Introduction into molecular mechanics and molecular dynamics simulations

Sabine Reißer and Tomáš Kubař
(Dated: July 19, 2021 at 11:59am)

CONTENTS

I. INTRODUCTION

A. Linux

We work under Linux, a free operating system that is less common than Windows though, but widely used for scientific applications. In contrast to the well-known mouse-controlled operation of desktop PCs, Linux is a command-based system, that is, users control the computer mainly via a text console. This is much more efficient and faster, but requires some practice. Various information and tutorials on Linux/Unix are available on the Internet, Therefore, the following is given only a minimal overview of the most common commands that are needed in the course of the tutorial.

In the Linux shell, you may first practice the most basic Unix commands, for example:

- `whoami` – Displays the username under which you are logged in, that is, *student* at the moment. Different users have different permissions and access to system resources which we should not detail here.
- `pwd` – Displays the current directory, at the moment the home directory, in which you have full permissions to read and write and where you should store all files for the tutorial.
- `date` – gives time and date
- `ls` – displays the contents of a directory. Your home directory should be empty at the moment (or contain files from the previous users). There are several modes of display for `ls`, and they may be selected with so-called switches. For example, `ls -l` outputs additional information about files, such as user rights, or `ls -ltrh` sorts the files so that the last created file is at the bottom of the list.
- `man command` – Displays the manual for a specific command. If you want to know which options (switches) a particular command accepts, you can learn this with `man`.
- `mkdir name` – Creates a directory. In your home directory for the exercises, create a subdirectory called exercises.
- `cd name` – Switches the working directory. Switch to the directory just created with `cd exercises`. The name of the destination directory can be specified relative to the home directory, or using the complete path, like `cd /home/student/exercises`. With `cd ..` you may change to the directory above the current directory. With `cd` without any name, change back to your home directory, while `cd -` returns to the previous directory. A single dot `.` stands for the current directory.
- `cp oldname newname-or-location`, `mv oldname newname-or-location`, `rm name` – File operations to copy, move (rename) or delete files; note that renaming and moving a file or directory is the same thing. `cp name1 name2` creates a copy of the file name1 under name2. You can also specify different directories. Use `mv name1 name2` to move or rename a file; `name2` is either the destination directory or the new file name. Files may be deleted with `rm name`, with caution because delete has no security question and cannot be undone! Complete directories can be deleted with `rm -r folder_name` – this is a quite dangerous command.
- `tar` – Compress, uncompress and archive files. With this command, files can be packed into archives and compressed, for example to make it easier to give to somebody else, or to make backup copies.
- `cat`, `more`, `less`, `head`, `tail name` – Viewing text files. `cat` outputs the entire file to the screen at once; for longer files, you can use `more` or `less`, which allows page-by-page scrolling. `head` and `tail` output the first and last ten (or another number of) lines of the file, respectively.
- `wc name` – Counting the lines, words, and characters in a text file.
- `grep` – Browse files. With `grep pattern file` the file is searched for lines in which the pattern (sequence of characters) occurs. For example,

  ```
  grep HA /usr/local/run/gromacs-5.0-dftb-v6a-plumed/share/gromacs/top/*/ffbonded.itp
  ```

  searches all lines of all force field files for binding parameters that relate to a particular type of hydrogen atom.
- `gedit`, `vim`, `nano`, `pico`, `emacs` – Text editors. A lot of text editors are available to edit the content of text files. As part of the exercises, it is sufficient to use the simple editor `gedit`, for example. Those who want to continue working on Linux systems should, sooner or later, become familiar with one of the more powerful editors like `emacs` or `vim/gvim`. Try out a text editor with the PDB file `1HSG.pdb`.

- `alias name = 'command'` – Assigns an abbreviation to a command. For example, you could define a shortcut for the detailed file list command as `alias ll = 'ls -ltrh'`
- `history` – Command history. You can also use the ↑ key to return to previously typed commands.
- With Ctrl-Shift-T, you can open a new tab in the shell. This is handy when a simulation is running and you want to continue working on another task.
- Also very convenient is the possibility to select text with the left mouse button, and then paste it with the middle mouse button (push the wheel if there is no middle button). This is often faster than the Windows-like sequence of Ctrl-C Ctrl-V, which works in modern Linux systems also.
- Use Ctrl-R to find a pattern in the command history.
- Tab allows you to automatically complete file names or commands.
- `mc` – Midnight Commander, a very handy file manager similar to Norton Commander or Total Commander. `mc` contains a simple text editor that can be accessed by pushing F4. We would like to recommend using `mc` as often as possible.
- `>` – The 'greater than' character directs the output from a program or command to a file. If you want to save the contents of the current directory in a text file, do so with `ls > what-is-here.txt`
- `|` – The vertical bar directs the output to the standard input of another program or command. This so-called pipe can also be used several times on the command line. For example, you can count how many times you logged on to the machine as a user 'student': `last | grep "student" | wc -l`

If you NEVER worked under Linux before, take your time – even an hour or so – and go through a little online course, for example under http://www.ernstlx.com/linux90bash1.xhtml. This will make it much easier for you to do complete our tutorial.

B. Procedure

All simulations will be performed on our computer cluster (172.22.90.209) The first task is to login to the cluster:

```
ssh -X tutorial@172.22.90.209
```

with 'tutorial' as password.

Next, login to one of the computing nodes which are reserved for practical course by typing:

```
loginX
```

where X is your group number. All participants in group 1 use the computing node 'gtx03' which has 40 physical cores and one graphic card, group 2 uses 'gtx04' with the same specifications. Once logged in on the computing node type `bash` and change to the working directory:

```
cd /scratch/tutorial
```

and create a new working directory with your name and switch to it

```
mkdir NAME
cd NAME
```

Copy the files for the part you are working on to your directory; note, the trailing dot is a part of the command:

```
cp -r /data/dmaag/Praktikum/part* .
```

In case that, at some point during the tutorial, you want to copy files between the server and your local machine, feel free to use the following command on the local machine:

```
scp tutorial@172.22.90.209:/scratch/gtx0X/tutorial/NAME/file your/machine/
```

C. Visualization with VMD

One of the most important techniques for molecular simulations is the visualization of structures and trajectories. We will use the molecular viewing program VMD, which is freely available under `http://www.ks.uiuc.edu/Research/vmd`. Before we build our own molecules, let us first look at an X-ray crystal structure that somebody has created and published. First, start the program from the shell

```
vmd
```

A viewing and a control window will open. Open the file browser under *File – New Molecule*. In the newly opened window select *Determine file type – Web PDB Download* and load the X-ray crystal structure file 1HEL – or another one, like for example a structure of your favorite protein. Thereafter, the crystal structure is displayed in the viewing window. Test the different functions of VMD with this example:

- Rotate, move and scale: With the left or right mouse button pushed down, the molecule can be turned. Press the `t` key to enter the translation mode, `s` for the scaling mode and `r` return into the rotation mode. `=` will reset the view.
- Selection of parts of the molecule: Open the viewport under *Graphics – Representations*. In the line *Selected Atoms* try different selections: "water", "backbone", "resid 1 to 10", "resname CYS", etc. Finish by selecting only the two cysteines number 30 and 115.
- Names and internal coordinates: Use the keys `1`, `2`, `3`, `4` to select and obtain information about single atoms, distances, angles and dihedral angles. Press `1` and select one of the sulfur atoms. The console displays some information about this atom. Continue to measure the distance (with `2`) between the sulfur atoms, the two S–S–Cβ angles (mode `3`) and the Cβ–S–S–Cβ dihedrals (mode `4`). If the screen is crowded with labels, you can delete them under *Graphics – Labels*.
- Various display options: Reset the selection to "all". Now choose "Secondary Structure" and "NewCartoon" as the color and drawing style. Choose a few more ways to represent the molecule and generate a view of your choice – for example, one in which the disulfide bridges are visible easily, or one in which the distribution of charged residues in the protein becomes apparent.

For a more detailed description of the (many different) display options of VMD, continue experimenting or check out the User Guide. VMD may be used create high quality images of molecular structure for publications via the *File – Render* menu. We will make use of the functionalities of VMD in the tutorial repeatedly, in order to visualize trajectories and compare structures.

D. Creating Molecules

In order to be able to even start geometry optimizations or molecular dynamics simulations, you essentially need two things: initial geometries and force field parameters. Geometries are usually derived from experimental data, e.g. X-ray or NMR structures, or from (coarse) modeling techniques. Force field parameters are usually part of the modeling package used, but they may be created manually or additionally in principle. For the preparation of the input files, we use two programs from the Amber simulation package: Leap and Antechamber. The latter serves to generate new molecules and adjust parameters, and is not used in this tutorial. We will use the graphical version of the former (xLeap) to build a peptide and save it as a PDB file:

```
xleap
```

First, load one of the Amber force fields (an improved version of the Amber99 force field) and look at some of the predefined units in it. (So-called units are atoms, molecules or parts of molecules, as well as parameters.)

```
source leaprc.ff14SB
list
desc ALA
desc ALA.1
desc ALA.1.3
edit ALA
edit TIP3PBOX
```

Practice viewing, selecting and editing molecules and atoms with the "edit" window of xLeap. The program provides various ways to create solvated and periodic systems. At the moment, however, we limit ourselves to isolated molecules.

For the following calculations, you now create the peptide WALP19 as a new unit `peptide` with the command

```
source leaprc.ff14SB
peptide = sequence { ACE GLY TRP TRP LEU ALA LEU ALA LEU ALA
LEU ALA LEU ALA LEU ALA LEU TRP TRP ALA NME }
```

In principle, one could also draw the molecule by hand into an empty unit and then define all atomic types and charges, but the prefabricated amino acid residues from the force field, of course, facilitate this work enormously. In addition, the peptide would have to be amidated at the C-terminus to obtain the genuine WALP19 peptide; we will not do it for the sake of simplicity.

As a starting structure for the simulations, we want to work with an idealized α-helix, therefore, the peptide must be folded accordingly. The secondary structure of a peptide is determined by the values of the dihedral angles φ and ψ, which are relatively flexible, in contrast to the rigid peptide bond connecting amino acid residues. Thus, different peptide conformations possess different values of φ and ψ; φ describes the rotation along the N–Cα bond (dihedral C(previous AA)–N–Cα–C), and ψ represents the rotational state on the bond Cα–C (dihedral N–Cα–C–N(following AA)).

We set the dihedral angles of all amino acids to values correspnding to an idealized α helix; find the right values in a book or in the internet. Then, complete the following command with the values you found:

```
impose peptide {1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21} {
{"N" "CA" "C" "N" value-for-psi}
{"C" "N" "CA" "C" value-for-phi}
}
```

WALP19 should be α-helical now. Save the structure with

```
savepdb peptide walp19.pdb
```

You can also save a series of commands in a text file for later reuse:

```
xleap -f filename
```

An example of this is the input file `leap.in`.

Further possibilities of xLeap can be found in the User Guide `Amber-Tools.pdf`.

Now look at the peptide with VMD and see if everything looks as expected. Also, view the file itself with a text editor of your choice. Try to answer the following questions:

- Which column means what?
- In which unit are the coordinates given?
- What does it mean that the peptide would have to be amidated at the C-terminus? What does the C-terminus look like in our simulations really?

E. Parametrization with pdb2gmx

The PDB file that we created contains only the structure of the molecule, but no atomic charges, masses, information on bonds etc. For the parameterization as well as for all further steps, we will use the MD simulation program Gromacs. Just like all other previously used programs, Gromacs is *open source* – that is, the source code is freely available and anyone can change it according to their needs.

We now use the Gromacs tool `pdb2gmx` to assign force field parameters:

```
gmx pdb2gmx -f walp19.pdb -ignh -o walp19.gro
```

We use the Amber99SB-ILDN force field and do not need water at first. The output file `walp19.gro` is a structure converted from the PDB file into Gromacs format. A look inside shows that the formats are in fact very similar.

More interesting is the file `topol.top`, which contains all force field parameters and the topology, i.e. the list of atoms and of the bonds between them. Lines with a semicolon `;` at the beginning are comments, typically some information for the user, and are ignored by Gromacs.

In this file the molecule types are defined under `[ moleculetype ]`, and this is the peptide K2 in our case. The atoms in the molecule are listed under `[ atoms ]`, in the order of the individual side chains. For each atom, there is the atomic type, the charge and the mass in columns 2, 7 and 8, respectively.

The bonds between atoms are defined under `[ bonds ]`. The first two numbers represent the indices of the atoms as defined under `[ atoms ]`. The force constants and equilibrium distances of bonds are defined in the force field, which was included at the beginning of the file:

```
; Include forcefield parameters
#include "amber99sb-ildn.ff/forcefield.itp"
```

The path may be given from the current working directory, or from a standard Gromacs directory, which is typically something like `/usr/local/run/gromacs-.../share/gromacs/top`. There you will find all force fields and the corresponding parameters. At the very bottom of the topology file (in the viewer program less, jump to the end of the file with `G`) we now find the composition of the system called "protein". For now, it only contains one molecule with the name "protein" under `[ molecules ]`.

II. MOLECULAR DYNAMICS SIMULATION

A. Simulate a peptide in water

In contrast to geometry optimization, a molecular dynamics simulation (MD) provides the temporal evolution of the molecular system, as realistically as possible. MD simulations tend to be computationally intensive and time consuming in actual use. Therefore, we will simulate only short time periods; longer trajectories for analysis can be found in the materials directory.

Create a folder `part2`. Copy the structure and topology files of the peptide K2 from the last exercise into this directory. In the following, you will place K2 into a water box, heat it up, equilibrate the system and start a short simulation.

In Gromacs, all simulations are prepared with the preprocessor `grompp` and executed with the program `mdrun`. `grompp` requires various input files that are assigned via switches. The most important are:

Switch	Filename suffix	What is it?
`-f`	`.mdp`	Simulation parameters
`-c`	`.gro`	Initial structure
`-p`	`.top`	Topology
`-n`	`.ndx`	Index file
`-o`	`.tpr`	Output – assembled run input file for `mdrun`

In the `.mdp` file, all variable simulation parameters are set. The index file with the suffix `.ndx` defines various groups of atoms within the simulated system. These groups may serve various purposes, and we will need them later when working with water and membranes. The switch `-v` stands for "verbose", and makes Gromacs print additional information on the screen; This switch is available for all Gromacs programs. Then, the `.tpr` file is passed to `mdrun` using the `-s` option. Alternatively, the `mdrun` option `-deffnm` (for "default filename") sets the same base name for the `.tpr` file as well as all output files. Note: All Gromacs programs understand the switch `-h` to display the help. A description and specification of the input and output files and of all options to the program are provided.

1. Solvation

First, place the peptide in a cubic simulation box:

```
gmx editconf -f walp19.gro -o walp19_newbox.gro -c -d 1.0 -bt cubic
```

See `gmx editconf -h` for the options you are using. Take a look at the new structure with VMD. When you type `pbc box` in the VMD console, you will see the edges of the box that was created.

Now fill the box with water. We use the TIP3P water model. The many different water models available can be found at `en.wikipedia.org/wiki/Water_model`.

```
gmx solvate -cp walp19_newbox.gro -p topol.top -cs spc216.gro -o walp19_sol.gro
```

See also the options under `gmx solvate -h`. The used water box SPC216 contains 216 water molecules and is equilibrated with the SPC model (the models SPC and TIP3P are very similar). The program tries to accommodate as many as possible of these small boxes in the peptide box. Look at the solvated box again with VMD.

If you now print the end of the topology with `tail topol.top`, you may find the number of water molecules added under `[ molecules ]`. At this point, we need to add the definition of the TIP3P water molecule to the topology. For this, use a text editor to add these three lines just above `[ system ]`:

```
;Include water topology
#include "amber99sb-ildn.ff/tip3p.itp"
#include "amber99sb-ildn.ff/ions.itp"
```

2. Equilibration

To prepare the system for a production simulation, a few equilibration steps are now required. First, perform an energy minimization. Use the file `em_steep.mdp` (has not been used before). You can execute the following commands:

```
gmx grompp -f em_steep.mdp -c walp19_sol.gro -p topol.top -o em.tpr
mdrunX -v -deffnm em
```

where X is your assigned GTX ID (1, 2, 3, or 4).

Such an energy-minimized structure can be heated up at a constant volume (NVT simulation). Prior to doing so, the peptide shall be restrained to its initial position, so that only the water molecules can move around it, and the sensitive peptide structure is not disturbed. To keep the peptide in place, introduce so-called position restraints for each peptide atom with `genrestr`:

```
gmx genrestr -f em.gro
```

Here you can consider the "heavy atoms" of the peptide, in other words everything except hydrogen atoms, which is the group "protein-H". As you can see in the newly created file `posre.itp`, the position of each heavy atom of the peptide will be restrained to its initial position, by means of a harmonic potential (elastic spring) with a force constant of 1000 kJ/(mol·nm^2). For more explanation, see Gromacs manual, section 4.3.1.

Now, read the comments in the file `nvt.mdp`. Here is some additional information:

- `define = -DPOSRES`
  From now on, the variable POSRES is defined, in order to switch on the position restraints that you have created. Look in the file `topol.top` for "POSRES": the conditional statement `ifdef` decides that if the variable is set, then the file `posre.itp` is included, and that means that the position restraints are working. At a later point, when you will be running an unrestrained MD simulation, you have to "comment out" or deleted the line `define = -DPOSRES` in the corresponding `.mdp` file.

- `gen_temp = 10.`
  At the beginning of the simulation, all atoms will be assigned stochastic (random) velocities. Their distribution corresponds to the Maxwell–Boltzmann statistics for the desired temperature. At 10 K, all of the atoms have low speed, and there is no atomic movement in awkward directions.

- `ref_t = 300. 300.`
  During the simulation, the velocities of atoms will be controlled by a thermostat so that their distribution remains close to the Maxwell–Boltzmann distribution for the desired temperature `ref_t`.

Like before, call the preprocessor and perform the actual simulation:

```
gmx grompp -f nvt.mdp -p topol.top -c em.gro -r em.gro -o nvt.tpr
mdrunX -v -deffnm nvt
```

As you see on the screen, the NVT equilibration may take around a minute.

Now use `gmx energy` and Xmgrace again, to inspect the temporal course of the temperature (similar to section I.E). Your plot should look something like Fig. 1 (left). You can see that the temperature is very low at 0 ps. The reason for this is the allocation of velocities according to the Boltzmann distribution for 10 K. The thermostat then heats the system up to the target temperature of 300 K during the first ca. 2 ps.

Use `gmx energy` to plot the potential and kinetic energy, which will look similar to Fig. 1 (right). After a small dip, the potential energy increases, thus illustrating the fact that the molecular system is no longer exactly in the energy minimum during the simulation. The shape of the kinetic energy dependence is the same as that of temperature. (Why?)

Next, the system has to be brought to the desired pressure of 1 bar. In this equilibration step, the pressure and the temperature are kept constant, therefore it is also called NPT equilibration.

Look at the file `npt.mdp`. The peptide is still treated with position restraints. The line

```
continuation = yes
```

specifies that the final velocities from the NVT equilibration step shall be used as the initial velocities in the NPT equilibration. There is a new command block to control the pressure:

```
pcoupl           = Parrinello-Rahman
pcoupltype       = isotropic          ; uniform in x,y,z directions
tau_p            = 0.5                ; relaxation time in ps
ref_p            = 1.0                ; desired pressure in bar
compressibility  = 4.5e-5             ; value for water in 1/bar
refcoord_scaling = com
```

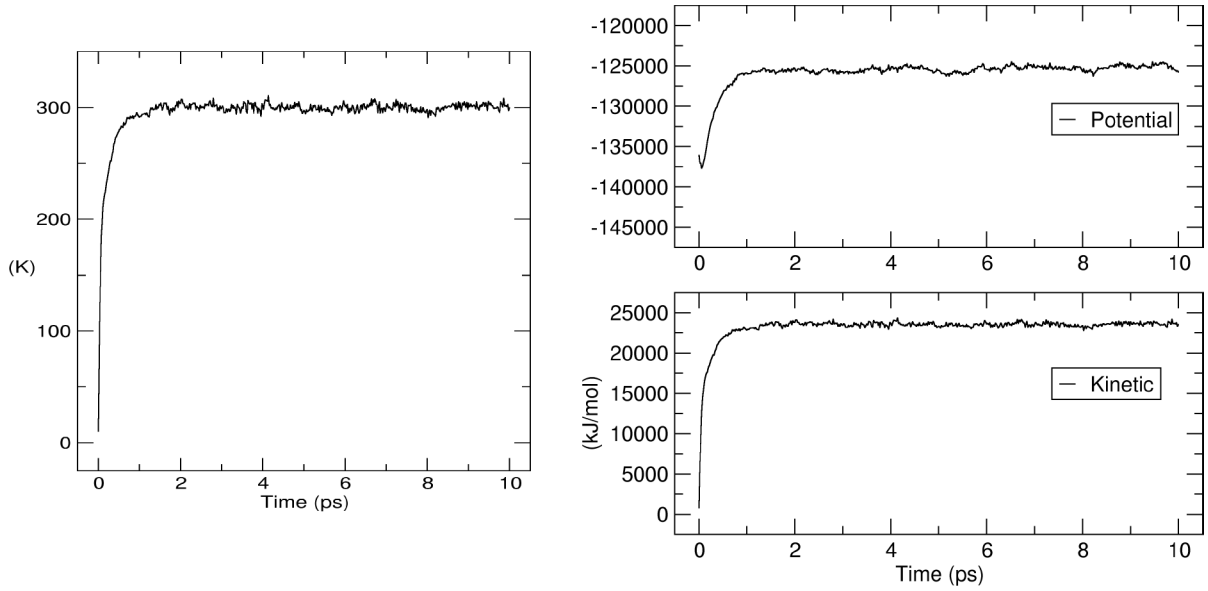

FIG. 1. NVT equilibration: The time course of temperature (left) and those of potential and kinetic energy (right).

To perform the NPT equilibration, use `grompp` and `mdrun` again:

```
gmx grompp -f npt.mdp -c nvt.gro -r nvt.gro -p topol.top -o npt.tpr
mdrunX -v -deffnm npt
```

This 100 ps long equilibration will take about 5 minutes. Then look again at `gmx energy` and Xmgrace for the evolution of temperature and density. (Or alternatively the volume. The information is the same as the density, which may be more informative being an intensive quantity.) Is the system equilibrated sufficiently? What is the magnitude of fluctuation of temperature and density?

3. Production simulation

Now that you we have equilibrated the system properly, you may start an MD simulation for actual data production. The peptide shall no longer be restrained in that simulation. Look at the corresponding input file `md.mdp`. What has changed? How do you see that the peptide can now move freely? What time span is simulated (how many ps or ns)? Use `npt.gro` as the initial structure and start the simulation. The simulation will take about an hour.

B. Visualization and analysis with VMD

Now use VMD to visualize the trajectory. The easiest way is to pass first the initial structure and then the trajectory to VMD:

```
vmd npt.gro md.xtc
```

Experiment with the trajectory player in the "VMD Main" window.

First, hide the water molecules. The dynamics of the peptide is initially difficult to track because the rotation of the entire molecule obscures any conformational changes. This can be fixed in VMD using the *RMSD tool*. Under *Extensions – Analysis*, select the *RMSD Trajectory Tool* window. Select the peptide (with "protein") and choose *Align*. With that, the overall translation and rotation should be removed. Also, save the time series of RMSD in a file and plot it (you probably need to remove the first two lines of the file to do so).

Next, visually follow the changes of the dihedral angles φ and ψ in the peptide. (These dihedrals determine the secondary structure; φ is C(–1)–N–Cα–C, and ψ is N–Cα–C–N(+1)). Mark these angles in the torsion mode of VMD (key `4`). In the *Main* window under *Extensions – Analysis*, select the *Ramachandran Plot* window. In the new window under *Molecule*, you have to pick `npt.gro` again, then you should see the position of the currently displayed geometry of the peptide in the Ramachandran plot. Follow the movement of the peptide in this conformational space during the

simulation by playing the trajectory in the *Main* window. The conformations that are normally preferred by proteins are indicated, however, a single peptide may differ greatly. Which commonly known conformations correspond to the individual color-coded fields?

Additionally, try to create a video file (*Extensions – Visualization – Movie maker*). It may be necessary to install additional software on the computer, so get in touch with the TA for help with that.

C. RMSD and Ramachandran plot with Gromacs tools

To analyze the structure and dynamics, you can also conveniently use the corresponding Gromacs programs. They typically generate files with extension `.xvg`, which may be viewed with `xmgrace`.

So, the RMS deviation from the starting structure can be calculated, considering the group "Protein-H", with

```
gmx rms -s md.tpr -f md.xtc
```

A Ramachandran plot can be generated with

```
gmx rama -s md.tpr -f md.xtc
```

The resulting plot is quite crowded, so it may be a good idea to convert the data series to a 2D histogram. You may do so with the script `make_histogram.py` followed by `rama-histo-to-png.sh`, which creates an image file `rama-kcal.png`. (The scripts also require the files `gnuplot-deltag.in` and `viridis-tom.pal`.)

In which area of Ramachandran plot do you find the dominant conformation? Is there perhaps another conformation, which is less populated but still relatively distinct? Do these conformations correspond to any popularly known secondary structure elements?

D. Principal component analysis

Another useful analysis technique is the principal component analysis (PCA), also called the essential dynamics (ED), or in a more general context, the principal axis transformation. Within PCA, the collective motions of the atoms are identified on the basis of an analysis of fluctuations of atomic positions. These fluctuations are coupled as the vibration of an atom affects the vibrations of others, and this kind of coupling is what PCA quantifies. In biomolecules, it appears that the collective motions that involve large numbers of atoms are inherently linked to the functions of those molecules, therefore, a PCA analysis may provide valuable insight here.

The first step of PCA is the construction of the covariance matrix, which is a measure of the correlated movements of each pair of atoms. This matrix is subsequently diagonalized. Its eigenvectors describe the collective motions of the atoms, and each element of the eigenvector quantifies how much the coordinates of every single atom participate in the particular mode of motion. The corresponding eigenvalue then gives the amplitude of that mode of motion. Typically, most of the entire dynamics of a molecule is covered by few (five or so) first eigenvectors.

First, create a covariance matrix from the trajectory of the K2 peptide as input, and diagonalize it:

```
gmx covar -s md.tpr -f md.xtc -o eigenvalues.xvg -v eigenvectors.trr -xpma covar.xpm -mwa
```

The switch `-mwa` requests mass-weighted analysis (rather than pure position weighting). Take a close look at the first eigenvector:

```
gmx anaeig -first 1 -last 1 -s md.tpr -f md.xtc -v eigenvectors.trr -eig eigenvalues.xvg -extr ev1.pdb
```

The switch `-extr` is used to find in the trajectory the minimal and maximal values along the eigenvector(s). Then, the eigenvector(s) can be inspected visually with VMD (`vmd ev1.pdb`).

III. EXTENDED SAMPLING METHODS

In many situations, it is far too inefficient to generate configurations of the molecular system using a free MD simulation. The sampling of the configuration space would just take way too long – it would take prohibitively long to discover all relevant configurations (e.g., conformations of a peptide) with the correct probabilities in the trajectory. Extended sampling methods have been developed in order to resolve this serious problem.

A. Preparation of the system – a dipeptide

We will study a very simple system – a dipeptide in aqueous solution. By 'dipeptide' we mean an amino acid residue with capping groups CH_3CO– (acetyl) and –$NHCH_3$ (methylamino), to yield a chemically complete molecule. Such a dipeptide has a complete pair of angles $\varphi - \psi$, and it therefore makes sense to set up a Ramachandran plot.

To make the exercise more interesting, each participant may simulate a different amino acid, so that everyone will obtain a different result. At the end, the results will be compared and discussed.

The preparation of the system follows Secs. I D, I E and II, and is summarized here:

1. Build the dipeptide with `xleap`. This is the point where `ALA` **can be replaced with another amino acid!**

   ```
   source leaprc.ff14SB
   pep = sequence { ACE ALA NME }
   savepdb pep pep.pdb
   quit
   ```

2. Create a topology, using the AMBER99SB-ILDN force field and the TIP3P water model.

   ```
   gmx pdb2gmx -f pep.pdb -ignh -o pep.gro
   ```

3. Set up a cubic box and solvate.

   ```
   gmx editconf -f pep.gro -o pep.edit -c -d 1 -bt cubic
   gmx solvate -cp pep.edit -o pep.box -cs spc216.gro -p topol.top
   ```

4. If you have a charged amino acid, electro-neutralize the system. Select the group that contains water for `genion`.

   ```
   echo > dummy.mdp
   gmx grompp -f dummy.mdp -p topol.top -c pep.box -o dummy.tpr
   gmx genion -s dummy.tpr -o pep.ion.gro -p topol.top -neutral
   ```

With that, the system has been set up, and a topology file containing water (and an ion if needed) is ready.

Then, the molecular system has to be equilibrated. The procedures follow those in section II again, consisting of the following steps. (If there are no counterions, which is the case with uncharged amino acids, use `pep.box` rather than `pep.ion`).

```
# Short energy minimization.
gmx grompp -f em_steep.mdp -c pep.ion.gro -p topol.top -o em_steep.tpr
mdrunX -deffnm em_steep

# Heat up to 300 K.
gmx grompp -f nvt.mdp -c em_steep.gro -p topol.top -o nvt.tpr
mdrunX -deffnm nvt

# Switch on a barostat, and equilibrate until the density remains constant.
gmx grompp -f npt.mdp -c nvt.gro -p topol.top -o npt.tpr
mdrunX -deffnm npt
```

B. Free (unbiased) simulation

1. Run it

First, perform a free, unbiased MD simulation (that means, no extended sampling yet). You may use the input file `md_dipeptide.mdp`, which is provided. Prepare the simulation considering `npt.gro` as a starting structure:

```
gmx grompp -f md-dipeptide.mdp -c npt.gro -p topol.top -o md.tpr
mdrunX -v -deffnm md
```

2. Analysis – free energies

In the first step, calculate the dihedral angles along the trajectory:

```
gmx rama -s md.tpr -f md.xtc -xvg none
```

This calculation produces a Ramachandran plot file `rama.xvg` on the output.

Then, use the script `make_histogram.py` to construct a 2D histogram of the dihedrals φ and ψ. Finally, this can be converged to a free energy surface with the script `rama-histo-to-png.sh`, which produces the free energy plot in an image file `rama-kcal.png`. Free energy in kcal/mol is color-coded.

C. Umbrella sampling

1. Intro

The purpose of these simulations is to calculate the potential of mean force for the rotation along the ψ angle of your dipeptide by means of umbrella sampling simulations (US). US uses additional harmonic potentials to force the simulated system into regions of the reaction coordinate (here, that is the ψ angle) that would hardly be reached in a free MD simulation. In this particular case, we wish to sample the entire range from -180° to $+180^\circ$. We will divide this interval into steps of 10°, resulting in 37 individual simulations usually called windows. In each of the windows, the minimum of the harmonic potential is located in the center of the window, and the molecular system is forced to remain in that region of the reaction coordinate. However, the force constant of the harmonic potential has to be chosen such that the sampled regions of neighboring windows overlap; you will need to check that once the simulation is finished.

2. Run it

Change to the `UMBRELLA` directory. From the directory where you have run the equilibration, copy the topology file (`.top`) and the final structure file (`npt.gro`) to the new directory. In addition, you will need an `.mdp` file for an NPT simulation of 100 ps, and one such file may be downloaded from the tutorial website (file `md-npt.mdp`). The US procedure will be controlled by the script `run_umbrella_sampling.sh`, available from the tutorial website. Edit the script file with a text editor: you need to adjust the file names of the equilibrated structure and of the topology, so change `EQUIL-GRO` and `TOPOL-TOP` to match the names of your files. Also, you need to complete the definition of the dihedral angle ψ on line 24; for that, replace `C-ATOM` and `N-ATOM` with the corresponding atom numbers.

The script will run a series of Gromacs simulations, one for each of the 37 windows. It will actually use the Plumed plugin to introduce the additional harmonic potential; the necessary input files for Plumed will be generated by Plumed in every window automatically.

Start the script `run_umbrella_sampling.sh`; the job may take an hour to complete. You may estimate the exact duration easily: a window is 100 ps long, meaning that the entire sampling will be 3.7 ns. From the known computational cost of the equilibration, you can get a rather accurate estimate of how long the US will take.

3. Analysis – free energies

As soon as the simulations in all of the windows have finished, you need to confirm that the entire range of the reaction coordinate has been covered. The values of ψ along all of the windows are stored in files `colvar.xvg` in

the directories `GRAD_<PSI>`, and the corresponding histograms are in files `colvar-dist.xvg`. You may plot all of the histograms in one diagram conveniently with

```
xmgrace GRAD_*/colvar-dist.xvg
```

Do the neighboring histograms overlap sufficiently? If they do, it means that the definition of windows and the choice of force constant of the additional potentials were correct.

Also, inspect the trajectories from the simulations to see how the rotation along the Cα–C bond was sampled. It is east to visualize the final structures from the individual simulations: Save those structures in a single file

```
cat $(for i in {-180..180..10};
do echo -n "GRAD_$i/md.gro "; done) > test.gro
```

and view the file `test.gro` with VMD. Create a representation that only contains the heavy atoms of the peptide with the selection of "protein and not hydrogen". Then, align all of the structures with the initial structure; this can be performed by clicking *Extensions – Analysis – RMSD Trajectory Tool* and then hitting ALIGN. After that, if you play the trajectory, you should be able to see one full rotation along the Cα–C bond. Is it the case?

Finally, the desired free energy dependence on the dihedral angle ψ may be obtained with a program that contains the implementation of the Weighted Histogram Analysis Method (WHAM). On input, this program needs – for each of the windows – one file containing the time course of the dihedral; these files were produced during the simulations under the name `colvar.xvg` in each directory. Also, an additional file is needed, which contains the list of files that should be used for analysis, together with the parameters of the additional potential (force constant and the energy minimum). This file is provided (`metafile`); note, the values of force constant are converted from the value of 1000 kJ/mol/rad^2 (Gromacs+Plumed units) to kcal/mol/deg^2 (human units).

The WHAM program accepts the following arguments:

```
wham [P/Ppi/Pval] hist_min hist_max num_bins tol temperature numpad metadatafile freefile
```

- periodicity of the reaction coordinate:
  `P` considers a periodicity of 360° (our case here); `Ppi` – 180°; `Pval` – arbitrary periodicity of `val`, e.g. `P90.0`
- `hist_min` and `hist_max` define the range of reaction coordinate to calculate the free energy
- `num_bins` – number of bins
- `temperature` – in Kelvin units (typically 300)
- `tol` – convergence criterion (tolerance, try to set it to 0.0001 or less)
- `numpad` – only for aperiodic and 2D calculations; shall not be zero
- `metadatafile` – is `metafile_kcal`
- `freefile` – the output file name

The temporary results from the individual iterations of WHAM are printed on the standard output. The PMF is printed every 100 steps in this form:

```
-177.500000   2.994034   0.564643
-172.500000   3.895723   0.393350
-167.500000   4.923043   0.260562
-162.500000   5.917272   0.174905
-157.500000   6.888434   0.118498
-152.500000   7.776316   0.083008
-147.500000   8.773800   0.055648
-142.500000   9.650452   0.039157
 ...
```

The first column is the reaction coordinate, and the second is the calculated free energy in kcal/mol. Copy the final PMF into a new text file (e.g. `delta_g.xvg`), only keep the first and second columns, and visualize with `xmgrace delta_g.xvg`.

- Check if the obtained minimum of free energy is in accordance with what you observe in a free MD simulations. For this, create a histogram of the angle ψ for a free NPT simulation. How does the angle behave during the simulation? Does it stay in the global minimum for the entire duration of the simulation, or is the energy barrier overcome?

- Do(es) the obtained energy barrier(s) appear meaningful? What about the shape of the barrier(s)? Is it rather round and smooth, or are there any "sharp edges"?
- Compare the obtained positions of the minima with the expected values of ψ for idalized secondary structures, α-helix, β-sheet, PPII-helix, etc.

4. Optional – test for hysteresis

If you encounter any sharp barriers, it means that the windows in that area did not quite converge. A possibility to resolve this problem, or even just an easy test to see if this problem appears, is to repeat the calculation in the opposite direction, e.g., going from 180° to −180°.

Do this – it is easy! To not lose your previous results, do the following in a newly created directory. Simply modify the range of variable `i` in the script `run_umbrella_sampling.sh`, and then run the script again, and calculate the free energy once more.

Plot the free energy curve together with the previous one in one common diagram. Do(es) the barrier region(s) look the same as it(they) did before?

D. Metadynamics

1. Intro

In a metadynamics simulation, an additional, time-dependent 'artificial' energy function (biasing potential) is added to the potential energy function of the molecular system. This serves the purpose of bringing the molecule out of the energy minimum, in which it is currently located. The biasing potential function takes the form of a sum of many Gaussian functions, which are added one after the other during the simulation. The Gaussians are functions of one or several, simple or complicated function(s) of atomic coordinates; these are called the collective variables (CV) or reaction coordinates. Examples of CV:

- dihedral angles φ and ψ in a peptide
- distance of a ligand from the center of the binding pocket
- gyration radius of a protein molecule.

An important feature of the metadynamics method is that the sum of the Gaussians converges to the negative of the free energy (ΔF or ΔG), in the course of the simulation. The critical prerequisite for success is the appropriate choice of CV. An appropriate, natural choice of reaction coordinates for a dipeptide are the dihedral angles φ and ψ, so that the free energy in the form of a Ramachandran plot is obtained as a result.

The Gromacs package itself cannot perform metadynamics. However, a utility (plugin) called Plumed has been developed that complements the desired functionality. Practically, the parameters of the metadynamics method are provided in an additional input file, like the file `wt-metad.dat` here. You need to adjust the atom numbers in the definitions of dihedrals φ and ψ to match your dipeptide:

```
phi: TORSION ATOMS=5,7,9,C-ATOM
psi: TORSION ATOMS=7,9,C-ATOM,N-ATOM
METAD ...
  LABEL=metad
  ARG=phi,psi
  PACE=100
  HEIGHT=0.625
  SIGMA=0.349,0.349
  GRID_MIN=-pi,-pi
  GRID_MAX=pi,pi
  FILE=HILLS
  BIASFACTOR=7.
  TEMP=300.
... METAD
PRINT STRIDE=100 ARG=phi,psi,metad.bias FILE=plumed.xvg
```

First, the dihedral angles φ and ψ are defined by atomic numbers of the atoms C, N, Cα, C and N. Then the metadynamics is activated: Additional gaussians are added as functions of φ and ψ, in every hundredth step of the MD, and the height w and width σ of the Gaussians

$$w \cdot \exp\left[-\frac{(\varphi(t)-\varphi^*)^2+(\psi(t)-\psi^*)^2}{2\sigma^2}\right]$$

are 0.625 kJ/mol and 0.349 rad = 20°. The latter two options request the so-called well-tempered variant of metadynamics: Here, the additional Gaussians become smaller and smaller over time (w is reduced automatically), to ensure a better convergence of free energy. The exact value of the bias factor should be adjusted to the system to be simulated, and it may be a good idea to try multiple values. However, a rule of thumb recommends a value of half of the expected energy barrier in $k_\mathrm{B}T$ units ($1\,k_\mathrm{B}T = 2.5$ kJ/mol at 300 K). The last line specifies the data to be written out during the simulation.

2. Run it

First, prepare a Gromacs simulation in exactly the same as for a usual MD simulation. It is a good idea to to choose the number of steps in `md.mdp` such that the simulation will run overnight and finish early the next morning. Prepare everything in your respective "LoginX"

```
gmx grompp -f md.mdp -c npt.gro -p topol.top -o md.tpr
```

After that, carry out the actual metadynamics simulation:

```
mdrunX -deffnm md -plumed wt-metad.dat -v
```

3. Analysis – free energies

The analysis requires the file `HILLS`, which contains the positions and heights of the Gaussians. As mentioned above, the sum of all of the Gaussians corresponds to a mirror image of the free energy as a function of the collective variables used. In your case, you may obtain the free energy as a function of the dihedral angles $\varphi - \psi$, actually the Ramachandran plot, with the following command:

```
plumed sum_hills --bin 180,180 --min -pi,-pi --max pi,pi --hills HILLS
```

The resulting landscape of free energy, which is written in the file `fes.dat`, is best represented graphically. Use the script that has been provided (file `./fes-dat-to-png.sh`) to generate an image in PNG format (new file `fes-kcal.png`). The free energy in kcal/mol is color-coded.

Which conformation(s) do you see in the Ramachandran plot? Compare the plot with one that you obtained from the analysis of the free, unbiased simulation above. Are all of the minimum-energy basins visible in the plot obtained from the free simulation? Also, are the minimum depths and barrier heights in both plots equal?

Compare your result with the plots that the other students obtained (for the different amino acids).

IV. QM/MM SIMULATION

The most obvious limitation of empirical force fields (MM) is that covalent bonds never break or form in simulations, so no chemical reactions can be described. This limitation can be released by using a combined QM/MM method: the region where the chemistry happens is described using a quantum chemical method, and the large remainder of the system is calculated with MM as usual. The Nobel Prize for Chemistry 2013 was awarded in part for the development of these hybrid QM/MM schemes.

In this exercise, we will investigate a small chemical reaction in a small molecule, namely the proton transfer in a malonaldehyde molecule, see Fig. 2. The advantage is that the simulation will run fast, in fact, even faster because an efficient QM method will be used: the semi-empirical density functional theory method DFTB3. You can look up the technical details of DFTB3, but they are not needed to perform and analyze the simulations. DFTB3 is implemented in a program called DFTB+, which is linked to a modified Gromacs installation, so no external program is needed to perform the quantum-chemical calculations.

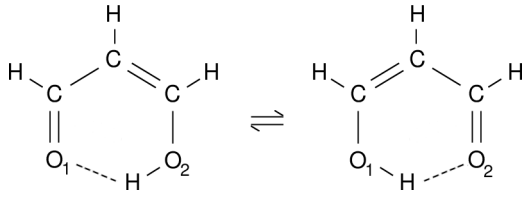

FIG. 2. Malonaldehyde – intramolecular proton transfer.

A. Equilibration with MM

First, the molecular system, a solvated malonaldehyde molecule, has to be equilibrated using a standard MM force field. (This also means that there may not be any proton transfer in the equilibration – but that does not matter.) You can use the supplied topology (file `mal.top`), as well as the structure in `mal.gro`. Start by placing the malonaldehyde molecule in a cubic water box, and proceed with the usual equilibration steps. The required `.mdp` files can be taken from Sec. III, in which you can modify the number of steps to keep the simulations quite short (below 100 ps).

```
gmx editconf ...
gmx solvate ...
gmx grompp -f steep.mdp ...
mdrunX -deffnm steep
gmx grompp -f nvt.mdp ...
mdrunX -deffnm nvt
gmx grompp -f npt.mdp ...
mdrunX -deffnm npt
```

B. Preparation

The most important question in any QM/MM simulation is how to divide the molecular system between QM and MM regions. Here it is very simple: a malonaldehyde molecule is so small that it can be described entirely with QM, and the aqueous solvent is then calculated using the MM force field. Such QM/MM settings have to be reflected in the topology of the system: No energy within the QM region is calculated with MM, thus all force field terms describing the QM region have to be removed.

To prepare a QM/MM-compliant topology, copy the existing topology to a new file `mal-qmmm.top`, and edit it with a text editor (e.g., `gedit`) as follows: Delete sections *angles*, *dihedrals*, and *pairs*. Furthermore, change the types of all bonds from 1 to 5, and delete the parameters. The section shall look like this:

```
 [ bonds ]
; ai   aj   funct      b0          kb
  1    2     5
  2    4     5
...
```

The topology is ready. No intramolecular interactions within the malonaldehyde molecule are calculated with the force field. The charges of the QM atoms can remain in the topology file, because they will be set to zero by Gromacs automatically. The interactions between the QM and MM regions are calculated in separate computations: the QM method takes covers the electrostatic (charge–charge) interactions, and it considers the charges of the QM atoms that are obtained from the Mulliken analysis, which is performed in the quantum chemical calculation. On the other hand, the usual Lennard-Jones interaction for QM–MM is evaluated with the MM routines of Gromacs, therefore atomic types have to be assigned to the QM atoms as well, and Lennard-Jones parameters have to be supplied.

C. QM/MM simulation itself

You will need the file `long.mdp`, which contains several additional options in order to set up QM/MM:

```
QMMM                    = yes
QMMM-grps               = MAL
QMmethod                = RHF
QMMMscheme              = normal
QMbasis                 = STO-3G
QMcharge                = 0
QMmult                  = 1
MMChargeScaleFactor     = 1
```

First, it is determined that a QM/MM simulation is to be run, and the QM region is defined. 'MAL' refers to the name of a group in the index file (here, `index.ndx`). The index file needs to be prepared with

```
gmx make_ndx -f npt.gro
```

and be provided to `grompp` later. Also required is additional information regarding the quantum-chemical calculation. While the following three options are ignored (no RHF/STO-3G calculation is performed), it may be important to specify the charge (and spin multiplicity) of the QM region. You will be dealing with a charge-neutral system in a single electronic state, therefore `QMcharge = 0` and `QMmult = 1`.

You will be using the DFTB3 method implemented in the DFTB+ program, therefore, you need an input file for DFTB+. The file name must always be `dftb_in.hsd`, and the file may contain the following:

```
Geometry = {
  TypeNames = C O H
  TypesAndCoordinates {
    2   0.0   0.0   0.0
    1   0.0   0.0   0.0
    3   0.0   0.0   0.0
    1   0.0   0.0   0.0
    3   0.0   0.0   0.0
    1   0.0   0.0   0.0
    3   0.0   0.0   0.0
    2   0.0   0.0   0.0
    3   0.0   0.0   0.0
  }
}
Hamiltonian = DFTB {
  SCC = Yes
  MaxAngularMomentum {
    C = "p"
    O = "p"
    H = "s"
  }
  SlaterKosterFiles = Type2FileNames {
    Prefix = "/home/tkubar/DFTB/3ob-3-1/"
    Separator = "-"
    Suffix = ".skf"
  }
```

```
  ThirdOrderFull = Yes
  HubbardDerivs {
    C = -0.1492
    O = -0.1575
    H = -0.1857
  }
  HCorrection = Damping { Exponent = 4.0 }
}
Analysis = { CalculateForces = Yes }
Options = { WriteDetailedOut = No }
```

The block 'TypesAndCoordinates' contains the list of QM atoms, specifying the elements and atom coordinates. While the elements have to follow the same sequence as they do in the Gromacs topology file, the coordinates in the file are ignored and may thus be set to arbitrary values like zeroes safely. DFTB requires a set of parameter files, which will be sought in the directory specified under 'SlaterKosterFiles – Prefix'. At this point, check it that directory exists; it should contain a number of `.skf` files.

The Gromacs version that you will be using is also linked with Plumed. Enable this version with the command

```
gromacs-qm
```

Just like with usual MM simulations, the Gromacs preprocessor has to be called first, followed by the actual simulation. An important difference here is that the double-precision versions of all Gromacs programs will be used. This is simple to do – always call `gmx_d` rather than `gmx`:

```
gmx_d grompp -f long.mdp -c npt.gro -p mal-qmmm.top -n index.ndx -o long.tpr
export GMX_DFTB_CHARGES=1
export GMX_QMMM_VARIANT=1
gmx_d mdrun -deffnm long -v > mdrun.out
```

Make sure to use the QM/MM topology `mal-qmmm.top`. The free QM/MM simulation will take somewhat longer, ca. 40 min for a simulation of 100 ps with a time step of 1 fs.

D. Analysis

First, inspect the trajectory visually (`vmd npt.gro long.trr`). You can hide the water (*Graphics – Representations* and 'not water' for selected atoms), or directly look at the `.xtc` file (`vmd mal.gro long.xtc`). Do you see a proton transfer? If the original O–H bond becomes unusually long, it is already broken. You can change the display to 'Dynamic bonds' in order to always see the currently existing O–H bond.

In order to quantitatively follow the proton transfer process, we should now choose a suitable reaction coordinate – a quantity that we can calculate from the coordinates of the atoms, and which describes the reaction well. With such a simple proton transfer, the difference between the two O–H distances is a good reaction coordinate.

Now, you need to measure these distances and calculate their difference, all that along the trajectory from the QM/MM simulation. You can use the Plumed program for this task with advantage. Note that Plumed has been designed to facilitate working with various reaction coordinates. Use a text editor to create an input file for Plumed under the name `diffdist.dat`. Here, both distances are defined, then their difference, which is written into a file:

```
d1: DISTANCE ATOMS=1,9
d2: DISTANCE ATOMS=8,9
d:  COMBINE ARG=d1,d2 COEFFICIENTS=1,-1 PERIODIC=NO
PRINT ARG=d FILE=diffdist.xvg
```

Run Plumed as follows:

```
plumed driver --mf_xtc long.xtc --plumed diffdist.dat --timestep 0.01
```

View the resulting file with `xmgrace diffdist.xvg`. What do you see?

In a QM/MM simulation, it is also interesting to look how the electron density evolves in time. In DFTB3, electron density is represented by partial charges of the individual QM atoms. Gromacs has written the atomic charges into a file which you can visualize with `xmgrace -nxy qm_dftb_charges.xvg` (atomic charges on the y-axis vs. number of MD steps on the x-axis). Each curve corresponds to the charge of one QM atom, so there are nine curves in total. Do you see any connection between the time course of the charges and any proton transfer events?

We are often not really interested in the time course of a quantity itself, but rather, the probability with which the quantity takes the different values. Use the analysis tool of Gromacs to create a histogram (= probability density) of the difference of the O–H distances:

```
gmx analyze -f diffdist.xvg -dist diffdist-histo.xvg -bw 0.005
```

The program reports the mean value and the standard deviation, and writes the histogram into a new file, which you can view with `xmgrace`. What would be the average, and what kind of histogram would you expect for the proton transfer in malonaldehyde, assuming that your simulation is long enough? Are your expectations fulfilled?

E. Metadynamics

In order to obtain converged free energies, it may be necessary to perform long simulations. Especially with QM/MM simulations, this may take prohibitively long computing times. This problem can also be solved by using extended sampling techniques. In the last exercise, you will investigate the proton transfer energetics with metadynamics.

To do so, use the existing `.tpr` file, which you had prepared for the free simulation. In addition, you will need an input file for Plumed, in which a metadynamics simulation is requested. Recall that you already ran a metadynamics simulation – for a dipeptide in Sec. III D. Now, however, it will be the well-tempered variant of metadynamics, combined with QM/MM. Create an input file under the name `wt-metad.dat` with the following contents:

```
d1: DISTANCE ATOMS=1,9
d2: DISTANCE ATOMS=8,9
d:  COMBINE ARG=d1,d2 COEFFICIENTS=1,-1 PERIODIC=NO
METAD ...
  LABEL=metad
  ARG=d
  PACE=100
  HEIGHT=0.625
  SIGMA=0.02
  FILE=HILLS
  BIASFACTOR=5.
  TEMP=300.
... METAD
PRINT STRIDE=100 ARG=d,metad.bias FILE=plumed.xvg
```

Execute the metadynamics simulation by additionally passing that file to `mdrun`:

```
gmx_d mdrun -deffnm long -plumed wt-metad.dat -v > mdrun.out
```

Once finished, or even while the simulation is still running, the file `HILLS` can be analyzed to construct the free energy curve (ΔG). Like before, this can be done with the Plumed program:

```
plumed sum_hills --bin 150 --min -0.15 --max 0.15 --hills HILLS
```

The resulting free energy in the file `fes.dat` can be viewed with `xmgrace`.

Does the curve agree with the histogram from the free simulation? Does the curve show the expected symmetry? How high is the barrier? If we want to calculate the rate of proton transfer from the barrier height, we might underestimate the rate. Why?