Home Assignment 3: Na^+ solvation in water (20p)

Nicklas Österbacka
nicklas.osterbacka@chalmers.se

February 2022

In this assignment you will study the solvation of a sodium ion in water using *ab initio* molecular dynamics (AIMD) at finite temperature. Much of the focus will be on reasoning and analysis of results; running calculations is fruitless unless you take a good look at the results! Good resources for this assignment are the ASE documentation pages for molecular dynamics and the Wikipedia page on the concept of **radial distribution functions**.

You will get some first-hand experience in running AIMD with ASE and GPAW, and in particular you will see that it is relatively expensive to run. **Plan ahead. The calculations will take quite a while to run, even at this relatively modest level of theory.** (That's why we're doing LCAO with a fairly small basis set!)

Check the course Git for two relevant AIMD trajectories and the corresponding temperature at each timestep. **You do not have to write an elaborate report, but please include your scripts as well as your trajectories/logfiles when handing in on Canvas. Your answers must be clear and well-motivated.**

Task 1: Running the calculations (6.5p)

You have been given a trajectory containing snapshots from an AIMD simulation of water, as well as the temperature at each time step. Due to the simulation cell's finite size the temperature naturally fluctuates despite the use of a thermostat, but thermal equilibrium should nevertheless have been reached at some point. When does this occur in the trajectory? How can you tell?

Select an appropriate snapshot from the trajectory and insert a sodium ion into it. This can, for instance, be accomplished by using ASE's graphical user interface to visualize the trajectory, using the drop-down menu to edit the structure, and finally saving it to a file.

Run AIMD for 2 ps starting from the structure you made[1]. Use the Nosé-Hoover thermostat with a target temperature of 350 K and the PBE exchange-correlation functional. An appropriate characteristic timescale for the thermostat is 20 fs for this system; **do not forget to specify a `logfile` name to get the temperature at each timestep and attach a trajectory writer to make sure that you get a snapshot at each timestep**:

```
atoms = read('someStartConfiguration.xyz')

calc = GPAW(
            ...
            mode    = 'lcao',
            xc      = 'PBE',
            basis   = 'dzp',
            symmetry= {'point_group': False}, # Turn off point-group symmetry
```

[1]This should take up to 300 core-hours, i.e., the expect the total runtime to be up to (300 hours)/(num. cores used), assuming you save the configuration to the trajectory file every timestep, and your timestep choice is appropriate. **Please plan ahead.** While you are waiting for the calculation to finish you can start on Task 2.

```
            charge  = 1, # Charged system
            txt     = 'output.gpaw-out', # Redirects calculator output to this file!
        )

atoms.set_calculator(calc)

from ase.units import fs, kB
dyn = NPT( # Some MD method
        ...
        atoms,
        temperature_K = 350,
        timestep = 1000*fs, # This is not an appropriate timestep, I can tell you that!
        ttime = 20*fs, # Don't forget the fs!
        externalstress = 0, # We don't use the barostat, but this needs to be set anyway!
        logfile = 'mdOutput.log', # Outputs temperature (and more) to file at each timestep
    )

trajectory = Trajectory('someDynamics.traj', 'w', atoms)
dyn.attach(trajectory.write, interval=1) # Write the current positions etc. to file each
    timestep
dyn.run(10) # Run 10 steps of MD simulation
```

Use GPAW's LCAO mode with a *dzp* basis set and the PBE exchange-correlation functional. Turn off the use of point-group symmetries by including `symmetry = {'point_group': False}` in GPAW's input[2]. The default parameters for the other settings are sufficient for our purposes, but **do not forget to specify the system's total charge**.

The Nosé-Hoover thermostat is implemented in ASE as a special case of the NPT ensemble. To use it, use the NPT MD class and set `pfactor` *to* `None` *and* `externalstress` *to 0, giving us NVT dynamics.*

In addition, please answer the following questions in the report:

- Care should be taken when placing the ion, but why? What would happen if you placed it too close to a water molecule? (I would not recommend trying it out; intuition should be enough here!)
- What is an appropriate timestep for MD simulations for this system? **Motivate.**
- Very briefly explain how the Nosé-Hoover thermostat keeps the temperature constant in an MD simulation. *(There are many thermostats; make sure that your explanation describes the correct one!)*
- The Nosé-Hoover thermostat is not actually ergodic. Explain what the term "ergodic" means in the context of molecular dynamics and why it's an important concept.
- Despite this, the Nosé-Hoover thermostat is considered to be an excellent choice for simulating systems at finite temperature. How is the ergodicity problem solved?
- DFT gives the energy as a function of the density, but we need *forces* to do MD. How do we calculate forces within the framework of DFT?
- What else could be used than DFT for force calculations? Give an example! What are some of the advantages/disadvantages compared to DFT?

Task 2: Analysis of results (7.5p)

An important concept in statistical physics and atomistic materials science is the concept of the radial distribution function (RDF), the definition of which you have surely seen during another course. (If you need a refresher on the concept, its Wikipedia page is excellent!) To analyze your simulation and determine the ion's solvation shell you need to compute the RDF $g_{\alpha\beta}(\mathbf{r})$ between the sodium ion and the water molecules. Write a Python script that calculates this. For multi-species simulations like this one, the total RDF is not

[2]Most DFT codes use symmetries to reduce the computational cost of simulations. GPAW will crash with an error if some atomic configuration violates point-group symmetry, which is clearly a problem if it occurs in the middle of an MD simulation...

necessarily appropriate; summing only over the relevant atoms is oftentimes better. **Explain the algorithm, preferably with the relevant code snippet inserted in the report.**

You are of course allowed to use Numpy, Scipy, and ASE here, but ***not*** *any ready-made implementations of the RDF in ASE. The only thing you really need from each snapshot is the distance between the relevant species, which can be extracted easily with the* `ase.Atoms.get_distances` *method.* Do *not* forget to take periodic boundary conditions into account!

Hint: What are the relevant species here? How are the water molecules distributed around the ion?

Was the 2 ps simulation enough to establish thermal equilibrium?

The water molecules are oriented in a certain manner, both around the ion and far away from it. Explain why!

The (first) solvation shell of a solvated ion is defined as the integral of the RDF between the ion and water molecules up to the first minimum. Calculate this for both your own trajectory and for the longer one provided by us, which you may find on the course Git. Was the short simulation enough to capture the solvation? (You should calculate this only for the equilibrated part of the trajectory; if yours was not equilibrated, calculate and compare it anyway.)

The experimental value of this integral is around 5. Do the simulation results agree well with this? What does this number tell us about the solvated ion?

Task 3: Reflection and contemplation (6p)

Figure 1 shows the RDF between oxygen atoms in a water AIMD simulation performed with different exchange-correlation functionals as well as from experiments. Based on this figure, which functional do you reckon is best-suited for simulating water?

Compute the RDF between oxygen atoms after thermal equilibrium for the (sodium-free) water trajectory you have been given. How well does it agree with the experimental curves in Figure 1? Does it exhibit quantitative or qualitative agreement/discrepancy?

Balancing physical validity with computational cost is key to any computational science. The dynamics with 24 molecules (hopefully!) gives a reasonable value for the coordination number, but reducing the number of water molecules to e.g. 7 would make the simulations much cheaper. Would this smaller simulation cell capture the same physics? Why? **Clear motivation is required for full points.**

Even if we consider an infinitely large system, there is no guarantee that our computational results would agree with experiments. What theoretical approximations are made in the variant of DFT used here? What numerical approximations have been made? How can we improve upon these approximations?

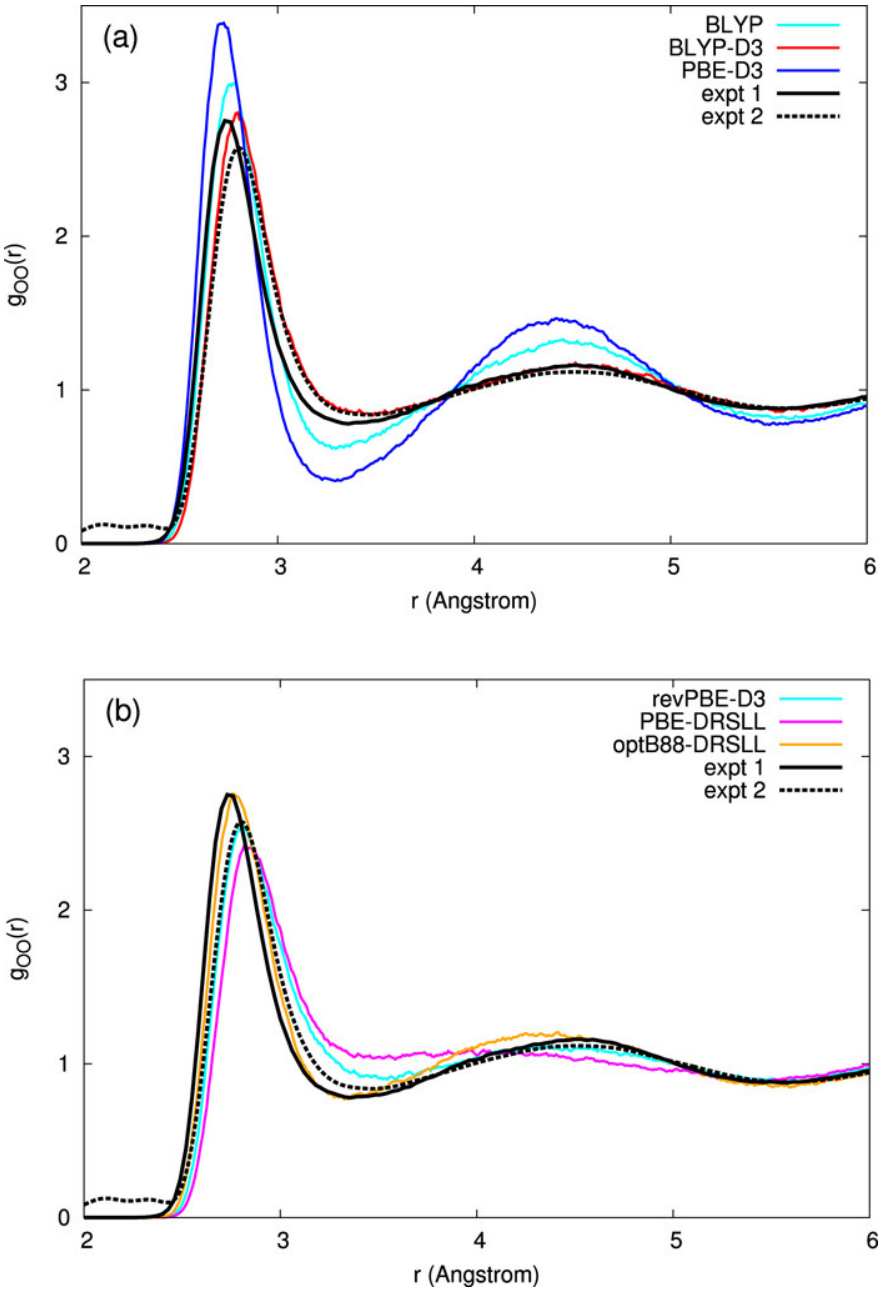

Figure 1: The RDF between oxygen atoms in water. From `https://doi.org/10.1063/1.4944633`.