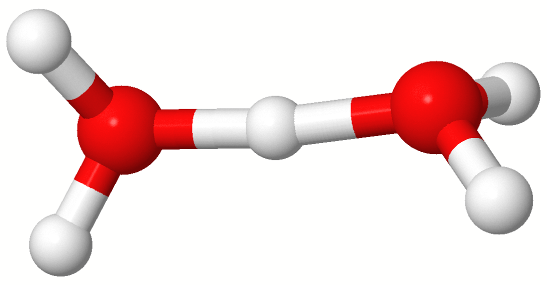
Advanced Materials Modelling *ab-initio* molecular dynamics with FHI-aims

Skoltech, May 12, 2020



Tutorial III: *Ab initio* Molecular Dynamics Manuscript for Exercise Problems

Prepared by Debalaya Sarker

with help of: Zhong-Kang Han and and Sergey V. Levchenko

Introduction

With this tutorial, we aim at introducing you to state-of-the-art techniques for sampling the potential energy surface of a system at constant energy (microcanonical ensemble) and at constant temperature (canonical ensemble). The technique we adopt is molecular dynamics (MD), that can be thought as old as modern physics itself, since we will *propagate* in time a system of particles, given initial positions and momenta, by *numerically integrate* Newton's equations of motion for the system.
Indeed the original formulation of Newton's second law is [1]: "*Lex II: Mutationem motus proportionalem esse vi motrici impressae, et fieri secundum lineam rectam qua vis illa imprimitur.*", in modern English, "Law II: The change of momentum of a body is proportional to the impulse impressed on the body, and happens along the straight line on which that impulse is impressed." In symbols:

$$\int_{\Delta t} \boldsymbol{F} \mathrm{d}t = \Delta \boldsymbol{p} \tag{1}$$

Where the vector $\boldsymbol{F}$ is the force acting on a body, $\boldsymbol{p}$ is its momentum, Δt is the time along which the force is applied. The whole l.h.s. expression is called *impulse.*
This technique allows one to sample the potential energy surface of a system at constant energy (micro-canonical ensemble) and at constant temperature (canonical ensemble). As long as the *ergodic hypothesis* is valid, we will be, in principle, able to relate the time averaged parameters (or properties), to the thermodynamic averages over the corresponding statistical ensemble.

The kind of MD we are interested in is Born-Oppenheimer MD, where the degrees of freedom of the nuclei are considered to be decoupled from the degrees of freedom of the electrons (Born-Oppenheimer approximation). On top of this approximation, usually one treats the nuclei as classical particles. Therefore, the forces between them are evaluated by solving the (ground-state) electronic structure problem for a given number of electrons and for a given position of the nuclei. In this tutorial, the electronic structure problem is solved via Density Functional Theory (DFT) within the FHI-aims code and the nuclei problem is managed by the i-PI code. i-PI works through a client-server paradigm, where the *ab initio* code, in this case FHI-aims, is the client that provides the calculation of interatomic forces, and i-PI is the server that provides the evolution of the equations of motion that sample the desired ensemble.

The tutorial contains two exercises; all of them adopts as guinea-pig the gas-phase (isolated) $H_5O_2^+$, which some of you may recognize as the Zundel (cat)ion. This system is one of the (protonated) water clusters that are thought to be the "building blocks" of liquid water. [2]

The **first**, and quick, introductory exercise presents the machinery we will use through the tutorial. In particular, you will learn how to setup the i-PI and FHI-aims inputs.

The **second** exercise poses the problem of choosing the right settings for obtaining a stable and reliable molecular dynamics trajectory. In particular we will investigate the effect of the self-consistent cycle and force convergence thresholds.

The **third** exercise deals with the choice of a thermostat for simulating the canonical ensemble, i.e., the contact between your system of interest and a thermal reservoir. In this tutorial we will focus on the so-called stochastic thermostats. Other implementations, for example based on extended Lagrangians exist as well, and will be briefly explained in the exercise.

All the useful files and scripts for this tutorial (together with solutions) are located in:

```
$HandsOn/tutorial_MD
```

In this tutorial we use python 2.7; please type in your working directory:.

```
source $py27/activate
```

This will switch to this version.

Molecular Dynamics: a client/server approach

Exercise 1: Getting started

In this exercise we are going to familiarize ourselves with the syntax of **i-PI** input files. As mentioned previously, the i-PI program works through a client-server architecture. It can use INTERNET or UNIX sockets, that allow the system to be simulated on the same machine or on a different machine, as long as the calculation can communicate with the server. Let's take a look at the input files. The simulation example here will consist of one Zundel cation in the canonical (NVT) ensemble at 300K.

`Server`: An example of an input file for i-PI can be found in
`$HandsOn/tutorial_MD/exercise_1/input.xml`.
It is an xml file, which is quite intuitive to learn. Please take your time to understand the keywords that are there and consult the i-PI manual found in `$HandsOn/tutorial_MD/doc`. We will be using UNIX sockets here, which is the most convenient way to use the code when running both servers and clients in small desktop (or laptop) machines. For that, we have to specify only the `<address>`, which can simply be a string containing a name of your choice. For internet sockets, one would have to provide the relevant IP address for the `<address>` field and a number for the `<port>` field.

`Client`: The keyword to add to the `control.in` file of FHI-aims is
`use_pimd_wrapper UNIX:<address> <portnumber>`
where`<address>` should be substituted by the name of the socket you choose, and the port can be any number since it does not play a role for UNIX sockets. The address that goes in the `control.in` file should match the one in the `input.xml` file of i-PI. The `geometry.in` is only provided for the initialization of FHI-aims, so in order to avoid inconsistency problems always check that the **atoms are listed in the same order as in the i-PI input file**. (Here, we did that for you ;))

Note that in all subsequent exercises you will be free to choose `<address>` as you want, and you should please change it in your input files (`control.in` and `input.xml`) so that your server and your client always match!

Let's run an **i-PI+FHI-aims** Molecular Dynamics simulation! **Instructions**

- Open a terminal at the current directory and launch i-PI by typing:
```
i-pi input.xml
```
- At this point i-PI should start and parse the input file. At the bottom of the output on the screen it should say:
```
Created unix socket with address localhost
@ForceField:  Starting the polling thread main loop.
```
This means i-PI has started properly, has created the UNIX socket, and is waiting for the communications from the clients that take care of the force evaluations.
- Now we can launch FHI-aims. Open a second terminal, either manually or by typing ctrl+shift+t, and enter the command:
```
mpirun -np 4 $HandsOn/bin/ipi.aims.x
```
Then FHI-aims should start and you will see some outputs.
- Now switch to the terminal where i-PI is running, notice that i-PI has built the connection with FHI-aims with the following message,

```
@SOCKET: Client asked for connection from .  Now hand-shaking.
@SOCKET: Handshaking was successful.  Added to the client list.
```

and started the Molecular Dynamics simulation. It should also print on screen information about the time taken for each MD step.

- What we are going to do now is to kill FHI-aims (don't worry, you shall not be sued for that). Simply switch to the terminal where FHI-aims is running and press `ctrl+c`. Now look at whether i-PI is still running. Notice that although the evolution of the MD is paused, i-PI itself does not die but instead continues to run and waits for a new client to take over. Now start FHI-aims again by typing:

  ```
  mpirun -np 4 $HandsOn/bin/ipi.aims.x
  ```

  What happens to i-PI now?
- What if one stops i-PI? Trigger a soft exit of i-PI by typing ctrl+c at the terminal where it is running, or create a file named EXIT in the folder where i-PI is running (you can use the bash command `touch EXIT`). Watch how i-PI responds, and how FHI-aims reacts. Think about what are the advantages of a clean exit when a MD program stops unexpectedly.
- Take a look at all the output files written by i-PI. You should have the file `ex1.out` that describes the system properties, `ex1.pos.xyz` that records the atomic trajectories, and `RESTART` that contains all the information to restart the simulation.

Timing: ~20 minutes total

The microcanonical ensemble

Exercise 2: The importance of the SC convergence criteria

In this exercise, we will investigate the importance of the self-consistency convergence criteria when simulating the microcanonical ensemble. The input files for i-PI can be found in the folder `exercise_2`. Please note that the initialization in i-PI is done with a previously thermalized geometry.

Instructions

- First, build an input file (`control.in`) for FHI-aims using the LDA (**`pw-lda`**) functional and no spin polarization (**`spin none`**). Please don't forget to specify the charge (**`charge 1.0`**) and use the `"light"` numerical and basis set standards for the species. Add the following line to tell FHI-aims that you need to compute the forces: `compute_forces .true.`
  Please refer to the manual for more information about these flags. We are going to refer to this `control.in` as **"default"**.
  You can either create the `control.in` file from scratch or modify the one provided in the previous exercise. In any case, do not add any flags that we do not mention. They are not needed and might hinder the performance of the calculation.
- Add the line corresponding to the i-PI communication that was shown in the previous exercise in the `control.in` file and assure that you specify the same address in the `input.xml` file.
- Start a 0.15 ps MD run in the microcanonical ensemble, using a 0.0005 ps (Δt = 0.5 fs) time step. For that, open the provided `input.xml` and change the lines `timestep` and `total_steps` accordingly. You also need to specify the dynamics mode to be `'nve'`. The file chosen for the initialization is a check-point file containing positions, velocities, and some other settings from a previous simulation where this molecule was thermalized, using the `'nvt'` mode - that is what the `'chk'` mode in i-PI means. In this case, initializing from the checkpoint file means that i-PI will read positions and velocities from it.
- After everything is set up, start the simulation in the following way

In order to start the run type:

```
i-pi input.xml > output_i-PI &
```

- (Wait 5 seconds in order to let i-PI do the initialization).
  ```
  mpirun -np 4 $HandsOn/bin/ipi.aims.x > output_FHI-aims &
  ```
- The symbol & puts the run in the background, so that the output file is created, but the terminal is free for other use.
- The simulations are time consuming. Meanwhile you might want to open another terminal window and check the progress by typing:
  ```
  tail -f <name-of-output>
  ```
  Press Ctrl+c to exit.
- ATTENTION: do not start another FHI-aims run simultaneously.
  That would slow down BOTH calculations considerably.

- When the previous calculation is over, run another simulation, keeping all parameters mentioned above but changing the name of the output and also changing to the following “loose” self-consistency convergence criteria:

```
sc_accuracy_rho 1E-2
sc_accuracy_eev 1E-1
sc_accuracy_etot 1E-2
sc_accuracy_forces 1E-1
```

- Finally, run another simulation with “extreme” self-consistency criteria:

```
sc_accuracy_rho 1E-7
sc_accuracy_eev 1E-6
sc_accuracy_etot 1E-9
sc_accuracy_forces 1E-7
```

IMPORTANT: Never use these settings in real production calculations. They are meant only for this pedagogical exercise.
Reminder: Change the output prefix in the i-PI input in order to not overwrite any output files.

When the simulation is done, plot the total energy, which is called `conserved` in the i-PI language for `NVE` simulations (Why?), vs. the simulation time in `xmgrace` (or, any other graphing platform of your choice) from the i-PI output file.

Can you see how the energy drifts with the “loose” settings? Find out what were the default criteria for these thresholds that were applied in the first simulation of this exercise (hint: you can find them in the `FHI-aims` output). How do they compare with the “loose” and “really accurate” settings, with respect to this energy drift?

Ideally, there should be no energy drift, since the energy is conserved in the micro-canonical ensemble. The reason for this drift is that we leave the true Born-Oppenheimer surface if we don’t converge well our electronic structure; this leads to an unphysical (and undesirable) energy drift. Last but not least, check the total simulation time for each run. You can find it at the end of the corresponding `FHI-aims` output file. Do you understand what you observe? Do you understand now which compromise has to be fulfilled when deciding the self-consistency convergence criteria?
Timing: ~20 minutes total

The canonical ensemble

Exercise 3: Testing thermostats

Most “real-life” experiments cannot be done in a situation where the energy is explicitly kept constant, but where other quantities like the average temperature or pressure are maintained instead. As you may know, an ensemble where the temperature is kept constant is the a canonical ensemble. In order to simulate this ensemble, the system has to be coupled to a heat bath. From a statistical mechanics point of view, the average kinetic energy in the canonical ensemble follows the equipartition theorem, which says that it is equally distributed on the various degrees of freedom of the system. Therefore, the momenta $\mathbf{p} = m\mathbf{v}$ follow the Maxwell-Boltzmann (MB) distribution:

$$P(|\mathbf{p}|) = \left(\frac{\beta}{2\pi m}\right)^{3/2} \exp\left(-\beta|\mathbf{p}|^2/(2m)\right). \tag{2}$$

The instantaneous temperature $\bar{T}$ is given by the relation $\bar{T} = \frac{2\langle K\rangle}{3k_B} = \frac{\sum_i^N |\mathbf{p}_i|^2/m_i}{3Nk_B}$, where $\langle K\rangle$ is the average kinetic energy of the system, m_i is the mass of atom i and N is the number of atoms. This means that the temperature is not constant but can (and should) fluctuate around the average value. The theoretical standard deviation is $\sigma^2 = \frac{2T^2}{3N}$. where T is the target temperature. T should also be equal to the mean of the distribution, i.e. $T = \frac{2\langle K\rangle}{3Nk_B}$, where $\langle K\rangle$ is the average value of the kinetic energy.

We will here focus on stochastic schemes to simulate thermostats in molecular dynamics. We will include a brief description of them below, as well as the idea behind other types of thermostats that will not be studied in this exercise, but that are nevertheless quite popular.

1. Extended Lagrangian approach: the Nosé-Hoover thermostat [5, 6].
   Equations of motion derived from the Lagrangian of the system conserve the total energy of the system. One can write an *extended* Lagrangian, by adding fictitious degrees of freedom, such that the overall total energy is conserved but the atomic subsystem can span ensembles other than microcanonical. With the Nosé-Hoover Lagrangian, the atomic subsystem samples the canonical ensemble. The equations of motion of the Nose-Hoover thermostat are:

$$\begin{aligned} \dot{\mathbf{r}}_i &= \mathbf{p}_i/m_i && (3)\\ \dot{\mathbf{p}}_i &= -\frac{\partial \mathscr{U}\left(\mathbf{r}^N\right)}{\partial \mathbf{r}_i} - \frac{\Pi \mathbf{p}_i}{Q} && (4)\\ \dot{\eta} &= \frac{\Pi}{Q} && (5)\\ \dot{\Pi} &= \left(\sum_i \frac{\mathbf{p}_i^2}{m_i} - \frac{g}{\beta}\right) && (6) \end{aligned}$$

   where g is the number of degrees of freedom of the system, $\mathscr{U}$ is the potential energy, Q the “thermostat mass”, and $\mathbf{p}_i$ and m_i the momenta and masses of the ith particle of the system, respectively. The conjugated momentum Π of the extra coordinate η acts as a fluctuating drag parameter to the atomic subsystem. The conserved energy associated to the equations of motion is:

$$\mathscr{E} = \sum_i \frac{\mathbf{p}_i^2}{2m_i} + \mathscr{U}\left(\mathbf{r}^N\right) + \frac{1}{2}\frac{\Pi^2}{Q} + g\frac{\eta}{\beta} \tag{7}$$

2. Stochastic velocity-rescaling thermostat (Bussi-Donadio-Parrinello)[7].

   In this algorithm, a deviation of the instantaneous kinetic energy is corrected in the following way:

$$dK = \left[\overline{K} - K(t)\right] \frac{dt}{\tau} + 2\sqrt{\frac{K(t)\overline{K}}{N_f \tau}} \xi(t) \tag{8}$$

   where $\overline{K}$ is the target kinetic energy, $K(t) = p^2(t)/2m$ is the instantaneous kinetic energy, τ is the relaxation time of the thermostat, N_f is the number of degrees of freedom, and ξ is a white noise term (the derivative of a Wiener process [1]) that obeys $\langle \xi(t)\xi(t') \rangle = \delta(t - t')$.

   In practice the trajectory is first propagated for one time step with e.g. a velocity-verlet integrator and the new velocities are calculated as usual. Then, the new kinetic energy K is evaluated and the velocities are rescaled by a factor α such that:

$$\begin{aligned}
\alpha^2 &= e^{-\Delta t/\tau} + \frac{\overline{K}}{N_f K}\left(1 - e^{-\Delta t/\tau}\right)\left(R_1^2 + \sum_{i=2}^{N_f} R_i^2\right) \\
&+ 2e^{-\Delta t/2\tau}\sqrt{\frac{\overline{K}}{N_f K}\left(1 - e^{-\Delta t/\tau}\right)} R_1
\end{aligned}$$

   where the R_i's are independent random numbers from a Gaussian distribution with unitary variance[2].

   For this thermostat a conserved pseudo-Hamiltonian $\tilde{H}(t)$ can be defined:

$$\tilde{H}(t) = H(t) - \int_0^t \left(\overline{K} - K(t')\right) \frac{dt'}{\tau} - 2\int_0^t \sqrt{\frac{K(t')\overline{K}}{N_f \tau}} \xi(t')$$

   where $H(t)$ is the total energy of the atomic system.

   The Bussi-Donadio-Parrinello thermostat yields the correct distribution of K, does not have ergodicity problems, does not perturb the dynamics, and its accuracy and efficiency is rather independent of τ.

3. Nuclear quantum effects: the colored noise thermostat.

   The colored noise thermostat is an extension of a Langevin thermostat; indeed it is also called Generalized Langevin Equation (GLE) thermostat. The classical Langevin thermostat is expressed through the following differential equation for the momentum (here in one dimension, without loss of generality):

$$\dot{p}(t) = -\gamma p(t) + \sqrt{2m\gamma T}\xi(t) \tag{9}$$

   where γ is a (friction) parameter and $\xi(t)$ is a stochastic variable distribute as a Gaussian white noise as above.

   The Langevin thermostat is constructed via a Markovian (i.e. memoryless) stochastic differential equation. Its extension, which leads to the colored noise thermostat, is constructed via introducing auxiliary degrees of freedom $\mathbf{s}$ to the dynamics. These extra degrees of freedom model a Markovian process in higher dimensions, but give

[1] An example of a Wiener process $W(t)$ is the Brownian motion (you might have heard about it...). $W(t)$ has the following characteristics: $\xi(0) = 0$; $W(t)$ is continuous; the increments are independent and $W(t_2) - W(t_1)$ is a Gaussian with average 0 and $\sigma = t_2 - t_1$.

[2] Note that $\sum_{i=2}^{N_f} R_i^2$ can be drawn directly from a suitable Gamma distribution

rise to non-Markovian dynamics when the fictitious degrees of freedom are integrated out. The equations of motion are:

$$\begin{aligned}
\dot{r} &= p/m & (10)\\
\begin{pmatrix}\dot{p}\\ \dot{\mathbf{s}}\end{pmatrix} &= \begin{pmatrix}-V'(R)\\ 0\end{pmatrix} - \mathbf{A}_p\begin{pmatrix}p\\ \mathbf{s}\end{pmatrix} + \mathbf{B}_p\left(\boldsymbol{\xi}\right), & (11)
\end{aligned}$$

where $\boldsymbol{\xi}$ is an array of uncorrelated Gaussian noises, $V'(R)$ is the gradient of the potential and the $\mathbf{A}_p$ and $\mathbf{B}_p$ are matrices that obey the relation

$$\mathbf{A}_p\mathbf{C}_p + \mathbf{C}_p\mathbf{A}_p^T = \mathbf{B}_p\mathbf{B}_p^T, \tag{12}$$

where $\mathbf{C}_p$ is the covariance matrix defined as $\mathbf{C}_p = \langle (p,\mathbf{s})^T(p,\mathbf{s})\rangle$. By integrating out the $\mathbf{s}$ degrees of freedom, one gets dynamics of a non-Markovian process in the physical variables, with the EOM given by

$$\begin{aligned}
\dot{r} &= p/m & (13)\\
\dot{p} &= -\frac{\partial V}{\partial r} - \int_{-\infty}^{t} Q(t-\tau)p(\tau) + \zeta(t), & (14)
\end{aligned}$$

where $\zeta(t)$ is a correlated noise and $Q(t-\tau)$ is a frequency dependent memory kernel which depends on $\mathbf{A}_p$. The fluctuation-dissipation theorem (FDT) (and canonical sampling) is obeyed if $\langle\zeta(t)\zeta(0)\rangle = k_BTQ(t)$. However, the FDT can be broken and one can enforce quantum statistic to some selected degrees of freedom (note: FDT is equivalent to equipartition, thus breaking FDT implies breaking equipartition). This is the way we will apply the thermostat in this exercise.

In the same spirit as the stochastic velocity rescaling thermostat, the colored noise thermostat, defines a conserved pseudo-Hamiltonian:

$$\tilde{H} = H - \sum_i \Delta K_i \tag{15}$$

where ΔK_i is the change in kinetic energy due to the action of the thermostat at the i-th time-step, and the sum is extended over the past trajectory.

Instructions part 1: Running the simualtions

- You will find some templates for the input files in the Exercise 4 folder. For this exercise we will simulate $H_5O_2^+$.
  You should use a time step of **0.5 fs**, which corresponds to the following line in the i-pi `input.xml` file,
  `<timestep units="femtosecond"> 0.5 </timestep>`
  and we will be simulating all systems at 300 K:
  `<temperature units="kelvin"> 300 </temperature>.`
  Since we are simulating a canonical ensemble, the dynamics mode will be set to `"nvt"`:
  `<dynamics mode="nvt">`

- The `control.in` file provided in this folder is tailored to run fast (and inaccurate) simulations, so that this exercise can be done within the time frame proposed here. **Please do not use this type of `control.in` file for real production calculations**.
- Two of these thermostats, namely the Stochastic Velocity Rescaling and the Langevin ones, have parameters that you can play with. In order to obtain reasonable results, one should provide an educated guess for the value of these thermostats' parameters. In order to realize to which extent such user-given parameters can influence a simulation, you are asked to try different values for these parameters, as explained below.

  1. Stochastic velocity-rescaling thermostat (svr)

     ```
     <thermostat mode="XXX">
     <tau units="femtosecond"> xxx </tau>
     </thermostat>
     ```

     Here tau (τ) is a relaxation time of the thermostat, and is given in femtoseconds. Its value has to be chosen by the user (i.e., you!). The performance of the thermostat depends on the value of τ. In order to gauge the influence of this parameter, we ask you to test two different values for τ : one value which should yield a correct behavior, for example $\tau = 20$, and one more extreme value, for example $\tau = 500$ or $\tau = 0.002$.
  2. langevin

     ```
     <thermostat mode="langevin">
     <tau units="femtosecond"> xx </tau>
     </thermostat>
     ```

     Just like for the SVR thermostat, you have to choose a value for τ.

Instructions part 2: Analysing the results

- Use the script `get_properties.py` to extract different properties, like the temperature, the kinetic energy and the potential energy:

  ```
  python get_property.py property ex4.out
  ```
  where 'property' is a placeholder for the property you want to extract. You can choose from 'temperature', 'kinetic_md', 'potential', 'conserved', 'kinetic_md(H)' and 'kinetic_md(O)'. This will output a single-column file containing the desired property without any change of units. Plot each of these quantity. Is it what you were expecting ?
- Let us analyze a bit further the output. Use the script `cumul_average.py` in order to obtain the the cumulative average, that is, the average temperature over time computed at each step of the simulation. You can do this by running the following script:

  ```
  python cumul_avg.py temperature.dat xxx
  ```
  Here we assumed that 'temperature.dat' is your single-column file containing the temperature of the system and xxx means that the first xxx data points are discarded.

Please try different xxx values and see how the cumulative average change. Do you understand why? Can we assert that the system is correctly thermalized?

How do the different thermostats you used perform? Are any of those working better or faster?

- Please repeat the previous item for the potential energy (i.e first extract the data from the `ipi-output` and then compute the cumulative average). Do you see a different behaviour? Can we assert that the system is correctly thermalized? From which point?
- You can compute the correlation function and correlation time by typing:

```
python get_corr-time.py temperature.dat xxx
```

Here we assumed again that 'temperature.dat' is your single-column file containing the temperature of the system and 'xxx' means that the first xxx data points are discarded. The script will print in the screen the correlation time and will produce a file with the correlation function.

Plot the autocorrelation function against time for the thermostats that you have been working with. What are their correlation time ? (How) does it change with respect to thermostat τ parameter ?

- Repeat the same for the potential energy, does it behave similarly or not?
- Use the script `get_avg.py` in order to obtain the average and standard deviations and then compute the average temperature and potential energy expressed as `'mean_value'` $\pm$ `'uncertainty'`. The syntax, as usual, is:

```
python get_avg.py temperature.dat xxx
```

Time estimation: ∼1h30m

References

[1] I. Newton, *Philosophiæ Naturalis Principia Mathematica*, London (1687)

[2] D. Marx, M. E. Tuckerman, J. Hutter, and M. Parrinello Nature **397**, 601 (1999)

[3] J. Kolafa, Mol. Simul. **18**, 193 (1996)

[4] Kühne, Thomas D. *et al.*, Phys. Rev. Lett. **98**, 066401 (2007)

[5] D. Frenkel and B. Smit, *Understanding Molecular Simulation: from algorithms to applications*, second edition, Academic Press 2002.

[6] S. Nosé, J. Chem. Phys. **81**, 511 (1984). W.G. Hoover, Phys Rev. A **31**, 1695 (1985)

[7] G. Bussi, D. Donadio, and M. Parrinello, J. Chem. Phys. **126**, 014101 (2007)

[8] M. Ceriotti, G. Bussi, M. Parrinello, JCTC **6**, 1170 (2010) `https://epfl-cosmo.github.io/gle4md/`

[9] G. Martyna, M. Klein, M. Tuckerman, J. Chern. Phys. **97**, 2635 (1992)

[10] M-P. Gaigeot, M. Martinez, and R. Vuilleumier, Mol. Phys. **105**, 2857 (2007)

[11] J. Borysow, M. Moraldi, and L. Frommhold, Molec. Phys. **56**, 913 (1985)

[12] R. Ramirez, T. Lopez-Ciudad, P. Kumar, and D. Marx, J. Chem. Phys. **121**, 3973 (2004)