Exercise 6 – Computational Materials Engineering

Prof. M. Moseler – WS 2020/2021

Goals:

- Understand molecular vibrations.
- Understand how to choose a proper time step in molecular dynamics simulations.
- Understand the total energy of a system.
- Understand energy conversion in the NVE ensemble.
- Understand the instantaneous temperature of a molecular system.

This exercise sheet is an extension of sheet 5. We consider the dynamics of two atoms interacting via the usual Morse potential

$$u_M(R) = D_0\left[e^{-2\alpha(R-R_0)} - 2e^{-\alpha(R-R_0)}\right]. \quad (2)$$

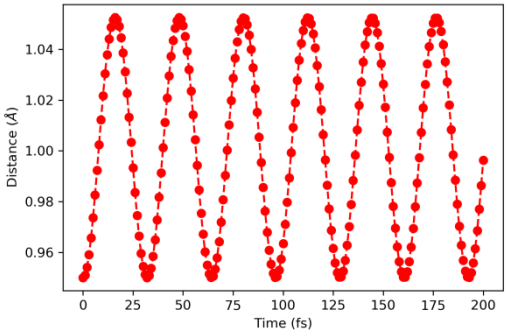
Throughout the whole sheet we use $\boldsymbol{D_0 = 1\ \text{eV}}$, $\boldsymbol{\alpha = 1\ \text{Å}^{-1}}$ **and** $\boldsymbol{R_0 = 1}$ Å.

It is recommended to copy the files of exercise sheet 5 into a new directory and save all the scripts in that folder.

Tasks

Task 1: Open the *dimer_dynamics.py* script, set the initial positions of the two atoms to $\vec{r}_1 = (0, 0, 0)$, $\vec{r}_2 = (0, 0, 0.95)$ and run it for 200 steps. Open the created *md.xyz* file with OVITO and observe the dynamics of the dimer.

Task 2: Download *plot_distance_template.py* from ILIAS. The script reads in all configurations included in *md.xyz* and prints the distances between the atoms along the trajectory. Modify the script and plot the distance between the atoms as a function of time. Label the axes accordingly. The resulting plot could look like this:



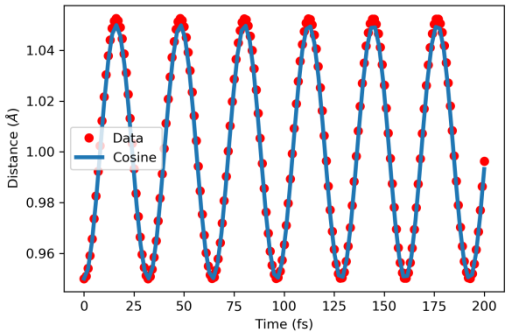
Task 3: If the distance between the two atoms is not too far from the equilibrium distance R_0, the energy around the minimal energy can be approximated by a parabolic function, i.e. our problem simplifies to a harmonic oscillator. For the starting conditions used (no initial velocities) the solution for the distance d between the two atoms is given by a cosine function of the form

$$d(t) = A \cdot \cos(\omega t) + B.$$

What is the meaning of A and B and what is their numerical value? The frequency ω can be calculated using the second derivative of the Morse potential evaluated in the energy minimum.

$$\omega = \sqrt{\frac{2k}{m}} \qquad \text{with} \qquad k = \frac{d^2 u_M(R)}{dR^2}\Big|_{R=R_0}.$$

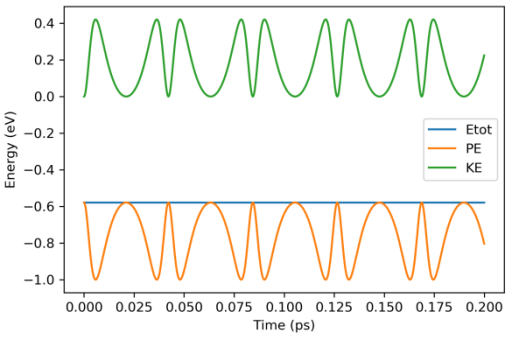
Calculate ω in units of [1/fs]. The conversion factor from $\sqrt{\frac{\text{eV}}{\text{Å}^2\, u}}$ to 1/fs is 0.0982269475. Add the cosine function with the numerical parameters you obtained to the graph of Task 2.



Task 4: Rerun the simulation again for 200 fs, but this time with a time step of $\Delta t = 10$ fs. Inspect the trajectory with OVITO and recreate the graph of Task 3 for the new results. What is going wrong here?

Task 5: Reset the time step to $\Delta t = 1$ fs and rerun the simulation for the intial conditions $\vec{r}_1 = (0, 0, 0)$, $\vec{r}_2 = (0,\ 0, 0.5)$. Recreate the graph of Task 3 for the new results. Why is the cosine function not fitting the data anymore?

Task 6: Rerun the simulation with $\Delta t = 0.1$ fs for 200 fs with the initial conditions of Task 5. When running a molecular dynamics simulation, ASE automatically creates a log files that contains the total energy, the kinetic energy, the potential energy and the temperature of the system. Inspect the *md.log* file. Create a new python script that reads in *md.log* and plots the total energy, the kinetic energy and the potential energy as a function of time. What is the relationship between the energies? *Hint:* the *loadtxt* function from *numpy* has a keyword *skiprows* that you can use to not read in the first line of the log file.



Task 7: Calculate the temperature from the kinetic energy in the log file using the formula given in the lecture and compare to the temperature values listed in the log file. Note that your accuracy is limited as the output in the log-files is rounded.

Results:

Task 2:

1. Append distances to a list-object
2. Create a time list-object (with same number of entries as the distance list-object)
   201 entries, because we start from time = 0fs and end at time = 200fs.
3. Plot time against distance and define some plot-styles
   (e.g. 'ro--' = *red* color, *o* markers, --dashed line)

Task 3:

1. A = Amplitude, B = R_0 = Minimum-Energy distance (rest-position)
2. Calculate $\omega = \sqrt{\frac{2k}{m}}$, with $k = \frac{d^2u_M(R)}{dR^2}\Big|_{R=R_0}$, *m=1.008u*

$$\frac{du_M(R_{KJ})}{dR_{KJ}} = D_0 2\alpha\left[-e^{-2\alpha(R-R_0)} + e^{-\alpha(R-R_0)}\right]$$

$$\frac{d^2u_M(R_{KJ})}{dR_{KJ}^2} = D_0 2\alpha^2\left[2e^{-2\alpha(R-R_0)} - e^{-\alpha(R-R_0)}\right]$$

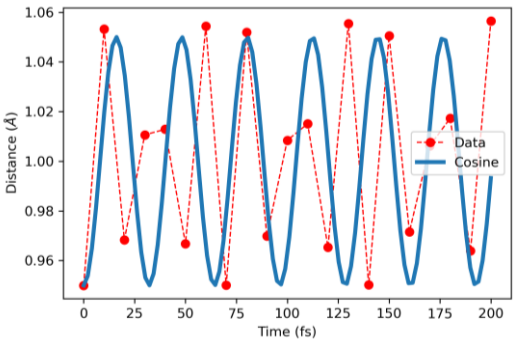
$$\omega = \sqrt{\frac{2D_0 2\alpha^2[2e^{-2\alpha(R-R_0)} - e^{-\alpha(R-R_0)}]}{m}}$$

$$\omega = 0.195673\ fs^{-1}$$

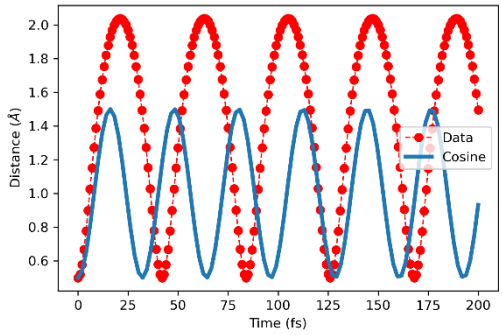
Add cosine function:

- Define cosine function (e.g. def cosine_func(x, A, omega, B):)
- Define xdata-values from 0 to 200 fs
- Plot xdata against cosine_func(xdata)

Task 4:



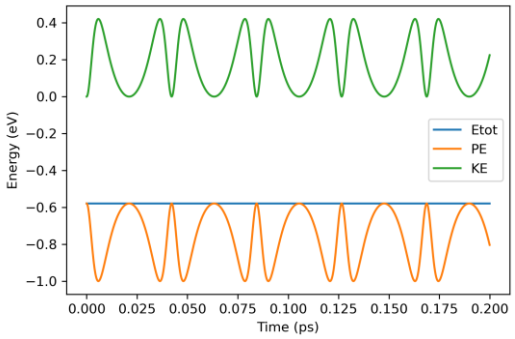
Task 5:



Task 6:

Plot Energy script:

1. Read time, total Energy, potential Energy, kinetic Energy and Temperature from Md_fine.log (np.loadtxt())
2. Plot Energies against time (plt.plot)



Task 7:

Formula is given in Lecture 6:

$$T = E_{kin}/(\frac{3N}{2}k_b)$$

Include formula in Plot Energy script and compare with the Temperature.