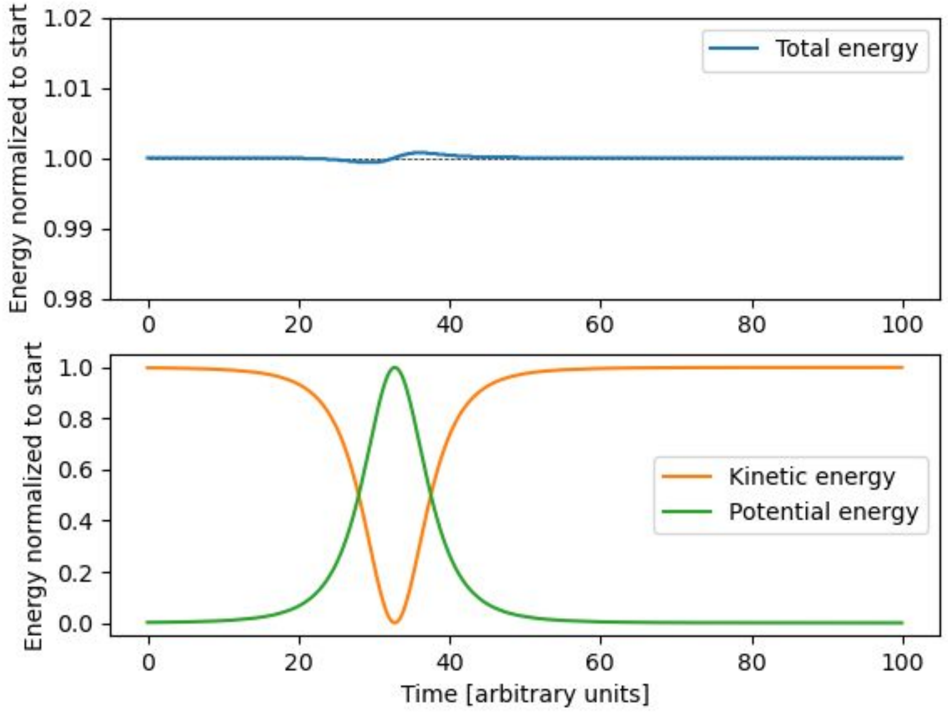
Energy conservation



$dt = 0.1$

(arbitrary units)

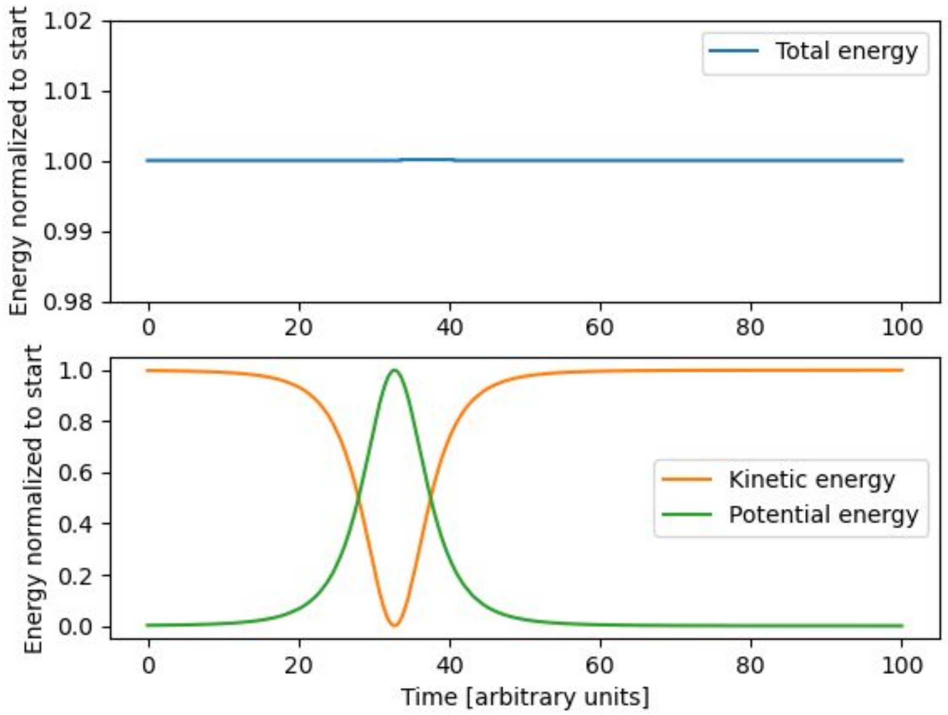
Energy conservation



$dt = 0.01$

(arbitrary units)

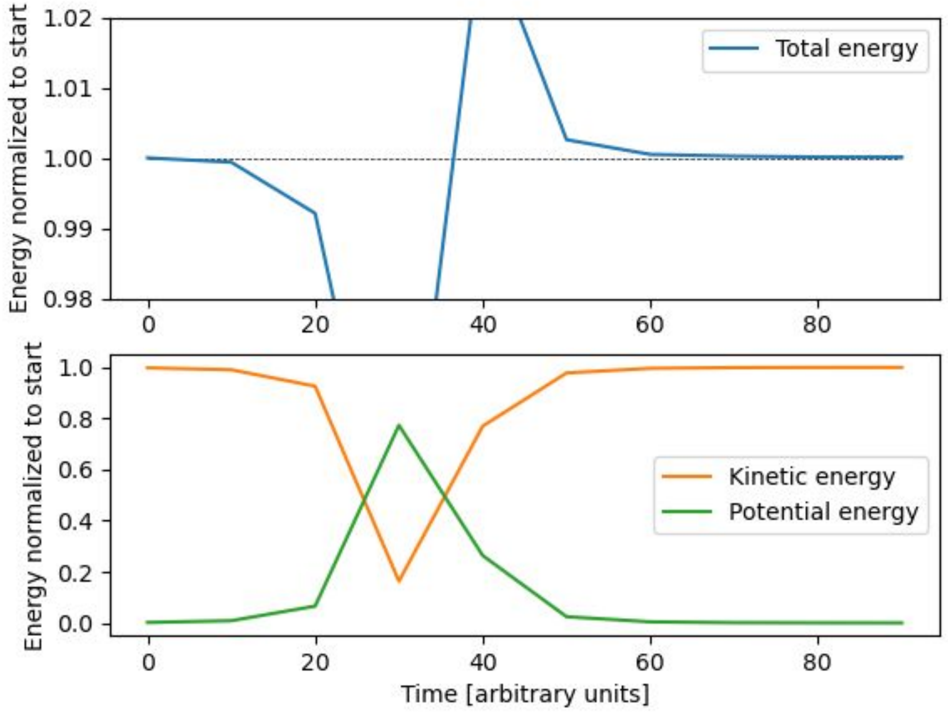
Energy conservation



dt = 0.001

(arbitrary units)

Energy conservation



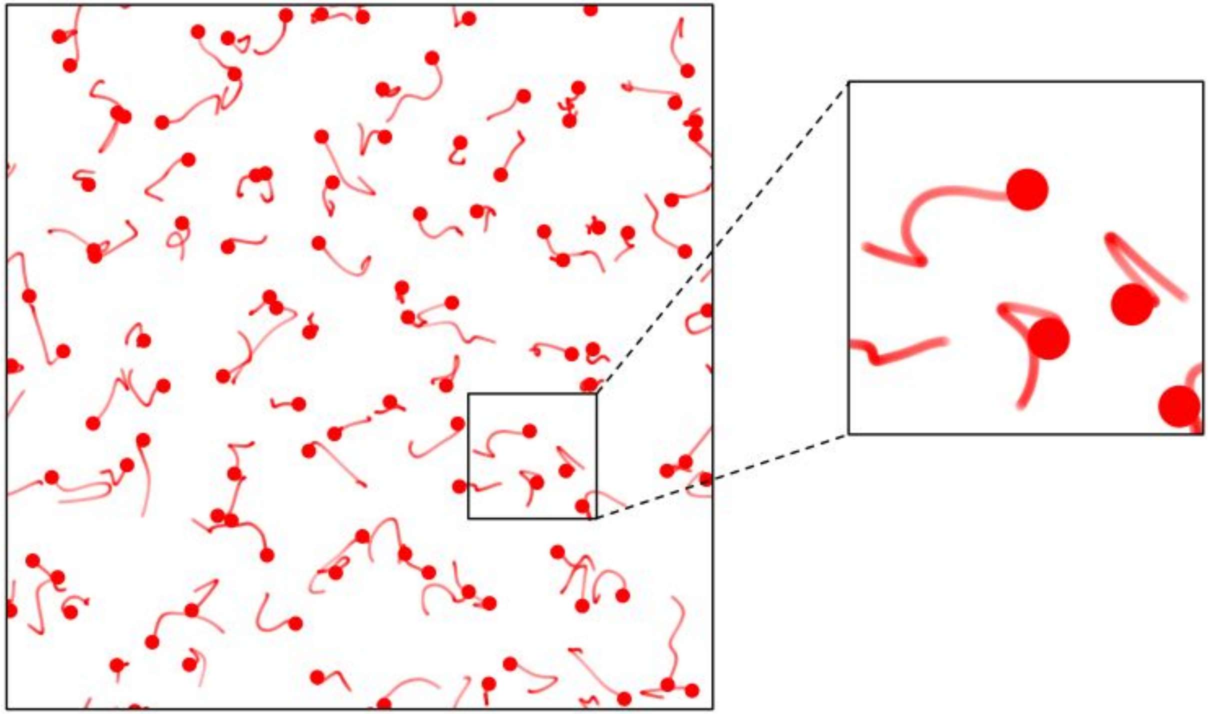
dt = 1.0

(arbitrary units)

Energy conservation

Any questions?

Where we are at



Project goal: Simulate different phases

We want to simulate **solid, liquid and gaseous Argon close to equilibrium**.

Project goal: Simulate different phases

We want to simulate **solid, liquid and gaseous Argon close to equilibrium**.

This means we want to create a system of particles with a “good” set of positions and velocities in a given volume.

Project goal: Simulate different phases

We want to simulate **solid, liquid and gaseous Argon close to equilibrium**.

This means we want to create a system of particles with a “good” set of positions and velocities in a given volume.

How do we get the starting state (the *Initial Conditions*) of our system based on an **input density and temperature** ?

Setting the density

Setting the density

We **know the mass** of each atom.

We **know how many atoms (particles)** we want in our simulation.

$$\rho = N \times m_{\mathrm{Ar}} / L^3$$

Setting the density

We **know the mass** of each atom.

We **know how many atoms (particles)** we want in our simulation.

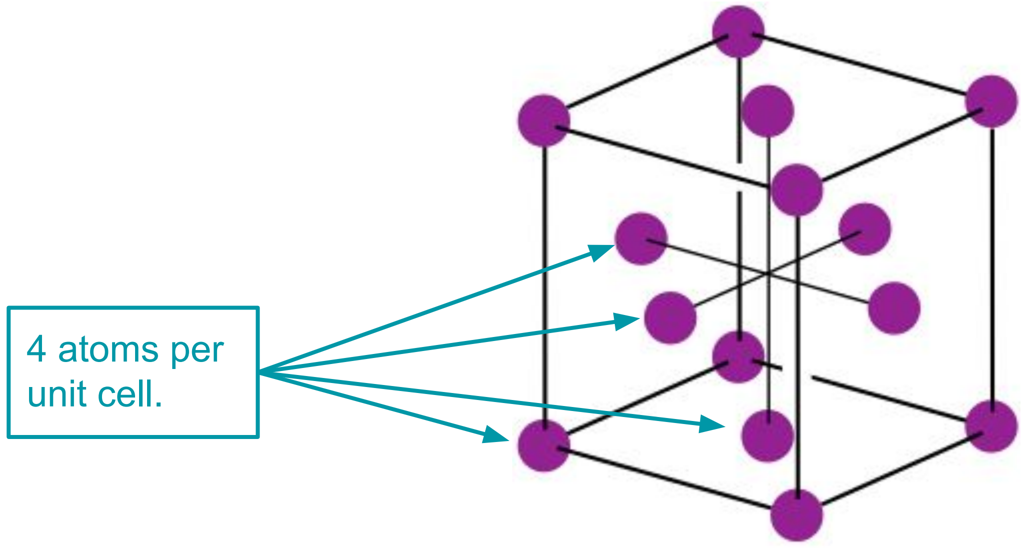
$$\rho = N \times m_{\mathrm{Ar}} / L^3$$

→ We can set the **volume** of the simulation to match the density we want.

Question for you: What is the expression for ρ in our dimensionless units?

FCC distribution of atoms

Argon atoms arrange themselves in crystals that follow a face-centred cubic pattern:



A whole crystal is created by replicating the base units in the *xyz*- directions.

Practical simulation size

To get a representative enough model, we will simulate a lattice of $3\text{x}3\text{x}3$ base FCC units. That's $N = 3^3 \times 4 = 108$ particles.

$\rightarrow$ We now have a way of setting the positions and simulation volume to reproduce the *density* we want.

Questions?

Setting the temperature

Setting a temperature

Let’s go back to statistical physics. We want a **Maxwellian distribution for the velocities**. That means, we have a **Gaussian distribution** for each component of the velocity vector:

$$p\left(v_x\right) \sim e^{-mv_x^2/(2k_BT)}$$

(same for v_y and v_z)

Question for you: What does this look like in our units?

Setting a temperature - Drawing random v’s

That looks relatively straightforward. **But it won’t work**.

Unless we are very very very lucky, the system will be out of equilibrium.

To get towards equilibrium, it will need to exchange kinetic energy for potential energy (and vice-versa). That can take a *really really* long time.

We need to do something to do bring the system closer to equilibrium.

Setting the velocities

Considering the degrees of freedom in the system, statistical physics tells us we should get a kinetic energy at equilibrium close to:

$$E_{\text{kin}}^{\text{target}} = (N-1)\frac{3}{2}k_B T$$

Note: Why N - 1 and not N? Total momentum is conserved, hence “blocking” one of the degrees of freedom.

Re-scaling the velocities

We can hence compute the total actual kinetic energy in the system and use this to re-normalize ***all*** the velocities.

1) Compute:

$$\lambda = \sqrt{\frac{(N-1)\,3k_BT}{\sum_i m v_i^2}}$$

Equilibrium target

Current kinetic energy

2) Re-scale ***all*** the velocities by the same constant: $\mathbf{v}_i \rightarrow \lambda \mathbf{v}_i$

Algorithm to set the initial velocities

1) Draw Maxwellian velocities.

For the pythonistas: `numpy.random.normal()`

2) Run the simulation for *a bit*.

3) Measure the kinetic energy.

$$E_{\text{kin}}^{\text{target}} = (N-1)\,\frac{3}{2}k_BT$$

4) If the energy is *far from* the expected equilibrium value, rescale the velocities.

5) Go back to (2). Repeat *a few* times.

Algorithm to set the initial velocities

1) Draw Maxwellian velocities.
2) Run the simulation for *a bit*.
3) Measure the kinetic energy.
4) If the energy is *far from* the expected equilibrium value, rescale the velocities.
5) Go back to (2). Repeat *a few* times.

For you to work out by trial and error.

$$E_{\text{kin}}^{\text{target}} = (N-1)\frac{3}{2}k_B T$$

Fourth Milestone

- Implement the initial conditions such that you can simulate the system at any user-specified **density** and **temperature**.

- Implement the relaxation method to drive the system towards equilibrium. (Derive the equations in dimensionless units!)

- Try to simulate a 3x3x3 “unit” of the Argon FCC structure ($N = 108$ atoms).

Fourth Milestone

You should be able to start a simulation that takes as input a *density* and a *temperature* only (for the 108 particles setup based on an FCC lattice).

The code should then:
- generate the distribution of particle positions,
- draw random Maxwellian velocities,
- and then apply the procedure to reach equilibrium.

Once equilibrium is reached, run for a while and observe the system.