Computer Simulations - Report of project 2.

Complete velocity Verlet algorithm for Molecular Dynamics simulation in an enclosed box

Pál Balázs
Eötvös Loránd Tudományegyetem

2019. december 11.

Abstract

For the second project of the Computer Simulations (ELTE Physics MSc) course I created a molecular dynamics simulation, using the velocity Verlet integration. I also implemented Verlet's pair-list method, along with his Lennard-Jones potential cut-off method. Sadly because of unknown reasons, I could only managed to run the simulation with periodic boundary conditions, instead of both periodic and closed conditions. I also considered the measurement of some thermodynamic quantities, which I did successfully, and compared them with literary data as well. Problems and limits of the simulation are also discussed.

I. INTRODUCTION

In the short description I've already discussed all the details regarding the theoretical background and implemented features. In this document I'll only focus on the discussion of those features, which were actually programmed and tested and won't copy and repeat the whole theoretical and technical details from the short description.

II. DESCRIPTION OF THE SIMULATION

II.1. FEATURES AND FUNCTIONALITY

For the second project I implemented a working molecular dynamics simulation, which intends to simulate the motion of a fairly small number ($N < 1000$) of interacting particles inside a box with periodical boundary conditions at a given T temperature, using the velocity Verlet method. The expected behaviour of simulation was to correctly illustrate the Brownian motion of some interacting particles and the temperature oscillation of its gas around a given temperature.

The interaction between the particles are defined with the truncated and shifted Lennard-Jones potential

$$V_c^{LJ}(r) = \begin{cases} 4V_0\left[\left(\frac{\sigma}{r}\right)^{12} - \left(\frac{\sigma}{r}\right)^{6}\right] & \text{if } r < r_c \\ 0 & \text{if } r \geq r_c \end{cases} \quad (1)$$

Where the `r_c` cut-off distance was chosen to be

$$\texttt{r\_c} = 2.5\sigma \quad (2)$$

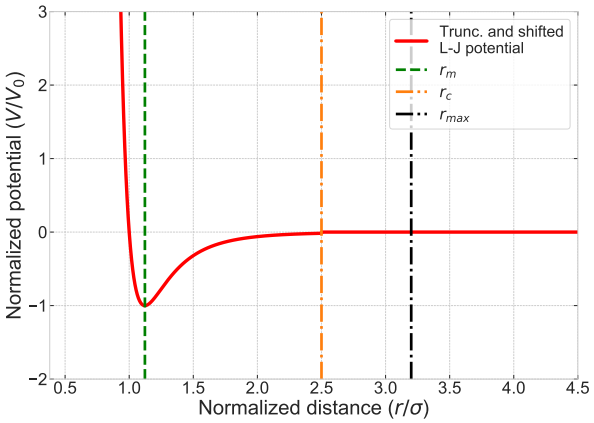


Figure 1: The truncated and shifted Lennard-Jones potential

I also implemented Verlet's pair-list method. This means, that I stored the indeces of every particle-pairs, which was situated closer to each other, than a chosen `r_max` distance. When the simulation calculates the interaction forces between particles, it only loops over this list, instead of calculating forces between all particles in the simulation. I've chosen a widely used value of this distance to be

$$\texttt{r\_max} = 3.2\sigma \tag{3}$$

Since the particles moves at a finite speed, it needs some steps, before a non-list element wanders into the interaction sphere of $\texttt{r\_c} = 2.5\sigma$. This means we only need to update the pair-list only after some `update_interval` number of iterations. I've chosen this number to be

$$\texttt{update\_interval} = 10 \tag{4}$$

To regularize the system, I've rescaled the velocities at every 200th step, to ensure the simulation never blows up.

Besides these mentioned values, I've used the following constants for my simulation to create the graphs in this report:

Quantity	Value
Particle number (N)	64
Temperature (T)	50 K
Step size (dt)	0.001
Particle density (ρ)	1.2

The particle density ρ was used to distribute the particles evenly in a box (discussed in Sec. V.)

II.2. USED MEASURE

To choose correct initial velocities, I've used two type of methods. One, the `gasdev()` function from the *Numerical Recipes* book (Press et al., 2007) which I've already mentioned in the short description and which I've used last semester, when tried to replicate a molecular dynamics simulation in `C++` and two, a built-in function in Python's NumPy library, named `numpy.random.normal()`. Both of them generates normally distributed random numbers, with $\sigma = 1$ value.

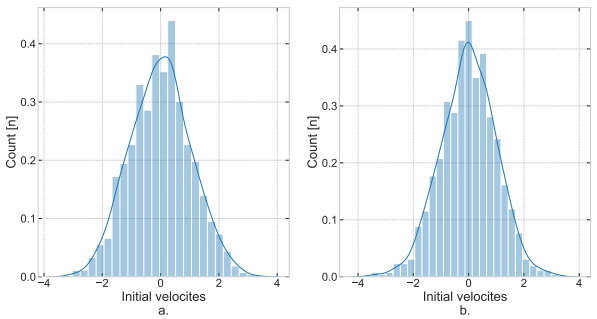


Figure 2: Comparing the Box-Müller algorithm with NumPy's built-in `numpy.random.normal()` function.
a. Box-Müller algorithm
b. NumPy's `numpy.random.normal()`

Running the simulation with either models, did not seem to make any difference, since both of their outputs are corrected with the already discussed transformation in the short description.

Physically, the velocity distribution of particles in a gas could be described by the discussed Gauss distribution:

$$P(v) = \left(\frac{m}{2\pi k_B T}\right)^{3/2} \cdot e^{-\frac{m\left(v_x^2 + v_y^2 + v_z^2\right)}{2k_B T}} \tag{5}$$

However, the two mentioned methods generates such velocity values, which correspond to the measure

$$k_B = N_A = \sigma = m = T = 1. \tag{6}$$

To make the calculations consistent, I've used this measure throughout everywhere in the simulation, except the $T = 1$ condition. Where the velocities are rescaled by a λ constant

$$\lambda = \sqrt{\frac{2\left(N-1\right)k_B T}{\sum_{i=1}^{N} m\boldsymbol{v}_i^2}} \tag{7}$$

I've used the following form for this constant:

$$\lambda = \sqrt{\frac{2\left(N-1\right)T}{\sum_{i=1}^{N} \boldsymbol{v}_i^2}} \tag{8}$$

where T is the temperature and could be freely given at the start of the simulation, despite the fact, that either at the `gasdev()` or `numpy.random.normal()` step we ignore its value. However this does not affect the simulation in such way, which we should deeper investigate in this course.

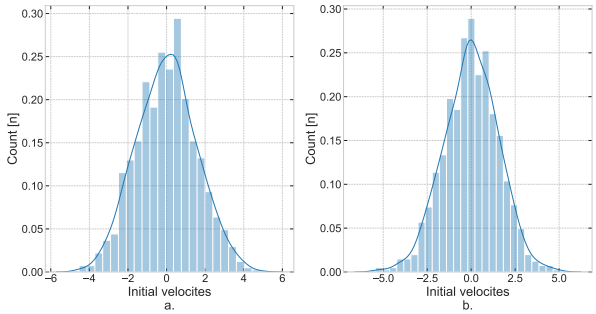


Figure 3: Comparing the distribution of the transformed and rescaled velocities for $T = 1$, initially generated by the Box-Müller algorithm and NumPy's built-in `numpy.random.normal()` function.
a. Box-Müller algorithm
b. NumPy's `numpy.random.normal()`

III. DIRECT RESULTS

The model was proved to be considerably robust in a way, that it always gave physically meaningful results. During runtime in every step I've measured the values of numerous quantities. I plotted the changing of the values which could be seen from Fig. (4) - Fig. (8). These measured values were the following:

1. Kinetic energy:

$$K = \frac{1}{2}\sum_i m_i \cdot \boldsymbol{v}_i^2 = \frac{1}{2}\sum_i \boldsymbol{v}_i^2 \tag{9}$$

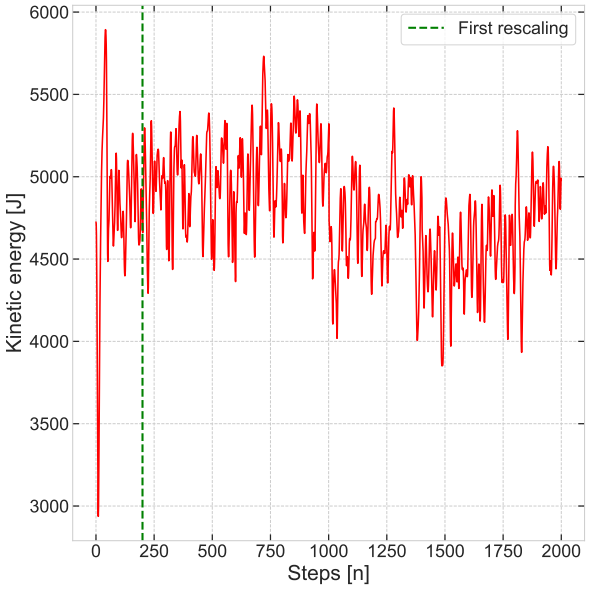


Figure 4: Kinetic energy change of the system with 64 particles for 2000 steps with dt= 0.001 step size.

2. Potential energy:

$$V = \sum_i 4 \cdot V_0 \cdot \left[\left(\frac{\sigma}{|\boldsymbol{r}_i - \boldsymbol{r}_j|} \right)^{12} - \left(\frac{\sigma}{|\boldsymbol{r}_i - \boldsymbol{r}_j|} \right)^{6} \right] \tag{10}$$

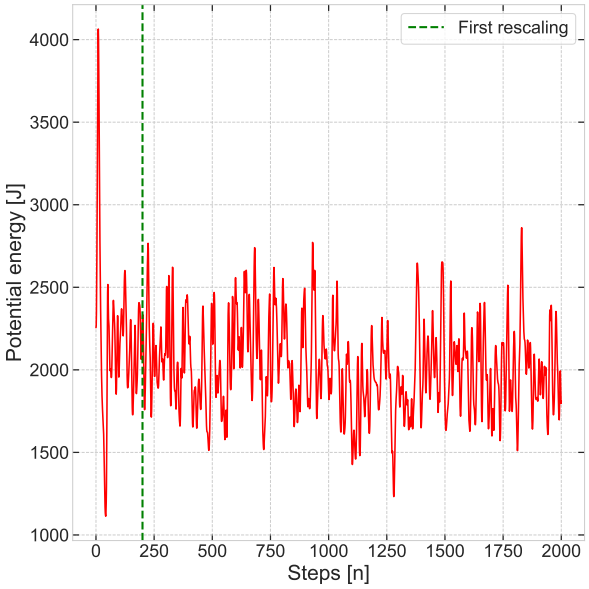


Figure 5: Potential energy change of the system with 64 particles for 2000 steps with dt= 0.001 step size.

3. Total energy, which is simply

$$H = K + V \tag{11}$$

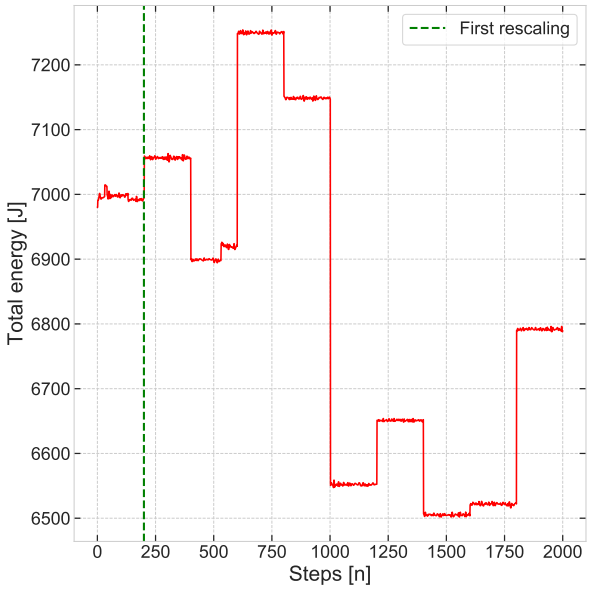


Figure 6: Total energy (K + V) of the system with 64 particles for 2000 steps with dt= 0.001 step size.

This quantity should be always constant. It could be well seen on Fig. (6), that energy remains totally stable and constant during the run, and only changes its value instantaneously, when the velocities are rescaled after every 200th step to regularize the system.

4. Energy dimensional quantity in Virial theorem, needed for the calculation of the system's pressure:

$$\sum_{i<j} \boldsymbol{r}_i \boldsymbol{F}_{ij} \tag{12}$$

Where the values of F_{ij} was already discussed in the short description:

$$\boldsymbol{F}(r) = -\boldsymbol{\nabla} V^{LJ}(r) = \frac{24V_0}{r^2}\left[2\left(\frac{\sigma}{r}\right)^{12} - \left(\frac{\sigma}{r}\right)^{6}\right]\boldsymbol{r} \tag{13}$$

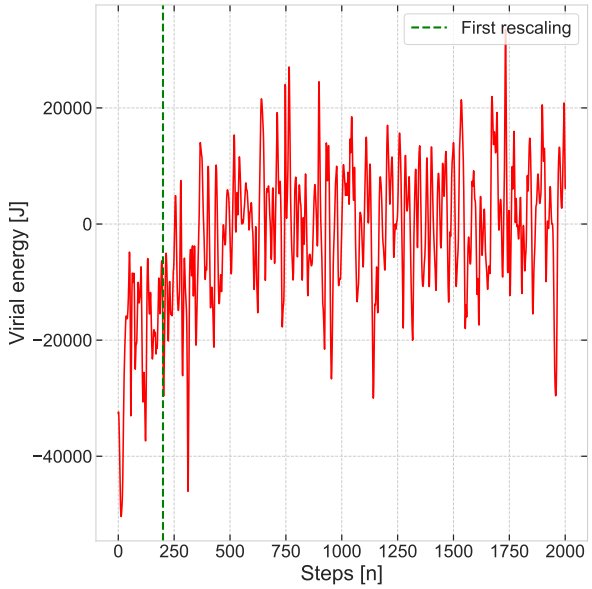


Figure 7: "Virial energy" change of the system with 64 particles for 2000 steps with dt= 0.001 step size.

5. And lastly, the instantaneous temperature of the system:

$$T = \frac{\sum_i m_i \cdot \boldsymbol{v}_i^2}{3 \cdot (N-1)} = \frac{\sum_i \boldsymbol{v}_i^2}{3 \cdot (N-1)} \qquad (14)$$

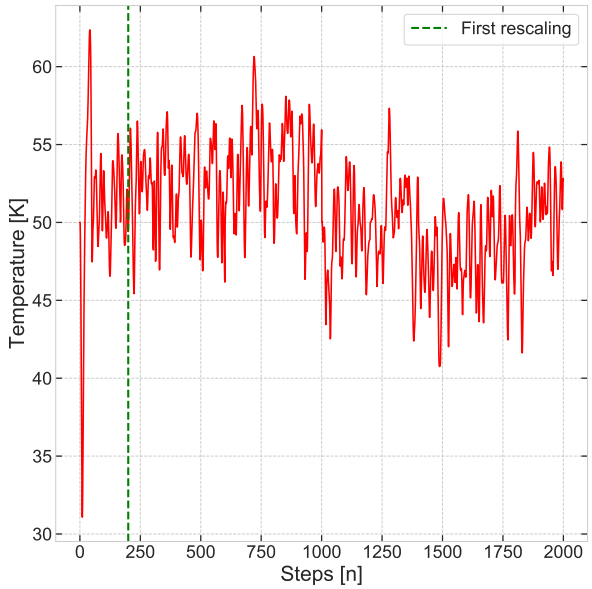


Figure 8: Temperature change of the system with 64 particles for 2000 steps with dt= 0.001 step size.

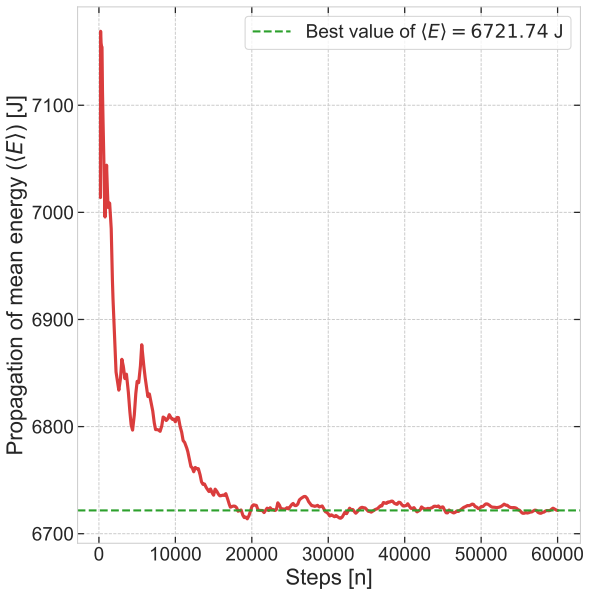
I did also tracked the coordinate-, velocity- and acceleration changes of the individual particles, and made a nice visualization, which should be helpful to understand the behaviour of the system. For 2 000 and 20 000 steps I plotted the Brownian oscillation and motion of the system's particles, which could be seen on Fig. (15) and Fig. (16). For a more pleasurable visual experience, some animations could be seen on my YouTube channel[1] inside my "Molecular Dynamics" playlist. Unfortunately these animations were made by me last semester, and thus they don't exactly represent my current results, since I've changed the used code and method a lot since then. However I still wanted to mention them, because they could be still very helpful to understand, what is really happening in this simulation.

IV. MEASURED THERMODYNAMIC QUANTITIES

Using the direct results and data from the simulation I could approximate the values of some thermodynamic quantities also. This goal was attempted by me last semester, but all I've got was totally bad, and meaningless results. Fortunately this semester I've succeeded to calculate some of these quantities and compare them to literary data as well and it was a pretty good match. Namely, these quantities were the molar heat capacity, the pressure and the compressibility factor of the system. To get more robust and accurate results, I've kept the simulation running for 60 000 steps to let values which are approximated by their average over time to converge. The first quantity which I've measured was the variance of the total energy:

$$\left[\langle E^2 \rangle - \langle E \rangle^2\right]$$

To make this quantity converge, I've monitored the changes of the $\langle E \rangle$ and $\langle E^2 \rangle$ expected values. When I've seen, they've converged without a doubt, I stopped the simulation. It was at $\approx 60\,000$ steps.



[1]Link to my YouTube channel: https://www.youtube.com/channel/UCBDSB7PdQ3E919WSBsTy7cQ

Figure 9: Expected value of total energy of the system with 64 particles for 60000 steps with dt= 0.001 step size.

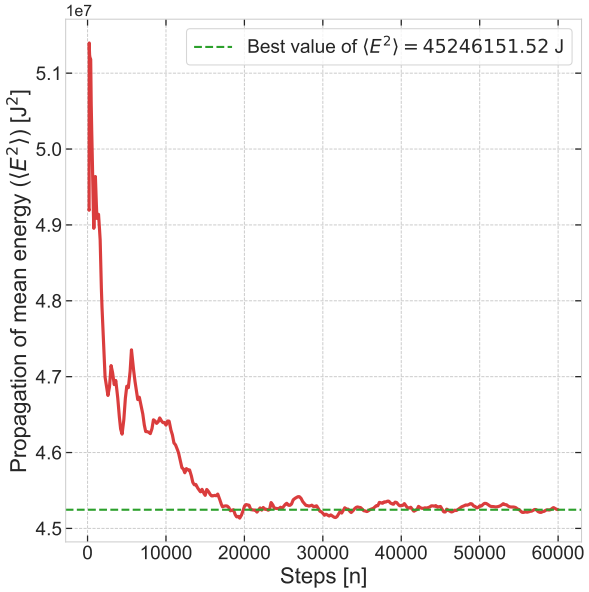


Figure 10: Expected value of squared total energy of the system with 64 particles for 60000 steps with dt= 0.001 step size.

Using these quantities I've successfully measured an approximate value for the variance of the total energy.

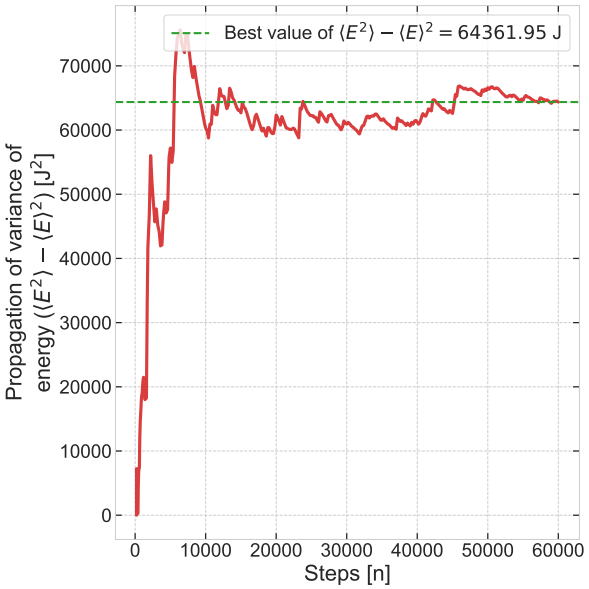


Figure 11: Change in the variance of energy of the system with 64 particles for 60000 steps with dt= 0.001 step size.

First, the molar heat capacity was calculated, which definition as discussed in the short description:

$$C_V = \left.\frac{\partial E}{\partial T}\right|_V = \frac{1}{k_B T^2}\left[\langle E^2\rangle - \langle E\rangle^2\right] \qquad (15)$$

Which in our measure is the following:

$$C_V = \frac{1}{T^2}\left[\langle E^2\rangle - \langle E\rangle^2\right] \qquad (16)$$

Using the instantaneous temperature values as T and the variance values seen on Fig. (11), a propagating value for C_V could be plotted, which could be seen on Fig. (12).

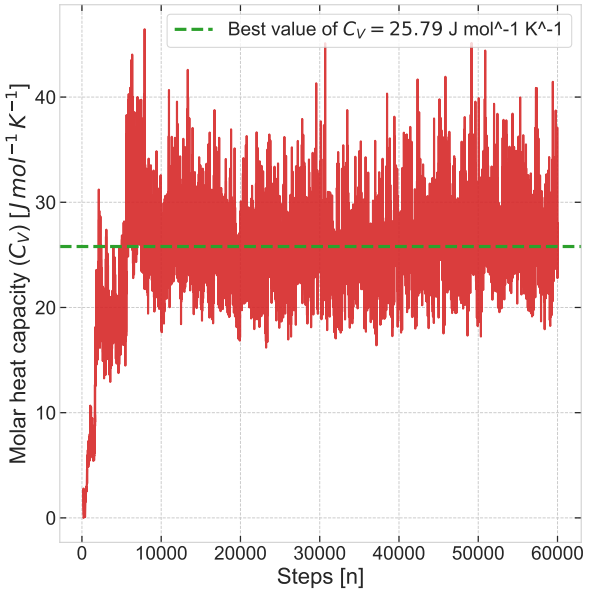


Figure 12: Molar heat capacity on constant volume of the system with 64 particles for 60000 steps with dt= 0.001 step size.

When using the best value for the variance of energy and the mean of the instantaneous temperatures as a value for T, then an exact/mean value could be given, which my best result for the C_V:

$$C_V \approx 25.79\ \frac{\mathrm{J}}{\mathrm{mol\ K}} \qquad (17)$$

In literary data, this value for simple gases is in the $10^1 - 10^2$ range, so this value fits very well into this interval.

Next the pressure of the system could be evaluated, which definition is the following:

$$p = \frac{Nk_BT + \frac{1}{3}\left\langle \sum_{i<j} \boldsymbol{r}_i \boldsymbol{F}_{ij}\right\rangle}{V} \qquad (18)$$

Which in our measure becomes simply

$$p = \frac{NT + \frac{1}{3}\left\langle \sum_{i<j} \boldsymbol{r}_i \boldsymbol{F}_{ij}\right\rangle}{V} \qquad (19)$$

The instantaneous pressure of the system could be also given, using the instantaneous temperature values, along with the instantaneous mean Virial energy. This quantity could be seen on the Fig. (13).

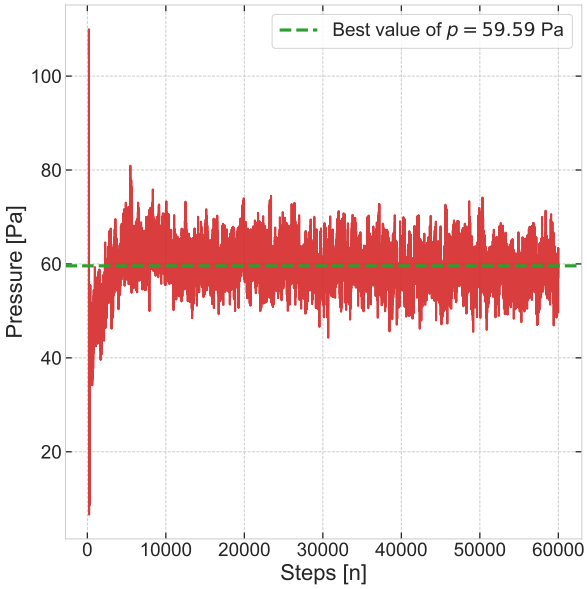


Figure 13: Pressure of the system with 64 particles for 60000 steps with dt= 0.001 step size.

Also an exact/mean value could be given for this quantity, using the same method seen at the case of the C_V:

$$p \approx 59.59 \text{ Pa} \tag{20}$$

According to the ideal gas law, for this particular system (T, V, N) this value should be $59.941Pa$, which is very close to my result.

Finally, the compressibility factor Z could be given, where Z by definition in our measure is

$$Z = \frac{pV}{Nk_BT} = \frac{pV}{NT} \tag{21}$$

I observed, that its value propagates toward the ideal gases' ($Z \to 1$) as the simulation runs longer and longer. This tendency could be seen on Fig. (14).

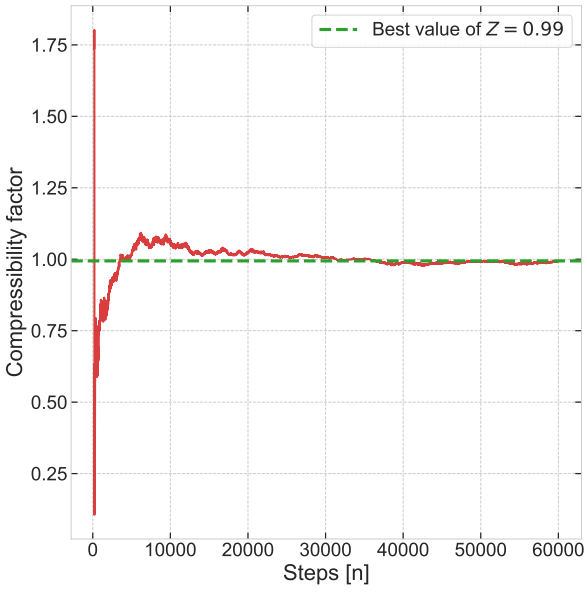


Figure 14: The changing of the compressibility factor of the system with 64 particles for 60000 steps with dt= 0.001 step size.

V. PROBLEMS, LIMITS AND THEIR SOLUTIONS

Simulations, using the Lennard-Jones potential tends to be highly unstable, when particles somehow (mostly at the start of the simulation) manage to get too close to each other. It is clear from the Lennard-Jones potential's definition that in this case the affected particles are getting inexplicably large velocities, which immediately breaks the simulation. To avoid this problem, initially I've placed the particles evenly distributed into a lattice inside the simulated volume. This made the simulation totally robust for the setting of its initial conditions.

The other problem I've encountered with was merely technical in its nature. To optimize runtime, I've used Python's `numba` library, to compile my functions into machine code. Without this, the simulation is pretty slow, but implementing this, it raises serious obstacles. First, `numba` treats global variables as runtime-constants and forbids the overwriting of them. Since I've treat the pair-list as a global variable it made it impossible to update it after every 10 steps. The simulation with `numba` only works well, if the close pairs never changes at all. Thus in this for the simulation could handle large number of particles, but only with `numba` deactivated. It also makes it impossible to run the simulation with bounded boundary conditions, since the pair-list could, and will be changing for much smaller number of particles too during the running.

VI. DISCUSSION

Finally, I've managed to make a working molecular dynamics simulation, which raised great obstacles in my path last semester. I measured some thermodynamic quantities, which correlated with literary data as well and thus – except some smaller problems – I've achieved my goals declared in the short description.

[1] William H Press et al. *Numerical recipes 3rd edition: The art of scientific computing*. Cambridge university press, 2007.

APPENDIX A. - FIGURES

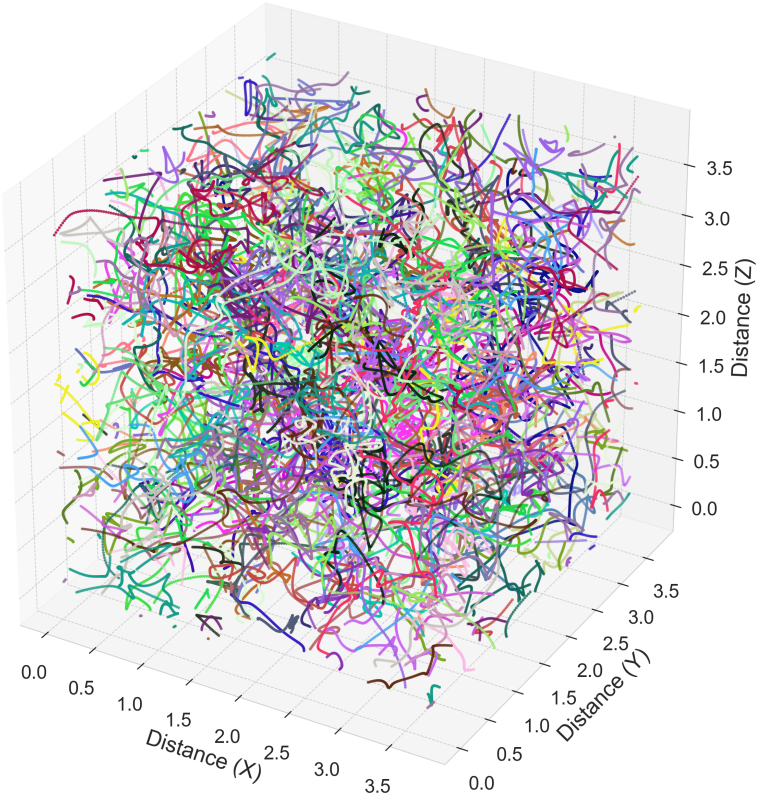


Figure 15: The trajectories of 64 particles for 2000 steps of the simulation with dt= 0.01 step size.

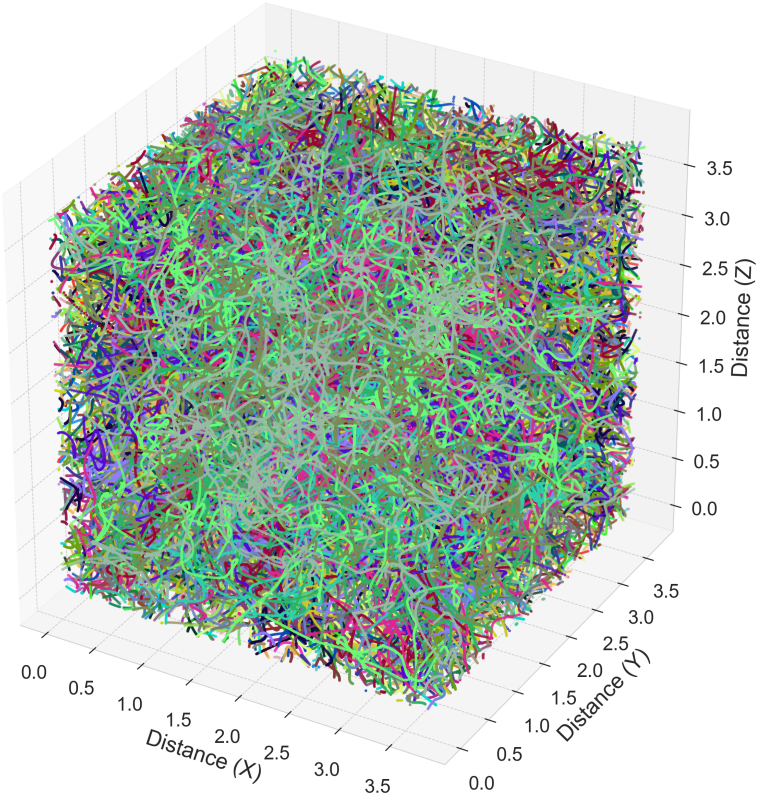


Figure 16: The trajectories of 64 particles for 20000 steps of the simulation with dt= 0.01 step size.