Computational Physics Sommersemester 2025

Exercise Sheet 02
LJ fluid-1: microcanonical ensemble in 2D

Matthias Müller, Elias Jedam May 12, 2025

Datum:
May 12, 2025

Contents

Before talking about the results or concrete implementations, we first have to explain how one has to implement the given values into the code (which seems to be trivial, but we still managed to waste $\sim$ 4 hours before we finally got there...).
We converged all given values to the (dimensionless) LJ units by setting the characteristic energy ϵ, length σ, and particle mass m of the system to unity. From this, the other properties can be derived:

The reduced temperature is defined as $T^* = \frac{k_B T}{\epsilon}$, where k_B is the Boltzmann constant in consistent units (e.g., kcal/mol/K). For example, with $T = 300$ K, $k_B = 0.001985$ kcal/mol/K, and $\epsilon = 0.2977$ kcal/mol, we obtain $T^* \approx \frac{0.001985 \cdot 300}{0.2977} \approx 2.0$.

The reduced timestep is defined using the LJ time unit $\tau = \sigma\sqrt{m/\epsilon}$, where σ is in nm, m is the particle mass in kg, and ϵ is the energy in joules. For $\sigma = 0.188$ nm, $m = 39.95$ g/mol $\approx 6.636 \times 10^{-26}$ kg, and $\epsilon = 0.2977$ kcal/mol $\approx 2.067 \times 10^{-21}$ J, this gives $\tau \approx 1.065$ ps. A physical timestep of $\Delta t = 1$ fs then corresponds to a reduced timestep of $\Delta t^* = \Delta t/\tau \approx 0.00094$.

All observables in the upcoming discussions will be given in these LJ units, which can easily be remapped to "real" units in the way we just described.

1 Part a

To determine the lattice constant a_{lat} we calculated:

$$a_{lat} = \frac{1}{\sqrt{\rho}} \tag{1}$$

with

$$\rho = \frac{N}{L^2} = \frac{n^2}{L^2} \tag{2}$$

You can find the according code snippet in the file part_a.py.

To find the box lenght L we calculated:

$$L = 2\sigma n \tag{3}$$

A snapshot can be seen in fig. 1, where the diameter of each particle is set to $1\,\sigma$.

2 Part b

In order to assign the velocities we used the numpy.random.uniform(-1,1) function. After that we substracted the mean value from each velocity. You can find the implementation also in part_a.py.

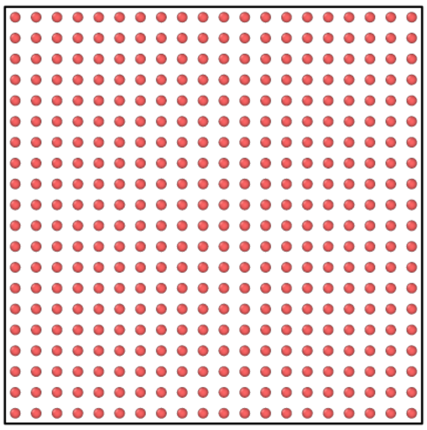
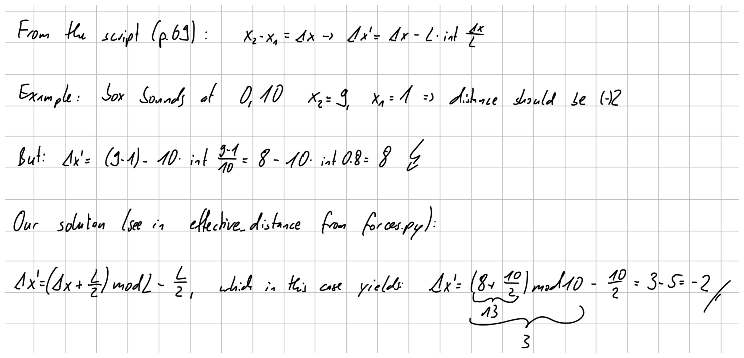


Figure 1: Snapshot of the initial setup



From the script (p.69): $x_2 - x_1 = \Delta x \rightarrow \Delta x' = \Delta x - L \cdot \text{int} \frac{\Delta x}{L}$

Example: box bounds at $0, 10$ $x_2 = 9$, $x_1 = 1 \Rightarrow$ distance should be $(-)2$

But: $\Delta x' = (9-1) - 10 \cdot \text{int} \frac{9-1}{10} = 8 - 10 \cdot \text{int}\, 0.8 = 8$ ↯

Our solution (see in effective_distance from forces.py):

$\Delta x' = (\Delta x + \frac{L}{2}) \bmod L - \frac{L}{2}$, which in this case yields: $\Delta x' = \underbrace{\underbrace{(8 + \frac{10}{2}}_{13}) \bmod 10}_{3} - \frac{10}{2} = 3 - 5 = -2$ //

Figure 2: Example that our distance measurement yields correct solutions while the one from the script doesn't work.

3 Part c

The PBC's were implemented in the wanted way by "reboxing" the particles after every step in the simulation. This can be found in the function *periodic_boundary_function* in the file **boundary_conditions.py**.
However, this is not sufficient as the PBC's also have to be implemented in the distance calculation for the forces as can be seen in the function *effective_distance* from **forces.py**. We first wanted to use the formula from the script, but we then changed it as we believe it is faulty as can be seen in fig. 2.

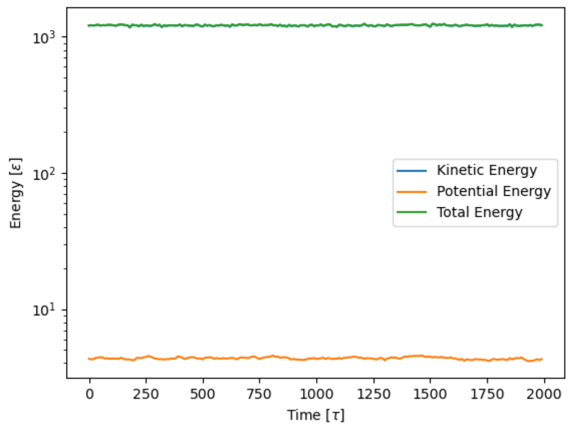
4 Part d

The force calculation for the LJ-cut potential can be found in *lj_cut_force* from **forces.py**, the implementation of the thermostat in *velocity_rescaling_thermostat* from **thermostats.py**.
The output-generating function is *export_data* from **data_io.py**. Here, we

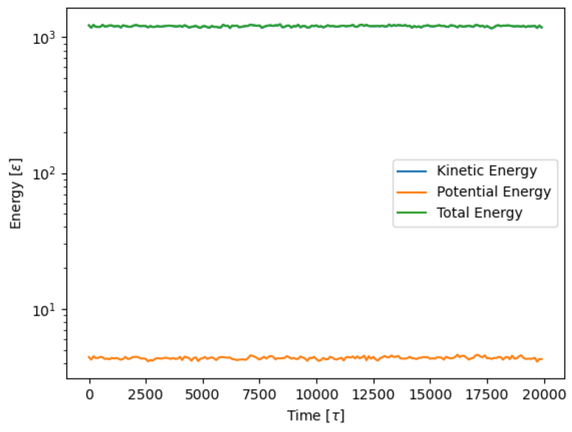
decided to keep the output in the classic ovito format, meaning that only properties of individual particles are exported, which can be selected via the parameter selected_properties. The energy calculation happens afterwards in post-processing (*compute_ke*, *compute_pe* and *compute_e* from **tools.py**).
If needed, this might easily be adapted in such a way that the energy calculation happens during the simulation and is written in the original data, however one has to keep in mind that this will increase the necessary computational power during the simulation process.

5 Part e

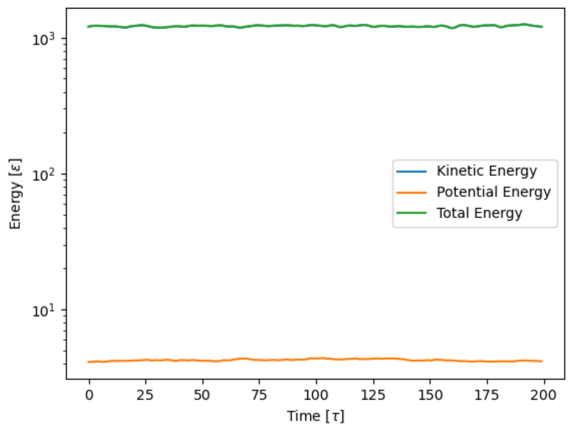
Now to the discussion of our results: first of all, we take a look at the data from the simulation with $r_{\text{cut}} = 2.5\,\sigma$ and $\Delta t = 1$, $1 \cdot 10^{-1}$ and $1 \cdot 10^{1}$ fs after having equilibrated the system in the requested manner.
When plotting the time evolution of $\hat{U}$, $\hat{K}$ and $\hat{E}$ as seen in fig. 3, one might at first suspect that the energy fluctuates more for larger Δt. However, a closer look shows that this is just due to the larger total time which was covered during the 20000 steps.



(a) $\Delta t = 1\,\mathrm{fs}$



(b) $\Delta t = 0.1\,\mathrm{fs}$



(c) $\Delta t = 10\,\mathrm{fs}$

Figure 3: Time evolution of the energy with a logarithmic $y-$axis.

This observation is supported when looking at the mean, standard deviation, and variance as displayed in the tables 1 to 3: all corresponding properties are in the same order of magnitude for all different time steps Δt, so it really seems like even a time step of $\Delta t = 10\,\text{fs}$ isn't too coarse for this system, thus it should be used to gain information on larger time scales in fewer time; one might even try to increase Δt even slightly more, but this would have to be researched further.

	Mean	STD	VAR
U	4.344	0.084	0.007
K	1209	13	177
E	1214	13	178

Table 1: Mean, STD and VAR of the energies for $\Delta t = 1\,\text{fs}$.

	Mean	STD	VAR
U	4.232	0.069	0.005
K	1214	14	208
E	1218	14	208

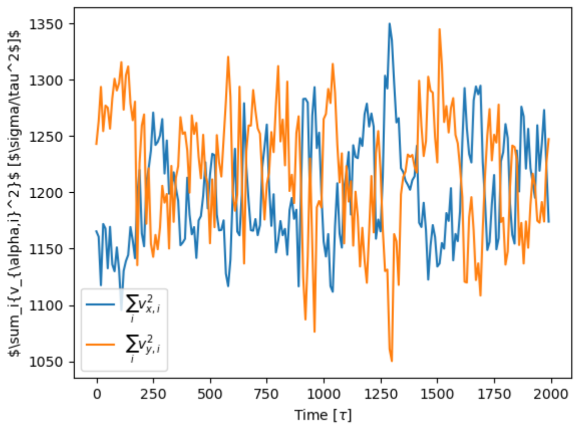
Table 2: Mean, STD and VAR of the energies for $\Delta t = 0.1\,\text{fs}$.

	Mean	STD	VAR
U	4.341	0.095	0.009
K	1205	16	243
E	1209	16	244

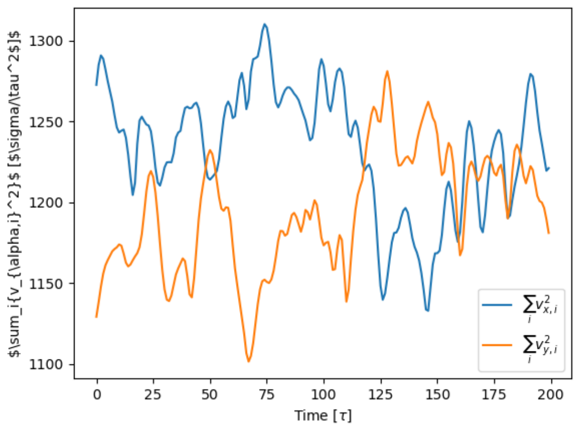
Table 3: Mean, STD and VAR of the energies for $\Delta t = 10\,\text{fs}$.

This overall impression is confirmed when looking at the velocity: while at first glance, the system looks the most stable at $\Delta t = 0.1\,\text{fs}$, this is just due to the covered time scale.

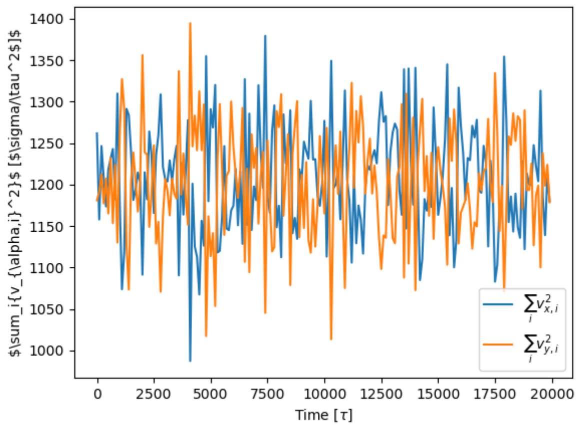
Overall, we see that all choices of Δt leed to stable results describing "real physics", i.e. the energy is conserved and the momenta cancel out.



(a) $\Delta t = 1\,\mathrm{fs}$



(b) $\Delta t = 0.1\,\mathrm{fs}$



(c) $\Delta t = 10\,\mathrm{fs}$

Figure 4: Time evolution of the velocities.

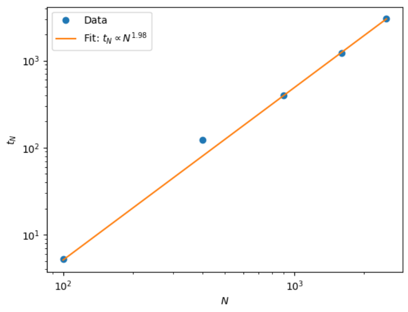


Figure 5: Fit for estimating the performance in a double–logarithmic plot to visualize the power law.

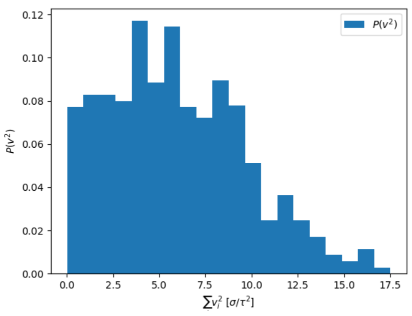


Figure 6: Velocity distribution.

6 Part f

Unfortunately, `njit` yielded errors that we couldn't resolve; we therefore just used `prange`.

With this, we obtained the parameters $\alpha \approx 1.98$ and $c \approx 5.58 \cdot 10^4$ for the fit $t_N = c \cdot N^\alpha$ as can be seen in fig. 5. This was surprising to us, as we would have suspected a value of $\alpha \ll 2$ as our loop for the force calculation only sums over the upper triangle of the matrix as discussed in the last tutorial. But probably the tested n where still too small so that the logarithm didn't have a strong influence.

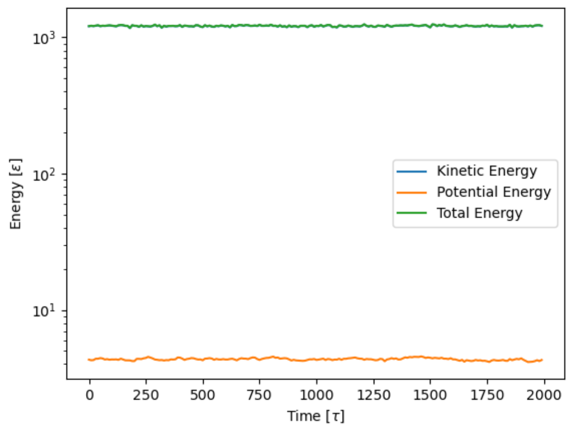
When computing how many years $n = 1000$ particles per side would take, we ended up at a value of 13653 years...

7 Part g

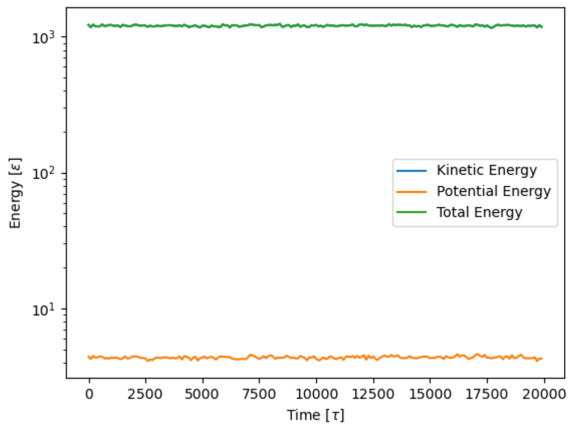
We then plotted the velocity distribution $P(v^2)$ for the data from $n = 20$, which clearly had the shape of a Maxwell–Boltzmann distribution as can be seen in fig. 6

8 Part h

When varying the cutoff as displayed in fig. 7, we can see that the energy of the system does not change compared to fig. 3(a), where r_{cut} was $2.5\,\sigma$; this means that the even the small cutoff of $2.5\,\sigma$ doesn't seem to alter the physics or the stability of the system, thus it is a valid assumption to save computational power.



(a) $r_{\text{cut}} = 4.0\,\sigma$

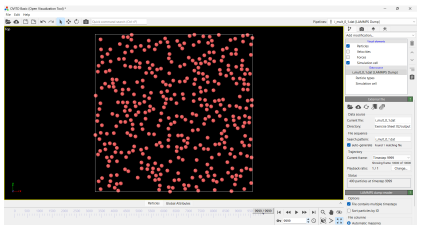


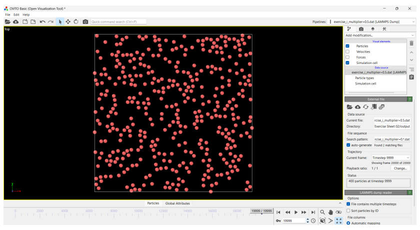
(b) $r_{\text{cut}} = 3.25\,\sigma$

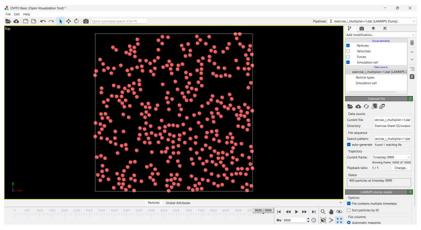
Figure 7: Time evolution of the energy with a logarithmic $y-$axis.

9 Part i

According to the LJ potential a higher ϵ results in a stronger potential. This means the particles should be more attractive.


(a) $\epsilon = 0.1 k_B T_0$


(b) $\epsilon = 0.5 k_B T_0$


(c) $\epsilon = 1.5 k_B T_0$

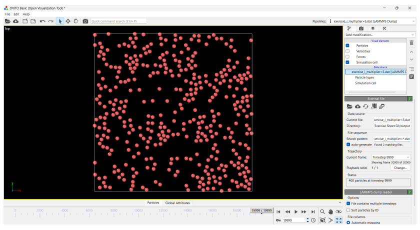

(d) $\epsilon = 5 k_B T_0$

Figure 8: Different patterns for varying ϵ.

As you can see in the pictures this results in a more structured pattern. The particles form bigger groups the higher the epsilon. We took the last picture from each simulation to compare.