Introduction into molecular mechanics and molecular dynamics simulations

Sabine Reißer and Tomáš Kubař

I. INTRODUCTION

A. Linux

This tutorial uses the Linux computers in seminar room 110, bldg 30.41. 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.

Log on the computer with 'student' as user name and 'student' as password. The computer starts a graphical X-Windows interface. **Please note:** The computer never needs to be shut down, and you never have to log off during the tutorial. To be able to enter text commands, you then start a so-called shell, a window linked to a command interpreter. This can be found in the menu under Favorites. In the shell that opens, 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. 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.

C. 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 K2 as a new unit `peptide` with the command

```
source leaprc.ff14SB
peptide = sequence { LYS ILE ALA GLY LYS ILE ALA LYS ILE ALA GLY LYS ILE ALA }
```

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 K2 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} {
{"N" "CA" "C" "N" value-for-psi}
{"C" "N" "CA" "C" value-for-phi}
}
```

K2 should be α-helical now. Save the structure with

```
savepdb peptide k2.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?

D. 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 k2.pdb -ignh -o k2.gro
```

We use the Amber99SB-ILDN force field and do not need water at first. The output file `k2.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 ]`.

E. Geometry optimization

Geometry optimizations search for the nearest energy minimum starting from an initial geometry. These methods cannot overcome energy barriers and do not include correction for different entropies of states. Nevertheless, they are useful tools for optimizing modeled structures or preparing for MD simulations.

In Gromacs all simulations including geometry optimization 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. Look at the file `em_steep_vacuo.mdp`, explanations can also be found here. 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.

Now call the preprocessor:

```
gmx grompp -f em_steep_vacuo.mdp -c k2.gro -p topol.top -o em_steep_vacuo.tpr
```

It is not really necessary to explicitly specify the file extensions, because Gromacs adds these automatically. The following command thus performs exactly the same operation:

```
gmx grompp -f em_steep_vacuo -c k2 -p topol -o em_steep_vacuo
```

If there is no error message, start the geometry optimization with

```
gmx mdrun -v -deffnm em_steep_vacuo
```

The switch `-v` stands for “verbose”, and makes Gromacs print additional information on the screen; This switch is available on all Gromacs programs. `-deffnm` means “default filename” and sets the same base name for 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.

Plot the course of potential energy with `gmx energy`, and view the file with Xmgrace:

```
gmx energy -f em_steep_vacuo.edr -o em_steep_E_pot.xvg
xmgrace em_steep_E_pot.xvg
```

(Alternatively, this can be prompted directly with the switch `-w` for `gmx energy`).

Now compare the different minimization algorithms available. Make two copies of the file `em_steep_vacuo.mdp` under the names `em_cg.mdp` and `em_bfgs.mdp`. In the files, change the ‘integrator’ from `steep` for steepest descent to `cg` for conjugate gradients or `l-bfgs` for the Broyden–Fletcher–Goldfarb–Shanno algorithm (Be careful, the first character in `l-bfgs` is an ‘ell’, not ‘one’). Perform these geometry optimizations as above. Remember to change the output file names accordingly.

Compare the course of potential energy, by passing Xmgrace all three `.xvg` files as arguments. With the addition `-legend load` to the `xmgrace` command, you can display a legend. Additionally, compare the resulting structures (in `.gro` files) visually using VMD. Make use of the capability of VMD to load several different structures in VMD and view them simultaneously (File – New molecule . . .). Did you obtain roughly the same structure with all of the algorithms, or are there any differences?

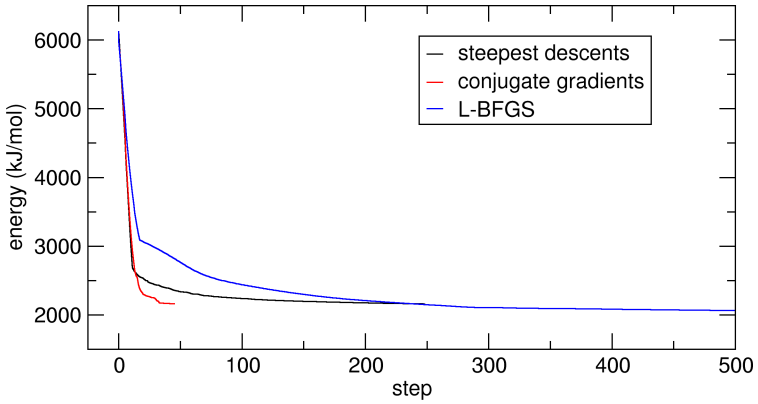

FIG. 1. Comparison of the various minimization algorithms. They take largely different numbers of steps to arrive in the nearest local minimum. The minima need not even be the same (e.g., L-BFGS leads to a minimum that is deeper in energy).

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 `part3`. Copy one of the geometry-optimized structures of the peptide K2 and the topology file from the last exercise into this directory. In the following, you will place the peptide K2, created and geometry-optimized in the previous part, into a water box, heat it up, equilibrate the system and start a short simulation.

1. Solvation

First, place the peptide in a cubic simulation box:

```
gmx editconf -f em_steep_vacuo.gro -o k2_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 k2_newbox.gro -p topol.top -cs spc216.gro -o k2_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"
```

Now, the system has to be electro-neutralized because the peptide carries a positive charge. (Question: Which amino acids are responsible for this?) For this, you will need an arbitrary `.mdp` input file; in fact, an empty file is good enough, so you may generate one with the command `echo` easily. Then, create a `.tpr` file, which you pass to the program `genion` to have the ions added:

```
echo > dummy.mdp
gmx grompp -f dummy.mdp -o dummy.tpr -p topol.top -c k2_sol.gro
gmx genion -s dummy.tpr -o k2_ion.gro -p topol.top -neutral
```

This program replaces the required number of water molecules with Cl^- ions. Indicate which of the groups designates water ("SOL", probably number `13`).

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 k2_ion.gro -p topol.top -o em.tpr
gmx mdrun -v -deffnm em
```

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
gmx mdrun -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. 2 (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. 2 (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?)

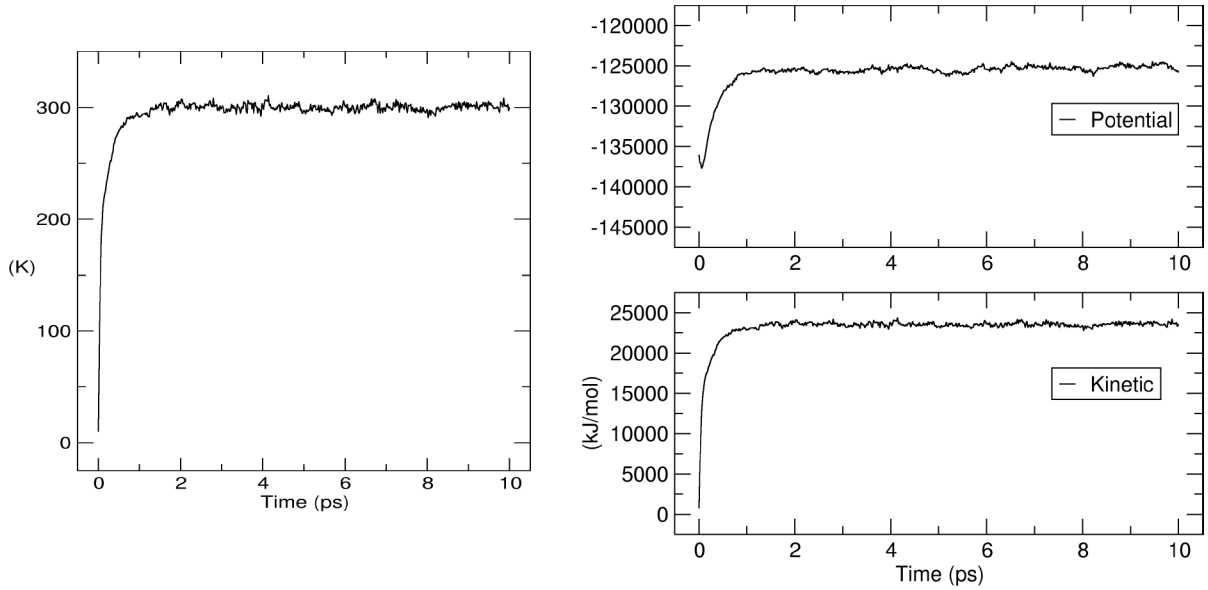

FIG. 2. NVT equilibration: The time course of temperature (left) and those of potential and kinetic energy (right).

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
```

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

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

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. Ramachandran plot and time series of dihedral angles

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
```

Also, generate the time series of the dihedral angles φ and ψ with the Gromacs program `gmx angle`. For that, you need an index file in which the atomic indices belonging to a dihedral angle are listed. Complete the list that has been started in the file `dihedrals.ndx`. Then inspect the file `calc_dihedrals.sh`. Here, `gmx angle` is called for each dihedral angle in a loop. Run the script with the required input files.

Plot the output files and compare with the results that you obtained previously with VMD. In addition, you may measure the angle over a peptide bond (ω), which is defined as Cα–C–N(+1)–Cα(+1). Does it remain close to 180° as it should?

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`).

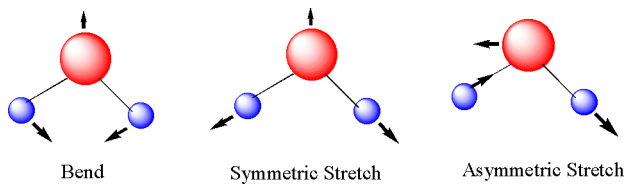

FIG. 3. Vibrational degrees of freedom of a water molecule.

E. Normal mode analysis

Being in a sense complementary to MD simulations, normal mode analysis (NMA) provides an idea of the conformational space of a molecule. In addition, the configuration entropy of the molecule may be calculate with NMA within a harmonic approximation. Entropy is usually difficult to access, so any approach to estimate it is valuable. The NMA procedure consists of three steps:

- minimization of the conformal energy as a function of the coordinates (geometry optimization),
- calculation of the Hessian (Hesse matrix), which contains the second derivatives of the potential energy with respect to the coordinates of atoms, and
- diagonalization of the Hessian to obtain the normal modes of vibration (eigenvectors of Hessian) and the vibrational frequencies (calculated easily from the eigenvalues of Hessian).

We will calculate the normal modes of the K2 peptide molecule. A certain limitation is that it is probably only reasonable to perform NMA for an isolated K2 molecule, rather than for the full solvated system; the reason is that it is probably not meaningful to search for an energy-minimized structure of such a complex solvated molecular system. It is namely very important for NMA that the structure of the molecule is as close as possible to a genuine minimum of potential energy.

Use the structure and topology of the K2 peptide in vacuo from Sec. I E. For all of the calculations involved in NMA, it is crucial to use Gromacs programs compiled in *double precision*, which is requested by calling `gmx_d` instead of just `gmx`.

The first calculation is a BFGS minimization:

```
gmx_d grompp -f nma-em-bfgs.mdp -c em_bfgs.gro -p topol.top -o nma-em-bfgs.tpr
gmx_d mdrun -v -deffnm nma-em-bfgs
```

It is now possible to calculate the Hessian for the structure that has been energy-minimized in double precision. The calculation of the Hessian is requested by setting `integrator = nm` in the `.mdp` file. Here, the input structure has to be read with maximum possible accuracy, and there is a problem with `.gro` files: the coordinates in these files are typically rounded to three decimal places, and such an accuracy is insufficient for the calculation of normal modes. To resolve this problem, coordinates have to be read from the last frame of the complete *double precision* trajectory file from the minimization. This file has a suffix `.trr`, and it is in a binary format (not human readable). The `.trr` file can be read in using the switch `-t` with `gmx_d grompp`:

```
gmx_d grompp -f nma.mdp -c nma-em-bfgs.gro -t nma-em-bfgs.trr -p topol.top -o nma.tpr
gmx_d mdrun -v -deffnm nma
```

After this procedure, the file `nma.mtx` contains the Hessian matrix, which has a dimension $3N \times 3N$ with N being the number of atoms in the molecule (of which each has 3 cartesian coordinates). Diagonalize the Hessian with

```
gmx_d nmeig -f nma.mtx -s nma.tpr -last -1
```

This procedure results in a set of $3N$ eigenvalues in `eigenval.xvg` and a set of $3N$ corresponding eigenvectors in `eigenvec.trr`, where each vector has $3N$ elements. This can be understood as a collection of vectors attached to each of the atoms, and each of those vectors defines the direction along which the respective atom moves in the given mode of motion, see Fig. 3 for a simple example. The first six eigenvalues should be close to 0 and correspond to the three translational and three rotational degrees of freedom, which are of no real interest. All other eigenvalues should be positive, while any negative eigenvalues signify that the structure of the molecule corresponds to a saddle point on the energy surface rather than to a minimum. You can check the values in the file `eigenval.xvg`.

Finally, visualize several modes of motion of the K2 peptide molecule. For this purpose, a trajectory file must be constructed from the eigenvector(s):

```
gmx_d anaeig -first 7 -last 7 -v eigenvec.trr -nframes 50 -max 5. -s nma.tpr -extr mode7.pdb
```

With this command, the number "7" stands for the ordinal number of the mode to be visualized; The 7th mode should be the first vibration. The generated PDB file can be opened and viewed with VMD. Repeat the procedure for several other modes of motion.

III. ADVANCED SIMULATIONS – PEPTIDE IN A MEMBRANE

A. Prepare initial structures and force fields

The file `DOPC_303K.gro` contains an equilibrated solvated lipid bilayer. How many lipid molecules are there in each layer? How many carbon atoms do the DOPC fatty acid chains have? From the box coordinates, calculate the area per lipid for this lipid at 303 K.

We also need a structure and a topology file for the peptide K2 here. Copy the structure `k2.gro` that you built in Sec. I C into your working directory, and also copy the topology `topol.top` from Sec. II A 1; remove water from the topology.

So far, we have run MD simulations with the Amber99SB-ILDN force field. For the lipids, we now use new Stockhom lipid force field, which is compatible with Amber force fields. All required force field parameters for the Stockholm lipids are contained in the file `forcefield.ff.zip`. Uncompress the folder with `unzip`, and find the topology file for DOPC under the name `DOPC.itp`.

B. Orient the peptide and the lipid membrane

First, we must rotate the peptide molecule so that its helical axis is aligned with the z direction. That may in fact be the case, as you can check with VMD. However, if it is not the case, achieve such an orientation with the following command:

```
gmx editconf -f k2.gro -o k2_edit.gro -princ -rotate 0 90 0
```

and select "System". Check the alignment of the peptide in the VMD again.

Now you have to set up the lipid bilayer. First, you need a `.tpr` file that only contains the lipid. There will be no real simulation for now, so you may use any `.mdp` file for `grompp`, and an empty file such as `dummy.mdp` will work. You also need to remove all of the water from the lipid structure `DOPC_303K.gro`, and adjust the number of atoms on the second line. Caution: the last line with the box size shall remain. You also need a topology file for the pure lipid system, and you may create one under the name `topol_DOPC.top`:

```
#include "forcefield.ff/forcefield.itp"
#include "DOPC.itp"
[ system ]
; name
DOPC
[ molecules ]
; Compound  #mols
DOPC         128
```

Assemble the desired `.tpr` file with the command

```
gmx grompp -f dummy.mdp -c DOPC_303K.gro -p topol_DOPC.top -o dummy_DOPC.tpr
```

If you receive warnings, ignore them with the option `-maxwarn [number]`. Note: This is the only step in which you use the empty `.mdp` file and the lipid topology file.

In the second step, remove the periodicity with `trjconv`:

```
gmx trjconv -s dummy_DOPC.tpr -f DOPC_303K.gro -o DOPC_whole.gro -pbc mol -ur compact
```

and select "System".

Now, inspect the last line of the new file `DOPC_whole.gro`: there are the $x/y/z$ lengths of the simulation box. You need to orient the K2 peptide in the same cell and place the molecule in the center of the box:

```
gmx editconf -f k2_edit.gro -o k2_newbox.gro -c -box [x-length] [y-length] [z-length]
```

C. Insert the peptide into the membrane

First, download the script InflateGRO renamed to `inflategro.pl`:

```
wget cbp.ipc.kit.edu/joomla/downloads/CB-praktikum-WS2016/materials/part3/inflategro.txt
mv inflategro.txt inflategro.pl
```

Combine the structure files of peptide and lipid:

```
cat k2_newbox.gro DOPC_whole.gro > system.gro
```

Remove the unnecessary lines (box vector from the peptide structure and the first two lines of the lipid structure), and update the number of atoms on the second line.

For InflateGRO, it is recommended to apply very strong position restrain on the positions of the heavy (non-hydrogen) atoms of the peptide. This is a way to avoid any movement of the peptide during the energy minimizations that are performed during the InflateGRO run. In order to do that, add a new `# ifdef` section to the topology file `topol.top`; that section will refer to a particular set of position restraints. At the same time, insert the topologies of lipid, water and counterions:

```
; Include position restraint file           ; was here originally
#ifdef POSRES
#include "posre.itp"
#endif
; Strong position restraints for InflateGRO ; added newly
#ifdef STRONG_POSRES
#include "strong_posre.itp"
#endif

; Include DOPC topology                     ; added newly
#include "DOPC.itp"
; Include water and counterion topology     ; added newly
#include "amber99sb-ildn.ff / tip3p.itp"
#include "amber99sb-ildn.ff /ions.itp"
```

Finally, the force field for the lipid should be inserted into the topology file (at the top):

```
#include "amber99sb-ildn.ff/forcefield.itp" ; was here originally
#include "forcefield.ff/ffnonbonded.itp"    ; added newly
#include "forcefield.ff/ffbonded.itp"       ; added newly
```

Create a new definition of position restraints as follows; select "Protein-H" when prompted:

```
gmx genrestr -f k2_newbox.gro -o strong_posre.itp -fc 100000 100000 100000
```

Then, obtain an `.mdp` file that you previously used for energy minimization, such as `em_steep.mdp`. In this file, add the following line to turn on the new position restraints:

```
define = -DSTRONG_POSRES
```

Finally, follow the instructions for InflateGRO that are listed in the script itself. Scale the positions of the lipid molecules by a factor of 4:

```
perl inflategro.pl system.gro 4 DOPC 14 system_inflated.gro 5 area.dat
```

- `system.gro` – coordinates that shall be scaled
- `4` – scaling factor. A value > 1 means inflation, < 1 means shrinking
- `DOPC` – reside name of the molecules for which coordinates shall be scaled
- `14` – cutoff radius (in Å) to search for the lipid molecules to be deleted
- `system_inflated.gro` – output coordinate file
- `5` – grid parameter (in Å) for the calculation of the area per lipid
- `area.dat` – output file with the area per lipid, useful to estimate if the final structure is already suitable

Notice how many lipid molecules have been deleted and update the entry under `[ molecules ]` in the topology file accordingly (last line: `DOPC 128` or less).

Perform an energy minimization (`gmx grompp` and `gmx mdrun`). To do this, use the above modified file `em_steep.mdp`. Use `em_steep.gro` as the name for the final structure.

Then shrink the system by scaling the coordinates of lipid molecules by a factor of 0.95,

```
perl inflategro.pl em_steep.gro 0.95 DOPC 0 system_shrink_1.gro 5 area_shrink_1.dat
```

and carry out another energy minimization.

For the shrinking, it is absolutely necessary to set the cutoff value to 0, otherwise you would delete further lipid molecules! The gradual shrinking is very easy to script using a `for` loop, for instance. After ca. 28 repetitions of scaling with a factor of 0.95, you will reach an area per lipid that is close to the experimental value – check that!

Before introducting the solvent, increase the length of the simulation box in the z-direction to 8 nm. To do that, simply overwrite the last number on the last line of the corresponding `.gro` file manually.

D. Solvate with water

To solvate a single protein molecule with `gmx solvate` is very easy. However, it is more difficult to solvate a membrane protein. The problem may be that the program tends to fill any 'gaps' between the fatty acid tail with randomly placed water molecules. This can be avoided with a little trick: Copy the Gromacs file `vdwradii.dat` (it can be found with the `locate` command) to the current directory, and change the value of carbon from 0.17 to 0.5. This will convince the program `gmx solvate` on the basis of the increased van der Waals radius that all carbons are very large. Consequently, any water molecules will be very unlikely to be squeezed in between fatty acid tails.

After solvation, verify with VMD that there are not many water molecules in the hydrophobic core of the lipid bilayer. Then, the few remaining water molecules in the hydrophobic core shall be removed manually. For this, view the structure in VMD and create a representation in which only water molecules can be seen, and changed the view to 'orthographic'. After switching to Label–Atom mode (press `1`), click on those water molecules that have penetrated deep into the membrane. Take a note of the residue numbers of molecules in VMD; since they match those in the `.gro` file, it is easy to consequently remove the disturbing water molecules manually in the `.gro` file. Then you also need to adjust the number of atoms at the top of the `.gro` file as well as the number of water molecules at the bottom of the topology file.

Note that using an increased radius for carbon can also create artificial voids, especially whenever the peptide protrudes from the membrane into the solvent (because the peptide also contains carbon), and also near the lipid head groups (for the same reason). Therefore, it will be necessary to check if any voids have disappeared after the equilibration, which you will perform in the next step below.

Importantly, delete the file `vdwradii.dat` from the working directory before starting the equilibration.

Also, remember to electro-neutralize the system with 4 Cl^- counterions prior to equilibration. Do that with the program `gmx genion` in the same way as in Sec. II A 1.

In all of the following simulations, you will need a so-called index file. You may create this with the program `gmx make_ndx`, which reads in the final structure file (with water and counterions):

```
gmx make_ndx -f ion.gro
```

In addition to the proposed atomic groups, you should also define a group combined from Protein and DOPC. Do this with the command `1 | 13` (if groups `1 Protein` and `13 DOPC` were predefined). The group name can remain `Protein_DOPC`. The newly created index file will then be passed to the Gromacs preprocessor with an additional option `gmx grompp ... -n index.ndx`.

E. Equilibrate

Now, create the parameter files for the equilibration. To do so, copy the `.mdp` files from Secs. II A 2 and II A 3 into your current working directory. Change the thermostat settings in `nvt.mdp` so that two separate coupling groups are introduced, and the reference temperatures are brought from the initial 10 K to the desired 303 K:

```
; Temperature coupling is on
tcoupl            = V-rescale
tc-grps           = Protein_DOPC  Water_and_ions
tau_t             = 0.1           0.1
ref_t             = 303.          303.

annealing         = single        single
annealing-npoints = 3             3
```

```
annealing-time    = 0. 50. 100.   0. 50. 100.
annealing-temp    = 10. 303. 303. 10. 303. 303.
```

In addition, add a command to remove the translational motion of the entire system:

```
; COM motion removal
; These options remove the translation of the system through the box
nstcomm   = 1
comm-mode = linear
comm-grps = System
```

Also turn off restraints by removing the line `define = -DPOSRES`. Change the simulation time to 100 ps (what is the required number of time steps?).

Apply the same changes to the file `npt.mdp` also. Additionally, change the time constant of the temperature coupling to 0.5 ps for both coupling groups, and introduce a pressure coupling section:

```
pcoupl = Parrinello-Rahman
pcoupltype = semi-isotropic
tau_p = 1.0
ref_p = 1.0 1.0
compressibility = 4.5e-5 4.5e-5
```

As you can see, the pressure in the directions $x - y$ is now adjusted independently of the direction z, which is called a semi-isotropic coupling. (It may become necessary to make the pressure coupling even weaker, with a relaxation time of up to 2 ps.) Change the simulation time to 1 ns and request the coordinates be written out every 2 ps.

(On my laptop with Gromacs 2018.4: NVT: 100 ps in 24 min, NPT: 1 ns in 4 h.)

F. Perform production MD simulation at 303 K

Modify also `md.mdp` accordingly. Set the simulation time to 10 ns, and write out the coordinates every 10 ps. Now execute the MD simulation.

For the analysis, you may either use your own trajectory file from the simulation that has run over the Christmas break, or alternatively, download a longer trajectory `md.xtc` and the corresponding `md.tpr` file.

G. Analyze

- Calculate the time course of the distance of the center of mass of the peptide from the center of the membrane:
  `gmx distance -f ...xtc -s ...tpr -n ...ndx -oav distave.xvg`
  `  -select 'com of group "Protein" plus com of group "DOPC"'`
- Compute the temporal course of helicity with `gmx helix`.
- Use `gmx bundle` to calculate the time course of the angles between the CA–CB bonds of the alanines from the normal direction of the bilayer (z-axis).
- Calculate the density of the four components of the molecular system (polar lipid head groups, hydrophobix lipid tails, water, and the peptide) along the z-axis. Use a script from the tutorial website (all five files in directory `part3/DENSITY`). Plot the resulting files with `xmgrace histo_*xvg`, and discuss your observations.

V. LIGAND DOCKING WITH AUTODOCK

A. Ligand Docking

MD simulations are not the only option to study biomolecular interactions. Despite all the approximations in the force fields and steadily increasing computer performance, one very important class of biochemical problems remains difficult to simulate: the so-called docking of a ligand to a receptor. The question is in which conformation and with what binding strength small molecules bind to macromolecular (protein) receptors. Non-covalent binding (interaction) is considered typically.

This question is crucial because one of the key functions of proteins is to recognize and bind chemical compounds, in order either to catalyze chemical reactions (enzymes) or to respond to the binding of the ligand in some way (sensor proteins). Ligand binding is not only important for the fundamental research of cell function; it is in fact one of the fields where computer modeling finds practical applications.

So-called *in silico* drug design is a branch of pharmaceutical research that aims at predicting the strength, geometry and specificity of binding of trial chemical substances to target proteins. The target proteins are commonly associated with one or more diseases, and the goal of drug development is typically either to inhibit or to activating these proteins. Importantly, three-dimensional structures of many hundreds of such target proteins are known, for example, from X-ray crystallographic analysis. The problem that is that it is extremely resource-intensive to make and purify the target protein in large quantities and to assess the activity of many trial substances (typical industrial compound databases contain tens of thousands of substances). In additiona, many new trial substances have to be synthesized anew, and all these procedures combined turn out to be extremely time-consuming and expensive, so that the whole study of a certain target protein may take years of time and billions of dollars.

Ligand docking calculations work as a filter for this process. The goal is to estimate – qualitatively – the strength of binding of a large number of trial compounds with simple and fast calculation methods, and perform the costly experiments only for the most promising candidates. Now, the calculation should not be an MD simulation because even the most efficient MD-based methods for calculating free energies are still far too demanding. Ideally, a result should be obtained in minutes for a receptor–ligand pair.

Ligand docking programs are implementations of such efficient algorithms. There are a variety of different procedures to search for geometries of receptor–ligand complexes, and ways to estimate the binding energy (so-called scoring functions). Several docking programs are widely used, like the free programs Autodock and Dock, as well as commercial variants like FlexX, Glide and Gold. Most currently used programs are of comparable performance, and they are quite often capable of distinguishing coumpounds that bind to the receptor from those that do not. That being said, a solution to the docking problem that would be both universally useful and practically efficient, is not available yet.

The aim of this exercise is to dock the drug Indinavir to its target protein, a protease from the HIV virus, with the help of the program Autodock. It is greatly simplified when compared to a realistic docking study: There is only one candidate ligand, and we know beforehand that it binds well. Still, the exercise should demonstrate the principles of ligand docking. Autodock and the graphical interface Autodock Tools (ADT) are installed on our computers.

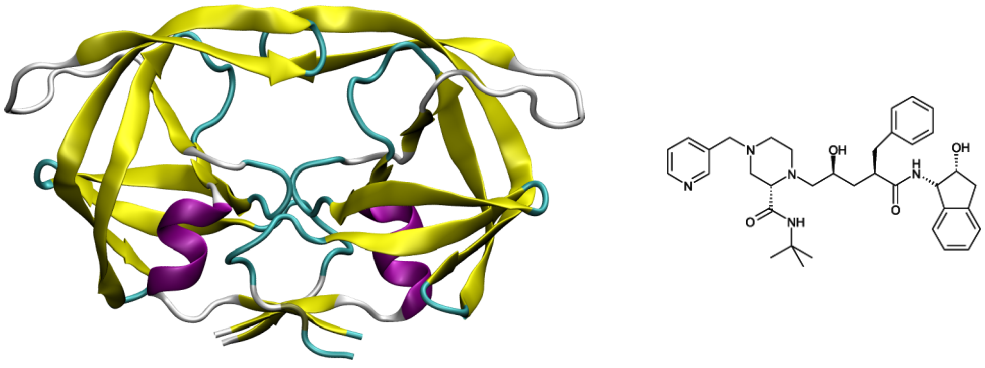

FIG. 4. Left: The Structure of the HIV Protease I. Right: Indinavir, an approved drug for the inhibition of the HIV-I protease.

The content of this experiment is based on the tutorial "Using AutoDock 4 with AutoDockTools" by Ruth Huey and Garrett M. Morris of The Scripps Research Institute, La Jolla, CA.

B. Visualization of docked structures

Autodock can be operated completely from the command line, but a graphical interface may be more comfortable, so we will use that in our tutorial. Start by creating a directory `docking`, in which all of the files for this exercise shall be stored. Change to this directory and start the graphical user interface of ADT:

```
adt
```

Pick *Autodock 4.2* in the small selection window. The *ADTools window* that has opened can be used to prepare all calculations and to view the results.

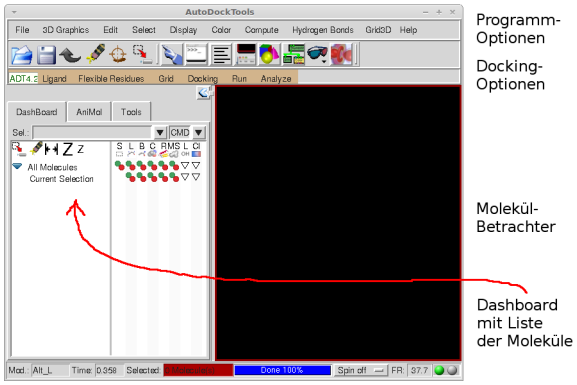

FIG. 5. The main window of ADT; most of the analysis can be performed starting from here.

Before you start a new calculation on your own, you may view the results from a typical docking calculation, which somebody performed previously. To do so, open the file `ind4.dlg` in a text editor first. This contains the entire output from a docking calculation. This includes information about the algorithm used, parameter settings, analysis of the ligand, and at the end of the file, the structures and binding strengths of the predicted positions (poses) of the docked ligand.

It is clearer to display the results graphically. In ADT, select *Analyze – Dockings – Open* and open the same file again. A message will pop up indicating how many individual ligand positions were found. It is possible to view the individual positions with *Analyze – Conformations – Play*, although this is not very helpful yet because the receptor structure is missing at the moment. A better overview of the calculated positions and their energies can be opened with *Analyze – Conformations – Load*. The example has two clusters of placements, one with eight and the other with two individual ligand positions. Click on the individual solutions to obtain some data about the positions and to rotate the ligand into the particular position.

More information about the distribution of ligand positions among the clusters can be obtained with *Analyze – Clusterings – Show*. Now, inspect the geometrical position of the ligand in the receptor, for the different clusters. To do so, load the protein structure with *Analyze – Macromolecule – Open*, file name `hsg1.pdbqt`. Check if the ligand really lies in the binding pocket of the receptor. Recall that you can switch between the ten ligand positions with *Analyze – Conformations – Load*.

Many different analyses could follow. The results from docking that you have at hand may already be sufficient, or you may wish to repeat the calculation with another choice of parameters, or rather to determine the specific interactions with the binding pocket in order to design even-stronger-binding ligands. Alternatively, one could use the geometries of the receptor–ligand complex as a starting point for other calculations such as MD simulations. The exact goals and means of a docking study depend on the properties of the receptor.

In the following, you will perform a rather typical docking calculation to predict the geometry of a protein–ligand complex.

C. Prepare the receptor

The first step is to prepare the receptor binding pocket. Since a typical study will dock many different ligands to a common receptor, the preparation may be somewhat more laborious; the idea is that these steps need to be performed only once, prior to the actual docking, and the additional effort pays of later as it will make the docking of each ligand easier or faster. In this step, a known three-dimensional structure of a target protein will be converted into a `.pdbqt` file, which is needed for an Autodock calculation.

As a structure of the receptor, we will use the X-ray crystal structure of the HIV protease I from the complex with indinavir, which is stored in the PDB database under PDB ID 1HSG. (You can also download this structure file yourself from `www.rcsb.org`.) Clearly, this will make the calculation much easier because the receptor is already in the optimal conformation to bind indinavir. We say that we re-dock a known ligand, instead of a "blind docking" of a new ligand, and an obvious advantage is that we already know where the binding pocket lies. The procedures for blind docking would be the same, however, except that critical analysis of the receptor structure would be needed to find a potential binding pocket.

Open the file `1HSG.pdb` with *File – Read Molecule*. (Alternatively, you may first view the PDB file with VMD or a text editor). Note that the crystal structure still contains co-crystallized water molecules and the ligand, and we will need to remove both, in order to perform docking calculation with an empty binding pocket. To do this, first select the molecules to be removed: *Select – Select From String*. In the window that opens, select `HOH*` under *Residue*, and select `*` under *Atom*, and add them; then select `MK1*` under *Residue*, and select `*` under *Atom*, and add them. In the viewing window, the ligand and the water molecules are marked in yellow. Identify the residue names of the molecules to remove (ligand and waters), and select them. Delete the selected atoms with *Edit – Delete – Delete atom set*; this operation cannot be reverted!

Now we have the structure of the receptor, but there are no hydrogen atoms, as is usual in X-ray crystal structures. Add these by *Edit – Hydrogens – Add*. Finally, the receptor shall be converted into a `.pdbqt` file, and also atomic charges shall be obtained. Click *Grid – Macromolecule – Choose* and pick `1HSG` (probably the only choice if no other structure files are open). In the dialog, select `1HSG.pdbqt` as the file name. Open this file with a text editor to see that the format is similar to a `.pdb` file. Note that there are atomic charges in addition to the usual information in a PDB file, and that ADT has only considered polar hydrogen atoms.

The preparation of the receptor will be completed at a later stage with the creation of the potential files, but the ligand molecule has to prepared before that. Click *Hide* (in the dashboard at the bottom of the ADT window) for the representation of 1HSG in order to clear the display.

D. Prepare the Ligand

Like for the receptor, it is required to specify topology files, atom types and atomic charges for the ligand molecule. Note that the same steps would be necessary to prepare an MD simulation; the difference is that with Autodock, they are performed on a simpler, and faster, level of modeling. In addition, it must be determined which dihedral angles of the ligand should be considered flexible. In contrast, the receptor structure is considered completely rigid in this exercise.

Start by loading a PDB file for Indinavir: *Ligand – Input – Open*. Select PDB as the file format before you can open the file `ind.pdb`. A message appears in the display window saying that ADT assigned Gasteiger charges and removed non-polar hydrogens.

Next, select the freely rotatable bonds with *Ligand – Torsion Tree – Choose Root* and *Ligand – Torsion Tree – Choose Torsions*. Accept the suggested selection criteria to define rotatable bonds. Alternatively, test how the selection of bond changes if you allow rotations around amide bonds. The choice of the "root atom" only affects the computational efficiency.

In order to make the ligand molecule more rigid in the docking, it is possible to set the maximum number of flexible bonds under *Torsion Tree – Set Number of Torsions*. This value should remain at the maximum of 14 (or 16 with amide bonds) however. Save the definition of the ligand with *Ligand – Output – Save as PDBQT* under the file name `ind.pdbqt`. For a more realistic study of hundreds of ligands, these steps can be automated in a script. In addition, there are various possibilities to automatically transfer chemical structural formulas directly into 3D structures. If you only have few ligands to prepare, manual preparation is good enough.

E. Create the interaction potentials of the receptor

Autodock considers the structure of the receptor in two ways in fact: partly as an atomistic model, and partly by way of potential functions for non-bonded interactions – van der Waals and electrostatics. The purpose of this is to accelerate the calculation of energy during docking of every ligand: the potential around the receptor is always the same, so it is enough to calculate every potential once at the beginning of the study, and use it in the calculation of all of the (possibly many) ligands. Autodock represents these potentials with a grid model: the potentials are computed for points that form a cubic lattice, and are interpolated between them.

The setup of potentials requires the knowledge of the atom types present in the ligand(s), and these are easy to find out if the ligand is already loaded in ADT. Pick `ind` under *Grid – Set Map Types – Choose Ligand* for indinavir.

Now, it has to be specified which part of the receptor should be considered as the binding region (pocket). Without any a priori knowledge, you can specify a box that includes the entire receptor molecule, but this leads to very slow calculations. For our exercise, give the box dimensions as 60, 60, 66 and a grid point spacing of 0.375. The center of the box shall be 16.0, 24.0, 4.0. Check whether the box contains the empty binding pocket of the enzyme, and adjust the location or size of the box accordingly. Close the grid window with *File – Close Saving Current*, otherwise your selection will be lost. Save the box definition with *Grid – Output – Save GPF* under the file name `1hsg.gpf`. You can check the new file with *Grid – Edit GPF*.

This grid parameter file shall be given to the program Autogrid in order to have the potentials files calculated. Switch to the console in the directory that contains all of the previously created files. Adjust the paths and filenames accordingly, and start the calculation of potentials with

```
autogrid4 -p GPF filename -l output filename &
```

This may take several minutes. Inspect the log file and make sure that a series of `.map` files were created (one for each atom type of ligand, one for electrostatics, etc.).

F. Run the docking calculation

At this point, the receptor and ligand files shall be selected for docking. Under *Docking – Macromolecule – Set Rigid Filename*, select the file you created, `1HSG.pdbqt`, and then under *Docking – Ligand – Choose*, select the ligand `ind`, for which a summary will be displayed again. Then set the parameters of the calculation under *Docking – Search Parameters – Genetic Algorithm* to choose one of Autodock's docking algorithms. The details of the calculation can be defined in the following window. The values that we will use are in fact too small values for a realistic application, in order to accelerate the calculation. Set *Number of GA Runs* to 20, *Population Size* to 100 and the number of energy evaluations to *short* (250,000). The remaining parameters server fine adjustments of the genetic algorithm and may be left at their default settings. Save the settings with *Docking – Output – Lamarckian GA* under the file name `1hsg-ind.dpf`. Inspect the resulting docking parameter file with *Docking – Edit DPF*.

Switch to the text console and start the actual Autodock calculation, which may take a few minutes, with

```
autodock4 -p 1hsg-ind.dpf -l 1hsg-ind.dlg
```

When the calculation is complete, there will a log file that contains the docked ligand position(s).

G. Analyze

Visualize the results from the docking calculation with *Analyze – Dockings – Open*. Save the best docking suggestion as a PDB file; the easiest way is to copy the relevant `ATOM` lines from the `.dlg` file. (Unfortunately, the coordinate export function of Autodock produces PDB files incompatible with VMD or other modeling programs).

Check:

- How many of the suggested ligand positions are within the binding pocket? Is this surprising?
- How many clusters of positions have been found? Is the number of dockings attempts enough for good clustering?
- How do the energies of the placements differ?
- What is the best predicted free enthalpy of binding? Is it in a range expected for a pharmaceutical inhibitor?
- How different is your calculated ligand position from the X-ray crystal structure?
- Which important interactions (contacts) between the ligand and the binding pocket can be identified?

VI. CALCULATION OF ELECTROSTATICS

A. Electrostatic potential on the molecular surface

In this expercise, an implicit solvent model will be used to calculate free energies of solvation and electrostatic potentials on surfaces of molecules. This could be performed with fully atomistic models as well though, but it would require exremely long simulations because the results woule have to be averaged over an overwhelming number of the relevant orientations of water molecules.

In an implicit solvent model, there is no simulation box filled with explicit, individual water molecules. Instead, rather complex potential energy functions are added to the force field, and they introduce forces on the atoms of the solute that approximate the forces that real water would exert on the solute. Thus, the dynamics of the solute should be the same as if the solute was embedded in a real, equilibrated solvent.

While so-called generalized Born models are popular for use in MD simulations, continuum dielectric models are usually used for static structures. These are based on the Poisson–Boltzmann equation, which is a physico-chemical extension of the Poisson equation from the theory of electrostatics. For a molecular structure, the energy (and forces or potentials) is calculated as if the molecule consisted of a nonconducting material with a low dielectric constant, in which point charges are embedded. This molecular model is immersed in a dielectric with a high dielectric constant. Despite the approximations involved in this model (dielectric constants are actually macroscopic quantities without molecular correspondences, partial charges are not well defined, etc.), the electrostatic potentials and solvation energies of molecules may be obtained with acceptable accuracy, provided that a good empirical parameterization is available.

B. The APBS software

You may read the description of a typically used lattice model in `PBS.pdf` for preparation. We use the freely available program APBS (`http://www.poissonboltzmann.org/apbs`). The calculations of the molecular surface and of electric potentials require the knowledge of atomic radii and of atomic charges, respectively. View these in the file `/usr/local/run/pdb2pqr-linux-bin64-2.0.0/dat/AMBER.DAT`. The format specifies:

```
<RESIDUE NAME> <ATOM NAME> <CHARGE in e> <RADIUS in Å>
```

First, combine this information with the atom coordinates in a `.pdb` file using the program `pdb2pqr`:

```
pdb2pqr --assign-only --ff=AMBER k2.pdb k2.pqr
```

View the resulting `.pqr` file with a text editor, and check if there are any warnings; also, the total charge is displayed at the top. If it is +4 and there are no warnings, it means that all charges have been read correctly.

Now, visualize the charge distribution by VMD; open the file `k2.pqr`. Find the interface to APBS under *Extensions – Analysis – APBS Electrostatics*. Go to the settings under *edit*, and set the entry "Working Directory" to point to your current working directory. Click *OK* and then *Run APBS*. In the following dialog, click *OK* and close the window *APBS Tool*. Create a new representation under *Graphical Representations*, and set *Coloring = Volume* and *Drawing = Surf*. Go to the Trajectory tab and set the range of the color scale to −10 through 10. You can make the visualization even more vivid by selecting *Transparent* for 'Material' and creating another representation in *Licorice*.

Judge on the following problem solely on the basis of your calculation: What will be the most likely orientation of the peptide when inserted into a lipid bilayer? Does that correspond to your experience and results from Sec. III?

C. Free Solvation Energy

Finally, calculate the electrostatic contribution to the solvation free energy of K2. Edit the input file `apbs.in` – replace the capitalized words with appropriate numerical values. Save the file and run the standalone APBS program.

```
apbs apbs.in
```

What is the value of the free energy of solvation?

D. Comparison with a mutant

Perform the same procedure with a slightly modified peptide: exchange two lysines in the K2 sequence for leucines. Compare the results that you obtained for K2 and for its mutant.

VII. 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 C, I D 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
gmx mdrun -deffnm em_steep

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

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

B. Free energies with 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 $+180°$. We will divide this interval into steps of 5°, resulting in 73 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

Create a new directory for the US simulations. From the directory where you have run the equilibration, copy the topology file (`.top`) and the final structure file (`.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 25; 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 73 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 7.3 ns. From the known computational cost of the equilibration, you can get a rather accurate estimate of how long the US will take.

3. Analyze

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 $(for ((i=-180; i<=180; i+=5)); do echo -n "GRAD_$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). **Possible kJ–kcal chaos here! If the resulting curve is ugly, multiply the force constant in metafile by 4.184!**

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`
- `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 kJ/mol **(or 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. 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?

C. 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.

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

After that, carry out the actual metadynamics simulation:

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

3. Analyze

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

D. Hamiltonian Replica Exchange

1. *Intro*

This method, also called 'HREX' for short, serves the same purpose as metadynamics – to discover all relevant structures of the molecular system within a shorter simulation time – but the approach is very different. In HREX, the molecular system is simulated in multiple copies, so that several simulations of the same molecule are running simultaneously; these simulations are called the replicas. The HREX simulation follows this protocol:

- First, a *hot region* is defined as a relatively small part of the molecular system. The search for relevant structures shall focus to the *hot region*.
- Each replica uses a different force field: the force field parameters (nonbonded interactions and dihedral angles) *in the hot region* are scaled by a factor f, $0 < f \leq 1$. This means that these interactions become weaker in the hot region, and free energy barriers are reduced. Each replica uses a different factor, and one replica is simulated with the unperturbed, original force field – thus, $f = 1$ for this replica.
- During the simulation, it is attempted in regular intervals (e.g., after every 1000th step) to exchange the coordinates and velocities between a pair of replicas (hence 'replica exchange'). In each of such attempts, the probability of exchange of replicas i and j is calculated as

$$\mathcal{P} = \exp\left[\frac{-V_i(r_j) + V_i(r_i) - V_j(r_i) + V_j(r_j)}{k_\mathrm{B}T}\right],$$

  where V_i and V_j are potential energy functions with force field parameters corresponding to replicas i and j, and r_i and r_j are the current coordinates in these replicas. Then a random number $0 < \rho < 1$ is generated, and if $\mathcal{P} > \rho$, then the coordinates and velocities of the replicas i and j are exchanged. If $\mathcal{P} < \rho$, no action is taken.

Despite the exchange events, correct canonical probabilities of conformations are still guaranteed in all of the replicas. The exchange events allow for the individual sets of coordinates to move between the replicas, and by virtue of that, even rather high energy barriers may be overcome via a detour through the simulations with energies scaled down. Thus, the canonical probability distribution in the unperturbed replica will establish much faster than in a standard MD simulation. See Fig. 6 for a typical pattern of coordinate sets being exchanged among replicas.

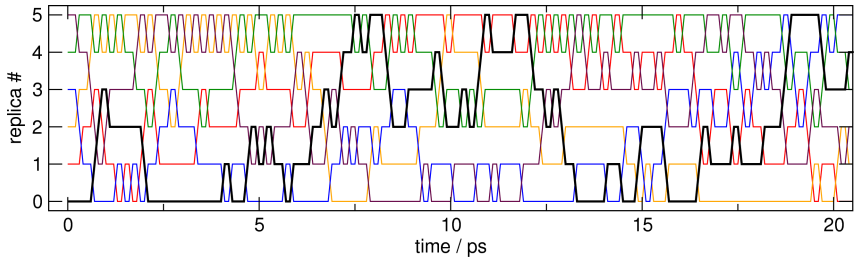

FIG. 6. History of a HREX simulation with 6 replicas, with exchange of coordinates&velocities attempted every 0.1 ps.

The idea of HREX is that any energy barriers are reduced because the underlying force field terms are scaled down. Therefore, the energy barriers will be overcome faster. With the equation for the rate of a process

$$k = A \cdot \exp\left[-\frac{E_\mathrm{A}}{k_\mathrm{B}T}\right]$$

in mind, it becomes apparent that a process may be accelerated either by reducing the activation energy – which is the case with HREX – or by raising the temperature. That is why we talk about *heating up* a part of the system, or the *hot region*, even though all simulations are performed at a normal temperature (usually 300 K). Typically, an 'effective' temperature of T/f is specified rather than the value of f itself.

An important difference between HREX and metadynamics is that no collective variables are needed to run HREX. It may be practical to introduce some collective variables in the analysis, but this has obviously no influence on the course of the actual simulation. Still, there is one choice that we have to make prior to running the simulation – namely, the *hot region* has to be defined, and that determines which force field terms will be scaled down. Typically, this is a small part of the molecular system. In this exercise, the dipeptide molecule will be the *hot region*, and its nonbonded terms and dihedral angles will be scaled down, while the solvent will exhibit full interactions in all replicas. Thus, the force field parameters of the water and possibly of the counterion will remain untouched.

2. Run it

We will simulate 4 replicas, with scaling factors of 1, 0.75, 0.56 and 0.42. What are the effective temperatures T/f?

First, you need to pre-process the topology with the `grompp` command to obtain a topology file that includes all of the force field parameters:

```
gmx grompp -f md.mdp -c npt.gro -p topol.top -pp processed.top
```

As mentioned above, all nonbonded interactions of the dipeptide shall be scaled. For this, the processed topology for the dipeptide must be modified slightly. To do so, add an underscore (`_`) to the atom type of every atomic of the dipeptide in the `[ atoms ]` section. For example,

```
1 HC 1 ACE HH31 1 0.1123 1.008; qtot 0.1123
```

shall be changed to

```
1 HC_ 1 ACE HH31 1 0.1123 1.008; qtot 0.1123
```

From this modified topology, it is now possible to create 4 different topology files, one for each value of the scaling factor. Do this by calling the following command, in which you replace `SCALE` by the desired value of the scaling factor, and replace `ORD` by values 0, 1, 2 and 3:

```
plumed partial_tempering SCALE < processed.top > topolORD.top
```

For example, `partial_tempering` with `1` for `SCALE` shall generate a non-scaled topology file `topol0.top`, and `partial_tempering` with `0.4` for `SCALE` shall generate a 'most scaled' topology file `topol3.top`.

NEW!

```
. /usr/local/run/gromacs-4.6.7-plumed-hrex/bin/GMXRC
```

Check in the `md.mdp` file that no barostat is applied, as pressure scaling may lead to instabilities (crashes) in replica exchange simulations. For each replica, create the corresponding `.tpr` file:

```
grompp -maxwarn 1 -f md.mdp -c npt.gro -p topolORD.top -o topolORD.tpr
```

Finally, create an empty file named `plumed.dat`, and start the HREX simulation with 4 replicas:

```
export PATH=/usr/local/run/openmpi-1.10.1-gcc-9.1.0/bin:$PATH
export LD_LIBRARY_PATH=/usr/local/run/openmpi-1.10.1-gcc-9.1.0/lib:$LD_LIBRARY_PATH
export LD_LIBRARY_PATH=/usr/local/run/gcc-9.1.0/lib64:$LD_LIBRARY_PATH
export LD_LIBRARY_PATH=/usr/local/run/plumed-2.4.1-openmpi/lib:$LD_LIBRARY_PATH
mpirun -np 4 gmx_mpi mdrun -v -plumed plumed.dat -multi 4 -replex 100 -hrex
```

The `-replex` option specifies how frequently an exchange between the replicas should be attempted; here, they will be attempted quite frequently – after every 100th step of MD.

3. Analyze

We are only interested in the behavior of the unperturbed molecule, so only the data from the replica with $f = 1$ (effective temperature 300 K) are needed. The analysis is identical to that of a normal simulation. Note that you cannot observe any continuous dynamics, because the exchange events introduce 'film tears' in the MD trajectory.

It is of interest to generate a Ramachandran plot for the dipeptide. First, create time series of dihedrals $\varphi - \psi$ with

```
gmx rama -s topol0.tpr -f traj0.trr
```

Use a text editor to remove the header (probably the first 29 lines) from the output file `rama.xvg`. Then, use a script to create the corresponding 2D distribution, which will be saved in file `rama.histo`:

```
./make_histogram.py
```

The histogram is now in fact a Ramachandran plot, and can be visualized with another script, which saves an image file under name `rama-kcal.png`:

```
./rama-histo-to-png.sh
```

The color encodes the Helmholtz free energy in kcal/mol (not Gibbs free energy because volume and not pressure is constant). If the color scale is too narrow (or too wide), you change customize the option `cbrange` in file `gnuplot-deltag.in`, and then run `./rama-histo-to-png.sh` again.

Which dominant conformations of the peptide do you see in the Ramachandran plot? Which has the lowest free energy and thus represents the global minimum? What are the energy barriers between the individual conformations? Do the results agree with those from the metadynamics simulation?

VIII. 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. 7. 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.

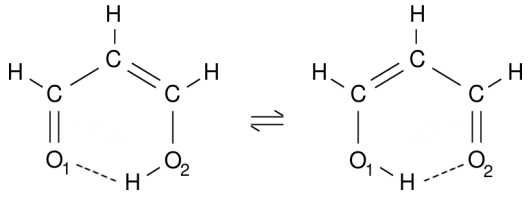

FIG. 7. Malonaldehyde – intramolecular proton transfer.

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. VII, 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 ...
gmx mdrun -deffnm steep
gmx grompp -f heat.mdp ...
gmx mdrun -deffnm heat
gmx grompp -f equil.mdp ...
gmx mdrun -deffnm equil
```

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 equil.gro
```

and be provided to `grompp_d` 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 = "/usr/local/run/gromacs-dftbplus/share/gromacs/top/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

```
. /usr/local/run/gromacs-dftbplus/bin/GMXRC
```

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 equil.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 equil.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. VII C. 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?