Tutorial for Using Integration of Density of States

Johannes Dietschreit and Laurens Peters
Ochsenfeld Group
University of Munich (LMU), Butenandtstr. 5–13, D-81377 München, Germany

A short molecular mechanics based example on how to use the IDS formalism published in Refs. [1] and [2] is given. We will simulate the vibrational free energy difference between capped serine and cysteine in solution.

1 Dependencies

It is important that you have a working version of the python libraries `numpy` [3, 4], `scipy` [5], and `MDAnalysis` [6, 7]. You will also need an molecular dynamics code. In the following, we assume that you have a working version of Gromacs [8–12], but any other MD code will do as well, like NAMD [13]. Visualization can be done with many packages, we will use VMD [14] as example.

2 Folder Structure

The tutorial contains three folders. One is named `python_scripts` which contains all the necessary scripts for the analysis after the MD simulations. The folders `serine` and `cysteine` contain the pdb of the respective molecule (the amino acids are capped at both ends), the input for minimization, and the input for the production run. We skip the equilibration for convenience, as there is no need for that in such a small system.

3 Preparing and Executing the small MD simulations

The next steps will be carried out explicitly for serine, please change the corresponding file names for cysteine.

1. Conversion to `gro`-format and selecting force field and water model. We selected AMBER99SB-ILDN [15] and SPC/E [16], but any other combination will work as well. NB: The final result will depend on your choice.

   ```
   gmx pdb2gmx -f serine.pdb -o serine.gro
   ```

2. Setting up the solvation box

   ```
   gmx editconf -f serine.gro -o box.gro -c -d 1.0 -bt octahedron
   ```

3. Filling the solvation box with water

   ```
   gmx solvate -cp box.gro -cs spc216.gro -o solvated.gro -p topol.top
   ```

4. Preparing minimization

```
mkdir mini
cp topol.top solvated.gro em.mdp mini/
cd mini/
gmx grompp -f em.mdp -p topol.top -c solvated.gro -o em.tpr
```

5. Running the minimization

```
gmx mdrun -v -deffnm em
```

6. Preparing the production run

```
cd ../
mkdir run/
cp mini/em.gro mini/topol.top run.mdp run/
cd run/
gmx grompp -f run.mdp -p topol.top -c em.gro -o run.tpr
```

7. Executing the production run

```
gmx mdrun -v -deffnm run
```

4 Carrying out the IDS analysis

We assume that the MDs have been carried out successfully. The file `run.trr` contains the coordinates and velocities that we need. One can look at these files with for example `gmx dump -f run.trr | less`.

First we will remove any overall translation and rotation the peptide has during the simulation and we will convert the binary file into textfiles for each atom that only contain the altered velocities. At the same time the velocity unit will be changed to what we have used originally in our quantum chemistry code ($\sqrt{E_h/amu}$). This is done by the `read_traj.py` script. Then we will calculate the velocity power spectrum and integrate over them. We assume that we are again in the folder serine. If you want to use a different folder structure please adjust the commands accordingly.

1. **Getting velocities per atom**. This might take a while as processing the `run.trr` is relatively slow. The `read_traj.py` script also works with dcd, then one needs to supply a coordinate and a velocity dcd, as they are created for example by NAMD.

   ```
   export PATH=$PATH:../python_scripts/
   mkdir vel
   cd vel/
   read_traj.py -t gro -c ../run/run.trr -p ../serine.gro -a 23
   ```

   `read_traj.py` reads in the first `-a` atoms of the file, `-t` sets the type of simulation engine that was used (either "gro" or "namd"), `-p` the parameter file (a `.gro` for Gromacs and a `.parm7`/`.parmtop` for NAMD), `-c` the `.trr` file or the coordinate-dcd, and `-v` the velocity-dcd in the case of NAMD.

2. **Calculating the spectra.** We use a damping to soften the spectrum, which is controlled with the `-d` and `-f` flags.

   ```
   cd ../
   mkdir spec
   cd spec/
   for atom in $(ls ../vel/); do SpecTraj.py -d true -f 1000 -t power ../vel/$atom >
   $atom; done
   ```

3. **Calculating the free energy differences.** Here we assume that the steps above have been carried out for both serine and cysteine. And we are now in the cysteine folder as we compute the reaction from serine to cysteine.

```
mkdir diff
cd diff/
for atom in $(ls ../spec/); do ThermoTraj.py -b false -e 1 -p 1
../../serine/spec/$atom ../spec/$atom > $atom; done
```

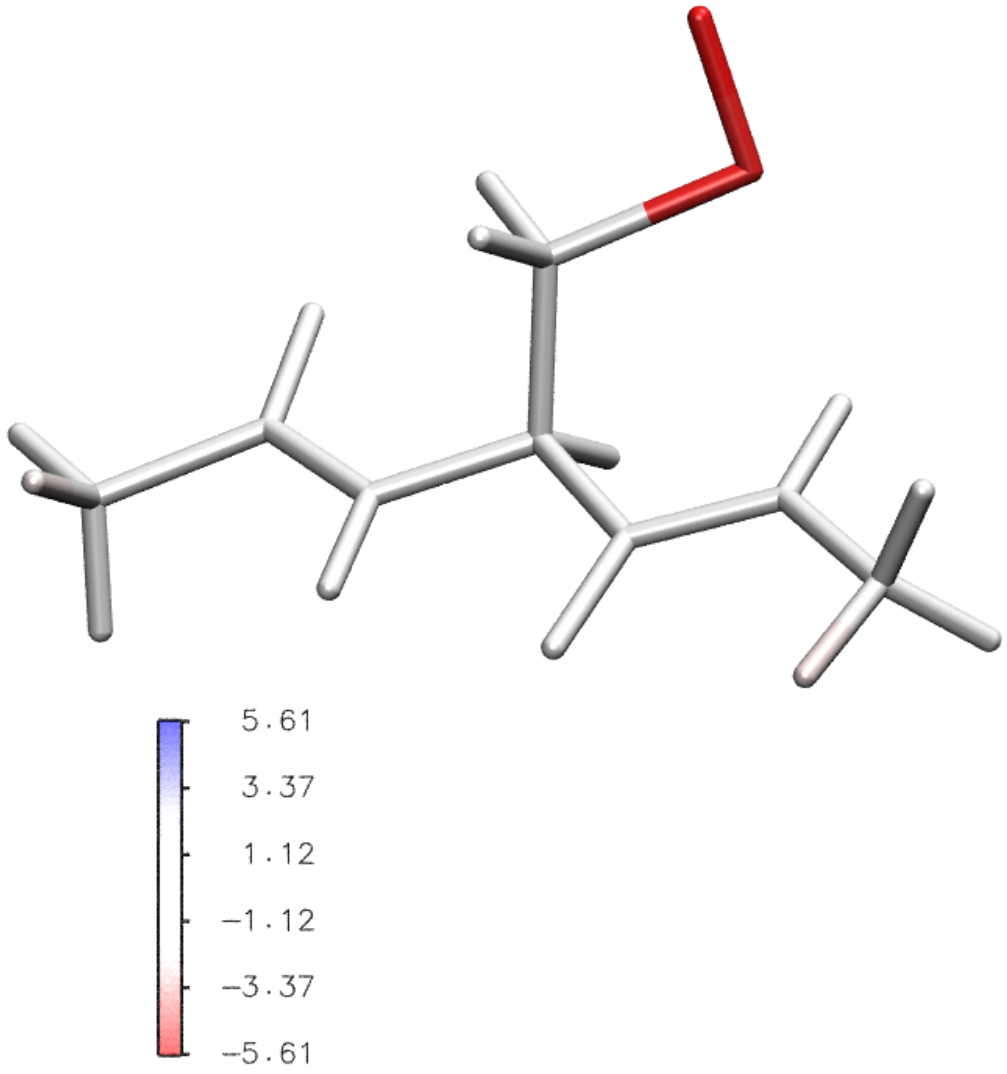
```
grep 'A(classic)' atom_*.dat > dA_list.dat
```

We have now a file that contains all the vibrational free energy differences per atom weighted with the classical weighting function.

4. Preparing visual inspection of results.

```
make_pdb.py -A dA_list.dat -o dA_sercys.pdb -s ../cysteine.pdb
```

5. Display results with VMD. The file `dA_sercys.pdb` can be opened with VMD and if the colouring method is chosen to be Occupancy, the atoms are coloured by their vibrational free energy difference (`Graphics -> Representations -> Coloring Method -> Occupancy`). For our picture we have added a fake atom that has an occupancy value such that the scale becomes symmetric around zero. The color scale is available via `Extensions -> Visualization -> Color Scale Bar`. Your result should look similar to this:



References

[1] L. D. M. Peters, J. C. B. Dietschreit, J. Kussmann, C. Ochsenfeld, *J. Chem. Phys. submitted* **2018**.

[2] J. C. B. Dietschreit, L. D. M. Peters, J. Kussmann, C. Ochsenfeld, *J. Phys Chem. Lett. submitted* **2018**.

[3] P. F. Dubois, K. Hinsen, J. Hugunin, *Computers in Physics* **May 1996**, *10*.

[4] P. F. Dubois, *Computing Science and Engineering* **Sept. 1999**, *1*, 66–73.

[5] E. Jones, T. Oliphant, P. Peterson, et al., SciPy: Open source scientific tools for Python, [Online; accessed ¡today¿], **2001**–.

[6] R. J. Gowers, M. Linke, J. Barnoud, T. J. E. Reddy, M. N. Melo, S. L. Seyler, D. L. Dotson, J. Domanski, S. Buchoux, I. M. Kenney, O. Beckstein in Proceedings of the 15th Python in Science Conference, (Eds.: S. Benthall, S. Rostrup), SciPy, **2016**, Chapter MDAnalysis: A Python package for the rapid analysis of molecular dynamics simulations, pp. 102–109.

[7] N. Michaud-Agrawal, E. J. Denning, T. B. Woolf, O. Beckstein, *J. Comput. Chem.* **2011**, *32*, 2319–2327.

[8] H. J. C. Berendsen, D. Van Der Spoel, R. van Drunen, *Computer Physics Communications* **1995**, *91*, 43–56.

[9] D. Van Der Spoel, E. Lindahl, B. Hess, G. Groenhof, A. E. Mark, H. J. C. Berendsen, *J Comput Chem* **Dec. 2005**, *26*, 1701–18.

[10] E. Lindahl, B. Hess, D. Van Der Spoel, *Molecular modeling annual* **2001**, *7*, 306–317.

[11] B. Hess, C. Kutzner, D. van der Spoel, E. Lindahl, *J Chem Theory Comput* **Mar. 2008**, *4*, 435–47.

[12] S. Pronk, S. Pall, R. Schulz, P. Larsson, P. Bjelkmar, R. Apostolov, M. R. Shirts, J. C. Smith, P. M. Kasson, D. van der Spoel, B. Hess, E. Lindahl, *Bioinformatics* **2013**, *29*, 845–854.

[13] J. C. Phillips, R. Braun, W. Wang, J. Gumbart, E. Tajkhorshid, E. Villa, C. Chipot, R. D. Skeel, L. Kale, K. Schulten, *J. Comput. Chem.* **2005**, *26*, 1781–1802.

[14] W. Humphrey, A. Dalke, K. Schulten, *J. Molec. Graphics* **1996**, *14*, 33–38.

[15] R. B. Best, G. Hummer, *J. Phys. Chem. B* **2009**, *113*, 9004.

[16] H. J. C. Berendsen, J. R. Grigera, T. P. Straatsma, *J. Phys. Chem.* **1987**, *91*, 6269?6271.