Computational Physics 2023–2024

Matthieu Schaller

This lecture can be taken either as a 3 or a 6 EC course.

The first six weeks are devoted to one project in which you code a Molecular Dynamics simulation of a system of Argon atoms and study its physical properties for different temperatures and densities.

In this second half of the lecture you work on two additional projects. The first of these projects (3 weeks) consists of a Monte-Carlo simulation of the Ising model to study magnetization. This system is much more straightforward to code but you have less time and have to focus more on an error analysis of your results (more than in the previous project where we focus more on other aspects).

Finally, for the last three weeks you can choose your own project from a large pool of possible themes. If there are some areas of physics that interest you most, please do come forward and we can discuss projects that suit you best.

The course assesment is based on the reports that you submit for each project, including the simulation programs that you wrote as well as a discussion and interpretation of the results.

Project 1

In our daily lives we are used to encounter different phases of matter. For instance, water can be in the form of ice covering lakes and canals, it is fluid when it comes out of the tap, and it turns to vapour when you boil it. In this assignment you write a simulation code to explore different phases of matter quantitatively for a simpler system: Argon atoms (water is surprisingly complicated and still an ongoing field of research). Through these simulations you will be able to experience the physics of different phases and investigate their behavior in a systematic and quantitative way.

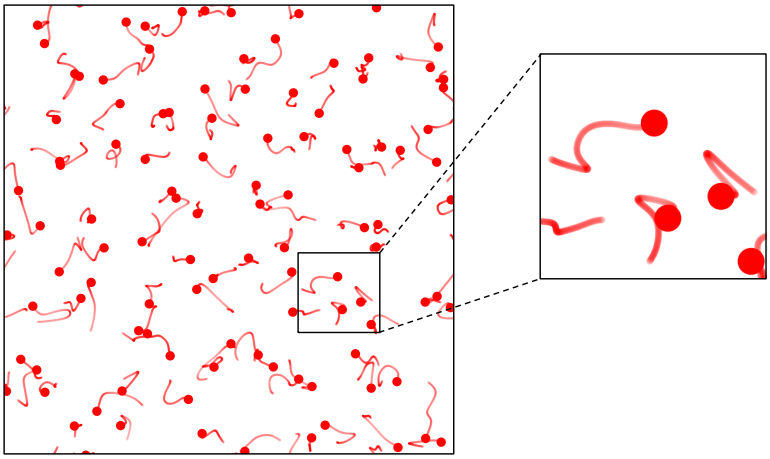

Figure 1: **MD simulation of a system of Argon atoms in a three-dimensional box with periodic boundary conditions.** The density ρ and temperature T correspond to the liquid state (specifically $\rho = 0.8$ and $T = 1.0$ in dimensionless units, see second lecture). Shown are the last 100 time steps in an equilibrated sample.

Molecular Dynamics Simulation Molecular dynamics (MD) simulations solve the equation of motion of a system of particles numerically. The method is indispensable as most systems of interacting particles cannot be solved analytically. For instance, whereas the trajectory of two celestial bodies that interact via the gravitational force can be solved exactly, there is already no general solution for the case of three bodies (three body problem). Before the advent of computers, the motion of e.g. Haley's comet was solved numerically (by hand) in order to be able to predict its next appearance. This was achieved by approximating its continuous trajectory by a discrete one, evolving the

system by small but finite time steps. The same is done in an MD simulation, often involving a huge number of interacting particles. An example can be seen in Fig. 1 which shows an MD simulation of 108 Argon atoms in a three-dimensional box. For each atom the last 100 positions can be seen. As small time steps have been chosen, the trajectories look almost continuous. In this project you will code this system yourself and study it in the gaseous, liquid and solid state. The particular example in Fig. 1 depicts a liquid.

The motion of each particle is governed by Newton's second law:

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

Of course, for a potential that only depends on the distance $r = \sqrt{x^2+y^2+z^2}$, we have $\nabla U(r) = \frac{dU}{dr}\frac{\mathbf{x}}{r}$. In the case of a potential that depends on the interaction between pairs of atoms, the force term on atom i is computed as

$$\mathbf{F}_i = \mathbf{F}(\mathbf{x}_i) = -\sum_j \nabla U(\mathbf{x}_i - \mathbf{x}_j), \tag{2}$$

where the sum runs over all the other atoms j in the system.

Numerical integration. In most cases these equations cannot be solved analytically or even with perturbation theory. In an MD simulation they are solved approximately. The general idea is to replace the evolution in continuous time by an evolution with finite time steps h. The simplest (often too simple) way to solve Eq. 1 is as follows. If $\mathbf{x}_n$ are the positions and $\mathbf{v}_n$ the velocities at time t_n, the positions and velocities at the next time point $t_{n+1} = t_n + h$ are given by

$$\mathbf{x}_{n+1} = \mathbf{x}_n + \mathbf{v}_n h \tag{3}$$

and

$$\mathbf{v}_{n+1} = \mathbf{v}_n + \frac{1}{m}\mathbf{F}(\mathbf{x}_n)\, h. \tag{4}$$

These so-called Euler methods for integrating ordinary differential equations (ODEs) have a major downside for the molecular dynamics simulations we are doing: they do not conserve the energy. But our goal is to simulate a system with conserved energy (the microcanonical ensemble), so we should look for a better algorithm. For now, we will nevertheless use this most simple approach. We shall pick up here again in the third lecture.

Different phases in a system of Argon atoms In the following you simulate phase transitions in a system of interacting particles that mimic Argon atoms. Depending on the externally imposed conditions (temperature and density) the system is either in a gaseous, liquid or solid state. Why do we choose Argon and not water, which is closer to our own experience? Water turns out to be very complicated, as the molecule is asymmetric and forms hydrogen bonds, etc. We want something simpler and a noble gas is a first choice as we do not have to worry about the formation of molecules. Historically, Argon (Ar) is the best studied system and that is why we chose this element here. *Question for you: Why not Helium?*

Unlike in an ideal gas system, the atoms interact. As Argon atoms are neutral, they do not interact via the Coulomb interaction. However, small displacements between the nucleus and the electron cloud give rise to a small dipole moment. Overall, the interaction between dipoles gives an attractive interaction that scales as $1/r^6$, where r is the distance between atoms (van der Waals interaction). On the other hand, at short distances there is a repulsion of quantum mechanical origin (Pauli repulsion). Both of these aspects are captured in the Lennard-Jones potential, which takes the form

$$U(r) = 4\epsilon \left(\left(\frac{\sigma}{r} \right)^{12} - \left(\frac{\sigma}{r} \right)^{6} \right). \tag{5}$$

The specific form of the repulsive $(\sigma/r)^{12}$-term has just historic reason (it is convenient as it is the square of the attractive term). For Argon the fitting parameters have been determined to be $\epsilon/k_B = 119.8\,\mathrm{K}$ and $\sigma = 3.405\,\text{Å}$.

System boundaries We want to simulate an infinite system of Argon atoms; i.e. not interactions of Argon with a wall for instance. But obviously, a computer cannot simulate infinite space. In practice, we will use a finite box of length L, with periodic boundary conditions (to mimic an infinite system). The periodic boundary condition has two important consequences:

- If a particle leaves the box, it reenters at the opposite side. *Question for you: What happens to the velocities?*
- The evaluation of the interaction between two atoms is also influenced by the periodic boundary conditions, as they introduce an infinite number of copies of the particles around the simulation box with which a given particle interacts.

That second point can lead to very complex situations depending on the shape of $U(r)$. Fortunately, the Lennard-Jones potential decays rather fast

with distance, namely like r^{-6}. That means even though the interaction of a given particle, say i, to another one, say j, contains infinite many terms (stemming from the original particle and its infinite number of copies), only one term contributes significantly. All others are much smaller as we will essentially always be in a situation where the box size L is significantly larger that the inter-atomic distance. It is only the copy (or the original) of particle j that is closest to particle i that needs to be accounted for; the rest can be safely neglected. This is called the *minimum image convention* and it is used in the following. Figure 2 depicts an example. It shows the original box with five particles in the center and images of the system all around (in pale colours). Even though the orange particle interacts with four particles and their infinitely many copies, only four interactions need to be accounted for. In this particular configuration only one of those four interactions is an interaction between a pair of two original particles.

Now, how do the particle positions evolve over time? This is explained above. Here you should implement the most simple scheme, Eqs. 3 and 4. The particles' positions and velocities can be most conveniently stored in `numpy` arrays. This allows then to represent the whole system of Eqs. 3, 4 and 5 in a very compact form in your code.

First Milestone

Code up and play around with this system.

Start by simulating the time evolution of a *few* particles (e.g. two particles on collision course or several particles with random initial positions and velocities) in a periodic box, add the forces due to the Lennard-Jones potential, employing the minimum image convention. Store the trajectories of the particles in a file and then display them (e.g. as a scatter-plot using `matplotlib`).

Hint: It's easier to start with a 2D system (since you can more easily visualise it), but plan to switch to 3D at a later stage.

Hint: Start small first. Two particles, then three and so on.

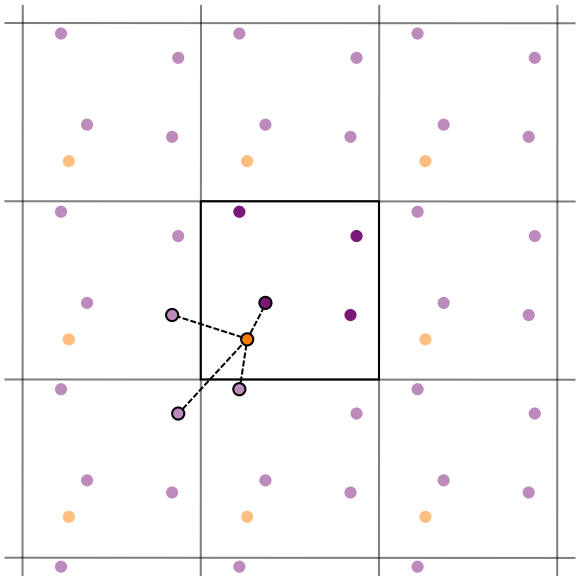

Figure 2: **The minimum image convention in 2D.** We consider a simulation with 5 particles (the central box). To explain what is going on, 8 additional identical boxes have been drawn here; they correspond to the periodic images of the simulation box. They are not themselves part of the simulation. In principle, the orange particle needs to interact with the 4 purple particles in the main box but also with *all other infinitely many* copies of these particles along each dimension in the tiling. Since $U(r)$ decays as r^{-6}, however, all these copies will contribute something very small, which we can safely neglect. The orange particle then only needs to interact with the closest "version" of each purple particle. This can either be the purple particle in the simulation box itself, or any of its periodically replicated "clones". The four interactions to consider here are shown as dashed lines.