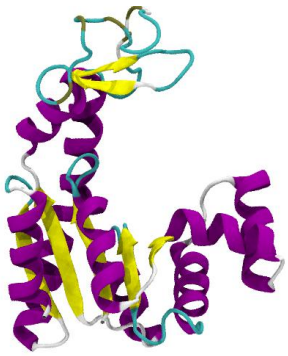
AdKGromacsTutorial Documentation

Release 2.0.1

Sean Seyler and Oliver Beckstein

Jul 03, 2018

Contents



Gromacs 5.x, 2016, 2018

Tutorial 2.0.1

Date Jul 03, 2018

CHAPTER 1

Objective

Perform an all-atom molecular dynamics (MD) simulation—using the Gromacs MD package—of the apo enzyme adenylate kinase (AdK) in its open conformation in a physiologically realistic environment, and carry out a basic analysis of its structural properties in equilibrium.

CHAPTER 2

Tutorial files

All of the necessary tutorial files can be found on GitHub in the Becksteinlab/AdKGromacsTutorial/tutorial directory, which can be easily obtained by git-cloning the repository:

```
git clone https://github.com/Becksteinlab/AdKGromacsTutorial.git
```

CHAPTER 3

Workflow overview

For this tutorial we'll use Gromacs (versions 5, 2016, 2018 should work) to set up the system, run the simulation, and perform analysis. An initial structure is provided, which can be found in the `tutorial/templates` directory, as well as the MDP files that are necessary for input to Gromacs. The overall workflow consists of the following steps:

3.1 Directory organization

The workflow for setting up, running, and analysing a simulation consists of multiple and rather different steps. It is useful to perform these different steps in separate directories in order to avoid overwriting files or using wrong files.

3.1.1 Create working directories

It is recommended that the following directory structure be used, as the tutorial steps through them sequentially:

```
coord/
top/
solvation/
emin/
posres/
MD/
analysis/
```

Create these directories using:

```
mkdir top solvation emin posres MD analysis
```

Description of directories

`coord` original PDB (structural) files

`top` generating topology files (`.top`, `.itp`)

solvation adding solvent and ions to the system

emin performing energy minimization

posres short MD simulation with position restraints on the heavy protein atoms, to allow the solvent to equilibrate around the protein without disturbing the protein structure

MD MD simulation (typically, you will transfer the `md.tpr` file to a supercomputer, run the simulation there, then copy the the output back to this trajctory)

analysis post-processing a production trajectory to facilitate easy visualization (i.e., using VMD); analysis of the simulations can be placed in (sub)directories under analysis, e.g.

```
analysis/RMSD
analysis/RMSF
...
```

The subdirectories depend on the specific analysis tasks that you want to carry out. The above directory layout is only a suggestion, but, in practice, some sort of ordered directory hierarchy will facilitate reproducibility, improve efficiency, and maintain your sanity.

Important: The command snippets in this tutorial assume the directory layout given above as the workflow depends on each step's being carried out *inside the appropriate directory*. In particular, *relative* paths are used to access files from previous steps. It should be clear from context in which directory the commands are to be executed. If you get a `File input/output error` from **grompp** (or any of the other commands), first check that you are able to see the file by just doing a `ls ../path/to/file` from where you are in the file system. If you can't see the file then check (1) that you are in the correct directory, (2) that you have created the file in a previous step.

3.1.2 Obtain starting structure

Note: The starting structure `coord/4ake_a.pdb` has been provided as part of the tutorial package, so the instructions that follow are optional for this tutorial. However, these steps provide an idea of what may be required in obtaining a suitable starting structure for MD simulation.

---

1. Download 4AKE the Protein Data Bank (PDB) through the web interface
2. Create a new PDB file with just chain A

   Modify the downloaded PDB file. For a relatively simple protein like AdK, one can just open the PDB file in a text editor and remove all the lines that are not needed.(For more complex situations, molecular modeling software can be used.)

- Remove all comment lines (but keep TITLE, HEADER)
- Remove all crystal waters (HOH)[1]
- Remove all chain B ATOM records.
- Save as `coord/4ake_a.pdb`.

[1] Often you would actually want to retain crystallographic water molecules as they might have biological relevance. In our example this is likely not the case and by removing all of them we simplify the preparation step somewhat. If you keep them, **gmx pdb2gmx** in the next step will actually create entries in the topology for them.

3.2 Generate a solvated protein system

3.2.1 Generate a topology

Using the modified PDB file (chain A of 4AKE with crystal waters removed), generate a topology file for the CHARMM27 force field together with the TIP3P water model using the gmx pdb2gmx tool:

```
cd top
gmx pdb2gmx -f ../coord/4ake_a.pdb -o protein.pdb -p 4ake.top -i protein_posre.itp -
↪water tip3p -ff charmm27
```

Note: Total charge -4.000e (in the next step we will add ions to neutralize the system; we need a net-neutral system to properly handle electrostatics)

3.2.2 Solvate the protein

Adding water

Create a simulation box with gmx editconf and add solvent with gmx solvate:

```
cd ../solvation
gmx editconf -f ../top/protein.pdb -o boxed.pdb -c -d 0.5 -bt dodecahedron
gmx solvate -cp boxed.pdb -cs spc216 -p ../top/4ake.top -o solvated.pdb
```

Attention: In order to reduce the system size and make the simulations run faster we are choosing a very tight box (minimum protein-edge distance 0.5 nm, `-d 0.5`); for simulations you want to publish this number should be 1.2...1.5 nm so that the electrostatic interactions between copies of the protein across periodic boundaries are sufficiently screened.

`gmx solvate` updates the number of solvent molecules ("SOL") in the topology file (check the `[ system ]` section in `top/system.top`)[1].

Adding ions

Ions can be added with the gmx genion program in Gromacs.

First, we need a basic TPR file (an empty file is sufficient, just ignore the warnings that gmx grompp spits out by setting `-maxwarn 10`), then run gmx genion (which has convenient options to neutralize the system and set the concentration (check the help!); gmx genion also updates the topology's `[ system ]` section if the top file is provided[1]; it reduces the "SOL" molecules by the number of removed molecules and adds the ions, e.g. "NA" and "CL").

```
touch ions.mdp
gmx grompp -f ions.mdp -p ../top/4ake.top -c solvated.pdb -o ions.tpr
printf "SOL" | gmx genion -s ions.tpr -p ../top/4ake.top -pname NA -nname CL -neutral␣
↪-conc 0.15 -o ionized.pdb
```

[1] The automatic modification of the top file by gmx solvate and gmx genion can become a problem if you try to run these commands multiple times and you get error messages later (typically from gmx grompp) that the number of molecules in structure file and the topology file do not agree. In this case you might have to manually delete or adjust the corresponding lines in `system.top` file.

The final output is `solvation/ionized.pdb`. Check visually in VMD (but note that the dodecahedral box is not represented properly).[2].

3.3 Energy minimization

In order to remove "clashes" (i.e. close overlaps of the LJ cores) we perform an *energy minimization*: Instead of a MD simulation we use an algorithm to change the coordinates in such a way as to reduce the total potential energy.

3.3.1 Set up and generate the run file

First, we will copy a file from the templates folder (provided in this tutorial) that tells Gromacs MD program *how* to do energy minimization:

```
cp ../templates/em.mdp .
```

Tip: Have a look at the MDP file to get a feel for what kinds of settings can be adjusted to suit one's needs. Individual parameters are explained in more detail in mdp options.

The `*.mdp` file contains the settings that dictate the nature of the simulation. For energy minimization, we will use the simple *steepest descent* minimizer (`integrator = steep` in `em.mdp`, which runs in parallel). Use gmx grompp (the GROMacs PreProcessor) to generate the run input file (TPR) from the run parameter file (MDP), coordinate file (the solvated system with ions; PDB), and the topology (TOP):

```
cd ../emin
gmx grompp -f em.mdp -c ../solvation/ionized.pdb -p ../top/4ake.top -o em.tpr
```

3.3.2 Perform energy minimization

The energy minimization is performed with gmx mdrun but by using the appropriate `integrator` option in the Run control options in the MDP file it has been instructed to do a energy minimization:

```
gmx mdrun -v -s em.tpr -deffnm em -c em.pdb
```

Ideally, the maximum force *Fmax* (gradient of the potential) should be < 1e+03 kJ mol^{-1} nm^{-2} (but typically anything below 1e+05 kJ mol^{-1} nm^{-2} works). See the screen output or the `em.log` file for this information.

Tip: The final frame of minimization (the structure in `em.pdb`) can be used as the input structure for further minimization runs. It is common to do an initial energy minimization using the efficient steepest descent method and further minimization with a more sophisticated method such as the *conjugate gradient* algorithm (`integrator = cg`) or the Newton-like *Broyden-Fletcher-Goldfarb-Shanno* (`integrator = l-bfgs`) minimizer. For details, see Run control options in the MDP file.

[2] For notes on how to visualize MD systems with VMD see the notes on *Trajectory visualization*.

3.4 Position-restrained equilibration

We first perform a short MD simulation with harmonic position restraints on the heavy protein atoms. This allows the solvent to equilibrate around the protein without disturbing the protein structure. In addition, we use "weak coupling" temperature and pressure coupling algorithms to obtain the desired temperatue, $T = 300$ K, and pressure, $P = 1$ bar.

3.4.1 Set up and generate the run file

We must first tell Gromacs *how* to perform our equilibration run in the same way that we did for the energy minimization step. This step requires the `top/protein_posres.itp` file with the default value for the harmonic force constants of 1000 kJ mol^{-1} nm^{-2}. The position restraints are switched on by setting the `-DPOSRES` flag in the `posres.mdp` file (see mdp options).

Create the run input (TPR) file, using the *energy minimized system* as the starting structure:

```
cd ../posres
cp ../templates/posres.mdp .
gmx grompp -f posres.mdp -o posres.tpr -p ../top/4ake.top -c ../emin/em.pdb -maxwarn 2
```

The mdp file contains cut-off settings that approximate the native CHARMM values (in the CHARMM program).

Weak (Berendsen) coupling is used for both temperature and pressure to quickly equilibrate. The protein and the solvent (water and ions) are coupled as separate groups. Gromacs provides a range of groups automatically (run `gmx make_ndx -f TPR` to see them) and we use the groups `Protein` and `non-Protein` (these particularly groups work since roughly Gromacs 4.5.3). If the standard groups do not work then you will have to create the groups yourself using `gmx make_ndx -f TPR -o md.ndx` (which would save them in a file `md.ndx`) and supply it to `gmx grompp -n md.ndx`.

3.4.2 Perform equilibration

Run the position restraints equilibration simulation with gmx mdrun:

```
gmx mdrun -v -stepout 10 -s posres.tpr -deffnm posres -c posres.pdb
```

Attention: Here the runtime of 10 ps is too short for real production use; typically 1 to 5 ns are used.

Generate a centered trajectory in the primary unitcell

In order to visually check your system, first create trajectory with all molecules in the primary unitcell (`-ur compact`; see also below the more extensive notes on *Trajectory visualization*):

```
echo "System" | gmx trjconv -ur compact -s posres.tpr -f posres.xtc -pbc mol -o␣
↪posres_ur.xtc
```

Visually check centered trajectory in VMD

If you have VMD installed then you can quickly visualize the system with the command

```
vmd ../emin/em.pdb posres_ur.xtc
```

If you don't have a **vmd** command available on the command line then launch VMD, load the `emin/em.pdb` file (*File → New Molecule...*), highlight your molecule 1 ("em.pdb") and load the `posres/posres_ur.xtc` trajectory into your *molecule 1*, *File → Load Data Into Molecule*. You should see that the first frame (from the energy minimization) looks as if the water is in a distorted box shape whereas all further frames show a roughly spherical unit cell (the rhombic dodecahedron).

3.5 Equilibrium molecular dynamics

3.5.1 Set up the production run

As usual, we must tell Gromacs what it will be doing using gmx grompp before we can perform our production simulation. Since we want to start our run where we left off (after doing equilibration), we prepare the TPR input file based on the last frame of the *Position-restrained equilibration* with **gmx grompp**:

```
cd ../MD
cp ../templates/md.mdp .
gmx grompp -f md.mdp -p ../top/4ake.top -c ../posres/posres.pdb -o md.tpr -maxwarn 3
```

The `md.mdp` file uses different algorithms from the *Position-restrained equilibration* for the temperature and pressure coupling, which are known to reproduce the exact *NPT* ensemble distribution.

3.5.2 Run the simulation

Run the simulation as usual with gmx mdrun:

```
gmx mdrun -v -stepout 10 -s md.tpr -deffnm md -cpi
```

This will automatically utilize all available cores and GPUs if available. The `-cpi` flag indicates that you want Gromacs to continue from a previous run. You can kill the job with `CONTROL-C`, look at the output, then continue with exactly the same command line

```
gmx mdrun -v -stepout 10 -s md.tpr -deffnm md -cpi
```

(Try it out!). The `-cpi` flag can be used on the first run without harm. For a continuation to occur, Gromacs needs to find the checkpoint file `md.cpt` and all output files (`md.xtc`, `md.edr`, `md.log`) in the current directory.

3.6 Trajectory visualization

Analysis are normally performed locally on a workstation, i.e. copy back all the files from the supercomputer to your local directory.

A typical analysis tasks reads the trajectory (XTC) or energy (EDR) file, computes quantities, and produces data files that can be plotted or processed further, e.g. using Python scripts. A strength of Gromacs is that it comes with a wide range of tools that each do one particular analysis task well (see the Gromacs manual and the Gromacs documentation).

3.6.1 Keeping the protein in one piece

If you just look at the output trajectory `md.xtc` in VMD then you will see that the protein can be split across the periodic boundaries and that the simulation cell just looks like a distorted prism. You should *recenter* the trajectory so that the protein is at the center, *remap* the water molecules (and ions) to be located in a more convenient unitcell representation.

We will use the gmx trjconv tool in Gromacs to center and remap our system.

Tip: **`gmx trjconv`** prompts the user with a number of questions that depend on the selected options. In the command line snippets below, the user input is directly fed to the standard input of **`trjconv`** with the `printf TEXT | gmx trjconv` "pipe" construct. In order to better understand the command, run it interactively without the pipe construct and manually provide the required information.

Center (`-center`) on the *Protein* and remap all the molecules (`-pbc mol`) of the whole *System*:

```
printf "Protein\nSystem\n" | gmx trjconv -s md.tpr -f md.xtc -center -ur compact -pbc␣
↪mol -o md_center.xtc
```

3.6.2 Pinning down a tumbling protein

It is often desirable to *RMS-fit* the protein on a reference structure (such as the first frame in the trajectory) to remove overall translation and rotation. In Gromacs, the gmx trjconv tool can also do more "trajectory conversion tasks". After (1) centering and remapping the system, we want to (2) RMS-fit (due to technical limitations in **`gmx trjconv`** you cannot do both at the same time).

RMS-fit (`-fit rot+trans`) to the protein *backbone* atoms in the initial frame (supplied in the TPR file) and write out the whole *System*:

```
printf "Backbone\nSystem\n" | gmx trjconv -s md.tpr -f md_center.xtc -fit rot+trans -
↪o md_fit.xtc
```

3.6.3 Check our modified trajectory

Visualize in VMD:

```
vmd ../posres/posres.pdb md_fit.xtc
```

Note: If you don't have a **`vmd`** command available on the command line then launch VMD, load the `posres/posres.pdb` file (*File* → *New Molecule...*), highlight your molecule 1 ("em.pdb") and load the `posres/md_fit.xtc` trajectory into your *molecule 1*, *File* → *Load Data Into Molecule*. You should see that the first frame (from the energy minimization) looks as if the water is in a distorted box shape whereas all further frames show a roughly spherical unit cell (the rhombic dodecahedron).

3.7 Analysis

A typical analysis tasks reads the trajectory (XTC) or energy (EDR) file, computes quantities, and produces datafiles that can be plotted or processed further, e.g. using Python scripts. A strength of Gromacs is that it comes with a wide range of tools that each do one particular analysis task well (see the Gromacs manual and the Gromacs documentation).

3.7.1 Basic analysis

As examples, we perform a number of common analysis tasks.

RMSD

The RMSD is the root mean squared Euclidean distance in *3N* configuration space as function of the time step,

$$\rho^{\mathrm{RMSD}}(t) = \sqrt{\frac{1}{N}\sum_{i=1}^{N}\left(\mathbf{r}_i(t) - \mathbf{r}_i^{\mathrm{ref}}\right)^2}$$

between the current coordinates $\mathbf{r}_i(t)$ at time *t* and the reference coordinates $\mathbf{r}_i^{\mathrm{ref}}$.

We compute the C_α **RMSD** with gmx rms with respect to the reference starting structure (the one used for creating the `md.tpr` file). Work in a separate analysis directory:

```
mkdir analysis/RMSD && cd analysis/RMSD
```

First we **create an index file** for the C_α atoms[1]. Use gmx make_ndx to create a file `ca.ndx` that contains the C_α atoms as an *index group*. Start **`make_ndx`** and use `md.tpr` as input; the output index file will be `ca.ndx`:

```
gmx make_ndx -f ../../MD/md.tpr -o CA.ndx
```

Use gmx make_ndx interactively by typing the following commands[2]:

```
keep 1
a CA
name 1 Calpha
q
```

(This sequence of commands only retains the "Protein" default selection, then selects all atoms named "CA", renames the newly created group to "Calpha", and saves and exits.)

You can look at `CA.ndx` and see all the index numbers listed under the heading `[ Calpha ]`.

Run **`gmx rms`**, using our newly defined group as the selection to fit and to compute the RMSD:

```
printf "Calpha\nCalpha\n" | gmx rms -s ../../MD/md.tpr -f ../../MD/md.xtc -n CA.ndx -
↪o rmsd.xvg -fit rot+trans
```

Note that the units are nm.

Plot the timeseries data in the `rmsd.xvg`[3].

[1] Actually, we don't need to create the index group for the C_α atoms ourselves because Gromacs automatically creates the group "C-alpha" as one of many default groups (other are "Protein", "Protein-H" (only protein heavy atoms), "Backbone" (N CA C), "Water", "non-Protein" (i.e. water and ions in our case but could also contain other groups such as drug molecule or a lipid membrane in more complicated simulations), "Water_and_ions". You can see these index groups if you just run gmx make_ndx on an input structure or if you interactively select groups in gmx trjconv, gmx rms, ...

However, making the "Calpha" group yourself is a good exercise because in many cases there are no default index groups for the analysis you might want to do.

[2] In scripts you can pipe all the interactive commands to gmx make_ndx by using the `printf ... | gmx make_ndx` trick:

```
printf "keep 0\ndel 0\na CA\nname 0 Calpha\nq\n" | \
     gmx make_ndx -f ../../MD/md.tpr -o CA.ndx
```

This will accomplish the same thing as the interactive use described above.

[3] If you use Python (namely NumPy and matplotlib) then you might want to use `gmx rms -xvg none` so that *no XVG legend information* is written to the output file

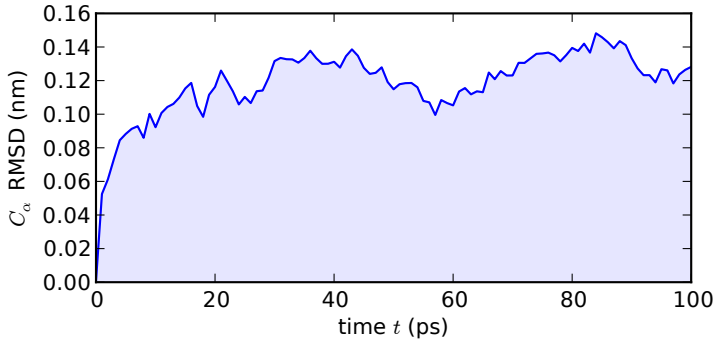


Fig. 1: Root mean square distance (RMSD) of the C_α atoms of AdK from the initial simulation frame.

RMSF

The residue root mean square fluctuation **RMSF** is a measure of the flexibility of a residue. It is typically calculated for the C_α atom of each residue and is then simply the square root of the variance of the fluctuation around the average position:

$$\rho_i^{\mathrm{RMSF}} = \sqrt{\left\langle (\mathbf{r}_i - \langle \mathbf{r}_i \rangle)^2 \right\rangle}$$

Use the `CA.ndx` file from the *RMSD* calculation with gmx rmsf

```
mkdir analysis/RMSF && cd analysis/RMSF
printf "Calpha\n" | gmx rmsf -s ../../MD/md.tpr -f ../../MD/md.xtc -n ../RMSD/CA.ndx -
↪o rmsf.xvg -fit
```

A plot[1] of ρ_i^{RMSF} versus residue number i shows the regions of high flexibility as peaks in the plot. Note that a 100-ps simulation might be too short to obtain a meaningful RMSF profile.

```
printf "Calpha\nCalpha\n" | \
    gmx rms -s ../../MD/md.tpr -f ../../MD/md.xtc -n CA.ndx \
            -o rmsd.xvg -fit rot+trans -xvg none
```

and you can easily read the data with `numpy.loadtxt()`:

```
import matplotlib.pyplot as plt
import numpy

t,rmsd = numpy.loadtxt("rmsd.xvg", unpack=True)

fig = plt.figure(figsize=(5,2.5))
ax = fig.add_subplot(111)
fig.subplots_adjust(bottom=0.2)

ax.fill_between(t,rmsd, color="blue", linestyle="-", alpha=0.1)
ax.plot(t,rmsd, color="blue", linestyle="-")

ax.set_xlabel("time $t$ (ps)")
ax.set_ylabel(r"C$_\alpha$ RMSD (nm)")

fig.savefig("rmsd_ca.png", dpi=300)
fig.savefig("rmsd_ca.svg")
fig.savefig("rmsd_ca.pdf")
```

[1] To plot in Python, make sure to *not* write xvg legend information to the output file (using the `-xvg none` flag)

```
printf "Calpha\n" | \
    gmx rmsf -s ../../MD/md.tpr -f ../../MD/md.xtc -n  ../RMSD/CA.ndx \
             -o rmsf.xvg -fit -xvg none
```

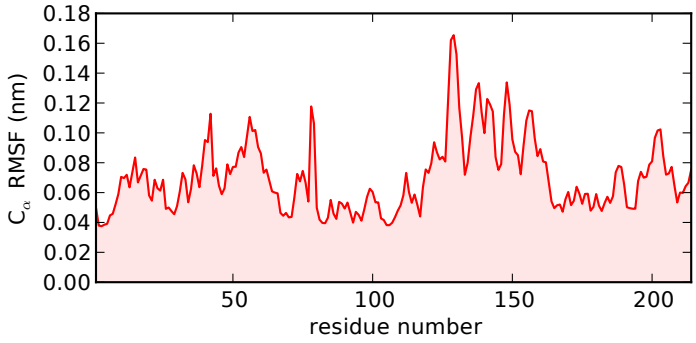


Fig. 2: Root mean square fluctuation (RMSF) of the C_α atoms of AdK.

Comparison with crystallographic B-factors

You can compare the RMSF to the isotropic atomic crystallographic B-factors, which are related by *[Willis1975]*

$$B_i = \frac{8\pi^2}{3}(\rho_i^{\mathrm{RMSF}})^2$$

(In this case you would want to calculate the RMSF for all heavy (i.e. non-hydrogen) atoms. You don't need to build and use a separate index file file: simply choose the default group "Protein-H" ("protein without hydrogens")).

Note: Gromacs RMSF are in units of nm and B-factors are typically measured in Å^2.

It is straightforward to write Python code that calculates the B-factor from the RMSF in `rmsf.xvg` and it is also easy to extract the B-factor ("temperatureFactor") from columns 61-66 in the ATOM record of a PDB file. (Left as an exercise. . .)

Distances

Distances can be a very useful observable to track conformational changes. They can often be directly related to real experimental observables such as NOEs from NMR experiments or distances from cross-linking or FRET experiments.

so that you can easily read the data with `numpy.loadtxt()`:

```
import matplotlib.pyplot as plt
import numpy

resid, rmsf = numpy.loadtxt("rmsf.xvg", unpack=True)

fig = plt.figure(figsize=(5,2.5))
ax = fig.add_subplot(111)
fig.subplots_adjust(bottom=0.2)

ax.fill_between(resid, rmsf, color="red", linestyle="-", alpha=0.1)
ax.plot(resid, rmsf, color="red", linestyle="-")

ax.set_xlabel("residue number")
ax.set_ylabel(r"C$_\alpha$ RMSF (nm)")
ax.set_xlim(resid.min(), resid.max())

fig.savefig("rmsf_ca.png", dpi=300)
fig.savefig("rmsf_ca.svg")
fig.savefig("rmsf_ca.pdf")
```

Here we calculate a simple distance between two C_α atoms as an approximation to the distance between two chromophores attached to the correspoding residues isoleucine 52 (*I52*) and lysine 145 (*K145*) used in a FRET experiment *[Henzler-Wildman2007]*.

First we need to create an index file containing the two groups:

```
mkdir -p analysis/dist/I52_K145 && cd analysis/dist/I52_K145
gmx make_ndx -f ../../../MD/md.tpr -o I52_K145.ndx
```

Use interactive commands like the following[1]

```
keep 0
del 0
r 52 & a CA
name 0 I52
r 145 & a CA
name 1 K145
q
```

to generate the index file `I52_K145.ndx`.

Then run gmx distance and compute the distance between the two atoms:

```
printf "I52\nK145\n" | gmx distance -s ../../../MD/md.tpr -f ../../../MD/md.xtc -n␣
↪I52_K145.ndx -o dist.xvg
```

The `dist.xvg` file contains the distance in nm for each time step in ps, which can be plotted[2].

(You can also use the centered and fitted trajectory `md_fit.xtc` as an input instead of `md.xtc` to make sure that the distance calculation does not contain any jumps due to periodic boundary effects, or use gmx mindist.)

See also:

[1] Note that one has to be careful when selecting residue ids in **`make_ndx`**. It is often the case that a PDB file does not contain all residues, e.g. residues 1–8 might be unresolved in the experiment and thus are missing from the PDB file. The file then simply starts with residue number 9. Gromacs, however, typically *renumbers residues so that they start at 1*. Thus, in this hypothetical case, a residue that might be referred to in the literature as "residue 100" might actually be residue 92 in the simulation ($N_{\rm res}^{\rm sim} = N_{\rm res}^{\rm PDB} - (\min N_{\rm res}^{\rm PDB} - 1)$). Thus, if you wanted to select the C_α atom of residue 100 you would need to select `r 92 & a CA` in **`make_ndx`**.

[2] To plot in Python, make sure to *not* write xvg legend information to the output file (using the `-xvg none` flag)

```
printf "I52\nK145\n" | \
     gmx distance -s ../../../MD/md.tpr -f ../../../MD/md.xtc \
                  -n I52_K145.ndx -o dist.xvg -xvg none
```

so that you can easily read the data with `numpy.loadtxt()`:

```
import matplotlib.pyplot as plt
import numpy

t,d,x,y,z = numpy.loadtxt("dist.xvg", unpack=True)

fig = plt.figure(figsize=(5,2.5))
ax = fig.add_subplot(111)
fig.subplots_adjust(bottom=0.2)

ax.fill_between(t, d, color="orange", linestyle="-", alpha=0.1)
ax.plot(t, d, color="orange", linestyle="-", label="I52-K145")

ax.set_xlabel("time $t$ (ps)")
ax.set_ylabel(r"C$_\alpha$ distance (nm)")
ax.legend(loc="best")

fig.savefig("d_I52_K145_ca.png", dpi=300)
fig.savefig("d_I52_K145_ca.svg")
fig.savefig("d_I52_K145_ca.pdf")
```

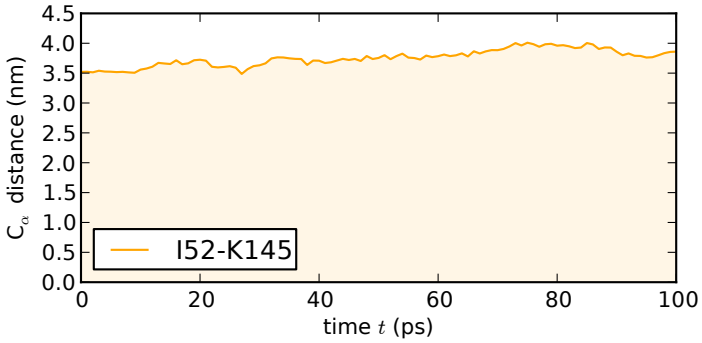


Fig. 3: Timeseries of the distance between the C_α atoms of I52 (NMP domain) and K145 (LID domain).

[Beckstein2009] for a discussion of FRET distances in AdK.

Radius of gyration

The radius of gyration measures the compactness of a protein structure.

$$R^2_{\mathrm{gyr}} = \frac{1}{M}\sum_{i=1}^{N} m_i(\mathbf{r}_i - \mathbf{R})^2$$

where $M = \sum_{i=1}^{N} m_i$ is the total mass and $\mathbf{R} = N^{-1}\sum_{i=1}^{N} \mathbf{r}_i$ is the center of mass of the protein consisting of N atoms.

The Gromacs tool gmx gyrate can be used to compute the radius of gyration for the whole protein (using the predefined "Protein" index group)

```
mkdir analysis/rgyr && cd analysis/rgyr
echo Protein | gmx gyrate -s ../../MD/md.tpr -f ../../MD/md.xtc -o gyrate.xvg
```

and the resulting time series in file `gyrate.xvg` can be plotted[1].

[1] To plot in Python, make sure to *not* write xvg legend information to the output file (using the `-xvg none` flag)

```
echo Protein | \
    gmx gyrate -s ../../MD/md.tpr -f ../../MD/md.xtc -o gyrate.xvg -xvg none
```

so that you can easily read the data with `numpy.loadtxt()`:

```
import matplotlib.pyplot as plt
import numpy

t,data,x,y,z = numpy.loadtxt("gyrate.xvg", unpack=True)

fig = plt.figure(figsize=(5,2.5))
ax = fig.add_subplot(111)
fig.subplots_adjust(bottom=0.2)

ax.fill_between(t,data, color="magenta", linestyle="-", alpha=0.1)
ax.plot(t,data, color="magenta", linestyle="-")

ax.set_xlabel("time $t$ (ps)")
ax.set_ylabel(r"protein $R_\mathrm{gyr}$ (nm)")

fig.savefig("rgyr.png", dpi=300)
fig.savefig("rgyr.svg")
fig.savefig("rgyr.pdf")
```

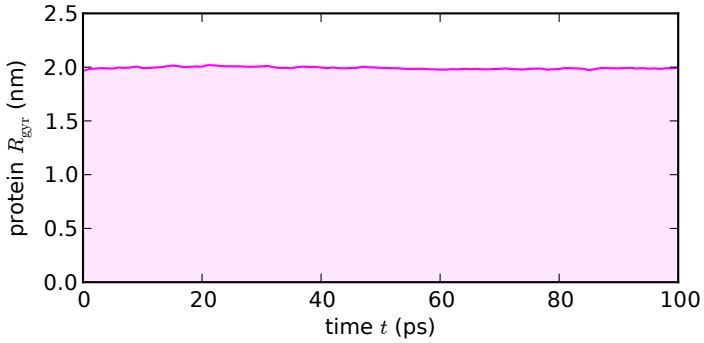


Fig. 4: Timeseries of the radius of gyration computed for the whole protein.

(In principle, R_{gyr} can indicate changes in conformation but the simulation time in our test trajectory is too short to reveal the large conformational transition that AdK can undergo *[Beckstein2009]*.)

3.7.2 More Gromacs tools

A number of interesting quantities and observables[1] can be calculated with Gromacs tools. A selection is shown below but you are encouraged to read the Gromacs manual and the Gromacs documentation to find out what else is available.

Selection of Gromacs analysis tools

The full list of Gromacs commands contains 98 different tools. A small selection of commonly used ones are shown here:

gmx energy basic thermodynamic properties of the system

gmx rms calculate the root mean square deviation from a reference structure

gmx rmsf calculate the per-residue root mean square fluctuations

gmx gyrate calculate the radius of gyration

gmx mindist and gmx distance calculate the distance between atoms or groups of atoms (make a index file with gmx make_ndx to define the groups of interest). **`gmx mindist`** is especially useful to find water molecules close to a region of interest.

gmx do_dssp Use the DSSP algorithm *[Kabsch1983]* to analyze the secondary structure (helices, sheets, . . .).

3.8 References

[1] "Observable" is used in the widest sense in that we know an estimator function of all or a subset of the system's phase space coordinates that is averaged to provide a quantity of interest. In many cases it requires considerable more work to connect such an "observable" to a true experimental observable that is measured in an experiment.

Bibliography

[Kabsch1983] Kabsch W, Sander C. Dictionary of protein secondary structure: pattern recognition of hydrogen-bonded and geometrical features. *Biopolymers*. 1983 22 2577-2637. doi: 10.1002/bip.360221211

[Willis1975] Willis & Pryor, Thermal vibrations in crystallography, Cambridge Univ. Press, 1975

[Henzler-Wildman2007] K. A. Henzler-Wildman, V. Thai, M. Lei, M. Ott, M. Wolf-Watz, T. Fenn, E. Pozharski, M. A. Wilson, G. A. Petsko, M. Karplus, C. G. Hübner, and D. Kern. Intrinsic motions along an enzymatic reaction trajectory. *Nature*, 450:838–844, Dec 2007. doi: 10.1038/nature06410

[Beckstein2009] O. Beckstein, E. J. Denning, J. R. Perilla, and T. B. Woolf. Zipping and unzipping of adenylate kinase: Atomistic insights into the ensemble of open/closed transitions. *J. Mol. Biol.*, 394(1):160–176, 2009. doi: 10.1016/j.jmb.2009.09.009.