Molecular Dynamics
FYS-3150

Gullik Vetvik Killie

October 30, 2014

Contents

Notes

- Is it not the Leapfrog algorithm?
- Need to invent some unit test to implement, ask for ideas
- Is it necessary to compute potential? Analytical form of force should be enough. Loss of computing speed when not caring about potential
- Dipoles? Should form crystals
- Magnetic force
- Different molecules
- Crystals mixed with liquids
- Unit Test, check that it stays stable if temp is zero, then forces should be zero for crystal configuration.

1 Task a: Periodic boundary conditions

This task was implemented in the function applyPeriodicBoundaryConditions() and the program can be found at the github page [1].

Since we have limited computational power we are only working on a small piece of argon material, to get away from boundary cases we simulate a system of infinite size by implementing periodic boundary conditions. This is done by confining the atoms in a box and each time a particle reaches the end of the box it is put to the other side.

2 Task b: Removing momentum

This task is implemented in the function removeMomentum().

Since all the atoms are given random velocities the net momentum is not zero and the cloud will drift. This is solved by calculating the total momentum of all the atoms, and then subtracting a fraction of the total momentum from each atom so the total momentum ends up being zero.

3 Task c: Putting the atoms on a lattice

This task is implemented in the function createFCCLattice()

The noble gas argon should have a stable lattice structure when a solid, by letting the system start in a stable situation we avoid a lot of energy to be infused into the systems temperature due to it minimizing potential energy. The implementation is done by going through several nodes, R_i, put on a grid in the box and then placing 4 atoms around each node. Let c_l be the length between nodes.

$$\mathbf{R}_{ij} = \mathbf{R}_i + \mathbf{r}_j \qquad j = \{1,2,3,4\} \qquad i = \{1,2,...,4\mathrm{N}_{\mathrm{atoms}}\}$$

$$\begin{aligned}
\mathbf{r}_1 &= 0\hat{\mathbf{i}} + 0\hat{\mathbf{j}} + 0\hat{\mathbf{k}} \\
\mathbf{r}_2 &= \frac{c_l}{2}\hat{\mathbf{i}} + \frac{c_l}{2}\hat{\mathbf{j}} + 0\hat{\mathbf{k}} \\
\mathbf{r}_3 &= 0\hat{\mathbf{i}} + \frac{c_l}{2}\hat{\mathbf{j}} + \frac{c_l}{2}\hat{\mathbf{k}} \\
\mathbf{r}_4 &= \frac{c_l}{2}\hat{\mathbf{i}} + 0\hat{\mathbf{j}} + \frac{c_l}{2}\hat{\mathbf{k}}
\end{aligned}$$

4 Task d: Implementing a Verlet integrator

Se subsection 8.1

5 Task e: Computing forces

See subsection 8.2

6 Task f: Calculating some statistical properties

6.0.1 Kinetic energy

The kinetic energy is found trough adding up the kinetic energy for all the atoms separately. It is straight forward and done in a function, in the statisticsSampler class, called sampleKineticEnergy(). It goes through all the atoms and adds it's contribution to the total kinetic energy.

$$E_k = \sum_{i=1}^{N_{atoms}} \frac{1}{2} m_i v_i^2$$

Then it also stores the instantaneous temperature of the substance which is given by the equipartition theorem $\langle E_K \rangle = \frac{2}{3} k_B N_{atoms} T$. By looking at the kinetic energy in a time-instant instead of averaging it we get an instantaneous temperature.

All of this is stored together with the potential energy to a text file "./statisticalResults/statisticalValues.tsv" which can be plotted by the python program plots.py.

7 Task g: implementing Neighbor cells

Table 7 shows the time spent computing with different amount of atoms in the system, as the system increases the time is increasing fast and we can see the need to implement neighbor cells. The time is increasing fast because the program has to calculate the forces between all the atoms and that is of the order $\mathcal{O}(N^2)$. By only calculating the nearby atoms, which will be the ones affecting each other the most, the time taken will mostly increase linearly.

N_{atoms}	4	32	108	256	500	864	1372 2048	
time (s)	0.003539	0.054251	0.274064	1.22426	3.96325	11.3712	28.3554	62.9434

Table 1: This table shows time to compute 100 timesteps with different amount of atoms in the model. The time is increasing much faster than linearly.

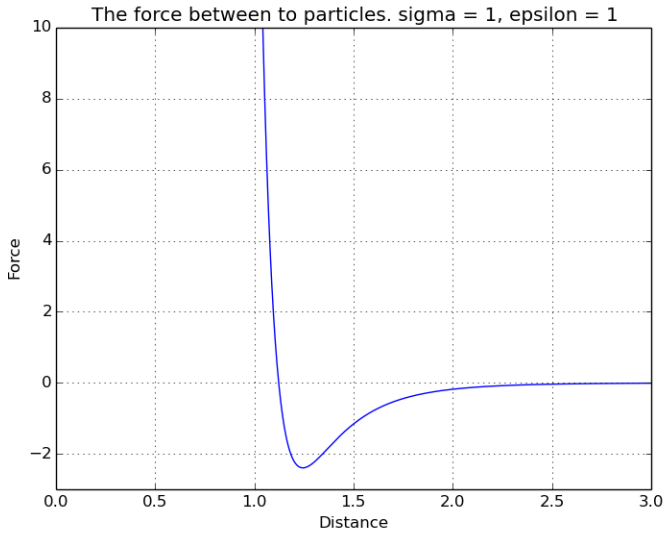

Figure 1: A picture over the force felt between two particles due to the Lennard-Jones potential. At approximately 3 σ of length the magnitude of the force closes to zero and we can ignore the contribution of atoms further away from each other.

The force is mostly ignorable outside a distance of 3σ, see figure 7.

- Divide box into smaller boxes with $l > 3\sigma$, and put the cell layer class between the system and atoms.

  $$\text{system} \rightarrow \text{cellbox} \rightarrow \text{atoms}$$

- For each box go through all the boxes neighboring boxes. Boxes should be stored in three lists as (i, j, k) and the neighboring boxes to box (i, j, k) are $(i-1, j-1, k-1), (i-1, j-1, k)$ and going through all the combinations to $(i+1, j+1, k+1)$
- Each atom needs an additional box assignment property
- Calculate the forces for all the atoms belonging to the neighboring boxes

8 Theory

8.1 Verlet integrator

The Integrator is a widely used integrator in molecular dynamics [3] because of it's properties as a symplectic integrator which means that it conserves areas in phase space very well and it

allows stable integration of the equation of motion [4]. Newton's second law for for a particle in our molecule ensemble reads:

$$m\frac{\partial^2 x_i}{\partial t^2} = F_i$$

$$\frac{\partial x_i}{\partial t} = v_i \text{ and } \frac{\partial v_i}{\partial t} = \frac{F_i}{m}$$

The Leapfrog algorithm is a slight continuation on the Verlet algorithm doing the step in two steps. Doing a Taylor expansion around both the step and half the step we get.

$$x_i(t+h) = x_i(t) + hx_i'(t) + \frac{h^2}{2}x_i''(t) + \mathcal{O}\big(h^3\big) \tag{1}$$

$$x_i'(t+\frac{h}{2}) = x_i'(t) + \frac{h}{2}x_i''(t) + \mathcal{O}\big(h^2\big) \tag{2}$$

Inserting equation (2) into (1) to obtain

$$x_i(t+h) = x_i(t) + hx_i'(t+\frac{h}{2}) + \mathcal{O}\big(h^3\big) \tag{3}$$

A Taylor first order expansion of the velocity produces the following

$$x_i'(t+\frac{h}{2}) = x_i'(t) + \frac{h}{2}x_i''(t) + \mathcal{O}\big(h^2\big) \tag{4}$$

By using equation (4), then (3) and then (4) again we obtain $x_i(t+h)$ and $v_i(t+h)$. The algorithm follows:

$$\begin{aligned}
v_i(t+\frac{h}{2}) &= x_i'(t) + \frac{h}{2}\frac{F_i(t)}{m} + \mathcal{O}\big(h^2\big) \\
x_i(t+h) &= x_i(t) + hx_i'(t+\frac{h}{2}) + \mathcal{O}\big(h^3\big) \\
v_i(t+h) &= v_i(t+\frac{h}{2}) + \frac{h}{2}\frac{F_i(t+\frac{h}{2})}{m} + \mathcal{O}\big(h^2\big)
\end{aligned}$$

8.2 Lennard-Jones Potential and force between two molecules

We will be using the Lennard-Jones potential to approximate the forces between the molecules, which work quite well, given it's simplicity, for neutral particles, especially noble gases, as we are dealing with in this study [2]. The formula is given below.

$$U(r_{ij}) = 4\epsilon\left[\left(\frac{\sigma}{r_{ij}}\right)^{12} - \left(\frac{\sigma}{r_{ij}}\right)^{6}\right]$$

At short distances the term to the twelfth power dominates and represents a repulsive force, Paulie exclusion principle, while at longer distances the term to sixth power dominates and represents the attractive van der Waal force. The σ is the distance at which the potential

is 0, while ϵ is the depth of the well.

Experimentally the following values for argon has been found: $\begin{cases} \epsilon/k_B & = 119.8\text{K} \\ \sigma & = 3.405\text{Å} \end{cases}$

The force felt between the molecules is given by the negative gradient of the potential.

$$\mathbf{F}(r_{ij}) = -\boldsymbol{\nabla} U(r_{ij})$$

The potential only has a nonzero derivative along $\mathbf{r}_{ij}$, the axis between then particles, so it is natural to evaluate the gradient in that coordinate system before projecting it onto the xyz coordinates used by the program.

$$\mathbf{F}(r_{ij}) = -4\epsilon \hat{\mathbf{r}}_{ij} \partial_{r_{ij}} \left[\left(\frac{\sigma}{r_{ij}} \right)^{12} - \left(\frac{\sigma}{r_{ij}} \right)^{6} \right]$$

$$\mathbf{F}(r_{ij}) = -4\epsilon \hat{\mathbf{r}}_{ij} \left[\left(-\frac{12}{\sigma} \right) \left(\frac{\sigma}{r_{ij}} \right)^{13} - \left(-\frac{6}{\sigma} \right) \left(\frac{\sigma}{r_{ij}} \right)^{7} \right]$$

Then it is projected onto the xyz coordinates, $F_k = \left(F \frac{r_{ij}}{|r_{ij}|} \right)_k = F \frac{k_{ij}}{|r_{ij}|}$, where k_{ij} is the distance between the particles in direction $k = \{x, y, z\}$, and $|r_{ij}|$ is the total distance.

$$F_k = \frac{24}{\sigma} \epsilon \left[2 \left(\frac{\sigma}{r_{ij}} \right)^{13} - \left(\frac{\sigma}{r_{ij}} \right)^{7} \right] \frac{k_{ij}}{|r_{ij}|}$$

8.2.1 Algorithm to implement force

The implementation of the force will be along the following steps:

- Calculate $r_{ij} = |\mathbf{r}_i - \mathbf{r}_j|$ for a particle pair
- Calculate $F_{r_{ij}} = \frac{24}{\sigma} \epsilon \left[2 \left(\frac{\sigma}{r_{ij}} \right)^{13} - \left(\frac{\sigma}{r_{ij}} \right)^{7} \right]$
- Calculate $F_x = F_{r_{ij}} \frac{k_{ij}}{|r_{ij}|}$ for xyz
- Add the force to both particles force account, halves the necessary computations. One positive one negative

A Unit scheme

The program uses a more natural set of units which let's Boltzmann's constant be 1.

$$1 \text{ mass unit} = 1 \text{ a.m.u} = 1.661 \times 10^{-27} \text{ kg} \quad (5)$$

$$1 \text{ length unit} = 1.0 = 1.0 \times 10^{-10} \text{ m} \quad (6)$$

$$1 \text{ energy unit} = 1.651 \times 10^{-21} \text{ J} \quad (7)$$

$$1 \text{ temperature unit} = 119.735 \text{ K} \quad (8)$$

All the other units are then expressed in term of these units.

Boltzmann constant translates between temperature and energy, in SI-units it is $k_B = 1.381 \times 10^{-23}$ J K^{-1}, which in the previously defined units becomes

$$k_B = 1.381 \times 10^{-23} \left(\text{ J K}^{-1} \right) \left(\frac{\text{energy unit}}{1.651 \times 10^{-21} \text{ J}} \right) \left(\frac{119.735 \text{ K}}{\text{temperature unit}} \right) \tag{9}$$

$$k_B = 1 \, \frac{\text{energy unit}}{\text{temperature unit}} \tag{10}$$

References

[1] Gulliks fys3150 github page
`https://github.com/Gullik/molecular-dynamics-fys3150`.

[2] Wikipedia `http://en.wikipedia.org/wiki/Lennard-Jones_potential`.

[3] Morten Hjorth-Jensen. Computational physics lecture notes fall 2014.

[4] Dominik Marx and Jurg Hutter. Ab initio molecular dynamics: Theory and implementation, 2009.