Dynamique moléculaire et simulation Monte-Carlo
5: Classical Molecular Dynamics (MD) Simulations

Bruno Rodriguez Carrillo
EPFL

May 10, 2021

MD Initialization and Temperature

1. Implement the initialization described in the Theory section in our ToyMD code in 'toy_md.py' in the section '##INITIALIZATION HERE##'. For this you should pick random velocities using 'r.random()' which will produce random numbers between '0' and '1'. You need to shift this random gaussian distribution appropriately and then multiply it such that the width of the velocity distribution matches the kinetic energy at the target temperature. Use the variables 'masses[i]' for the mass of particle 'i' and Boltzmann constant '0.0831415'. Remember that for each degree of freedom (e.g velocity in x direction) (5.22) holds.

   Since we know that equation (5.22) holds for each degree of freedom, we have for v_x^2:

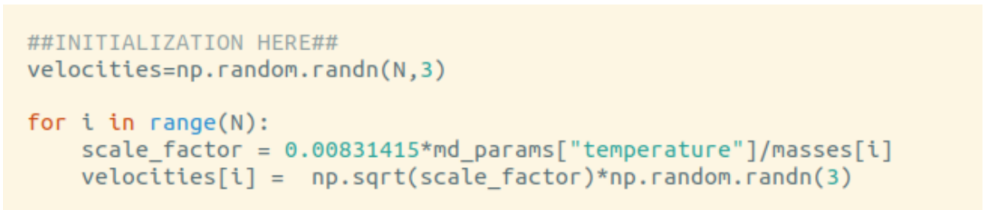
   $$E_{kin} = \frac{1}{2}k_B \cdot T = \frac{1}{2}m \cdot v_x^2 \Rightarrow v_x^2 = \frac{k_B \cdot T}{m}$$

   For v_x to follow a normal distribution with zero mean and variance $\sigma^2 = T$, we have

   $$v_x \sim \sqrt{\frac{k_B \cdot T}{m}} \cdot N(0,1)$$

   We use the same approach for v_y and v_y.

   Then a possible implementation is:



```
##INITIALIZATION HERE##
velocities=np.random.randn(N,3)

for i in range(N):
    scale_factor = 0.00831415*md_params["temperature"]/masses[i]
    velocities[i] =  np.sqrt(scale_factor)*np.random.randn(3)
```

   Figure 1: Velocity re-scale.

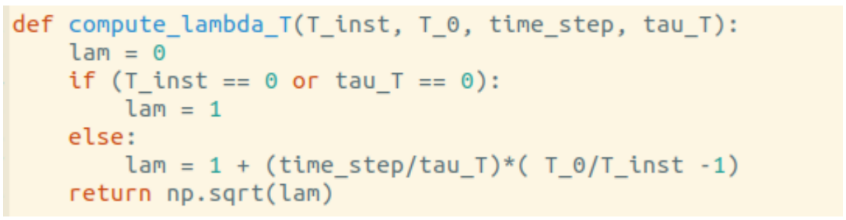
2. Implement the Berendsen thermostat in the 'toy_md.py' and 'toy_md_integrate.py' (change the 'compute_lambda_T' function) files.

   From equation (5.19), we have that:

   $$\lambda = \sqrt{1 + \frac{dt}{\tau} \cdot \left(\frac{T_0}{T} - 1\right)}$$

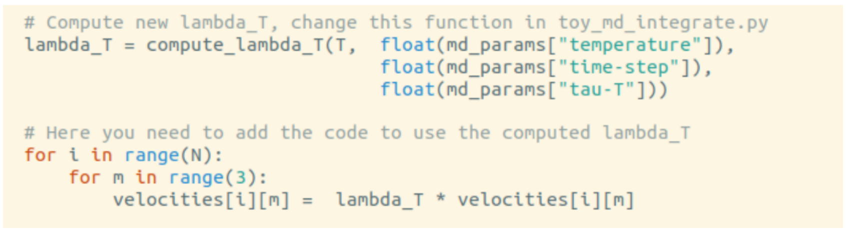
   Then we re-scale each velocity component as $v = \lambda \cdot v$ for each particle too.

   As a result, we have the following implementation:



```
def compute_lambda_T(T_inst, T_0, time_step, tau_T):
    lam = 0
    if (T_inst == 0 or tau_T == 0):
        lam = 1
    else:
        lam = 1 + (time_step/tau_T)*( T_0/T_inst -1)
    return np.sqrt(lam)
```

   Figure 2: Berendsen thermostat lambda computation



```python
# Compute new lambda_T, change this function in toy_md_integrate.py
lambda_T = compute_lambda_T(T,  float(md_params["temperature"]),
                                float(md_params["time-step"]),
                                float(md_params["tau-T"]))

# Here you need to add the code to use the computed lambda_T
for i in range(N):
    for m in range(3):
        velocities[i][m] =  lambda_T * velocities[i][m]
```

Figure 3: Berendsen thermostat velocity re-scaling

3. A better thermostat is the Andersen thermostat. It can be implemented as follows. Describe what problems this thermostat will present to us e.g for sampling of diffusion coefficients. What advantage does this thermostat have compared to the Berendsen thermostat?

   Based on [2], this method is closest to the experimental situation: the velocity alterations mimic particle collisions with the walls. The rate at which particles should undergo these changes in velocity influences the equilibration time and the kinetic energy fluctuations. If the rate is high, equilibration will proceed quickly, but as the velocity updates are uncorrelated, they will destroy the long time tail of the velocity autocorrelation function. Moreover, the system will then essentially perform a random walk through phase space, which means that it moves relatively slowly.

   Furthermore, the Nosé and the Andersen methods yield precise canonical distributions for position and momentum coordinates. However, for the Andersen method, it is not always clear at which rate the velocities are to be altered.

   On the other hand, Berendsen thermostat does not reproduce the canonical distribution.

Sampling Configurational Space

Open the resulting trajectory in VMD and plot the bond distance of a $C - O$ bond. What to you observe?

1. Plot the distribution of the CO and OO bond lengths during the simulation. How do the sampled values correspond to the values set in the force field?

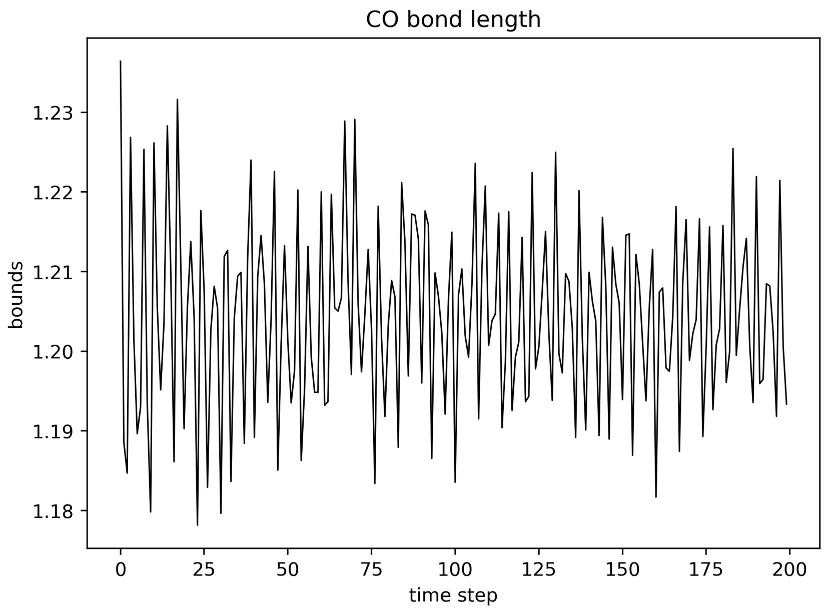


Figure 4: CO bond length.

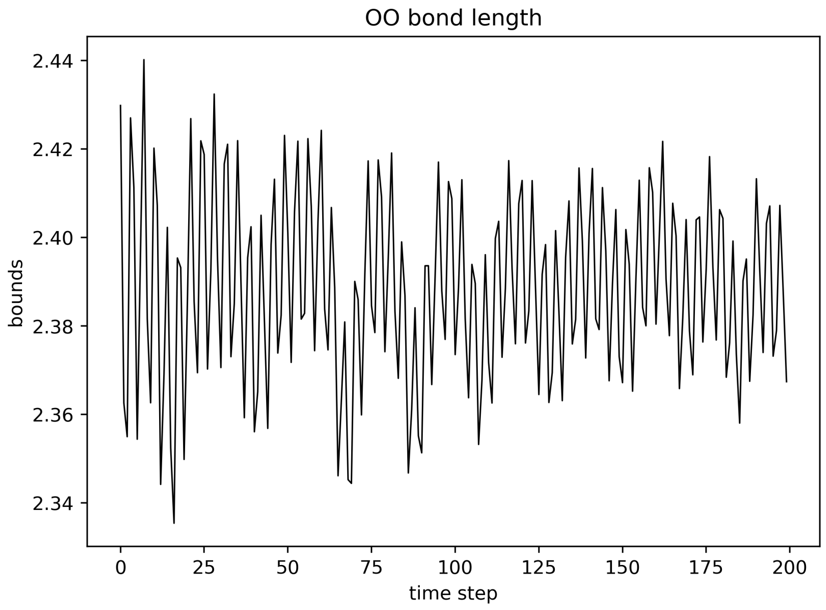


Figure 5: OO bond length.

From [1], we should bear in mind that sometimes we have to deal with large molecular systems, such as biopolymers. In these cases, the bond length of a pair of nearest neighbors does not change very much even though the angle between a pair of nearest bonds does. If we want to obtain accurate simulation results, we have to choose a time step much smaller than the period of the vibration of each pair of atoms. This costs a lot of computing time and might exclude the applicability of the simulation to more complicated systems, such as biopolymers.

The observed lengths correspond to the first term in equation (5.15). Such a force tries to keep the atoms of a fixed molecule together. This also means that the repulsion and attraction forces acting are such that the length of the bonds is kept constant throughout the simulation time. This is the reason why we observe an oscillating behavior around a value. If the length changed abruptly, it would probably mean that the molecule is destroyed because of other forces like the one generated by the Leonard Jones potential acting on molecules.

2. Visualize the radial distribution function of the CO_2 trajectory. What do you observe? Bonus: What would you observe for a heterogenous system (i.e a polar molecule solvated in liquid CO_2)?

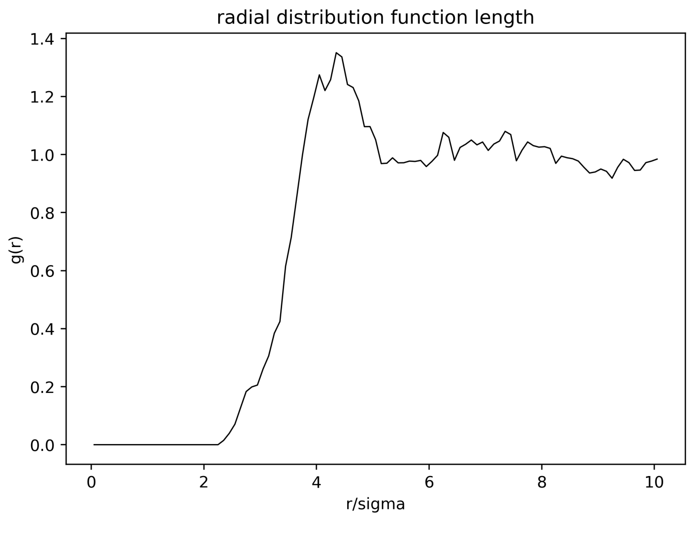


Figure 6: Radial distribution function of the CO_2 trajectory.

From [2], we have that the pair correlation function contains information concerning the local structure of a fluid. Suppose we were to sit somewhere in the fluid and watch the surroundings for some time, then, on average, we would see a homogeneous structure. Now, If we move along with a particular particle, however, and watch the scenery from this particle's perspective, we will find no particles close to us because of the strong short-range repulsion. Then we have an increase in density due to a layer of particles surrounding our particle, followed by a drop in density marking the boundary between this layer and a second layer, and so on. Because of the fluctuations, the layer structure becomes more and more diffuse for increasing distances and the correlation function will approach a constant value at large distances, as we observe in the plot above.

Timestep and coupling

1. Test the influence of the coupling parameter τ and timestep dt. You set both of those parameters in params.txt. Test 3 different settings of τ and dt and describe what you observe.

   When τ tends to infinity, we observe that the Berendsen thermostat is inactive since we have $\frac{dT}{dt} = 0$ and then we are just implementing velocity re-scaling by a factor of 1 ($\lambda = 1$); hence, temperature fluctuations are allowed. The reason why this happens is that the velocities are first generated according to the re-scaling $v_x \sim \sqrt{\frac{k_B \cdot T}{m}} \cdot N(0, 1)$. Thus temperature fluctuations will vary until they reach the appropriate value of a microcanonical ensemble. Even thought temperature fluctuates, we are not sampling the NVT ensemble but the isokinetic one. This is the case when $\tau >> dt$.

   On the other hand, too small values of τ will cause abrupt temperature fluctuations, in this case $dt >> \tau$. When $\tau = dt$, the Berendsen thermostat is a velocity scaling from one time step to another. As a result, the dynamics generated in this case is not realistic.

   We should also point out that the dt parameter is the one used during the Verlet algorithm and such a number determines in some sense the accuracy of our values. As it is well known, too small values of dt will generate more precise results but at the same time computational cost will grow.

Bibliography

[1] Tao Pang. *An Introduction to Computational Physics*. 2nd ed. Cambridge University Press, 2006. DOI: `10.1017/CBO9780511800870`.

[2] Jos Thijssen. *Computational Physics*. 2nd ed. Cambridge University Press, 2007. DOI: `10.1017/CBO9781139171397`.