Molecular Dynamics Simulation Tutorial

Sina Kazemi & Peter Güntert

Introduction

One of the principal tools in the theoretical study of biological molecules is the method of molecular dynamics simulations (MD). This computational method calculates the time-dependent behavior of a molecular system. MD simulations have provided detailed information on the fluctuations and conformational changes of proteins and nucleic acids. These methods are now routinely used to investigate the structure, dynamics and thermodynamics of biological macromolecules and their complexes. They are also used in the determination of structures from X-ray crystallography and from NMR experiments.

The goal of this course is to provide an overview of the theoretical foundations of classical molecular dynamics simulations, and to discuss some practical aspects of the method. Additionally, we want to investigate which information a MD run can provide. Even if a MD simulation is run for a shorter period of time than the underlying biological process, it can provide valuable information.

The following special symbols are used in this tutorial:

Further reading, mainly in the GROMACS manual, which is available from ftp://ftp.gromacs.org/pub/manual/manual-5.0.4.pdf.

Questions

Critical points.

The further reading and questions will be part of the topics for the exam.

Preparing the structure

Bovine pancreatic trypsin inhibitor (BPTI) is one of the smallest and simplest globular proteins. BPTI's function is the suppression of protein digestion, i.e., the breakdown of proteins into their peptide building blocks, by means of inhibiting the action of the enzyme trypsin, which is produced in the bovine pancreas. BPTI is a member of the family of serine protease inhibitors [1]. A hallmark of this class of proteins are the many conserved cysteine residues that form disulfide bonds stabilizing the three-dimensional structures. BPTI has a relatively broad specificity in that it can inhibit several kinds of digestive enzymes. BPTI is one of the most extensively studied globular proteins and has been investigated structurally by both X-ray crystallography and NMR spectroscopy. Furthermore,

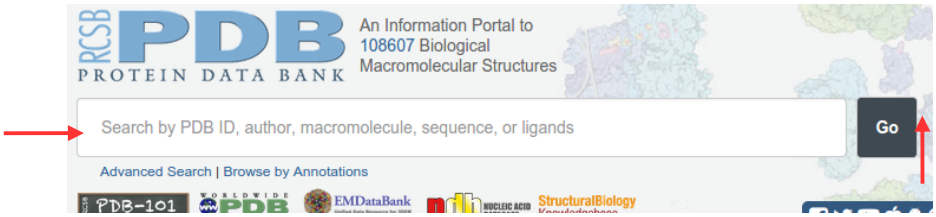
its protein folding pathway and dynamics have been investigated in great detail. Since BPTI forms complexes with the enzymes that it inhibits, it has also been the subject of studies investigating protein-protein interaction and molecular recognition.[1]

This case study is divided into three sections: First, the primary sequence and the three-dimensional structure of BPTI will be introduced. The second part will examine the thermal stability of BPTI that is related to its three internal disulfide bonds. Finally, in the last section, we study BPTI's function as trypsin inhibitor by investigating a trypsin-BPTI complex.

On the PDB-database homepage

http://www.rcsb.org

open the X-ray structure of BPTI with the PDB code 4PTI.



It is always advisable to take a look at the features of a structure you want to use.

Which method was used to determine the structure?

How precise is the structure?

How many residues and how many domains we are dealing with?

Especially, which additional atoms are to be found in the structure?

Depending on those additional atoms and if they are important for the current investigation additional molecules need to be parameterized and modeled for the simulation. However, here we want to simulate the system simply in water (and ions see below).

It is important keep in mind during the course of this tutorial that you always deal with two data systems. One is your local directory and the other is the remote directory on the cluster we use for the calculation. The terminal commands specified here are only for the remote system. For transferring files between the two systems use the drag and drop options of MobaXterm.

First create a directory for your files. The name of this directory must be your *Lastname* (or *LastnameFirstname*). The directory name must not contain blanks or other characters than the common letters of the English alphabet (no ä ö ü ß é etc.). This is necessary as you all use the same user account on the remote machine. Use the command (on the remote system)

[1] Ascenzi, Paolo, et al. The bovine basic pancreatic trypsin inhibitor (Kunitz inhibitor): a milestone protein. *Current Protein and Peptide Science* 4, 231-251 (2003).

```
mkdir MyName
```

to create the directory (replace '*MyName*' with YOUR own Lastname or LastnameFirstname) and

```
cd MyName
```

to make this directory your current working directory. Create another directory for the MD files we will create in the next steps. This is necessary as there will be many files created during this course and we want to keep an overview over these files. Henceforth, we will assume this 'working directory' is called 'bpti_md'. Use the command

```
mkdir bpti_md
```

to create the directory and

```
cd bpti_md
```

to make this directory your current directory. Download the PDB structure using the menu *Download Files > PDB File (Text)* which you find next to the PDB-ID (4PTI) on the website. Save or copy the PDB-file to a directory on your local system. Take a look at the file you just downloaded using PyMOL.

Which molecules can be found in this structure?

What can we find beside the protein? (Compare with the PDB-Entries!)

What is also inside the protein beside the protein own atoms?

Are there cysteines in the structure? Why is that important?

To start with the preparation, you need to copy the file 4PTI to the remote directory we just created 'my_name/bpti_md' (use MobaXterm). Additionally, please copy the following files from the directory '/home/guest/md_tutorial_files':

```
ions.mdp
md.mdp
mdout.mdp
minim.mdp
npt.mdp
nvt.mdp
GMX_EM
GMX_MD
GMX_NPT
GMX_NVT
```

to your directory.

As you observed before, the structure file contains besides the protein also other molecules that we need to remove before we start. For that you need to remove all lines that do not start with 'ATOM' as the first word. You could use any editor to do that but fortunately there is also a handy shell

command to help us:

```
grep "^ATOM" 4PTI.pdb > 4pti_processed.pdb
```

The grep command allows to filter all lines in the file that contain a certain sequence of characters. (In the expression ^ indicates that the following word hat to be at the beginning of the line to be filtered.) The result is written to the file after the > character.

Please keep in mind that even though terminal regions of the protein may be absent, which does not present a problem for dynamics. Incomplete internal sequences or any amino acid residues that have missing atoms will cause the translation to fail. These missing atoms or residues must be modeled in using other software packages. Also note that GROMACS cannot generate topologies for arbitrary molecules, just the residues defined by the force field (in the *.rtp files - generally proteins, nucleic acids, and a small number of cofactors, like NAD(H) and ATP). Fortunately, we do not face such problems with this structure.

Assembling the system

Now that all additional molecules are gone and we have verified that all the necessary atoms are present, the PDB file should contain only protein atoms, and is ready to be input into the first GROMACS module, pdb2gmx. The purpose of pdb2gmx is to generate three files:

1. The topology for the molecule.
2. A position restraint file.
3. A post-processed structure file.

The topology (topol.top by default) contains all the information necessary to define the molecule within a simulation. This information includes nonbonded parameters (atom types and charges) as well as bonded parameters (bonds, angles, and dihedrals). We will take a more detailed look at the topology once it has been generated. Execute the command:

```
gmx pdb2gmx -f 4pti_processed.pdb -o 4pti.gro -water TIP3P -ignh
```

You will see a list of force fields from which you need to select an appropriate one for the later simulation. The force field will contain the information that will be written to the topology file. This is a very important choice! You should always read thoroughly about each force field and decide which is most applicable to your situation. For this tutorial, we will use the 'AMBER03 protein, nucleic AMBER94' force field, so type 1 at the command prompt, followed by 'Enter'.

You have now generated three new files: 4pti.gro, topol.top, and posre.itp. 4pti.gro is a GROMACS-formatted structure file that contains all the atoms defined within the force field (i.e., H atoms have been added to the amino acids in the protein). The topol.top file is the system topology (more on this in a minute). The posre.itp file contains information used to restrain the positions of heavy atoms (more on this later).

You can always take a look at the GROMACS-formatted file in PyMOL using the command

```
editconf -f 4pti.gro -o 4pti.pdb
```

to convert the files into the PDB format.

Let's look at what is in the output topology (topol.top). Using a plain text editor, inspect its contents. After several comment lines (preceded by ';'), you will find the following:

```
#include "amber03.ff/forcefield.itp"
```

This line calls the parameters within the AMBER 2003 force field. It is at the beginning of the file, indicating that all subsequent parameters are derived from this force field. The next important line is [moleculetype], below which you will find

```
; Name        nrexcl
Protein_A     3
```

The name "Protein_A" defines the molecule name, based on the fact that the protein was labeled as chain A in the PDB file. There are 3 exclusions for bonded neighbors. More information on exclusions can be found in the GROMACS manual; a discussion of this information is beyond the scope of this tutorial.

The next section defines the [atoms] in the protein. The information is presented as columns:

```
[ atoms ]
;   nr       type  resnr residue atom  cgnr     charge        mass  typeB   chargeB   massB
; residue   1 ARG rtp NARG q +2.0
     1         N3      1    ARG      N      1  -0.670515      14.01   ; qtot -0.6705
     2          H      1    ARG     H1      2   0.473505      1.008   ; qtot -0.197
     3          H      1    ARG     H2      3   0.473505      1.008   ; qtot 0.2765
     4          H      1    ARG     H3      4   0.473505      1.008   ; qtot 0.75
     5         CT      1    ARG     CA      5   0.093903      12.01   ; qtot 0.8439
     6         HP      1    ARG     HA      6   0.024998      1.008   ; qtot 0.8689
```

The interpretation of this information is as follows:

nr Atom number

type Atom type

resnr Amino acid residue number

residue Amino acid residue name. Note that this residue was "LYS" in the PDB file; the use of the .rtp entry "LYSH" indicates that the residue is protonated (the predominant state at neutral pH).

atom Atom name

cgnr Charge group number. Charge groups define units of integer charge; they aid in speeding up calculations

charge Self-explanatory. The "qtot" descriptor is a running total of the charge on the molecule

mass Also self-explanatory

typeB, chargeB, massB Used for free energy perturbation (not discussed here).

Subsequent sections include [bonds], [pairs], [angles], and [dihedrals]. Some of these sections

are self-explanatory (bonds, angles, and dihedrals). The parameters and function types associated with these sections are elaborated on in Chapter 5 of the GROMACS manual.

Read Chapter 4 *Interaction function and force fields* (especially sections 4.1.1, 4.1.3, 4.2.1, 4.2.5, 4.2.12, 4.2.13 up to Eq. 4.63) on pp. 67–86 of the GROMACS manual.

The remainder of the file involves defining a few other useful/necessary topologies, starting with position restraints. The "posre.itp" file was generated by pdb2gmx; it defines a force constant used to keep atoms in place during equilibration (more on this later).

```
; Include Position restraint file
#ifdef POSRES
#include "posre.itp"
#endif
```

This ends the "Protein_A" moleculetype definition. The remainder of the topology file is dedicated to defining other molecules and providing system-level descriptions. The next moleculetype (by default) is the solvent, in this case SPC/E water. Other typical choices for water include SPC, TIP3P, and TIP4P. We chose this by passing "-water TIP3P" to pdb2gmx. Please also check the web page

http://www1.lsbu.ac.uk/water/water_models.html

for a summary of the many different water models.

```
; Include water topology
#include "amber03.ff/tip3p.itp"

#ifdef POSRES_WATER
; Position restraint for each water oxygen
[ position_restraints ]
;  i funct       fcx        fcy        fcz
   1    1       1000       1000       1000
#endif
```

As you can see, water can also be position-restrained, using a force constant (kpr) of 1000 kJ mol^{-1} nm^{-2}.

Ion parameters are included next:

```
; Include topology for ions
#include "amber03.ff/ions.itp"
```

Finally come system-level definitions. The [system] directive gives the name of the system that will be written to output files during the simulation. The [molecules] directive lists all of the molecules in the system.

```
[ system ]
```

```
; Name
Protein in water

[ molecules ]
; Compound        #mols
Protein_chain_A     1
```

Now that we have examined the contents of a topology file, we can continue building our system for the simulation. In this example, we are going to be simulating a simple system of a protein in aqueous solution. It is possible to simulate proteins and other molecules in different solvents, provided that good parameters are available for all species involved.

There are two steps in defining the box and filling it with solvent:

1. Define the box dimensions using the editconf module.
2. Fill the box with water using the solvate module.

For the purpose of this tutorial, we will use a simple cubic box as the unit cell. For globular systems the rhombic dodecahedron formed box is often used. The advantage is that its volume is ~71% of the cubic box of the same periodic distance, thus saving on the number of water molecules that need to be added to solvate the protein.

Read Chapter 3.2 *Periodic boundary conditions* on pp. 11–14 of the GROMACS manual.

Why this rhombic dodecahedron form is used and not others?

Which kind of geometric bodies can be used for this purpose?

Create a cubic box with the command:

```
gmx editconf -f 4pti.gro -o 4pti_newbox.gro -c -d 1.0 -bt cubic
```

The above command centers the protein in the box (-c), and places it at least 1.0 nm from the box edge (-d 1.0). The box type is defined as a cube (-bt cubic). The distance to the edge of the box is an important parameter. Since we will be using periodic boundary conditions, we must satisfy the minimum image convention. That is, a protein should never see its periodic image, otherwise the forces calculated will be spurious. Specifying a solute-box distance of 1.0 nm means that there are at least 2.0 nm between any two periodic images of a protein. This distance will be sufficient for just about any cutoff scheme commonly used in simulations.

We will fill this box containing the protein with water molecules using the GROMACS solvate command (to be entered on one line!):

```
gmx solvate -cp 4pti_newbox.gro -o 4pti_solv.gro -p topol.top
        -cs spc216.gro
```

The configuration of the protein (-cp) is contained in the output of the previous editconf step, and the configuration of the solvent (-cs) is part of the standard GROMACS installation. We are using spc216.gro, which is a generic equilibrated 3-point solvent model. You can use spc216.gro as the solvent configuration for SPC, SPC/E, or TIP3P water, since they are all three-point water models.

The output is called 4pti_solv.gro, and we tell solvate the name of the topology file (topol.top) so it can be modified.

At the end of your topology file you now find an entry like:

```
[ molecules ]
; Compound        #mols
Protein_chain_A     1
SOL         6943
```

In many cases proteins do not have a zero net charge.

Which factors lead to a non-zero net charge of the protein?

How could you calculate the net charge manually?

The tool for adding ions within GROMACS is called genion. What genion does is read through the topology and replace water molecules with the ions that the user specifies. The input is called a run input file, which has an extension of .tpr; this file is produced by the GROMACS grompp module (GROMACS pre-processor), which will also be used later when we run our first simulation. What grompp does is process the coordinate file and topology (which describes the molecules) to generate an atomic-level input (.tpr). The .tpr file contains all the parameters for all of the atoms in the system.

To produce a .tpr file with grompp, we will need an additional input file, with the extension .mdp (molecular dynamics parameter file); grompp will assemble the parameters specified in the .mdp file with the coordinates and topology information to generate a .tpr file.

An .mdp file is normally used to run energy minimization or an MD simulation, but in this case is simply used to generate an atomic description of the system. An example .mdp file (the one we will use) can be downloaded here.

We can assemble the .tpr file and get the net charge of the system in one step with the following command:

```
gmx grompp -f ions.mdp -c 4pti_solv.gro -p topol.top -o ions.tpr
```

In the outpout of this command you will find a note containing the net charge of the system. Look for lines similar to

```
NOTE 1 [file tmp.top, line 8487]:
  System has non-zero total charge: CHARGE
  Total charge should normally be an integer. See
  http://www.gromacs.org/Documentation/Floating_Point_Arithmetic
  for discussion on how close it should be to an integer.
```

where *CHARGE* is the net charge of your system.

Now we have an atomic-level description of our system in the binary file ions.tpr. We will pass this file to genion adding ions to neutralize the system.

In order to add positive ions use the command (on one line):

```
gmx genion -s ions.tpr -o 4pti_solv_ions.gro -p topol.top -pname NA
           -nname CL -np CHARGE
```

or to add negative ions use

```
gmx genion -s ions.tpr -o 4pti_solv_ions.gro -p topol.top -pname NA
           -nname CL -nn CHARGE
```

Which ions you should use for the system of BPTI?

The command will ask you about which group to be replaced by the ions. Please select the solvent group to be replaced. Take a look at the end of your topology file to check if the right ions where added to your system. Also, take a look at your entire system in PyMOL (use the command mentioned above to convert 4pti_solv_ions.gro in to the PDB file format).

Where are the ions positioned?

Are there any water molecules positioned within the protein?

Energy minimization

The system is now compiled and neutralized but before we can start an MD simulation we need to minimize the energy of the system to avoid steric clashes or inappropriate geometry.

Why is this so important?

What could happen if we don't minimize and equilibrate the system beforehand?

Read Chapter 3.10 *Energy minimization* on pp. 51–53 of the GROMACS manual.

The process for energy minimization is much like the addition of ions. We are once again going to use grompp to assemble the structure, topology, and simulation parameters into a binary input file (.tpr), but this time, instead of passing the .tpr file to genion, we will run the energy minimization through the GROMACS MD engine, mdrun.

Assemble the binary input using grompp using this input parameter file:

```
gmx grompp -f minim.mdp -c 4pti_solv_ions.gro -p topol.top -o em.tpr
```

Make sure you have been updating your topol.top file when running solvate and genion, or else you will get lots of nasty error messages ("number of coordinates in coordinate file does not match topology," etc).

Please check all the notes and especially error in the output of the program. Errors can occur if you use the wrong combination of input files or due to errors in the topology file. After generating the input file em.tpr you can minimize the energy of the system on the cluster using the command:

```
qsub GMX_EM
```

Actually, the command to invoke the structure calculation with GROMACS is '`gmx mdrun -v -deffnm em`' but we need to run all heavy calculations on the cluster nodes and not on the main node of the cluster. The script file GMX_EM only advises the queuing system how many nodes and CPUs to use and which program to start. You can also take a look at the script file with a text editor.

Our cluster has 16 CPUs per node. If you look into the GMX_EM file you find that we use 16 CPUs for each calculation only. In fact, it is not advisable to use more on this system as the calculation will not increase in speed but rather decrease. Do you have any idea why this is so?

For details about parallelization you may consult Chapters 3.16 *Parallelization* and 3.17 *Domain decomposition* on pp. 58–64 of the GROMACS manual.

This job will take a couple of minutes. You can check the progress of you job in two ways. You can check whether jobs are waiting in the queue, running on a cluster node, or have completed using the command

```
showq
```

Additionally, each calculation will generate a log-file with the same name as the binary file. In the case of the energy minimization for example where the binary file em.tpr is used, an em.log file will be written. When the calculation is finished the last line of this file will contain something like:

```
Finished mdrun on rank 0 Tue May 12 15:55:26 2015
```

In summary, after the run we will get the following files (this output file we will get for each mdrun respectively):

em.log ASCII-text log file of the EM process

em.edr Binary energy file

em.trr Binary full-precision trajectory

em.gro Energy-minimized structure

There are two very important factors to evaluate to determine if energy minimization was successful. The first is the potential energy. E_{pot} should be negative, and (for a simple protein in water) on the order of -10^5 to -10^6, depending on the system size and number of water molecules. The second important feature is the maximum force, F_{max}, the target for which was set in minim.mdp ('*emtol = 1000.0*') indicating a target F_{max} of no greater than 1000 kJ mol^{-1} nm^{-1}. It is possible to arrive at a reasonable E_{pot} with F_{max} > emtol. If this happens, your system may not be stable enough for simulation. Evaluate why it may be happening, and perhaps change your minimization parameters (integrator, emstep, etc).

The em.edr file contains all of the energy terms that GROMACS collects during energy minimization. You can analyze any .edr file using the GROMACS energy module:

```
gmx energy -f em.edr -o potential.xvg
```

The energy module writes a table of the desired value over course of the simulation. In the case of the energy minimization please select the potential value by entering '10 0'. You can select each set of quantities you want by entering its number and close the selection by entering a '0'. Afterwards the

output will be written to the file specified by the '-o file' parameter. To plot the result please use the Perl-script prepared by us for these purpose:

```
plotXVG.pl potential.xvg
```

The program will produce the following files

potential.csv	ASCII table file containing the data for the plot
potential.gpl	A script file for Gnuplot to generate the plot
potential.png	The picture of the plot that was automatically generated

The script file is not very complicated (feel free to take a look). It generates a file (.csv) more convenient for the program Gnuplot to read and automatically generated a simple Gnuplot script (You can manipulate this .gpl file to create really neat plots; please check the Gnuplot manual and demos on the Gnuplot homepage.) Take a look at the .png file on your local machine (again use MobaXterm file browser to open the file).

What do you see?

How does the potential energy of the system behave?

Is this an expected behavior during minimization?

Equilibration

The energy minimization ensured that we have a reasonable starting structure, in terms of geometry and solvent orientation. To begin real dynamics, we must equilibrate the solvent and ions around the protein. If we were to attempt unrestrained dynamics at this point, the system may collapse. The reason is that the solvent is mostly optimized within itself, and not necessarily with the solute. It needs to be brought to the temperature we wish to simulate and establish the proper orientation around the solute (the protein). After we arrive at the correct temperature (based on kinetic energies), we will apply pressure to the system until it reaches the proper density.

Remember the posre.itp file that pdb2gmx generated a long time ago? We're going to use it now. The purpose of posre.itp is to apply a position restraining force on the heavy atoms of the protein (anything that is not a hydrogen). Movement is permitted, but only at a substantial energy penalty the further the atom moves away from its original position. Position restraints thus allow us to equilibrate the solvent around the protein, without causing significant structural changes in the protein.

Read Chapter 4.3.1 *Position restraints* on pp. 86–87 of the GROMACS manual.

Equilibration is often conducted in two phases. The first phase is conducted under an NVT ensemble (constant Number of particles, Volume, and Temperature). This ensemble is also referred to as "isothermal-isochoric" or "canonical." The simulation time frame for such a procedure is dependent upon the contents of the system, but in NVT, the temperature of the system should reach a plateau at the desired value. If the temperature has not yet stabilized, additional time will be required. Typically,

50–100 ps should suffice, and we will conduct a 100 ps NVT equilibration for this exercise. Depending on your machine, this may take a while (less than five minutes on our cluster nodes).

We will call grompp and mdrun as two separate commands, just as we did for the energy minimization step:

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

qsub GMX_NVT
```

A full explanation of the parameters used can be found in the GROMACS manual, in addition to the comments provided. Take note of a few parameters in the .mdp file:

gen_vel = yes	Initiates velocity generation. Using different random seeds (gen_seed) gives different initial velocities, and thus multiple (different) simulations can be conducted from the same starting structure.
tcoupl = V-rescale	The velocity rescaling thermostat is an improvement of the Berendsen weak coupling method, which did not produce a correct kinetic ensemble.
pcoupl = no	Pressure coupling is not applied, i.e. the simulation is conducted at constant volume.

Read Chapter 3.5 *Temperatur: Berechnung, Stabilisierung* of the Skript *Einführung in die Moleküldynamiksimulation von Proteinen* and Chapter 3.4.8 *Temperature coupling* (especially the two sections *Berendsen temperature coupling* and *Velocity-rescaling temperature coupling*) on pp. 31–37 of the GROMACS manual.

You can check the resulting system configuration using PyMOL (the file nvt.gro contains the final system coordinates). Analyze the result of the equilibration by following the course of the temperature in the simulation. Use the 'gmx energy' command as described above (section Energy Minimization).

Examine the nvt.mdp file. Which temperature was set here?

Do we reach this temperature? Why is the system temperature not at this value at the beginning of the equilibration?

Do you have any idea why we do need to restrict the volume of the system?

Does the temperature stabilize during the simulation?

The previous step, NVT equilibration, stabilized the temperature of the system. Prior to data collection, we must also stabilize the pressure (and thus also the density) of the system. Equilibration of pressure is conducted under an NPT ensemble, wherein the Number of particles, Pressure, and Temperature are all constant. This ensemble is also called the "isothermal-isobaric" ensemble, and most closely resembles experimental conditions.

The .mdp file used for a 100 ps NPT equilibration can be found here. It is not drastically different from the parameter file used for NVT equilibration. Note the addition of the pressure coupling section, using the Parrinello-Rahman barostat.

Read Chapter 3.7 *Druck: Berechnung über Virialsatz, Stabilisierung* of the Skript *Einführung in die Moleküldynamiksimulation von Proteinen* and Chapter 3.4.9 *Temperature coupling* (especially the two sections *Berendsen pressure coupling* and *Parrinello-Rahman pressure coupling*) on pp. 37–39 of the GROMACS manual.

A few other changes:

continuation = yes We are continuing the simulation from the NVT equilibration phase

gen_vel = no Velocities are read from the trajectory (see below)

We will call grompp and mdrun just as we did for NVT equilibration. Note that we are now including the -t flag to include the checkpoint file from the NVT equilibration; this file contains all the necessary state variables to continue our simulation. To conserve the velocities produced during NVT, we must include this file. The coordinate file (-c) is the final output of the NVT simulation.

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

qsub GMX_NPT
```

Please check this simulation as the NVT before by looking at the output coordinates with PyMOL (see above). It is important to check the result with the 'gmx energy' command. Please check especially the pressure and the density of the system.

The pressure value fluctuates widely over the course of the 100 ps equilibration phase, but this behavior is not unexpected. The running average of these data are plotted as the red line in the plot. Over the course of the equilibration, the average value of the pressure is around 1 bar.

Why do you think the pressure is fluctuation so large?

What happens to the density of the system and is this behavior expected?

Let the molecules dance

Upon completion of the two equilibration phases, the system is now well equilibrated at the desired temperature and pressure. We are now ready to release the position restraints and run production MD for data collection. The process is just like we have seen before, as we will make use of the checkpoint file (which in this case now contains preserved pressure coupling information) to grompp. We will run a 1 ns MD simulation, the script for which can be found here.

```
gmx grompp -f md.mdp -c npt.gro -t npt.cpt -p topol.top -o md_0_1.tpr

qsub GMX_MD
```

This simulation over 1 ns should take around 20 minutes on our cluster nodes. (Please note that the output files will be name 'md_0_1.')

Read Chapter 3.4 *Molecular Dynamics* on pp. 15–45 (excl. section 3.4.6) of the GROMACS manual.

Extracting knowledge

So far, we mainly performed technical steps. Many decisions we made need to be tailored specifically to the needs of the system and simulation in question and good knowledge of this system is needed for getting reasonable results. However, the trajectory of this simulation contains the desired results and at this point we start to cope with the scientific problems that initiated the MD simulation. Therefore, the analysis of the trajectory is a very important step in a MD-simulation study and lead to the gain of knowledge.

What types of data are important? This is an important question to ask before running the simulation, so you should have some ideas about the types of data you will want to collect in your own systems. There are different possibilities to analyze a generated trajectory. Here a few basic tools will be introduced.

General properties

The first such tool is trjconv, which is used as a post-processing tool to strip out coordinates, correct for periodicity, or manually alter the trajectory (time units, frame frequency, etc.). For this exercise, we will use trjconv to account for any periodicity in the system. The protein will diffuse through the unit cell, and may appear to "jump" across to the other side of the box. To account for such actions, apply the following command to the whole system (on one line):

```
gmx trjconv -s md_0_1.tpr -f md_0_1.xtc -o md_0_1_noPBC.xtc
            -pbc whole -ur compact
```

We will conduct all our analyses on this 'corrected' trajectory. First, we want to take a look at the change of the system and the protein, respectively, during the MD run. For that we will use a visual inspection using PyMOL. Use the trajconv module to convert the trajectory to the PDB format executing

```
gmx trjconv -s md_0_1.tpr -f md_0_1_noPBC.xtc -o md.pdb
```

and copy the md.pdb file to your local machine for the inspection with PyMOL. In parallel we can observe the overall change in the conformation by the rmsd to the reference state.

```
gmx rms -s em.tpr -f md_0_1_noPBC.xtc -o rmsd.xvg
```

This module allows selecting the atom for the aliment. Please try different atoms sets (e.g. backbone and all heavy atoms) and store the results in different files (you can replace rmsd.xvg e.g. by rmsd_BB.xvg etc.) Afterwards you can generate a plot of the .xvg file using the Perl-script as described above.

The module gyrate allows you to check the radius of gyration of the system:

```
gmx gyrate -s md_0_1.tpr -f md_0_1_noPBC.xtc -o gyrate.xvg
```

What happens to the protein during this simulation?

Plot the radius of gyration and check for the changes. Check the system energies, temperatures and pressures using the energy module. Also take a look at the energy terms

```
Angle
Proper-Dih.
Improper-Dih.
LJ-14
Coulomb-14
LJ-(SR)
Coulomb-(SR)
```

What do we observe for these values?

Bubbles in the protein

In the first steps we inspected the crystal structure of BPTI more closely and detected that some water molecules are enclosed inside the protein.

Visualize the ions in the trajectory.

Is it possible to see these water molecules in the trajectory?

Are there as many internal water molecules as in the crystal structure? If not: why do you think that is and how can we overcome this? Are there other water molecules inside the protein?

Try to trace these water molecules during the trajectory. What happens to them? Which contacts to the protein do we see?

One way to look for internal water molecules with PyMOL is to show all water molecules and a surface representation of the protein. To do this, first select the protein and use the “extract object” action to make a separate object for the protein, which can then be represented by as surface. This surface can be clipped with the mouse wheel to reveal possible water molecules in the interior of the protein.

General structure stability

Another property that can be informative is the number of hydrogen bonds, either internally (protein-protein) or between the protein and the surrounding solvent. The presence or absence of a hydrogen bond is inferred from the distance between a donor-H - acceptor pair and the donor-H - acceptor angle. To calculate the hydrogen bonds, use the following command, and have a look at the output files:

```
g_hbond -f md_0_1_noPBC.xtc -s md_0_1.tpr -num hb_intern.xvg
```

In the program we can select which hydrogen bonds (between which groups) we want to observe. For the internal protein-protein hydrogen bonds in this example enter '1 1' as the selection. Check also the

number of hydrogen bonds to the solvent.

Why is the number of the hydrogen bonds (especially the internal ones) important?

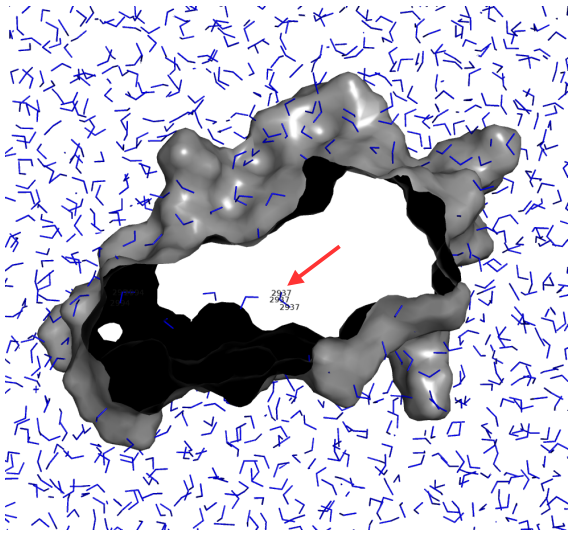
Why do we see fluctuations?

It is also possible to check the interactions of one part of the system to another. For example we want to track one water molecule. First chose a water molecule you want to track in PyMOL by selecting it. You can either see the residue id of the molecule in the top control window after selection or activate the label (see picture below).

After selecting a water molecule, you want to observe you need first to make an index file containing the selection by using the module make_ndx:

```
make_ndx -f md_0_1.gro -o index.ndx
```

Here you need to select the water molecule (in this example 2937 but the number is arbitrary) by entering the residue id. Then giving the new group (in this example 18) a name and quit:



```
> r 2937
18 r_2993              :     3 atoms
> name 18 inside_water
> q
```

Afterwards you can use the created index file to calculate contacts between this water and the protein

```
g_hbond -f md_0_1_noPBC.xtc -s md_0_1.tpr -num hb_intern.xvg -n index.ndx
```

Plot the result and analyze your observations. What happens to internal water molecules?

What happens to slightly buried water molecules or those in contact with surface residues?

Long(er)-term dynamics of calmodulin

BPTI is a highly stable protein with a denaturation temperature above 85 ºC. Very long trajectories are necessary to observe changes or transitions in this structure. Therefore, we now want to take a look at calmodulin, a protein in which conformational changes occur with higher probability.

As its name suggests, calmodulin is a calcium-modulated protein. It is abundant in the cytoplasm of all higher cells and has been highly conserved throughout evolution. Calmodulin acts as an intermediary protein that senses calcium levels and relays signals to various calcium-sensitive enzymes, ion channels and other proteins. Calmodulin is a small dumbbell-shaped protein composed of two globular domains connected by a flexible linker. Each end binds to two calcium ions. We have run a 10 ns simulation of the calcium-free state (PDB structure 1CFD) of calmodulin. Additionally, a simulation was performed in which the molecule was artificially brought into an off-equilibrium state to allow for more prominent conformational changes on a shorter timescale. This was accomplished by using the calcium-bound state (1CLL) as starting structure for an MD simulation. In the simulation, however, the two calcium ions were removed. In this manner we can observe the transition of calmodulin into its calcium-free state. For, first a 10 ns simulation was performed. This simulation was then extended to 100 ns.

The trajectories of these runs are available in the directories

/home/guest/md_tutorial_files/calmodulin_noCA_10ns (starting with calcium-free state)

and

/home/guest/md_tutorial_files/calmodulin_100ns (starting with calcium-bound state)

Make a new subdirectory of your working directory and copy the trajectory of the 10 ns simulation into it. Now compare the results of this simulation with the development in your previous 1 ns simulation of BPTI.

How do the energies and other quantities of interest develop?

Are there any significant structural changes?

Actually, the protein BPTI is a model system for the study of aromatic ring flips.[2,3] Proteins continuously undergo dynamic processes over a wide range of time scales. However, as mentioned before due to the high stability of BPTI, which renders ring flips very rare events on our MD time scale, and as calmodulin is more suited for other analysis we want to perform, we are going to use the calmodulin trajectories.

On which time scales do aromatic ring flips take place in real proteins?

Why do you think is the time scale like that?

Let us check whether ring flips of Phe/Tyr side chain can be observed in the 10 ns MD simulation of calmodulin. Ring flips are manifested in ~180º changes of the χ^2 torsion angle that describes rotations

[2] Wagner, G. et al. Reinvestigation of the aromatic side-chains in the basic pancreatic trypsin inhibitor by heteronuclear two-dimensional nuclear magnetic resonance. *J. Mol. Biol.* 196, 227−231 (1987).
[3] Weininger, U. et al. Ring flips revisited: ^{13}C relaxation dispersion measurements of aromatic side chain dynamics and activation barriers in basic pancreatic trypsin inhibitor. *Biochemistry* 53, 4519–4525 (2014).

of the C^{β}-C^{γ} bond in the side chains of the aromatic amino acids Phe and Tyr. For the analysis we want to use the long calmodulin trajectory (from 20 ns to 100 ns, called md_2_10).

The values of torsion angles against the simulation time can be plotted with the command

```
gmx chi -s md_2_10.tpr -f md_2_10.xtc -all -maxchi 2
```

This command produces many output files. Here we are interested in the chi2PHE*.xvg and chi2TYR*.xvg files that contain plots of the Phe/Tyr χ^2 torsion angles. If you find rare flipping events you can extract just a piece of the trajectory to observe the flip in PyMOL. E.g. if you observe a flip of χ2 of Phe13 between timestamps 29 ns to 30 ns you can extract only this part of the trajectory with the following command:

```
gmx trjconv -s md_2_10.tpr -f md_2_10.xtc -o flip_1.pdb
            -b 29000 -e 30000 -pbc whole -ur compact
```

Please keep in mind that PyMOL becomes very slow and uses up a lot of memory if you try to load trajectories with more than 100 conformers.

Can you detect Phe/Tyr ring flips in the 80 ns trajectory?

How can the result be rationalized?

Principal mode analysis of an MD trajectory

MD trajectories contain all details of the simulated molecular motion, which can be confusing and obscuring the essential motions. Therefore, methods have been developed to extract the “principal” or “essential” motions that have the largest amplitudes and involve the largest parts of the structure.

An often used but often poorly understood method of analysis is principal component analysis (PCA) of the trajectory. This method, which is sometimes also referred to as 'essential dynamics' (ED), aims at identifying large scale collective motions of atoms and thus reveal the structures underlying the atomic fluctuations. The fluctuations of particles in a molecular dynamics simulation are by definition correlated due to interactions between the particles. The degree of correlation will vary and notably particles which are directly connected through bonds or lie in the vicinity of each other will move in a concerted manner. The correlations between the motions of the particles give rise to structure in the total fluctuations in the system and for a macromolecule this structure is often directly related to its function or (bio)physical properties. Thus, the study of the structure of the atomic fluctuations can give valuable insight in the behavior of such macromolecules. However, it does require a certain level of understanding of linear algebra methods and multivariate statistics to interpret the results and identify the shortcomings of the method. In particular, the aim of principal component analysis is to describe the original data in terms of new variables which are linear combinations of the original ones. This is also the most important problem with PCA: it only offers an interpretation in terms of linear relationships between atomic motions.

Read Chapter 8.10 *Covariance analysis* on pp. 246–248 of the GROMACS manual.

You will perform PCA of the 10 ns calmodulin trajectory (in the directory ‘calmodulin_noCA_10ns’), following the recipe given at http://nmr.chem.uu.nl/~tsjerk/course/molmod/analysis1.html.

The first step in PCA is the construction of the covariance matrix, which captures the degree of collinearity of atomic motions for each pair of atoms. The covariance matrix is, by definition, a symmetric matrix. This matrix is subsequently diagonalized, yielding a matrix of eigenvectors and a diagonal matrix of eigenvalues. Each of the eigenvectors describes a collective motion of particles, where the values of the vector indicate how much the corresponding atom participates in the motion. The associated eigenvalue gives equals the sum of the fluctuation described by the collective motion per atom, and thus is a measure for the total motility associated with an eigenvector. Usually most of the motion in the system (>90%) is described by less than 10 eigenvectors or principal components.

The covariance analysis may give a substantial number of files. For that reason, it is better performed in a subdirectory below the directory of the 10 ns MD run of calmodulin:

```
mkdir COVAR
cd COVAR
```

The construction and diagonalization of the covariance matrix can be performed with the program g_covar. Issue the following command to perform the analysis. Note that the construction and diagonalization of a covariance matrix may take some time.

```
g_covar -s ../md_0_1.tpr -f ../md_0_1.xtc -o eigenvalues.xvg
        -v eigenvectors.trr -xpma covar.xpm -b 100
```

Choose ‘Backbone’ for the analysis. The sum of the eigenvalues is a measure of the total motility in the system. It can be used to compare the flexibility of a protein under different conditions, but it is hard to give a meaningful interpretation when comparing proteins of different sizes.

Now have a look at the covariance matrix itself, stored in covar.xpm. The covar.xpm file contains a special type of matrix representation that can be visualized with the following two commands.

```
xpm2ps -f covar.xpm -o covar
ps2pdf covar.eps
```

Both axes of the plot correspond to the protein sequence. The matrix shows the covariances between atoms. Red means that two atoms move together, whereas blue means they move opposite to each other. On the diagonal, the intensity of the red color indicating the amplitude of the fluctuations. Use a PyMOL picture of the protein with residue labels alongside the covariance matrix to identify secondary structure elements.

Which regions of the protein show correlated motions? How do they move with respect to each other and to the rest of the protein?

From the covariance matrix, you can see that groups of atoms move in a correlated or anti-correlated manner. This allows rewriting the total motion into collective motions of groups of atoms. As mentioned, the eigenvalues, stored in the file eigenvalues.xvg, indicate the total fluctuation explained by the corresponding eigenvector.

Plot the distribution of eigenvalues. What do you see?

The first five eigenvectors would typically capture the main motions, accounting for >80% of the

total motility. If the total fluctuation explained is lower, that suggests that there are no clear-cut collective motions.

To see what these eigenvectors actually mean, further analysis can be performed with the tool g_anaeig. To have a closer look at the first two eigenvectors issue the following command:

```
g_anaeig -s ../md_0_1.tpr -f ../md_0_1.xtc -v eigenvectors.trr
         -eig eigenvalues.xvg -proj proj-ev1.xvg -extr ev1.pdb
         -rmsf rmsf-ev1.xvg -first 1 -last 5
```

Choose again 'Backbone' for the analysis. The eigenvectors correspond to directions of motion. The option -extr extracts the extreme structures along the selected eigenvectors from the trajectory. Have a look at these structures by loading them into PyMOL. Download the file 'modevectors.py' from http://www.pymolwiki.org/index.php/Modevectors. Within PyMOL, execute the command File -> Run... -> modevectors.py.

```
modevectors ev11, ev11, 1, 2, factor=1, headrgb=(1,0,0),
            tailrgb=(1,0,0), cutoff=0.5, outname=ev1
```

Compare the vectors and the movement along the trajectory to the calcium-free from (1CFD) and the calcium-bound form (1CLL).

Visualize the first principal components. What do you see?

Could you see this "principal" motions from directly observing the trajectory?

Do the essential modes in this short simulation indicate the transition from one state to the other?

As mentioned before, calmodulin binds to various other proteins in the cell and is included in the regulation of many pathways. Typically, it wraps around its target, with the two globular domains gripping either side of it. One of its targets is calmodulin-dependent protein kinase II-alpha (1CM1). Compare this structure to the calcium-bound state of calmodulin (it is likely that you need to align the structures).

Which kind of movement does the head section (calcium binding domain) of calmodulin perform?

Is the movement suggested by the first principal component? If so, how do the vectors indicate that movement?

Folding of β-hairpin peptides

In recent years, some small peptides with specific secondary structure came into focus as they are simple-to-handle molecules that can be used in drug development and as a tool to interfere in regulatory cell processes.[4,5] This is because some small peptides can form secondary structures that are recognized by other proteins. In fact, you are already familiar with such a peptide from the

[4] Latham, P. W. Therapeutic peptides revisited. *Nat. Biotech.* 17, 755–757 (1999).
[5] Vlieghe, P. et al. Synthetic therapeutic peptides: science and market. *Drug Discov. Today* 15, 40–56 (2010).

exercise above. A peptide out of calmodulin-dependent protein kinase II-alpha was used to determine a bound state of calmodulin via X-ray crystallography. In this case, the peptide forms a helix that is stabilized by the binding to calmodulin.

The β-hairpin is an especially popular small model system. In contrast to an isolated α-helix, β-hairpins are structurally rather diverse. Additional aspects of hairpin structure receiving recent attention are differences in strand length and in the positioning of particular cross-strand residue pairs with respect to the β-turn.[6] Additionally, the right-handed twist present in β structures causes one face of the hairpin (to where the side chains of non-hydrogen-bonding residues are directed) to be rather open, while the other is sterically more "closed".

Due to these advantages, a number of efforts have been made to create stabile β-hairpin-forming peptides. Various computational methods have also been established for this purpose.[7] We want to take a closer look at such a peptide to investigate folding processes of polypeptides. In the following, we use a peptide that was optimized to form a β-hairpin structure. A folded structure of such a peptide is available in the PDB (1J4M). This structure was solved using NMR techniques.[7] However, unlike usual cases here only the average structure instead of a bundle of structures was published.

Please perform a 2 ns simulation of the folded peptide in a new directory following the steps of the BPTI example. After you copied the necessary files, please edit the 'nsteps' entry in the md.mdp file to set a 2 ns long simulation.

Compare the results to a 20 ns simulation starting from an unfolded form of the same peptide. This simulation was already performed for you. You find the resulting files in the directory /home/guest/md_tutorial_files/1j4m-fold.

To compare the two runs, extract both trajectories and investigate whether the molecule forms a stable β-hairpin and whether the trajectory starting from the unfolded structure folds into a hairpin.

What do you observe in the folding trajectory?

Can the native fold be established again? Is the calculation time sufficient for this small and fast-moving system?

It can be quite cumbersome to look over whole trajectories, especially if a long simulation was performed, which is the usual case for observing biologically interesting events. In particular, rare events, trends, or detailed movements might be missed just by visual inspection. You might have noticed this already in the analysis of the aromatic rings performed before. A more quantitative analysis of the trajectory would be interesting in this case.

Does a hydrophobic collapse take place?

Follow the compaction of the protein by plotting appropriate quantities. Which ones?

How close is the final structure of the run to the native peptide structure?

[6] Stanger, H. E. et al. Length-dependent stability and strand length limits in antiparallel β-sheet secondary structure. *Proc. Natl. Acad. Sci. USA* 98, 12015–12020 (2001).

[7] Pastor, M. T. et al. Combinatorial approaches: A new tool to search for highly structured β-hairpin peptides. Proc. Natl. Acad. Sci. USA 99 614-619 (2002).

Is the hydrophobic collapse accompanied by a decrease in energy?

What about entropy?

First, we can repeat an analysis that we already did for BPTI. Check the number of hydrogen bonds (using the command g_hbond, see above) but here only within the backbone of the peptide. Backbone-to-backbone hydrogen bonds indicate the formation of secondary structures.

In the folded state, most proteins form a rather compact packing of the amino acids. Especially, the hydrophobic side chains need to be buried within the folded structure. We can analyze the change in the surface area of the peptide during the folding process. Also, a (crude) estimation of the solvation energy is calculated.

```
gmx sasa -s md_0_2.tpr -f md_0_2.xtc -o sasa.xvg -odg solv.xvg
```

A standard method to detect secondary structure elements in a given 3D protein structure is to use the DSSP (Define Secondary Structure of Proteins) algorithm.[8] In different implementations this algorithm is used widely. E.g. in PyMOL whenever you open a protein structure to assign the secondary structure. The GROMACS tool set also implements a version of it. It can be used to visualize the secondary structure.

```
gmx do_dssp -s md_1_2.tpr -f md_1_2.xtc -o ss.xpm -sc count_ss.xvg
```

The ss.xpm file can be visualized in the same way as the covar.xpm file above.

Keep in mind that all of these analysis tools can also be used on any specific part of the protein by user-defined selections. To accomplish this, just create an index file with the command ‘make_ndx’ as described above and use this index file with the parameter ‘-n’.

[8] Kabsch, W. & Sander, C. Dictionary of protein secondary structure: pattern recognition of hydrogen-bonded and geometrical features. *Biopolymers* 22, 2577-2637 (1983).