Σ 19/1

Solutions to Problem Set 5

Fundamentals of Simulation Methods 8 ECTS
Heidelberg University WiSe 20/21

Elias Olofsson
ub253@stud.uni-heidelberg.de

December 9, 2020

1 Preparatory Work (8pts)

In order to rewrite all suitable equations of motion for N argon atoms in terms of the constant parameters

$$\sigma = 3.4 \cdot 10^{-10} \text{ m}, \tag{1}$$
$$\epsilon = k_B \cdot 120 \text{ K} = 1.65 \cdot 10^{-21} \text{ J}, \tag{2}$$
$$\mu = 6.69 \cdot 10^{-26} \text{ kg}, \tag{3}$$

we define the scaled and dimensionless variables

$$\vec{x}\,' = \frac{\vec{x}}{\sigma}, \qquad E' = \frac{E}{\epsilon}, \qquad m' = \frac{m}{\mu}, \tag{4}$$
$$\vec{v}\,' = \frac{\vec{v}}{\sqrt{\epsilon/\mu}}, \qquad t' = \frac{t}{\sigma\sqrt{\mu/\epsilon}}, \tag{5}$$

✓

noticing that we can combine our parameters σ, ϵ and μ to get the dimensions of time and velocity required to scale $\vec{v}$ and t. Furthermore, we use these relationships find the dimensionless equivalents of temporal and spacial derivatives

$$\frac{d}{dt} = \frac{1}{\sigma\sqrt{\mu/\epsilon}}\frac{d}{dt'}, \qquad \nabla = \frac{1}{\sigma}\nabla'. \tag{6}$$

Our equations of motions are

$$\frac{d\vec{x}_i}{dt} = \vec{v}_i, \tag{7}$$
$$m_i\frac{d\vec{v}_i}{dt} = \vec{F}_i^{\text{tot}} = -\nabla V_i^{tot} = -\sum_{j\neq i}\nabla V(|\vec{x}_i - \vec{x}_j|), \tag{8}$$

where we use V as the Lennard-Jones potential

$$V(r) = 4\epsilon\left[\left(\frac{\sigma}{r}\right)^{12} - \left(\frac{\sigma}{r}\right)^{6}\right], \tag{9}$$

between particles i and j. Thus, substituting the dimensionless variables into the first of the equations and expanding using the chain rule, we have

$$\begin{aligned}
LHS &= \frac{d\vec{x}_i}{dt} = \frac{dt'}{dt}\frac{d}{dt'}(\sigma\vec{x}_i{}') = \sqrt{\frac{\epsilon}{\mu}}\frac{d\vec{x}_i{}'}{dt'}, \qquad (10)\\
RHS &= \vec{v}_i = \sqrt{\frac{\epsilon}{\mu}}\vec{v}_i{}', \qquad (11)\\
\implies \quad \frac{d\vec{x}_i{}'}{dt'} &= \vec{v}_i{}', \qquad (12)
\end{aligned}$$

and doing the same for the second equation, we get

$$\begin{aligned}
LHS &= m_i\frac{d\vec{v}_i}{dt} = \mu m_i'\frac{dt'}{dt}\frac{d}{dt'}\left(\sqrt{\frac{\epsilon}{\mu}}\vec{v}_i{}'\right) = \frac{\epsilon}{\sigma}m_i'\frac{d\vec{v}_i{}'}{dt'}, \qquad (13)\\
RHS &= -\sum_{j\neq i}\nabla V(|\vec{x}_i - \vec{x}_j|) \qquad (14)\\
&= -\sum_{j\neq i}\nabla\left[4\epsilon\left[\left(\frac{\sigma}{r_{ij}}\right)^{12} - \left(\frac{\sigma}{r_{ij}}\right)^{6}\right]\right] \qquad (15)\\
&= -\sum_{j\neq i}\frac{1}{\sigma}\nabla'\left[4\epsilon\left[\left(\frac{\sigma}{\sigma r'_{ij}}\right)^{12} - \left(\frac{\sigma}{\sigma r'_{ij}}\right)^{6}\right]\right] \qquad (16)\\
&= -\frac{\epsilon}{\sigma}\sum_{j\neq i}\nabla'\left[4\left[\left(\frac{1}{r'_{ij}}\right)^{12} - \left(\frac{1}{r'_{ij}}\right)^{6}\right]\right] \qquad (17)\\
\implies \quad m_i'\frac{d\vec{v}_i{}'}{dt'} &= -\sum_{j\neq i}\nabla' V'(r'_{ij}), \qquad (18)
\end{aligned}$$

with the dimensionless Lennard-Jones potential

$$V'(r') = 4\left[\left(\frac{1}{r'}\right)^{12} - \left(\frac{1}{r'}\right)^{6}\right], \quad \checkmark \qquad (19)$$

and where $r'_{ij} = \|\vec{x}_i{}' - \vec{x}_j{}'\|$. Thus the dimensionless equations of motion for the N argon atoms in terms of their parameters σ, ϵ and μ, is given by eq.(12) and eq.(18-19).

For implementation specific details regarding the rest of the N-particle simulation, please see the attached `md.c` file. Furthermore, a `Makefile` for easy compilation of the program has also been included. With this file, the random number generator PCG in `pgc_basic.c` from `https://www.pcg-random.org/` that I have used for generation of random floating point numbers, is automatically linked during compilation.

I have modified the given code template so that it can fully utilize neighbor lists for more efficiently keeping track of each particle's nearest neighbors.

Thus there are two radii that can be set for the simulation, with the first being `rcut` which sets the potential to zero for all pairwise particle interactions more distant than this radius. The other one is `neighbor_cut` which determines the radius for which two particles are considered neigbors, and thus are added to one of the particle's list during construction.

Nice +1p

During all of these simulations, the cutoff radius for the potential has been large enough ($r_{\text{cut}} > \frac{\sqrt{3}}{2}L$) to not effectively change the behaviour of the code compared to the normal direct summation method, where all pairwise force-interactions are considered at all times, for all particles in the entire system. Thus we have a code with a complexity that scales as $\mathcal{O}(N^2)$ with N particles.

However, since all the preparations are done with the code, the complexity can greatly be reduced by reducing both of the radii, and tweak how often the neighbor lists are re-constructed. If these three parameters are balanced correctly, we can potentially get close to a linear complexity $\mathcal{O}(N)$ instead. But since the time for execution was only ∼15 s for the number of particles and iterations specified in the task, we can reasonably wait a few extra seconds and retain the full precision of direct summation method instead. → or increase the box size / more particles

→ You can get to $\mathcal{O}(N \cdot \log N)$ at best

2 Implementation and application (12 pts)

Instead of the normal leapfrog scheme, I have chosen to implement the integration in my code using the Velocity-Verlet scheme. The reason being that this scheme is a bit easier to overview than the half-shifted normal scheme, and it also has the benefit that positions and velocities are already calculated at the same time instances. Thus kinetic and potential energy calculations are straight-forward and no half-stepping forwards is required. We do however need to store the acceleration of the previous timestep, but since the C-code had already made room for this, an implementation was easy.

like most MD software

My implementation does not explicitly output instantaneous kinetic temperature to a file, contrary to the instructions. But this is no problem however, since we can easily convert the average kinetic energy per particle, which is in fact saved to a file, to kinetic temperature using the relation

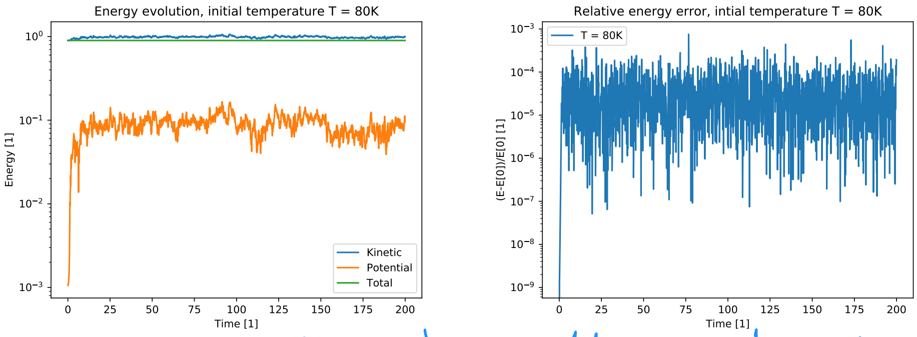
$$\frac{E_{\text{kin}}}{N} = \left\langle \frac{mv_i^2}{2} \right\rangle = \frac{3}{2}k_B T, \tag{20}$$

or rather the dimensionless version

$$\frac{E'_{\text{kin}}}{N} = \left\langle \frac{m'v_i'^2}{2} \right\rangle = \frac{3}{2}\frac{k_B}{\epsilon}T = \frac{3}{2}\frac{T}{120\ \text{K}}, \tag{21}$$ ✓

where we used eq.(2) for the last substitution.

Running the simulation for $\Delta t' = 0.01$, $N = 125$ for the initial temperature $T_{\text{init}} = 80\text{K}$, we plot the evolution of the energies and the relative error of the total energy, as can be seen in fig.(1). Here, we can see that the total energy is conserved, which is to be expected from the symplectic properties of the Velocity-Verlet scheme. Furthermore, we can also calculate the auto-correlation of the average absolute value of the velocity of the system, seen in fig.(2). → as discussed in tutorial, try substracting the mean first



✓

(a) → So almost everything

(b) Kinetic, as expected for an "ideal" gas

Figure 1 – Left: Time evolution of kinetic, potential, and total energy per particle of the system. Right: Time evolution of the relative error of the total energy of the system.

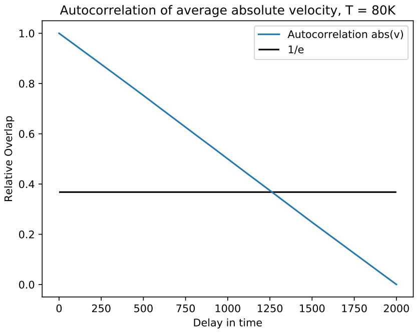
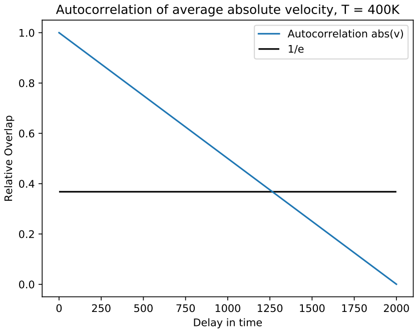


Figure 2 – Auto-correlation of the average absolute velocity, for an initial temperature of 80K.

Doing the same for a simulation with an initial temperature of 400K, we can plot the corresponding auto-correlation of the average absolute value of the velocities as seen in fig.(3).

To be honest, I'm not entirely sure on how to interpret these two graphs. Autocorrelation normally is supposed to make it easier to discern a periodic



Correct: do it particle by particle (since it's about "self" correlation) and average afterwards

Figure 3 – Auto-correlation of the average absolute velocity, for an initial temperature of 400K.

So did you average first?

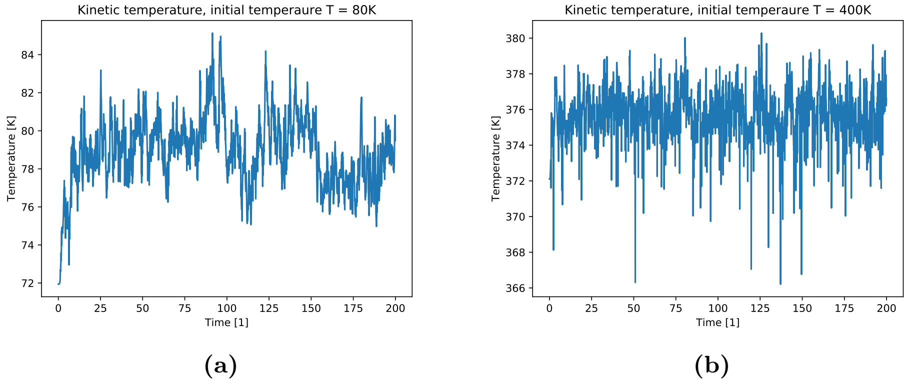
signal from a noisy background or similar, but I have no idea on how it's supposed to be used here, if I have calculated it correctly, or if I have something else wrong with my simulation in general.

A straight line on a steady descent I think should indicate a rather constant and uncorrelated signal, thus that the average absolute velocity is constant, and in extension that the kinetic temperature is constant. This makes sense to me, but having these unnaturally straight plots makes me question if I have done things correctly. ⇒ yeah they should not look like this

However, we can easily plot the kinetic temperature for both initial temperatures, using the calculated kinetic energies and the relation in eq.(21), which results can be seen in fig.(4). Since these two graphs are not unnat-

only on the autocorrelation since ✓

Good comments and analysis: not so long



(a) **(b)**

Figure 4 – Time evolution of the kinetic temperature for the system at initial temperature 80K and 400K, respectively.

urally constant, I believe that my auto-correlations from above cannot be correct, or at least that I have misunderstood something.

I have also unfortunately run out of time, and cannot complete the last part of this exercise, which was to determine the molar heat capacity. I have no direct idea in this very moment on how I would have calculated this, so I would deeply appreciate if this could be explained during the tutorial on friday, as well possibly how the auto-correlation should have been done, and what information we could draw from that.

Was messed up a bit from
our side as well, so only
-1p for not trying