September 13, 2024

Note! Copying .xyz files in Meson didn't work for me, so I had to copy them to build directory by hand.

Velocity-Verlet integrator

Lennard-Jones potential - Pauli repulsion and London dispersion (attraction) (REF)
Velocity-Verlet integration (REF)

$$\dot{\overline{v}}_i(t) = \frac{\overline{f}_i(t)}{m_i}$$
$$\dot{\overline{r}}_i(t) = \overline{v}_i(t)$$

The movement is approximated by assuming constant force and making small time steps. We take Newton's law and definition of velocity, using Taylor formula, we get the approximation equations. (REF Lec 2)

Allen, Michael P., and Dominic J. Tildesley, Computer Simulation of Liquids, 2nd edn (Oxford, 2017; online edn, Oxford Academic, 23 Nov. 2017)

Predictor step:

$$v_i(t + \Delta t/2) = v_i(t) + \frac{1}{2m_i} f_i(t)\Delta t$$
$$r_i(t + \Delta t) = r_i(t) + v_i(t + \Delta t/2)\Delta t$$

Corrector step:

$$v_i(t + \Delta t) = v_i(t + \Delta t/2) + \frac{1}{2m_i} f_i(t + \Delta t)\Delta t$$

Implementation — src/atoms.h, src/verlet.cpp, src/verlet.h, tests/test_verlet.cpp

Lec 2

Milestone 4. Lennard Jones potential with direct summation

How to compute forces from energy. By definition of pair potential:

$$E_{pot} = \sum_{i<j} V(r_{ij})$$

Lennard-Jones potential is sum of Pauli Repulsion (repulsive force) and London Dispersion (attractive force). These interactions act even on uncharged atoms.

(REF Lec 3 Müser, M. H., Sukhomlinov, S. V., and Pastewka, L. (2022). Interatomic potentials: achievements and challenges. Advances in Physics: X, 8(1). https://doi.org/10.1080/23746149.2022.2093129)

$$V_{ij}(r) = 4\epsilon \left(\left(\frac{\sigma}{r}\right)^{12} - \left(\frac{\sigma}{r}\right)^{6} \right)$$

By definition of energy, force is the derivative of energy by position.

$$\overline{f}_k = \begin{pmatrix} -\partial E/\partial x_k \\ -\partial E/\partial y_k \\ -\partial E/\partial z_k \end{pmatrix} = \sum_i \frac{\partial V}{\partial r_{ik}} \hat{r}_{ik} \text{ (unit vector)}$$

Let's compute for one of the dimensions.

$$\frac{\partial V_{ij}}{\partial x_k} = (\text{chain rule by vector length}) \frac{\partial V_{ij}}{\partial r_{ij}} \frac{\partial r_{ij}}{\partial x_k}$$

$$\frac{\partial r_{ij}}{\partial x_k} = \frac{\partial \sqrt{(x_j - x_i)^2 + \dots}}{\partial x_k} = (*)$$

Remember: $(\sqrt{u})' = \frac{1}{2\sqrt{u}} u'$

$$(*) = \frac{1}{2r_{ij}} \frac{(x_j - x_i)^2 + \dots}{\partial x_k} = \frac{1}{r_{ij}} \cdot (x_j - x_i) \cdot \left(\frac{\partial x_j}{\partial x_k} - \frac{\partial x_i}{\partial x_k} \right)$$

Here is a pair of atoms i and j. We want the force for atom k. If $k \neq i$ and $k \neq j$, the expression is 0.

$$(*) = \begin{pmatrix} (x_j - x_i)(\delta_{jk} - \delta_{ik})/r_{ij} \\ (y_j - y_i)(\delta_{ik} - \delta_{jk})/r_{ij} \\ (z_j - z_i)(\delta_{ik} - \delta_{jk})/r_{ij} \end{pmatrix}$$

$$k = j \;\to\; (x_k - x_i)(1 - 0)/r_{ik}, \;\; (*) = \hat{r}_{ik}$$

$$k = i \;\to\; (x_j - x_k)(0 - 1)/r_{kj}, \;\; (*) = \hat{r}_{jk}$$

When we take the derivative of energy, only the pairs, where one of the atoms is k, are left. The derivative is exactly the same for V_{ij} and V_{ji}.

$$\boxed{\overline{f}_k = -\sum_i \frac{\partial V}{\partial r} \hat{r}_{ik}}$$

Last piece in the puzzle is derivative of potential by distance.

$$\frac{\partial V}{\partial r} = 4\epsilon(\sigma^{12}(-12)r^{-13} - \sigma^6 \cdot 6 \cdot r^{-7})$$

First molecular dynamics simulation

In the first simulation, we don't care about physical units, and just set $m = 1, \sigma = 1, \epsilon = 1$. Simulation duration is $100\sqrt{m\sigma^2/\epsilon}$, the atom positions are saved each $1\sqrt{m\sigma^2/\epsilon}$.

Experimentally choosing the time step (multiplied by $\sqrt{m\sigma^2/\epsilon}$): 0.001 — OVITO visualization shows that most of atoms evaporate and fly into infinity; 0.00001 — most atoms stay together, while some still evaporate. We can also detect evaporation if potential energy increases and kinetic energy doesn't change. The total energy was plotted for different time steps in Fig. 1.

Implementation — src/lj_direct_summation.cpp, src/lj_direct_summation.h, tests/test_lj_direct_summation.cpp src/xyz.cpp, src/xyz.h, tests/test_verlet.cpp, milestones/04/main.cpp

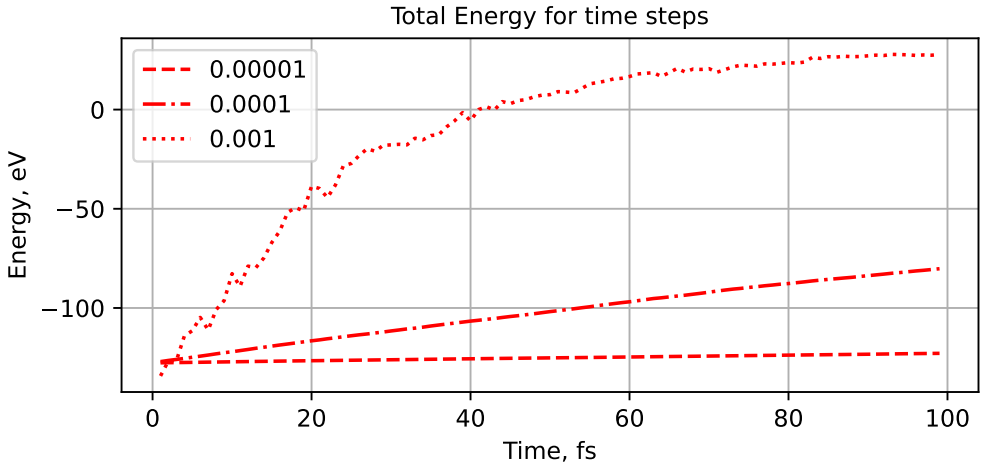

Figure 1: Total energy behaviour is better with very small time step.

Berendsen thermostat

As the molecular dynamics system evolves, potential energy decreases and kinetic energy increases, therefore we connect the system to a heat bath of constant temperature, which stabilizes the simulation. (REF Lec 4)

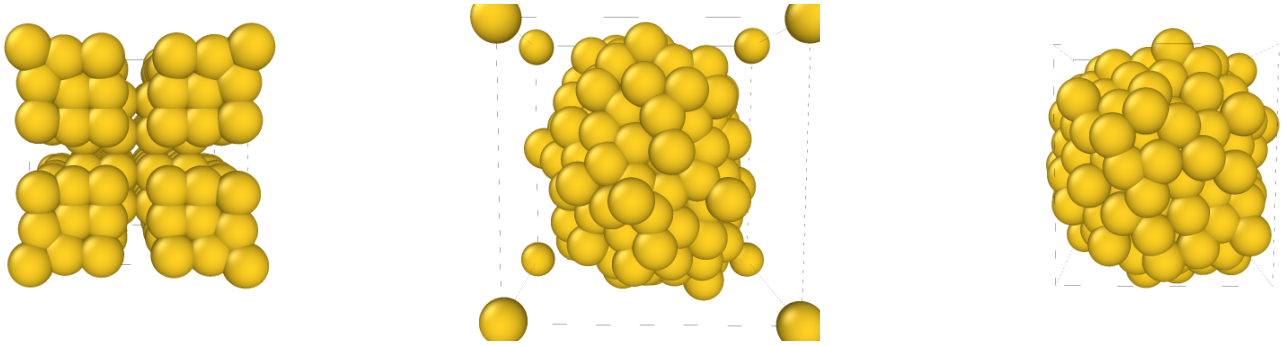

Figure 2: OVITO visualization with Berendsen thermostat. Without thermostat, atoms vaporize even with the smallest time step.

H. J. C. Berendsen, J. P. M. Postma, W. F. van Gunsteren, A. DiNola, and J. R. Haak. Molecular dynamics with coupling to an external bath. J. Chem. Phys., 81(8):3684–3690, 1984. URL https://doi.org/10.1063/1.448118.

To decrease the temperature, we scale the velocities of atoms by a factor:

$$v' = \lambda v,\ \lambda = \sqrt{1 + \left(\frac{T_0}{T} - 1\right)\frac{\Delta t}{\tau}}$$

$$E_k = \frac{1}{2}\sum_i m_i v_i^2 = \frac{3}{2} N K_b T \ \rightarrow\ T = \frac{2}{3}\frac{E_k}{K_b N}$$

T_0 is target temperature, $\Delta t = 0.0001\sqrt{m\sigma^2/\epsilon}$ from the first simulation, τ is relaxation time of the thermostat and should be much larger than the time step.

I create a cube of atoms of width 4 for this experiment. Choosing target temperature: 0.05 — visualization shows that the atoms are pushed apart into infinity in small groups; 0.005 — no vaporization.

Choosing Berendsen relaxation time: for the whole simulation, $1000\Delta t$ is required to keep the atoms from evaporating; initial stronger relaxation doesn't make a difference.

Testing the Berendsen implementation is very simple, because it modifies the speed of each atom separately, so it's enough to test the behaviour on just one atom. I tested that the temperature should exponentially approach the wished value, and in case the relaxation time is same as step time, it should change instantly.

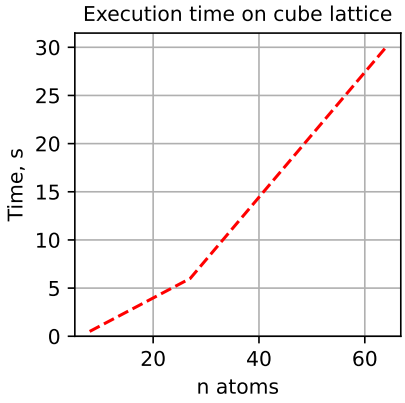

Figure 3: Time scales quadratically with number of atoms.

Implementation — src/lattice.cpp, src/lattice.h, src/thermostat.cpp, src/thermostat.h, tests/test_thermostat.cpp milestones/05/main.cpp

Milestone 6. Neighbor list

Cutoff radius is 3σ to allow speedup.

Implementation — src/neighbors.cpp, src/neighbors.h, tests/test_neighbors.cpp, src/lj.cpp, src/lj.h, tests/test_therm milestones/05/main.cpp

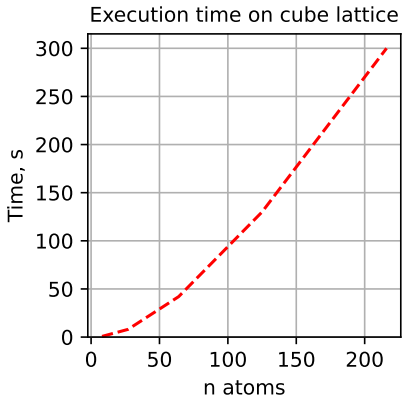

Figure 4: Time scales linearly with number of atoms.