Molecular Dynamics Project

FYS-MEK1110 - Mechanics

Tommy Myrvik and Anders Johansson

18th July 2018

Part 1 Introduction

In this project you will learn the basics of a simulation technique called molecular dynamics (MD). Molecular dynamics is a method actively used in research here at the Department of Physics, yet its basic principle can be understood and implemented with the background of a first-year physics student.

Molecular dynamics is based on the assumption that even atoms move according to the laws of Newton, given the correct model for interactions. The goal of this project is to model an argon gas, where the atoms interact according to the famous Lennard-Jones potential,

$$U(r) = 4\varepsilon\left(\left(\frac{\sigma}{r}\right)^{12} - \left(\frac{\sigma}{r}\right)^{6}\right), \tag{1}$$

where r is the distance between two atoms, $r = \|\vec{r}_i - \vec{r}_j\|$. σ and ε are a parameters which determine which chemical compound is modelled. This potential is a good approximation for noble gases.

1a) Understanding the potential

i. Plot the potential as a function of r with $\varepsilon = 1$ and $\sigma = 1$.

ii. The behaviour of $U(r)$ is vastly different for $r \ll \sigma$ and $r \gg \sigma$. Which term in the potential, equation (1), dominates in each case and what is the effect?

iii. Find and characterise the equilibrium points of the potential.

iv. Describe qualitatively the motion of two atoms which start at rest separated by a distance of 1.5σ. What if they start with a separation of 0.95σ? (hint: use the graph of the potential).

v. Characterise the shape of the potential around $r = 1.0\sigma$. Can you think of other force(s) with the same behaviour?

1b) Forces and equations of motion

i. Find the force on atom i at position $\vec{r}_i$ from atom j at position $\vec{r}_j$.

ii. Show that the equation of motion for atom i is

$$\frac{\mathrm{d}^2\vec{r}_i}{\mathrm{d}t^2} = \frac{24\varepsilon}{m}\sum_{j\neq i}\left(2\left(\frac{\sigma}{\|\vec{r}_j - \vec{r}_i\|}\right)^{12} - \left(\frac{\sigma}{\|\vec{r}_j - \vec{r}_i\|}\right)^{6}\right)\frac{\vec{r}_j - \vec{r}_i}{\|\vec{r}_j - \vec{r}_i\|^2}. \tag{2}$$

1c) Units

As you may remember from MAT-INF1100, numerical accuracy is reduced when computing with values which are many orders of magnitude apart. This is often an issue in physics, and molecular dynamics is no exception. For example, the mass of argon is smaller than 10^{-25} kg, while typical length scales are on the order of nanometres, 10^{-9} m.

The remedy is to change units so that most quantities are close to 1. From equation (1) on the previous page it is clear that σ and ε are the typical scale for length and energy.

i. Introduce the scaled coordinates $\vec{r}_i{}' = \vec{r}_i/\sigma$ and show that the equation of motion can be rewritten in terms of these coordinates as

$$\frac{\mathrm{d}^2\vec{r}_i{}'}{\mathrm{d}t'^2} = 24\sum_{j\neq i}\left(2\|\vec{r}_j{}' - \vec{r}_i{}'\|^{-12} - \|\vec{r}_j{}' - \vec{r}_i{}'\|^{-6}\right)\frac{\vec{r}_j{}' - \vec{r}_i{}'}{\|\vec{r}_j{}' - \vec{r}_i{}'\|^2}, \tag{3}$$

where $t' = t/\tau$ for a suitable choice of τ.

ii. What is the characteristic time scale τ, and what is its value for argon, which has $\sigma = 3.405$ Å (1 Å $= 1 \cdot 10^{-10}$ m), $m = 39.95$ u (1 u $= 1.66 \cdot 10^{-27}$ kg) and $\varepsilon = 1.0318 \cdot 10^{-2}$ eV (1 eV $= 1.602 \cdot 10^{-19}$ J)?

Part 2 Two-atom simulations

2a) Implementation

i. Write a function which solves equation (3) for two atoms and finds the positions and velocities of the atoms as a function of time. Implement three different integration methods: Euler, Euler-Cromer and Velocity-Verlet (see appendix C on page 5 for a description of the latter).

2b) Motion

i. Simulate the motion of two atoms which start at rest separated by a distance of 1.5σ. Use $\Delta t' = 0.01$.

ii. Plot the distance between the atoms as a function of time.

iii. How does the motion fit with your expectations from exercise 1a) on the preceding page?

2c) Energy

i. Plot the kinetic, potential and total energy as a function of time. Specify your choice of parameters.

ii. Should the total energy be conserved? Why, or why not?

iii. Does your program fulfill this? If not, what could be the cause?

iv. Simulate the same system as in exercise 2b) with the Euler, Euler-Cromer and Velocity Verlet algorithms for a total time $t' = 5.0$, and compare graphs of the total energy as a function of time.

2d) Different motion

i. Repeat exercise 2b) on the preceding page with an initial separation of 0.95σ. Explain your results.

2e) Visualisation

i. Extend your implementation such that it writes to an `xyz`-file at each timestep (see appendix A on the next page).

ii. Visualise the results of your simulations using Ovito (see appendix B on page 5).

Part 3 Large systems

3a) Implementation

i. Implement a solver of equation (3) on the previous page for N atoms, given initial positions and velocities.

ii. Use Newton's third law to reduce the number of force calculations.

iii. Reproduce your results for the 2-atom model from the previous section to verify your implementation.

As you will experience, it takes a lot longer to simulate N atoms than two — in fact the time increases as N^2. One very simple way to reduce simulation times is to look at the expression (or plot) of $U(r)$ and see that it goes very rapidly towards zero as r increases. This means that atoms far apart interact weakly, and the forces between them can be ignored.

iv. Extend your implementation such that atoms more than 3σ apart do not interact.

3b) Four atoms

i. Simulate the motion of four atoms starting at rest from the positions $[1, 0, 0]$, $[0, 1, 0]$, $[-1, 0, 0]$ and $[0, -1, 0]$.

ii. Visualise the results in Ovito, describe and explain the motion.

iii. Plot the potential, kinetic and total energy as a function of time, and comment on the energy conservation.

iv. Repeat the above exercises with a small perturbation in the initial positions, such that the first atom starts at $[1, 0.01, 0]$.

3c) Initialisation

While we are interested in simulating liquid argon, which will not be in an ordered structure, the simplest choice of initial positions is a regular crystal structure. Our choice of structure is the face-centred cubic lattice, as this is the crystal structure of solid argon.

The smallest repeating unit is called a unit cell, and each unit cell contains four atoms. When creating a structure of $N \times N \times N$ unit cells, the atoms should be placed at

$$\begin{bmatrix} i & j & k \\ i & 0.5+j & 0.5+k \\ 0.5+i & j & 0.5+k \\ 0.5+i & 0.5+j & k \end{bmatrix} \cdot d$$

where i, j and k run from 0 to $N-1$ and d is the size of the unit cell. This structure will contain $4N^3$ atoms in total.

i. Write a function which takes N and d as arguments and returns the positions of $4N^3$ on a face-centred cubic lattice.

ii. Verify your implementation by calling your function for $N = 4$, writing the resulting positions to an `xyz`-file and looking at the result in Ovito. Your system should contain $4 \cdot 4^3 = 256$ atoms.

Appendix

A Data file format

The `xyz`-format is a semi-standard format for storing data from molecular dynamics simulations. Each time step is stored in the following format, and there are no blank lines between timesteps:

- A line containing the number of atoms (an integer).
- An ignored line (this line is usually written as a header for the subsequent columns).
- One line for each atom, containing the atom type and the x-, y- and z-koordinates.

For two atoms simulated over three timesteps, where `xij` represents the x-coordinate of atom j at timestep i, the file would look like this:

```
2
type  x   y   z
Ar   x11 y11 z11
Ar   x12 y12 z12
2
type  x   y   z
Ar   x21 y21 z21
Ar   x22 y22 z22
2
```

```
type  x   y   z
Ar   x31 y31 z31
Ar   x32 y32 z32
```

B Visualisation

Files written in the `xyz`-format can be read using the Ovito visualisation tool. It can be downloaded and installed from `https://ovito.org/index.php/download`.

When the installation has finished, simply open Ovito and click "File" $\rightarrow$ "Load File" and choose your `xyz`-file. Edit the column mapping in the dialogue if necessary. When the atoms have appeared on your screen, check the box named "File contains time series" on the right-hand side, press ▷ and watch your atoms move around!

C Velocity-Verlet

The Velocity-Verlet integration scheme is based on a second order Taylor polynomial. With $\vec{r}_i(t)$ denoting the position of atom i at a time t, the second order Taylor expansions of position and velocity can be written as

$$\begin{aligned}\vec{r}_i(t+\Delta t) &\approx \vec{r}_i(t) + \vec{v}_i(t)\Delta t + \tfrac{1}{2}\vec{a}_i(t)\Delta t^2\\ \vec{v}_i(t+\Delta t) &\approx \vec{v}_i(t) + \vec{a}_i(t)\Delta t + \tfrac{1}{2}\vec{a}_i'(t)\Delta t^2.\end{aligned}$$

There is no explicit expression for $\vec{a}'(t)$. It can, however, be approximated using our old friend

$$\vec{a}\,'(t) \approx \frac{\vec{a}(t+\Delta t) - \vec{a}(t)}{\Delta t}.$$

Since the acceleration is independent of the velocity, the newly updated position, $\vec{r}(t+\Delta t)$, is sufficient to calculate $\vec{a}(t+h)$. Inserting this into the expression for $\vec{v}(t+\Delta t)$, we get

$$\begin{aligned}\vec{v}_i(t+\Delta t) &\approx \vec{v}_i(t) + \vec{a}_i(t)\Delta t + \tfrac{1}{2}(\vec{a}_i(t+\Delta t) - \vec{a}_i(t))\Delta t\\ &= \vec{v}_i(t) + \tfrac{1}{2}(\vec{a}_i(t) + \vec{a}_i(t+\Delta t))\Delta t.\end{aligned}$$

The discretised algorithm then becomes

$$\begin{aligned}\vec{r}_{i,j+1} &\approx \vec{r}_{i,j} + \vec{v}_{i,j}\Delta t + \tfrac{1}{2}\vec{a}_{i,j}\Delta t^2\\ \vec{v}_{i,j+1} &\approx \vec{v}_{i,j} + \tfrac{1}{2}\left(\vec{a}_{i,j} + \vec{a}_{i,j+1}\right)\Delta t,\end{aligned}$$

where $\vec{r}_{i,j}$ is the position of atom i at timestep j. In your implementation, you should avoid having to calculate the acceleration more than once per timestep.

Part 1: 2-atom model

Bare legger en liten mal her på oppgavene jeg har programmert så langt. Veldig overfladisk oppgavetekst, kan og bør endres underveis.

a)

Plot the LJ-potential curve. What does the different terms in the potential do?

b)

Find the force corresponding to the potential from a), and plot the result. For what distance r is this force 0? Is this force conservative? *What's different with this force compared to say Newton's Law of gravitation? What about the force from a spring?*

c)

Say we place one atom at $\vec{r_1} = [0, 0, 0]$ and the other at $\vec{r_2} = [1.5, 0, 0]$, with no initial velocities. Describe qualitaively what the motions of the atoms will look like. Support your arguments with the LJ potential curve.

If we place the second atom at $\vec{r_2} = [0.95, 0, 0]$ instead, what do you expect then? Explain by again using the potential curve from task a).

d)

Develop a code to simulate a system of two atoms with the LJ-forces being the only ones acting. Use the Euler-Cromer integration method (?). Simulate the system with the initial conditions (both cases) from task c), and plot the distance between the atoms r as a function of time. What do you see?

e) (optional)

Download a visualization tool (i.e. Ovito), and write a function that writes the positions of the atoms at every time step to a xyz.-file. Load this file into *Ovito* and describe what you see. Does this fit well with the assumptions you made in task c) ?

f)

Implement different integration methods so that you have the Euler, Euler-Cromer and Velocity-Verlet methods available for your simulations. Run simulations with the same initial conditions as in the first case in task c) ($\vec{r_1} = [0, 0, 0]$, $\vec{r_2} = [1.5, 0, 0]$) for all three methods, and plot the mechanical energy for all three with $\Delta t = 0.01$. Compare the results for all three methods. How does these methods perform in terms of energy conservation?

g) (I tvil om denne skal med)

Run simulations for all three methods implemented in the previous task, and find the an approximation to the largest time step (Δt) required to keep the integration from exploding. Whats the difference between these methods, and why does some of them perform better than others?

Part 2: N-atom model

h)

Generalize the program you wrote in the previous part for any number of atoms, N. Computing time increases drastically with system size, so some precautions should be made:

- Pair-wise forces: The LJ-potential describes forces *between* atoms, meaning that the forces acting on atom i from atom j is the same as the forces acting on atom j from atom i, just with opposite signs. Realizing this can cut the computations in half, in contrary to computing the same force twice.
- Hopefully when you plotting the LJ-potential earlier you saw that the forces converge towards 0 as $r \to \infty$. This means that atoms that are far away from each other share little to none influence, and are basically neglectable. Computing the forces between these atoms are hence a waste of computing power, and can be ignored by implementing a cut-off length - If the atoms are further away than this length, the forces are set to 0. For the LJ-potential this cut-off is usually set to 3.0 (LJ units). **Note:** The LJ-force at $r = 3.0\sigma$ is NOT exactly 0, meaning that with the cut-off implemented, the potential curve will do a small 'jump' at this point (messing up the conservation of energy, etc). A solution to this is to simply shift the whole potential curve by the value at the cut-off point, leading to a smooth curve (no jumps) with 0 forces at cut-off.
- Due to memory restrictions it may be a bad idea to keep positions and velocities for all time steps stored in arrays. MD-simulations typically need large system sizes (N) and small time steps (dt) to create decent results, potentially leading to really huge matrices. For the simulations we'll do in this project we *probably* won't need blue-screen inducing matrices (depending on the memory resources you have available), but if you're going to experiment further with different time steps and system sizes this *may* become a problem. A solution here is to just keep positions and velocities for two time steps at a time, and write out the data to a text-file instead. This way there is no memory constraints, only on your patience waiting for the simulations to complete.

Reproduce the results from the two-atom case with the generalized code for verification.

i)

Start with a 4-atom system. Place the atoms so that $\vec{r}_1 = 2\vec{i}$, $\vec{r}_2 = -2\vec{i}$, $\vec{r}_3 = 2\vec{j}$ and $\vec{r}_4 = -2\vec{j}$. Compare with the experiment with two atoms from earlier. What do you see? Visualize with Ovito, and explain the results. Calculate the potential and kinetic energy of the system, and comment on the conservation of energy.

j)

Do the same again, but now perturb *one* of the atoms, giving them a small positional component (say 0.1) in one of the directions where it originally was 0. Visualize the results, and describe the effect of this small perturbation. Comment again on the energy of the system.

k)

If you have played around with different Ns and dts you've probably seen that this system isn't very stable - the atoms start spreading around outside the initial box, and some might really gain speed and aim for infinity and beyond (especially if the initial positions are generated randomly). This is mainly due to two parts still missing: good initialization and boundary conditions on the simulation box boundaries. We will first implement the former:

It might seem like a reasonable idea to initialize the atoms uniformly in the simulation box by splitting the simulation box into N smaller, equally sized boxes (called *unit boxes*), and putting one atom at the center of each box (and to some extent this is reasonable). It can be shown however that the equilibrium state of a system of atoms interacting with a LJ-potential doesn't prefer this simple cubic structure, but rather a face-centered cubic structure (fcc). Instead of having just one atom in each unit box, we instead have *four*, which practically means that the number of atoms in the system should be a multiple of 4. In this structure, one atom is attached to the one of the corners of the unit box, and the other three are put at the center of the three walls connected to this first atom (see drawing).

Put drawing here

Write a function that generates this structure - visualize with a small system to see that the implementation is working as intended.

l)

The main reason that atoms occasionally escapes the system is the obvious one - we are allowing them to. In a MD-simulation we're for the most part simulating the behaviour of a *tiny* part of a much larger material, so the box we're simulating will in reality be surrounded by other similar boxes. So far we've not implemented anything telling the atoms what happens when they cross the boundaries of the simulation box, but we have a few options available:

- Reflective boundaries: A simple, but less realistic method is to simply reflect the atoms moving out of the box back in. This simulates a hard wall situated at the boundaries, and while it does conserve energy, it does a poor job of keeping the fcc structure of the system.

- Periodic boundary conditions: A more realistic option, where instead of the atoms being reflected back, they appear on the other side of the system.

BlaBla om at det også må regnes ut krefter på tvers av grensene.

Part 3: Let's do some science!

References

[1] Aneesur Rahman. "Correlations in the Motion of Atoms in Liquid Argon". In: *Physical Review* 136.2A (Oct. 1964), A405–A411. DOI: `10.1103/physrev.136.a405`. URL: `https://doi.org/10.1103%2Fphysrev.136.a405`.