Report to assignment 1
International Course for Computational Physics

Simulation of Argon particles in a box

Emma C. Gerritse, Sophie Hermans and Jason Emming
March 2015

Abstract

This report describes the results of the simulation of Argon particles under the influence of a Lennard-Jones potential, using Python. Up to a 1000 particles have been simulated in a box with periodic boundary conditions. From the positions, momenta and forces of all of the particles the total, potential and kinetic energies, the temperature, pressure and correlation function for the system and their corresponding errors have been calculated.

Signature: Date:

Lecturers: Dr. Jos Thijssen and Dr. Phil Duxbury **Year:** 2014-2015

Contents

1 Introduction

How particles in a system move and interact is what molecular dynamics strives to answer. Developing a computer simulation to accurately predict the behavior of matter has applications that are wide and far reaching. From answering how galaxies evolve to precisely modeling the folding of amino acids into complicated proteins, each system must involve many particles. The motion of these particles is characterized by their potential, which governs the force between them. It was the mission of this group to simulate the consequences of the potential between individual atoms of Argon gas.

2 Theory

In order to simulate a number of particles in a box, one first has to define a particle. The Argon atoms, in this case, have been modeled as point particles that have three properties: a position, a momentum and a potential.

2.1 Potential and force

The Argon particles exert a certain force on one another. They are net neutral particles, so it is a Lennard-Jones force that must be applied. This force can be derived from the Lennard-Jones potential given in equation 1.

$$V_{LJ} = 4\epsilon \left(\left(\frac{\sigma}{r} \right)^{12} - \left(\frac{\sigma}{r} \right)^{6} \right) \tag{1}$$

In this equation, the potential has both a long range attractive component as well as a short range repulsive one. r represents the distance between the two particles in question. The ϵ characterizes the strength of these interactions, while the σ defines the radius of an Argon particle and sets the range. The derivative of this potential then gives us the Lennard-Jones force, as shown in equation 2.

$$\hat{F}_{LJ} = 24\epsilon \left(\frac{\sigma^6}{r_{ij}^7} - 2\frac{\sigma^{12}}{r_{ij}^{13}} \right) \hat{r}_{ij} \tag{2}$$

Integrating the force for each particle and solving for its motion, we were able to simulate the movement of an N-body system of particles in a box.

2.2 Extracting physical parameters

Using our molecular dynamics model we observed the behavior of Argon particles as a function of their temperature. To determine the temperature we use the relationship between the velocities of the particles and their kinetic energy. This relation is given by the Maxwell-Boltzmann distribution in equation 3.

$$\frac{3}{2}(N-1)k_B T = \frac{1}{2}\sum_i m_i |\hat{v_i}^2| \tag{3}$$

We also determined the density of particles from a particular point as a function of distance, known as a pair correlation function. To do this, we counted the number of

particles that fell within a shell of radius r and shell width of δr. With is, we plotted a histogram of the particle locations, see figures 8, 9, 10.

Finally, we calculated the pressure of the particles in this volume, which follows from the virial relation given in equation 4.

$$pV = Nk_BT + \frac{1}{3}\sum_{i<j} \hat{r_{ij}} \cdot \hat{F_{ij}} \tag{4}$$

3 Computational theory

To simulate the problem of Argon particles in a 3D box, we have made a code in Python. Next to the theory of the needed physics, we also need some theory about the computational part, such as boundary conditions, see section 3.1, and initial conditions, section 3.2.

3.1 Boundary conditions

In our molecular dynamics code we implemented periodic boundary conditions. By doing so, we approximated a seemingly infinite system, while only simulating a small unit of the whole. As particles moved outside the boundary they re-emerged on the other side of the box, while still maintaining their momentum. The positions of the Argon atoms were checked every iteration of the program. If they had wandered beyond the length of the box, L, the modulo of their position, $\vec{x}$, was taken. This moved them to an appropriate position within the boundary conditions i.e. $\vec{x}_{new} = \vec{x} \bmod L$

The boundary conditions were also taken into account for the potential calculations. We had to determine the shortest distance from one particle to another in order to accurately gauge its magnitude. This must allow for distances to be measured through the edges of the box as well. This prevented any particle from suddenly feeling a strong increase in force, which had not been present before, when it moved across the boundary.

3.2 Initial Conditions

When the molecular dynamics simulation is running, the old positions, momenta and forces are used for every subsequent iteration, as described in section 3.4. For the first calculation however, some initial values are needed. If these initial conditions are chosen randomly, the programme may never converge to a physical solution. Therefore, we have to set our initial conditions carefully.

3.2.1 Initial Positions

As an initial condition for the positions of the particles we used a face centered crystal (FCC) lattice, see figures 1 and 2. One unit cell contains 4 particles located at $(0,0,0), (\frac{1}{2},\frac{1}{2},0), (\frac{1}{2},0,\frac{1}{2})$ and $(0,\frac{1}{2},\frac{1}{2})$.

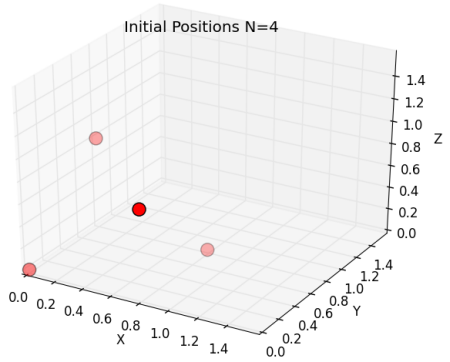

Figure 1: A FCC lattice as initial positions for 4 particles.

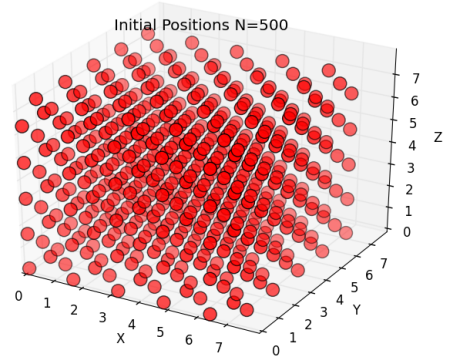

Figure 2: A FCC lattice as initial positions for 500 particles.

3.2.2 Initial Momenta

The initial momenta have been generated using a Gaussian distribution in the x-, y-, and z-direction, with $\mu = 0$ and $\sigma^2 = 2T$. This ensures a realistic physical distribution of the velocities. The zero mean gives a total momentum of exactly zero, if an infinite number of particles were to be simulated. Since we do not have an infinite number of particles, the average μ will never be exactly zero. Therefore we have to correct the momenta of all the particles in three dimensions by subtracting $\frac{1}{N}\sum_{i=1}^{N} p_i$, such that the average is equal to zero exactly.

3.3 Cut-off radius

When two particles are far apart, their force on each other is very small, therefore their contribution to the total force can be neglected. In our code we have defined a cut-off radius. When the distance between two particles is larger than this radius, we do not calculate the contributing force, but simply set it to zero to save computation time.

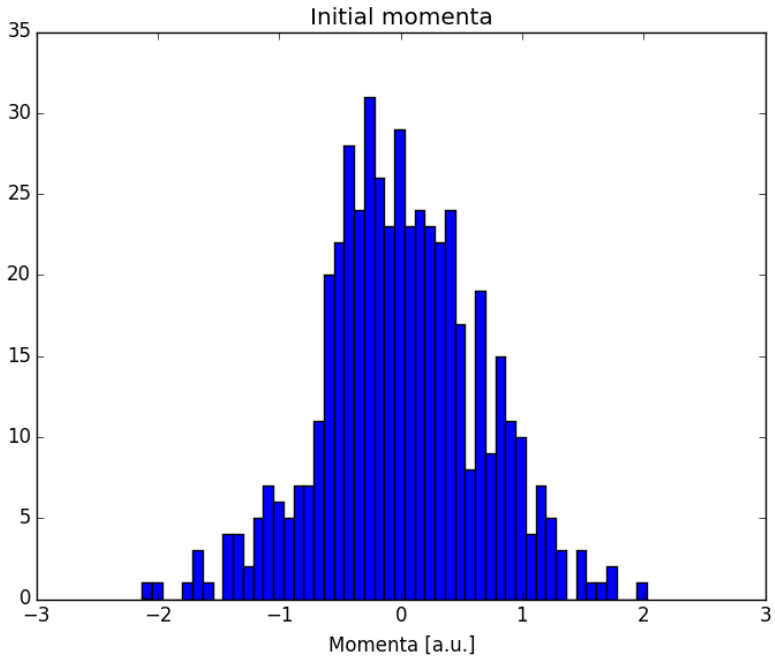

Figure 3: A histogram of the initial momenta for 500 particles. It follows a Gaussian distribution with $\mu = 0$ and $\sigma^2 = 2T$.

3.4 Verlet's Theorem

The positions of the particles are updated each iteration step. This can be done with simple Newtonian physics.

$$\vec{p} = \vec{p_0} + \vec{F}\Delta t \tag{5}$$

$$\vec{x} = \vec{x_0} + \vec{v_0}\Delta t \tag{6}$$

Where $\vec{F}$ is the force coming from the Lennard Potential between the particles, which is also updated each iteration step. In this simulation program the mass of the particles is set to 1, so $\vec{p} = \vec{v}$.

It turns out that this a rather crude approach, the positions and the momenta of the particles are only updated at integer time steps. A 'leapfrog' method such as the Verlet's Theorem would be more accurate, because this method uses the central difference for the velocity:

$$\vec{v} = \frac{d\vec{x}}{dt} = \frac{\vec{x}(t + \Delta t) - \vec{x}(t - \Delta t)}{2\Delta t} + O^2(\Delta t) \tag{7}$$

The computational error is now of the second order, compared to a first order error in the case of a backward or forward difference.
So we have used the Verlet's Theorem in our programme:

$$\vec{p} = \vec{p_0} + \frac{1}{2}\vec{F}\Delta t \tag{8}$$

$$\vec{x} = \vec{x_0} + \vec{v_0}\Delta t \tag{9}$$

$$\vec{F} = \sum_{i,j} \vec{F}(\vec{x_i} - \vec{x_j}) \tag{10}$$

$$\vec{p} = \vec{p} + \frac{1}{2}\vec{F}\Delta t \tag{11}$$

3.5 Data blocking

The most straight forward way to calculate the error in computationally simulated data, is to take use the central limit theory in calculating a mean and a standard deviation from that mean. However, this only works for uncorrelated samples. In reality, the data are often showing some memory, as is the case in Molecular Dynamics simulations. All the data within a range τ of a data point are correlated to that data point. If the time-step between data points Δt is smaller than τ, we can use a method called *data blocking*.

When using data blocking, the sample of data points is divided into subsequent data blocks. These blocks are of such size that the n^{th} data point of block i is not correlated with the n^{th} data point of block $i+1$. The minimal size satisfying this condition is the correlation time τ, and this is most commonly used as the block size. In order to get this τ, an autocorrelation function of the data is determined. The correlation time τ is the time at which the correlation drops below half of the autocorrelation at zero. For the error, we then use equation 12 where σ is the standard deviation and M the total number of data points.

$$e = \sqrt{\frac{2\tau}{M}}\sigma \tag{12}$$

3.6 Temperature renormalization

In physical experiments, often the temperature of the system is held at a fixed value. In order to simulate this our programme we keep the temperature fixed as an input parameter by using a temperature renormalization. In other words we want to assign a specific value to the kinetic energy. We do this by adjusting the momenta of the particles.

$$v' = \sqrt{\frac{T_{target}}{T}}v \tag{13}$$

Where T_{target} is the temperature you want to maintain and T the actual temperature in the system.

4 Results

4.1 Energy

The total energy of the system is defined by equation 14.

$$E_{tot} = E_{pot} + E_{kin} \tag{14}$$

Table 1: Overview of the mean values and errors of the energy of the system for 500 particles and 200 time steps.

	Mean value [a.u.]	Error [a.u.]
Total Energy	-3679.1	26.6
Potential Energy	-3834.7	28.2
Kinetic Energy	155.58	4.73

Table 2: Overview of the mean values and errors of the temperature of the system for 500 particles and 200 time steps.

	Mean value [a.u.]	Error [a.u.]
Temperature	0.210	0.011
Temperature (renormalized)	0.2048	0.0071

The total energy should be conserved for all the time steps. The results for this is plotted in figure 4.

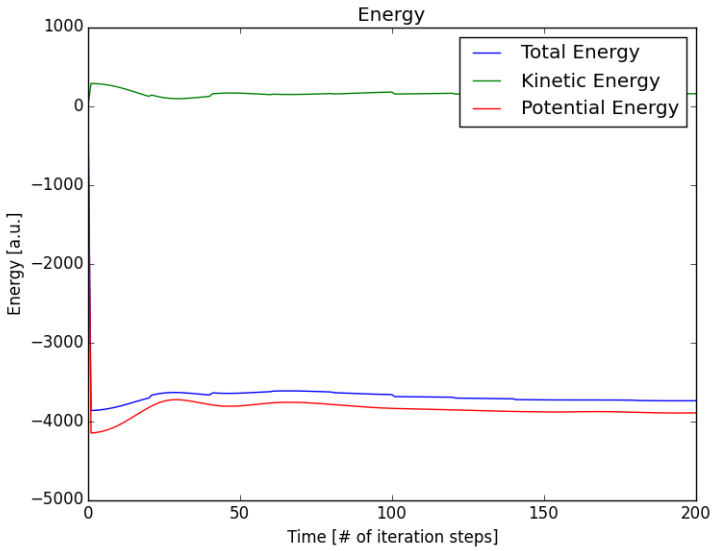

Figure 4: The total, potential and kinetic energies plotted for different time steps.

4.2 Temperature

As described in section 3.6, the temperature can be held constant using a temperature renormalization algorithm. The results for 500 particles and 200 time steps are plotted in figures 6 and 5 for with and without temperature renormalization respectively.

We have also calculated the average temperature with and without renormalization and the corresponding errors using the method described in section 3.5. The results are shown in table 2.

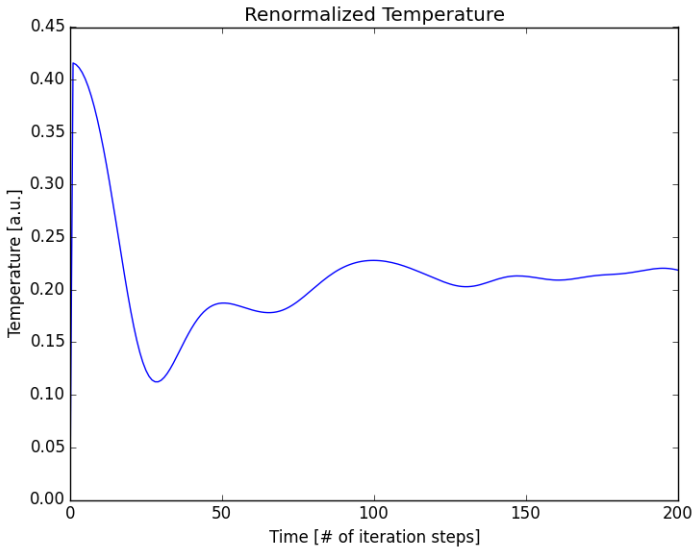

Figure 5: Temperature of the system with 500 particles for 200 time steps, without renormalization.

Table 3: Mean value and error of the pressure in the system for 500 particles and 200 time steps.

	Mean value [a.u.]	Error [a.u.]
Pressure	3.67	0.21

4.3 Pressure

We have calculated the pressure according to equation 4 for 500 particles and a temperature of $T = 0.2$ in arbitrary units, see figure 7. We have computed the average value and its error, see table 3.

4.4 Correlation function

With our programme we have computed the correlation function for three different states of the system; gas, liquid and solid. The results are plotted in figures 8, 9 and 10 respectively. The different states are obtained by changing the (renormalized) temperature.

4.5 Computation Time

To see how our programme handles a larger number of particles, we have measured the computation time for different number of particles/unit cells. The results can be seen in figures 11 and 12. We noticed that the first step takes a lot more time than the others, because it has to initialize the problem, i.e. the initial momenta and positions. That is the reason we have the computation time for 1 and 11 time steps.

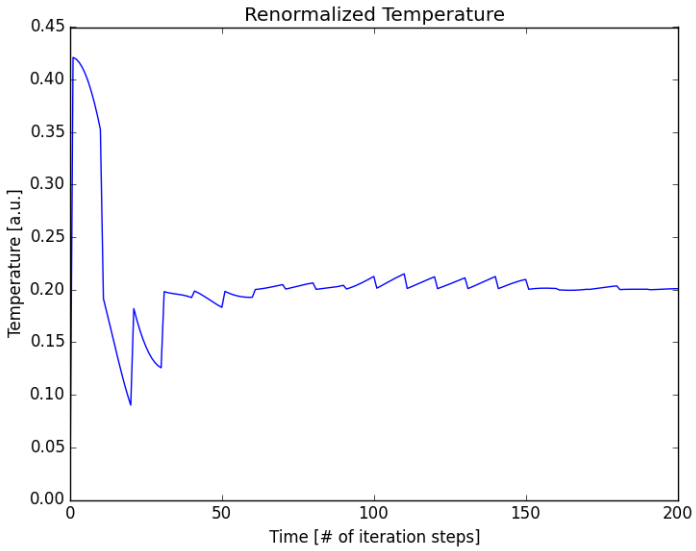

Figure 6: Temperature of the system with 500 particles for 200 time steps, with renormalization.

4.6 Animation

We were able to keep track of each particle?s positions following every iteration run and use this data to animate its motion in real time. We did this by plotting the x,y,z coordinates of each Argon atom on a 3D scatter plot. We continually updated its positions, while erasing the previous one. This gave the illusion of motion as the program ran.

By varying the temperature in the initial conditions we formed Argon in both a solid and gaseous states, see figures 13 and 14.

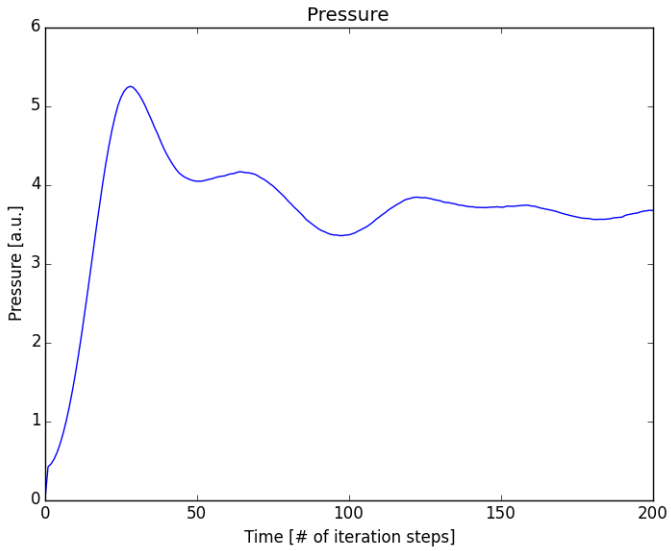

Figure 7: The pressure in the system for 500 particles and a temperature of $T = 0.2$ [a.u.].

5 Conclusions and Discussion

The Python-based molecular dynamics simulation described in this report can simulate at least 500 Argon particles in a box with periodic boundary conditions. This program has functionalities to fix or get the temperature, calculate the pressure and the correlation function, and animate the movement of the Argon particles in a visually appealing manner. During these simulations, the energy and momentum in the system are conserved to a reasonable degree.

Improvements on this program could be made by increasing the efficiency of the code, for example, exporting portions to Fortran to execute the time-consuming parts of the calculation, such as the force and distance calculations. Another improvement would be the extension of functionalities, like a heat capacity calculation module. A third improvement builds upon an assumption made at the beginning of the program, which is that all physical constants (the Boltzmann factor, the unit cell size, etc.) are unity. Implementing the correct physical values would make for easier use of the program, by enabling the user to input familiar values for the box length, temperature, pressure and other input parameters.

In the end, the simulation of Argon particles using this simulation is a useful exercise to gain more insight into the physical properties an intricacies of such an apparently simple system. It can visually show the difference of particle behavior in different phases of matter (solid, liquid and fluid).

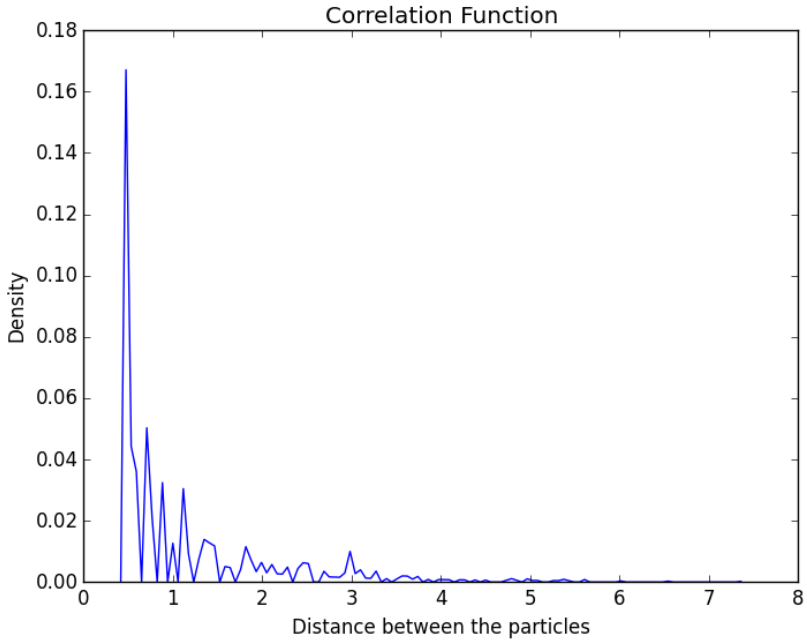

Figure 8: Correlation function of the gas state of the system with 108 particles for a temperature $T = 2$ [a.u.].

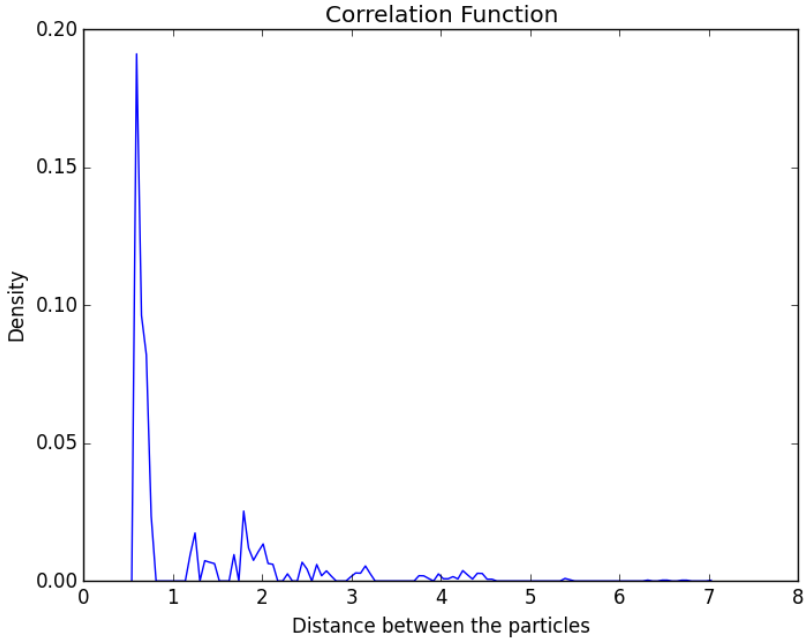

Figure 9: Correlation function of the liquid state of the system with 108 particles for a temperature $T = 2 \cdot 10^{-4}$ [a.u.].

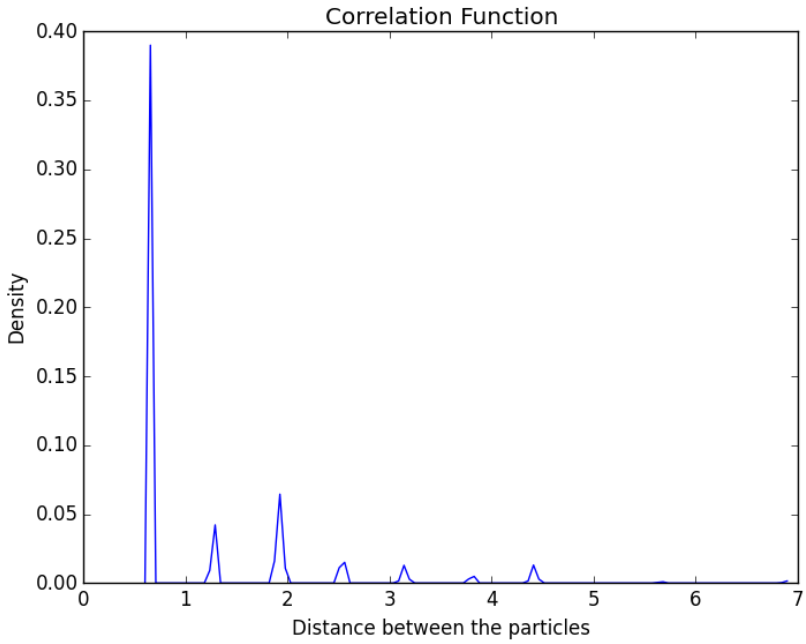

Figure 10: Correlation function of the solid state of the system with 108 particles for a temperature $T = 2 \cdot 10^{-6}$ [a.u.].

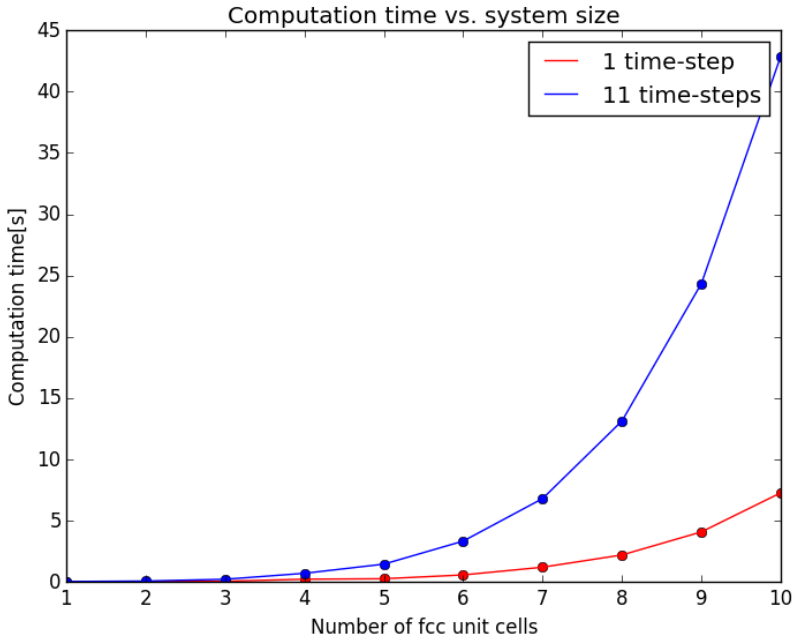

Figure 11: Computation time in seconds for an increasing number of unit cells.

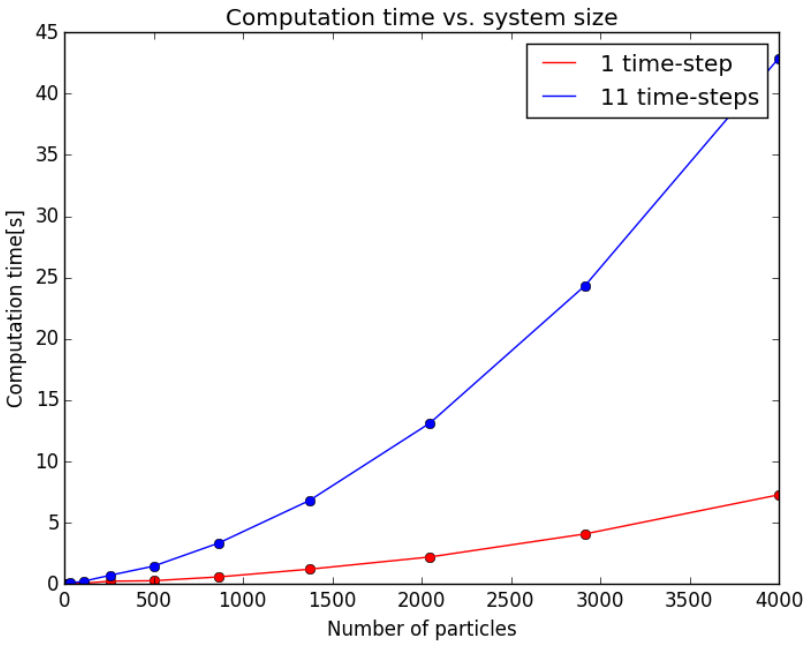

Figure 12: Computation time in seconds for an increasing number of particles.

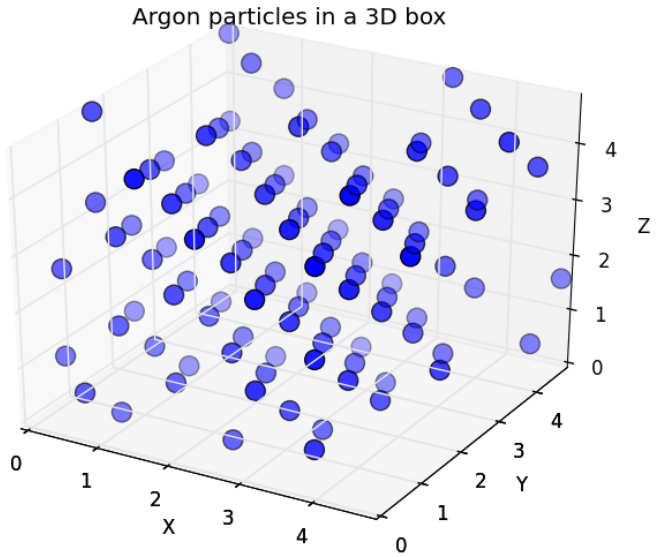

Figure 13: Animation solid

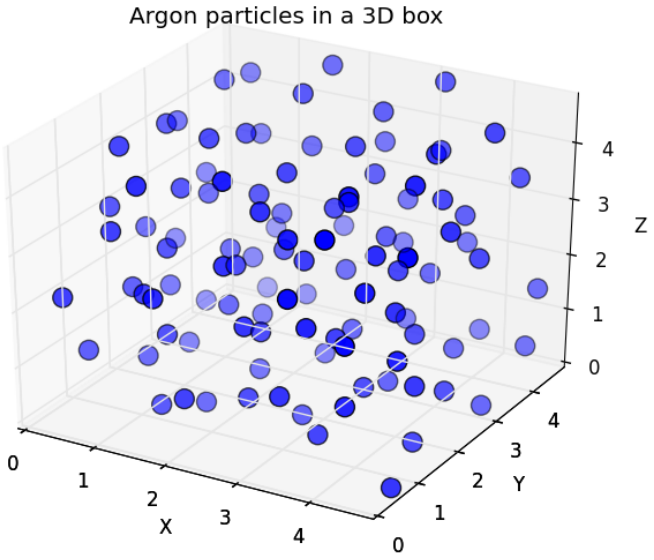

Figure 14: Animation gas

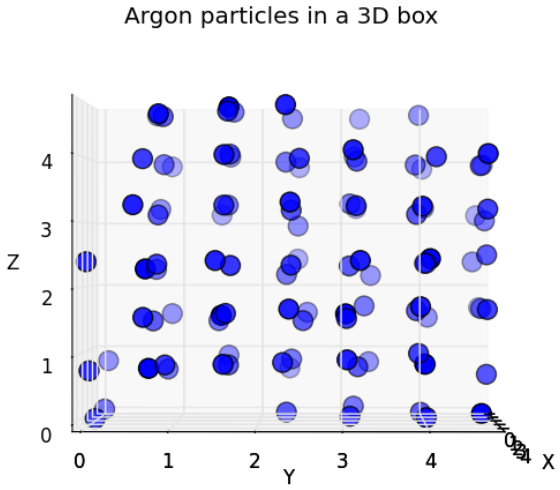

Figure 15: Animation gas wiggle