Requirements document

Nicollas Gillard Barroso, Rubén Martínez Ibañez and Iván Villegas Pérez

Autumn 2022

With the use of Python, so far we have been capable of modeling the behaviour of an ideal gas inside a box, without interatomic interactions. We now want to take one step further and add these. To do so, we start assuming our fluid is a solid. There is a given number of cells, each with four atoms in it, as it is assumed to be a FCC lattice type cell. The way atoms interact with each other will be determined by a Lennard-Jones potential, which is inversely proportional to the interatomic distance to the power of 6. If we were to analyze the interaction each atom has with the rest, the program would not be efficient, as it would take a vast amount of time to load and calculate such a number of processes. To avoid this, we define a function whose purpose is to only take into account atoms which are close enough.
As atoms' temperature will not be absolute zero ($T \neq 0$K), they will vibrate around its equilibrium position, until the temperature is large enough for it to lose its structure, hence it is melted.

A UML diagram can be seen in the attached file or in our public GitHub repository[1].

A sequence char can be seen in Figure 1.

Before creating our model, we first need to introduce the conditions of the system, such as the temperature, the number of cells, the density of the material, etc. A program we shall call "inputs.py" will have the purpose of reading this data and transforming it into usable units, which are the outputs of it. The units we will be using are natural units, hence sigma and epsilon will be equal to 1. The second code we are creating will be called "neighbours.py", which, as explained briefly before, will first obtain the relative distance between two atoms and, if the distance between two given atoms is large enough, when analyzing the behaviour of one of these atoms, this distant atoms will not be taken into account until a period of time large enough, hence, after knowing all the initial positions, in every loop it will interact with a few atoms until enough time has passed to other atoms to not be as negligible. The next program we want to create will be called "LennardJonesPotentials.py", which applies all the potentials and obtains the dynamics of the system. Finally, the program "initialization.py" joins all the previous program, obtaining the simulation we want, where we have the units to work with from the "inputs.py", each atoms only interacts with atoms from "neighbourlist.py" and obtains the potentials and dynamics from "LennardJonesPotentials.py".

List of requirements:

- Read the input variables from a file, in any order.
- Receive the temperature from any degree units and convert to Kelvin.
- Use reduced units, such as $\epsilon = 1$ and $\sigma = 1$.
- Implement the Lennard-Jones potential to every atom, so that they interact with each other.
- Initially consider an infinite solid at T=0, following a FCC lattice.
- Consider that the initial velocities are random, and the module follows the Maxwell-Boltzman distribution. Then compute an histogram.
- Print the input variables as an output file to check if they are correct.

[1] https://github.com/ivanvillegas7/Requeriments_Document/tree/main

- Print the initial positions and initial velocities in a text file.
- Print the potentials, the kinetic energy and the total energy in a text file.
- Compute a video of the simulation using a text file with the positions and velocities of each atom in each step.

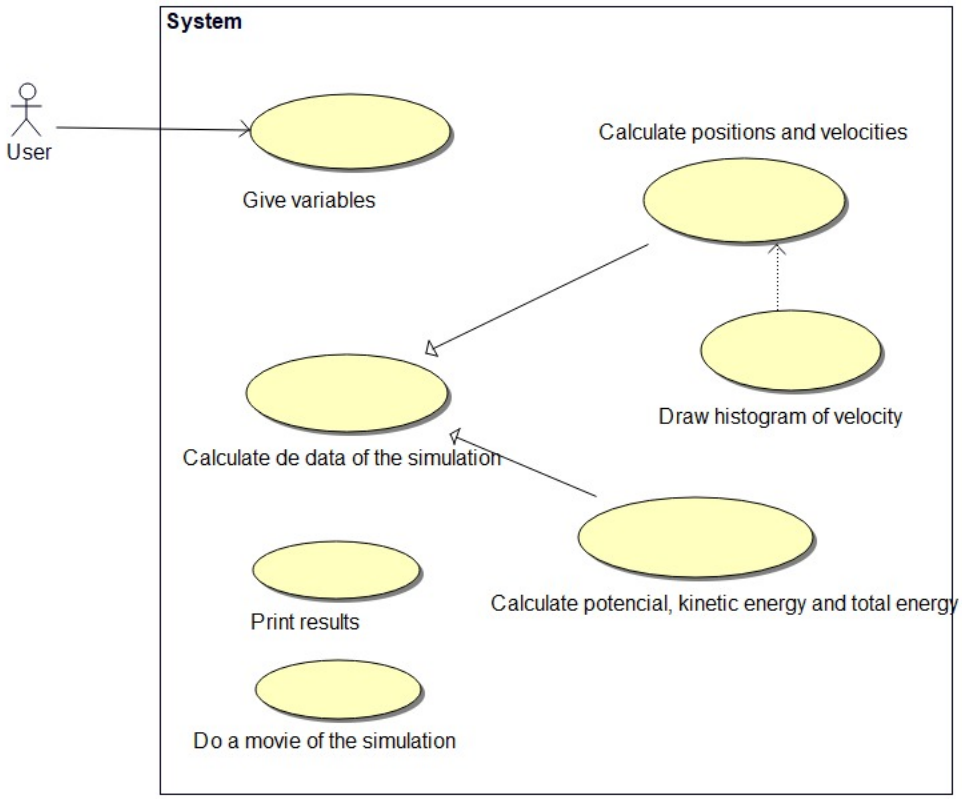

Figure 1: Sequence chart exemplifying one of the scenarios.