CSCBP: Exercise 3

# Protein Simulations & Properties

Document version: 12.10.2015

| | |
|---|---|
| Exercises week 1: | 20.10. or 22.10. |
| Exercises week 2: | 27.10. or 29.10. |
| Deadline for the report: | 06.11. |
| Contact: | annick.renevey@phys.chem.ethz.ch<br>HCI G223 |

**Summary**

During this third exercise you will perform molecular dynamics (MD) simulations of the 56-residue protein Gb88 in water at two temperatures using the GROMOS program, in order to investigate its fold and stability. The set up of the simulations is similar to that of the previous exercise concerning a $\beta$-peptide in methanol, except for the use of non-GROMOS (PDB) initial coordinates and the more careful equilibration protocol. The focus of this exercise is on: (*i*) the aspects of the topology definition specific to proteins; (*ii*) the generation of initial coordinates based on an experimentally derived three-dimensional structure; (*iii*) the thorough equilibration and thermalisation of the initial configuration prior to production; (*iv*) the analysis of the simulations in terms of observables relevant for proteins. For this exercise, you are asked to work in pair with another student, each of the two being responsible for a simulation at one of the two temperatures considered. You can then directly compare your results with those of your colleague.

## 1 Introduction

Proteins are one of the four major classes of biomolecules and represent a key component in numerous biological processes including structuring, catalysis, transport and signalling. They are linear polymers of the 20 natural amino-acid residues, the specific sequence of residues in a protein being referred to as its primary structure. In three dimensions, protein segments tend to adopt preferential local conformation (*e.g.* $\alpha$-helix or $\beta$-sheet) referred to as secondary-structure elements. In turn, these elements typically pack together to define what is called the tertiary structure of the protein. Last, when multiple proteins assemble to form a single protein complex, this is referred to as a quaternary structure.

With the improvement in sequencing techniques it has become relatively easy to establish the residue sequence of a protein. Experimental techniques such as X-ray crystallography (on protein crystals) and NMR spectroscopy (in solution), although far less trivial to apply, have also permitted the determination of the three-dimensional structures of numerous proteins. These structures (atomic coordinates) are typically deposited in databases, the most famous one being the Protein Data Bank (PDB). Benefiting from the availability of such structures, the field of protein simulation has advanced very rapidly over the last three decades, and received its first Nobel Prize in 2013. Key questions addressed in these simulations include protein dynamics, folding and conformational changes, protein-protein interactions and protein-ligand binding. These two weeks, you will be getting a crash-course in protein MD research!

As an example, we will work with a small protein called Gb88. This is one of the multidomain parts of the protein G, a cell wall protein from *Streptococcus*. The Gb88 domain binds to serum proteins in the blood. Their special camouflage strategy, as they cover themselves with host proteins, gives them a selective advantage to pathogenic bacteria. To investigate the protein

fold and stability, 24 mutations to the natural occurring version (wild type) of the protein were applied that resulted in the protein we will work today with. Circular dichroism spectra indicated that the melting temperature (unfolding) for the protein is around 74°C [3]. The detailed three-dimensional structure of the protein in solution was determined on the basis of NMR data [4]. This protein has already been considered in computational research at ETHZ [5].

In this exercise you will perform MD simulations of Gb88 in water at two temperatures: (*i*) at 25°C (298 K), the temperature at which the protein is stable (at equilibrium) and (*ii*) at 75°C (348 K), the temperature at which we expect the protein to unfold. The work for the **first week** is described in Section 2 (preparation of the GROMOS molecular topology and initial coordinate files, starting of the MD simulations). An overview of the work flow for this first week is provided in the Figure 1.

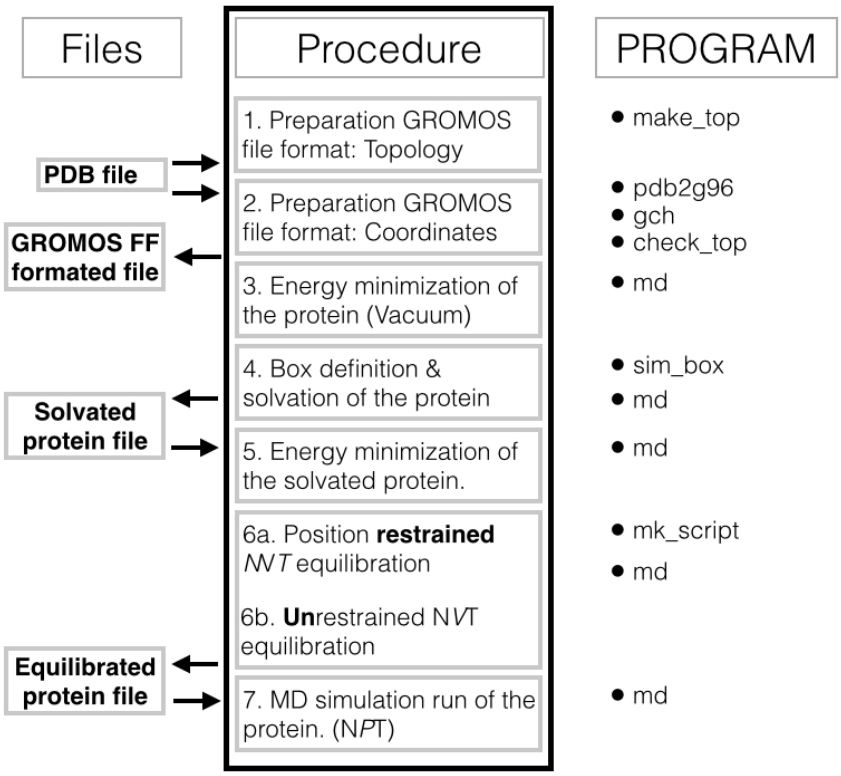


Figure 1: Overview of the GROMOS work flow for the present exercise.

The work for the **second week** is described in Section 3 (analysis of the MD simulations). At the end of the exercise, each student is expected to write a short **report** as described in Section 4. Note that contrary to the two preceding exercises, questions to be addressed in the report are asked on the flight (rather than gathered in Section 4.2). You will find these throughout the document labelled as **(Q...)**. But **additional thinking questions** are still listed at the end (in Section 4.3). Please answer the questions and write down your name and the name of the student you will work with. At the end of this document, you will also find an appendix (Appendix A: a list with all the files you need to successfully perform the exercise; Appendix B: a quick summary of pH and pKa for non-chemists) and a list of relevant references.

Note that when we need a text editor there, we will use "vim"[1], but you are of course free to use instead the text editor of your choice.

As usual, some of the files will be provided to you ready-for-use (but it is still a good idea to look inside!), other files will be provided to you in an incomplete form (*i.e.* with "`TO_DO`" inserts to be substituted by appropriate entries), and other files not at all (then it is all up to you to generate them!).

If you have any question regarding the exercise, the computational technique(s), or our field of research in general, **do not hesitate to ask us!**

# 2 Week One (Submitting Simulations)

## 2.1 The Protein Data Bank (PDB)

The three-dimensional structure of the protein Gb88 in aqueous solution has been determined on the basis of NMR data [3], and the coordinates deposited in the Protein Data Bank (PDB). More precisely, Nuclear Overhauser Effect (NOE) intensities have been recorded, which provide information on average hydrogen-hydrogen distances within the protein. Based on this information (along with a substantial amount of modelling!), a set of model three-dimensional structures have been optimized that reproduce the experimental data with similar accuracies. Note that the atomic coordinates only include protein atoms (no solvent). In the PDB, all structures have a unique identifier consisting of four letters or digits. The set of model structures for Gb88 based on the quoted NMR study have the identifier 2JWU.

To get these structures, go the website `www.rcsb.org` and download the corresponding record (in the right upper corner of the web page, option: PDB file (text) of the protein (2JWU)) to your working directory. Open the downloaded file with your text editor and have a look at it.

The file contains the coordinates of the atoms in your molecule in a format specific to the PDB (PDB format) which differs from the GROMOS format for atomic coordinates (note in particular the use of Ångstrom rather than nanometer as unit of length). The PDB file contains a lot of information regarding the protein in addition to the atomistic coordinates, such as the primary sequence and the corresponding experimental conditions.

**(Q1)** Find out and report the following information from the PDB file:

- Check that the set of structures is derived from NMR and that the literature source is [4].
- What were the experimental conditions during the NMR measurements? (temperature, pressure, solvent, pH?)
- Number of chains and the corresponding chain identifier
- Number of residues
- What is the amino-acid sequence of the protein?
- What are the secondary-structure elements of the protein? (Hint: on the webpage about 2JWU, look at the structure image on the right)
- Number of atoms in the protein
- Number of model structures in the file

[1] for more info, see `http://www.terminally-incoherent.com/blog/reference/vim-cheat-sheet/`

The file contains a set of model structures, and we will only consider the first of them for our simulations. To extract this model 1 structure, use the following command

```
awk 'NR==220, NR==1142' 2JWU.pdb > protein_model1.pdb
```

It is often a good idea to check first the consistency of your model structure. There are online servers (*e.g.* WHATIF, PROCHECK) that perform this task, and can spot *e.g.* missing residues or atoms, inconsistency of symmetries, unlikely covalent geometries, etc. The model 1 structure passes these simple consistency checks easily so they can be skipped in our case.

## 2.2 Visualization of the Protein

First let's examine the model structure of our protein using the graphics program PyMOL. Load the structure in PyMOL using

```
pymol protein_model1.pdb
```

The structure is initially represented in a 'line' representation. On the right sight of the main window, the object (protein) is listed and next to it several options allow you to change the representation. Have a look at the options and then, show the structure as cartoon by writing in the window

```
hide all
show cartoon
```

To color your protein according its secondary structure (ss), where we consider helices (h), $\beta$-sheets (s) and connections/loops (|+"), type

```
color red, ss h
color yellow, ss s
color green, ss |+''
```

To get a space-filling (CPK) model of the protein on top, type the following commands

```
show spheres
bg_color white
set sphere_scale, 0.9
set sphere_transparency, 0.6
```

To save an image of your protein representation, first improve the quality by typing

```
set opaque_background, off
ray 1000
```

Then export it using

```
png perfect_protein.png
```

**(Q2)** Insert the image into your report and describe the secondary elements of the protein

Now, before we start setting up and running simulations, there are three important points you should keep in mind concerning this structure. First, this is a *single static conformation* within an ensemble of conformations at equilibrium in solution, a conformation which was selected because it is expected to be a highly-populated member of this ensemble. Second, this expectation is only relevant for the ensemble of conformations corresponding to the *experimental conditions*

of the NMR experiment, *i.e.* in pure water at atmospheric pressure and close to room temperature (*e.g.* at higher temperature, the protein is expected to be predominantly unfolded!), and close to neutral pH (*e.g.* under acidic or basic conditions, it may unfold as well!). Third, the structure is *inferred based on NMR data* and not determined by the NMR data. A limited number of NOE-derived hydrogen-hydrogen distances is not sufficient alone to determine the coordinates of all atoms in the protein, and the structure refinement relies on the use of standard geometric parameters (bond lengths, angles, dihedrals and improper dihedrals, atomic excluded volumes), sometimes even on a potential energy function (simple force field) to complement the experimental information. This underdetermination by the experimental data is the very reason why a set of acceptable structures has been provided in the PDB file of Gb88 rather than a single one. To summarize, saying that 2JWU/model1, or the cartoon you just made of it, is "the structure of the protein Gb88" is a very common but also very imprecise statement, hiding a lot of the complexity of the situation. And when you forget too much of the complexity of a problem, it always ends up blowing up in your face at a later point...

The above discussion should also make the key strength of protein MD simulations (also valid for other biomolecular systems) very obvious: it permits to replace a discussion of conformational properties based on single structures by a discussion properly based on conformational ensembles.

### 2.3 Setting up the Directory Structure

In the Exercise 2, you have already become familiar with the set-up procedure for starting MD simulations of a peptide using GROMOS [1, 2]. For the protein, we will follow a similar preparation procedure. As mentioned before, in order to speed-up this preparation, a number files are directly provided to you, either ready-for-use or in an incomplete form.

```
ssh -X user@realbeaver.ethz.ch
cd
mkdir ex3
cd ex3
cp -r /usr/local/CSCBP/ex3/dir_for_students/* ./
```

There should now be **8 subdirectories** in your current directory:

- the **pdb** dir: contains the file `protein_model1.pdb`, *i.e.* the 2JWU/model1 structure you considered in the previous section with "END" as last line; this is our reference structure for the folded conformation of the protein (solute atom coordinates) in PDB format, and the one we will use to set-up the MD simulations.
- the **topo** dir: will be used to construct and store the topology information.
- the **coord** dir: will be used to convert the PDB coordinates (solute atoms) to the GROMOS coordinate format.
- the **min** dir: will be used to perform an EM of the initial coordinates (solute alone) in vacuum
- the **box** dir: will be used to solvate the protein into a box of water
- the **min_h2o** dir: will be used to perform an EM of the initial coordinates (solute+solvent) in solution
- the **eq** dir: will be used for further equilibration and thermalisation of the coordinates (solute+solvent) using MD

- the **md** dir: will be used for the production MD simulations at the two temperatures considered.

These directories will progressively fill up as we follow the set-up and simulation protocol described in detail in Sections 2.4. Today, we provide this tidy directory structure, but next time (and maybe after the CSCBP course), you might be on your own and should keep in mind to organize things yourself. There are two key advantages in maintaining such a clear directory structure (as well as a carefully thought file-naming system) when you set-up a simulation. First, you may easily generate over hundred files in the set-up. Keeping everything in a single directory would increase the likelihood of confusions and mistakes. Second, you will typically spend a day or two on the set up, but months on the runs. By the time you are done, you will have largely forgotten the set-up details. If you need to recheck things later (or someone else after you), this will be far easier when the structure is clear. Structuring your directories / file names should in any case not be difficult if you have a clear idea (flowchart) of the set-up protocol in your mind, which you should have in any case prior to starting anything.

## 2.4 Setting up and Running the Simulation

### 2.4.1 Topology

GROMOS++ PROGRAM NEEDED: `make_top`

INPUT FILES AVAILABLE: `54a7.mtb 54a7.ifp make_top_protein.arg`

OUTPUT FILES THAT WILL BE CREATED: `protein.top`

TO DO: complete the `make_top_protein.arg` file

Let's begin as usual with the construction of the molecular topology file using `make_top`.

⇒ For more details on the purpose and content of the topology file, refer to Exercise 1.

⇒ For more details on the construction of the topology file using `make_top`, refer to Exercise 2.

We are going to use the 54A7 version of the GROMOS biomolecular force field (molecular topology building block file `54a7.mtb` and interaction parameter file `54a7.ifp` provided), which includes building blocks and parameters for the 20 natural amino acids (in different protonation states for ionisable residues) as well as information on how to link them together through successive peptide bonds and to cap the chain with terminal groups. All we need is to know about the sequence of the protein and the protonation states of the ionisable residues, the nature of the chain termini and the counter-ions to be possibly added. Go into the `topo` dir

```
cd topo
```

Have a look at the `make_top_protein.arg` file

```
vim make_top_protein.arg
```

The main element missing in this file is the @seq parameter, where you have to provide the residue sequence of the protein, i.e. the GROMOS names of the 56 residues preceded by the N-terminal capping and the C-terminal capping (that is, in total, 58 words). You can start by

copying the sequence you read from the PDB file, adding NH (with the corresponding charge plus/min) at the start and CO (with the corresponding charge plus/min) at the end.

For most residues, the PDB name is the same as the GROMOS name, so you have nothing to change. But some residues are ionisable, i.e. they can bear a proton or not depending on the pKa of the functional group and the experimental pH. The same is true for the non-blocked capping groups R-NH2 and R-COOH which we will use here, and for which we can use the pKa of ethylamine and acetic acid, respectively, as estimates. If you are not a chemist, you can find some basic information on pH and pKa in Appendix B of this document. And for your convenience, indicative pKa's for the relevant residues are listed in Table 1.

Table 1: Indicative pKa values for the natural amino-acid residues. Note that the pKa of a specific residue within a protein may differ (shifts induced by the local protein environment of the residue). Source: Ref. [6].

| AA | pKa |
|---|---|
| ASP | 3.02 |
| GLU | 4.61 |
| Acetic acid | 4.76 |
| HIS | 6.99 |
| CYS | 6.18 |
| Ethylamine | 10.75 |
| LYS | 10.67 |
| ARG | 12.10 |

If the pH is lower than their pKa:

- The acidic residues **Asp** and **Glu** as well as **the free C-terminus** will be protonated and neutral (GROMOS names: ASPH, GLUH and COOH).
- The basic residues **Lys**, **Arg** and **His** as well as **the free N-terminus** will be protonated and positively charged (GROMOS names: LYSH, ARGH, HISH and NH3+).

If not:

- The acidic residues **Asp** and **Glu** as well as **the free C-terminus** will be deprotonated and negatively charged (GROMOS names: ASP, GLU and COO-).
- The basic residues **Lys**, **Arg** and **His** as well as **the free N-terminus** will be deprotonated and neutral (GROMOS names: LYS, ARG, HISA/HISB and NH2).

Note, **His** is a bit special in that it has two deprotonated forms depending on the location of the remaining proton, as illustrated in Figure 2. Then you have to think about which form to select (*e.g.* by looking at potential hydrogen-bonding partners in the protein structure). Luckily, we have no His in our protein Gb88. At very high pH, we also have to consider the protonation state of residue Cys. But there aren't any in Gb88.

Now, you should be able to modify your initial @seq appropriately (hint: besides replacing the NH (charged?) and CO (charged?) by something meaningful, there is only one name to change systematically to another one). When this is done, it is also important to determine the net charge of the protein given the selected ionization state. If the protein is non-neutral, you may have to decide whether to simulate a non-neutral system or to neutralize it by adding

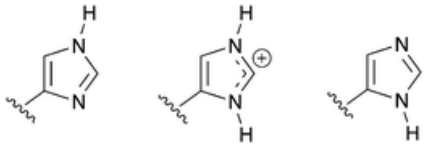

Figure 2: Histidine protonation states labelled HISA (H on ND1), HISH (protonated) and HISB (H on NE2).

counter-ions, which must then also be included into the topology file as seen in Exercise 2. You may also add counter-ions even for a neutral protein to mimic a finite ionic strength in the experiment. Here, we will add none. Answer the following questions:

**(Q3)** What pH will we consider in the simulations (same as in the NMR experiments)?
**(Q4)** List the 58 words you added after the @seq?
**(Q5)** What is the net charge of the protein?

Now make sure you have complemented everything else needed in `make_top_protein.arg` and build the topology file of the protein using `make_top`

```
make_top @f make_top_protein.arg > protein.top
```

Now the molecular topology file `protein.top` is generated. Open the file and check that it looks reasonable

```
vim protein.top
```

In particular, have a look at the number of atoms (solute)

```
/SOLUTEATOM
```

Now you should see:

```
SOLUTEATOM
# NRP: number of solute atoms
592
```

**(Q6)** Why is this number of atoms different from that in the PDB file?

(The answer to this question will become obvious in the next subsection; but don't miss the thrill of finding the answer by yourself right now!) At this point, you should in principle run `check_top` to perform a consistency check of your topology. For the sake of time, we will skip this step and go directly to the coordinate generation step

```
cd ../coord
```

### 2.4.2 Coordinates

GROMOS++ PROGRAM NEEDED: `pdb2g96`

INPUT FILES THERE: `pdb2g96.arg pdb2g96.lib`

OUTPUT FILES THAT WILL BE CREATED: `protein_g96.cnf`

The discrepancy between the number of solute atoms in the PDB file and in the GROMOS molecular topology file is a clear hint that there are at this point some compatibility problems between the two files. If you want them to live together as a happy couple, you'll have to iron these out. The possible sources of discrepancies can be the following:

1. Coordinates are in **Ångstrom** in the PDB file but nanometer in the GROMOS file.
2. There can be **extra hydrogen atoms** in the PDB file. This is typically the case for NMR structures, where all hydrogen atoms are included, even the aliphatic ones that GROMOS does not treat explicitly (united-atom representation).
3. There can be **missing hydrogen atoms** in the PDB file. This is typically the case for X-ray structures, where these atoms have a too low electron density to be detected.
4. The **ordering of the atoms** in the PDB file may differ from the one adopted in GROMOS files
5. There can be **missing residues** in the PDB file. For example, it is not uncommon that disordered segments in a protein (typically loops or termini) cannot be resolved in an experimental X-ray structure determination, and are absent in the PDB file. In this case, you would have to do some modelling to generate a good guess for the missing coordinates.
6. There may be small **typos and errors** in the PDB file.

Besides these, also remember that a PDB file may contain multiple structures for the same protein. This was the case for our file 2JWU (bundle of structures refined based on NMR data), and we already took care of that by selecting model1. It can also happen in X-ray structures when the different copies of the protein in the crystallographic unit cell are refined independently.

Discrepancies of types 1-4 above can be fixed automatically using the GROMOS++ program pdb2g96, which converts a PDB coordinate file into a GROMOS coordinate file based on a GROMOS molecular topology file, in such a way that the atom content and ordering in the created GROMOS coordinate file matches exactly that specified in the topology (by matching the names of the atoms in the topology with the ones of the PDB file. Note that as soon as agreement is reached, the atom name information in the coordinate file will be entirely ignored for all subsequent GROMOS operations, *i.e.* only the atom order in this file will matter, the names being extracted from the topology file. In case 2 above, pdb2g96 will have to add hydrogen atoms, the coordinates of which are not known experimentally. In this case, pdb2g96 will set the unknown coordinates to zero, and a second GROMOS++ program `gch` has to be used to construct these coordinates based on standard geometric constructions. Whenever discrepancies of types 5-6 above occur, they will have to be detected and repaired manually (good luck!).

In the case of our protein Gb88, we only have types 1 (atom needed reordering) and 3 (aliphatic hydrogen atoms to be mercilessly slaughtered), and pdb2g96 will do the job. Briefly review the argument file `pdb2g96.arg`

```
vim pdb2g96.arg
```

If you agree with the content of the file, you can continue with the execution of the command

```
pdb2g96 @f pdb2g96.arg > protein_g96.cnf
```

Now the GROMOS coordinate file `protein_g96.cnf` of the protein is generated

**(Q7)** Compare the records for residue 4 (Lys) in the PDB and GROMOS coordinate files and describe/explain the differences?

Let's continue with the next step, performing an EM in vacuum

```
cd ../min
```

### 2.4.3 Energy Minimization of the Protein (Vacuum)

GROMOS++ PROGRAM NEEDED: `md`

INPUT FILES THERE: `em_protein.imd em_protein.run`

OUTPUT FILES THAT WILL BE CREATED: `protein_min.cnf em_protein.omd em_protein.out`

Before placing the protein in a box and solvating it, the initial GROMOS coordinates generated based on the PDB file will be relaxed by energy minimization.

⇒ A more detailed explanation regarding the EM procedure can be found in Exercise 2!

Briefly check if all is correctly specified in the argument file `em_protein.run`

```
vim em_protein.run
```

and in the GROMOS input file `em_protein.imd`

```
vim em_protein.imd
```

If you agree with the content of these files, you can continue with the execution of the command

```
./em_protein.run
```

Now your protein is energy minimized (`protein_min.cnf`)

**(Q8)** Which algorithm is applied in the EM procedure? (Tip: check the `em_protein.imd` file)

**(Q9)** What is the total potential energy of the system before and after EM (look it up in `protein_min.out`)?

**(Q10)** What are the two main causes for the very high potential energy of the 2JWU/model1 structure in the GROMOS force field?

Let's continue with the next step, solvating the protein and creating a computational box

```
cd ../box
```

### 2.4.4 Solvating the Protein in a Water Box

GROMOS++/MD++ PROGRAM NEEDED: `sim_box md frameout`

INPUT FILES THERE: `sim_box_protein.arg h2o.cnf`

OUTPUT FILES THAT WILL BE CREATED: `protein_box.cnf protein_box.pdb`

Now the protein is ready to be placed into a box and solvated, for subsequent simulations under periodic boundary conditions.

⇒ A more detailed explanation regarding this procedure can be found in Exercise 2!

The box shape will be chosen rectangular (r), the simple point charge (SPC) water model[8] will be employed (as already specified in the topology file), the minimum solute-to-wall distance will be 0.9 nm and the minimum solute-solvent distance 0.23 nm. Thus, there will be no direct interactions between periodic copies of the protein, the closest surface atoms of two periodic copies being at least 1.8 nm apart (longer than the cutoff distance of 1.4 nm). A slightly larger distance would be advised considering that the protein can rotate in the box and that some water molecules will still interact with two periodic copies of the protein, but we want fast simulations for this exercise so we are less careful. Also we don't add ions (neutral protein, assumed zero ionic strength situation). The GROMOS++ program `sim_box` is used to generate the box and solvate the protein. Briefly check if all is correctly specified in the `sim_box_protein.arg file`

```
cat sim_box_protein.arg
```

If you agree with the content of the file, you can continue with the execution of the command

```
sim_box @f sim_box_protein.arg > protein_box.cnf
```

Now your protein is solvated in a water box (`protein_box.cnf`). To visualize how the protein is solvated execute the following command

```
frameout @f frameout_box.arg
mv FRAME_00001.pdb protein_box.pdb
pymol protein_box.pdb
```

**(Q11)** What is the volume of your protein? (hint: approximate the protein volume as a cylinder and use PyMOL Measurement's function to get the length and radius)

**(Q12)** What is the volume of your box?

Let's continue with the next step, relaxing the solute-solvent system in the computational box

```
cd ../min_h2o
```

### 2.4.5 Energy Minimization of the Protein in the Box

MD++ PROGRAM NEEDED: `md`

INPUT FILES THERE:
`protein_box.cnf em_protein_box.imd em_protein_box.run frameout_protein_h2o.arg`

OUTPUT FILES THAT WILL BE CREATED:
`protein_box_min.cnf em_protein_box.out`

During the immersion into the solvent, water molecules may still have been placed too close (clash) or too far (gaps) relative to the protein surface. In addition, their orientation towards the protein surface is not optimized. All these effects result in high potential energy contributions. If we were to immediately start with MD, this would result within a few steps into a huge effective temperature at the protein surface, and collisions leading to distortion in the protein structure (generally sanctioned by GROMOS in form of an unpleasant SHAKE failure!).

Therefore, we need to perform an equilibration of the solute-solvent system using EM. During this process, the solute atoms will be positionally restrained around their coordinates in the initial structure. This means that we keep them on a tight leach (harmonic spring) and prevent that they move too far away from their starting position, which corresponds to the (energy minimized) experimental structure. The solvent molecules, on the other hand, can move entirely freely.

The list of atoms to be positionally restrained must be specified in a file `protein_box.por`. The reference positions of these atoms must be specified in a file `protein_box.rpr`. You have to prepare these two files yourself, but it is easy. First copy the file `protein_box.cnf` file to the current directory

```
cp ../box/protein_box.cnf ./
```

Now make two clones of it with the required file names

```
cp protein_box.cnf protein_box.por
cp protein_box.cnf protein_box.rpr
```

Open the file `protein_box.por` in your text editor

```
vim protein_box.por
```

- Write in the TITLE block the text "list of solute atoms to be positionally restrained"
- Change the keyword POSITION at the beginning of the atom coordinate block into the keyword POSRESSPEC
- Delete all the solvent atoms

Now you your `protein_box.por` should look like:

```
TITLE
solute atoms to be positionally restrained
END
POSRESSPEC
# first 24 chars ignored
```

```
1 THR H1 1 5.144434523 5.236928112 4.193546264
......
......
56 GLU O1 591 0.316814628 0.079450732 1.428651374
END
```

When GROMOS reads this file, it will ignore entirely the coordinates and just look at the list of atoms. Next, open the file `protein_box.rpr` in your text editor

```
vim protein_box.rpr
```

- Write in the TITLE block the text "reference positions of solute atoms to be positionally restrained"
- Change the keyword POSITION at the beginning of the atom coordinate block into the keyword REFPOSITION

Now you your `protein_box.rpr` should look like:

```
TITLE
reference positions for restraining solute atoms
END
REFPOSITION
# first 24 chars ignored
1 THR H1 1 5.144434523 5.236928112 4.193546264
......
......
0 SOLV HW2 16135 2.1554.91829 0.562881411 0.408577477
END
```

When GROMOS reads this file, it will only use the coordinates of the atoms listed in `protein_box.por` (*i.e.* here, the solute atoms), and ignore all the rest. Now open the file `em_protein_box.imd` in your text editor

```
vim em_protein_box.imd
```

Look at the POSITIONRES block

```
POSITIONRES
# NTPOR NTPORB NTPORS CPOR
  1 1 0 2.5E4
END
```

This block takes care that the atoms are positionally restrained (NTPOR = 1) with a specified harmonic force constant (COPR).

**(Q13)** What are the units of the number 2.4E4 listed as CPOR? How does the value compare with *e.g.* the force constant for a C-C bond?
(Hint: check the force field parameter file `.ifp` for the C, CHn - C CHn bond) **(Q14)** What are the switches NTPORB and NTPORS (by now, you are GROMOS experts, so you should know where to find the answer yourself!)

Briefly check if all is correctly specified in the file `em_protein_box.run`

```
cat em_protein_box.run
```

If you agree with the content of the file, you can continue with the execution of the command (the '&' symbol puts the job on the background, in case you want to check something in the meanwhile)

```
./em_protein_box.run &
```

Now your solvated protein in the box is equilibrated (`protein_box_min.cnf`). Let's continue to the last step before the actual production run, the generation of initial velocities followed by MD thermalisation/equilibration

```
cd ../eq
```

### 2.4.6 Thermalisation and Equilibration

GROMOS++/MD++ PROGRAM NEEDED: `mk_script ene_ana md`

INPUT FILES THERE:
`eq_mkscript.arg eq.imd equilibrium.jobs mk_script.lib protein_box_min.cnf`

OUTPUT FILES THAT WILL BE CREATED:
`protein_*.imd protein_*.run protein.cnf protein_*.trc.gz protein_*.tre.gz`

In the previous exercise, we immersed the peptide in the solvent, performed an EM, generated random velocities appropriate for a temperature T and directly started the MD simulation at this temperature. For a protein, we want to be a bit more careful and apply a thorough thermalisation procedure. We will only let the protein loose when the system has been well equilibrated at the target temperature. This will avoid that the actual production simulation starts from a protein structure that looks already quite distorted (in a random fashion) relative to the experimental one. For this, we will use in combination a progressively increasing temperature and progressively decreasing position restraints on the solute atoms. Note that there will be **two** different target temperatures, 298 K and 348 K, one for each student of a pair.

The thermalisation procedure is greatly facilitated by the use of the GROMOS++ program `mk_script`, which allows the automatic generation of successive MD jobs that: (*i*) slightly differ in their input parameters; (*ii*) use the final configuration of one job as the starting configuration of the next one; (*iii*) automatically submit the next job upon completion of the previous one. Have a look at the `eq_mkscript.arg` input file

```
vim eq_mkscript.arg
```

The name you want to give to the jobs for thermalisation/equilibration is indicated after @sys, the pathway of your current working directory is indicated after @dir, the file specifying the number of heat-up steps and their differing input parameters after the @joblist and all the information regarding the protein system after @files. To understand how the work will be done, we have to look further at the files `eq.imd` and `equilibrium.jobs`.

The model input file `eq.imd` is a regular input file for the GROMOS program md. It specifies default input parameters for all the jobs. Only a few (8) of them will be later substituted by other values specific to each job, which will be indicated by the word "*overwritten*" below. This file is similar to the input file for the energy minimization `em_protein_box.imd`. The structure of these files has been already discussed in Exercise 2. So, we will only mention the most relevant blocks here.

```
********************
The INITIALISE block
********************

INITIALISE
# NTIVEL NTISHK NTINHT NTINHB NTISHI NTIRTC NTICOM NTISTI IG TEMPI
   1 3 0 0 1 0 0 0 145117 0.0
END
```

NTIVEL (overwritten) specifies if GROMOS++ should generate the initial velocities (1) or read them from the initial configuration file (0). NTISHK (overwritten) is used to enforce bond-length constraints (SHAKE) after reading the initial configuration. NTINHT and NTINHB are only used for Nose-Hoover thermo- and barostats and can be ignored in our case. Every time an atom leaves the periodic box and enters from the opposite site, the incident is recorded in the so-called lattice shift vectors. NTISHI (overwritten) makes sure that these vectors are initialized to zero. NTIRTC can be turned on for roto-translational constraints, which is not relevant in our case. NTICOM specifies if initial removal of centre of mass motion is required. NTISTI specifies whether to reset the stochastic integrals used in stochastic dynamics (SD) simulations. IG is the random number generator seed and TEMPI (overwritten) the initial temperature used to generate the Maxwell-Boltzmann distribution for generation of initial velocities.

```
********************
The STEP block
********************

STEP
# NSTLIM T DT
  10000 0.0 0.002
END
```

NSTLIM specifies how many steps we want to simulate, T the time offset at the start of the job, and DT is the integration time step. Here, you want to start at time 0 and to carry out a 20 ps simulation (10000 steps).

```
********************
The BOUNDCOND block
********************

BOUNDCOND
# NTB NDFMIN
   1 3
END
```

Here the periodic boundary conditions are specified. With NTB =1, rectangular PBC is selected. NTB is the number of uncoupled degrees of freedom.

```
********************
The MULTIBATH block
********************
```

```
MULTIBATH
# ALGORITHM:
# weak-coupling(0): use weak-coupling scheme
# nose-hoover(1): use Nose Hoover scheme
# nose-hoover-chains(2): use Nose Hoover chains scheme
# NUM: number of chains in Nose Hoover chains scheme
# !! only specify NUM when needed !!
# NBATHS: number of temperature baths to couple to
# ALGORITHM
  0
# NBATHS
  2
# TEMP0(1 ... NBATHS) TAU(1 ... NBATHS)
  60 0.1 60 0.1
# DOFSET: number of distinguishable sets of d.o.f.
  2
# LAST(1 ... DOFSET) COMBATH(1 ... DOFSET) IRBATH(1 ... DOFSET)
  592 1 1 16135 2 2
END
```

This block controls the thermostat. With `ALGORITM=0`, the weak-coupling scheme is selected. `NBATHS` specifies the number of temperature baths to couple the system to (we want 2, one for the solute and one for the solvent). `TEMP0` (overwritten) specifies the temperature for each bath and `TAU` the coupling time used in the weak-coupling method for each bath. `DOFSET` specifies the number of distinguishable sets of degrees of freedom. `LAST` points to the last atom for the set of degrees of freedom. `COMBATH` is the temperature bath to which the center of mass motion is coupled of this set of degrees of freedom. `IRBATH` is the temperature bath to which the internal and rotational degrees of freedom of this set of degrees of freedom are coupled.

**(Q15)** Why are the protein and solvent separately coupled to a heat bath?

```
**********************
The COMTRANSROT block
**********************

COMTRANSROT
# NSCM
  -1000
END
```

This block is needed to remove the center of mass motion (here, translation and rotation). Without this block it can happen that all the kinetic energy is converted to center of mass translation (flying ice cube problem). With `NSCM` specifies how often the center-of-mass (COM) motion is removed. If NSCM is $< 0$: translation and rotation motion are removed every `NSCM` th step. If NSCM is $> 0$: only translation motion is removed every `NSCM` th step.

```
**********************
The COVALENTFORM block
**********************
```

```
COVALENTFORM
# NTBBH: 0,1 controls bond-stretching potential
# 0: quartic form (default)
# 1: harmonic form
# NTBAH: 0,1 controls bond-angle bending potential
# 0: cosine-harmonic (default)
# 1: harmonic
# NTBDN: 0,1 controls torsional dihedral potential
# 0: arbitrary phase shifts (default)
# 1: phase shifts limited to 0 and 180 degrees.
# NTBBH NTBAH NTBDN
  0 0 0
END
```

Here the functional forms are specified for bond-stretching (`NTBBH`), bond-angle bending (`NTBAH`) and for torsional dihedral (`NTBDN`). The default options are chosen for all functional forms.

```
**********************
The WRITETRAJ block
**********************

WRITETRAJ
# NTWSE = configuration selection parameter
# =0: write normal trajectory
# >0: chose min energy for writing configurations
# NTWX NTWSE NTWV NTWF NTWE NTWG NTWB
  100 0 0 0 100 0 0
END
```

MD++ produces a massive amount of data, too much to store every step of a simulation. Therefore it is specified how often the coordinate trajectory (`NTWX`), the velocity trajectory (`NTWV`), the force trajectory (`NTWF`), the energy trajectory (`NTWE`), the free energy trajectory (`NTWG`) and the block averaged energy trajectory (`NTWB`) are written out. In the present case, we are only interested in the coordinates and energies. These are written out every 100th step. `NTWSE` functions as a 'second switch', since it defines the selection criterion for trajectories: If `NTWSE = 0`: the normal coordinate trajectory will be written, if `NTWSE > 0`: a minimum energy trajectory will be written.

```
**********************
The PRINTOUT block
**********************

PRINTOUT
#NTPR: print out energies, etc. every NTPR steps
#NTPP: =1 perform dihedral angle transition monitoring
# NTPR NTPP
  100 0
END
```

Similar to the WRITETRAJ block, `NTPR` specifies how often the information regarding the energies is printed to the output file. `NTPP` specifies if the dihedral angle transitions are also monitored and written to the output file.

```
**********************
The PAIRLIST block
**********************

PAIRLIST
# algorithm: standard(0) (gromos96 like pairlist)
# grid(1) (XX grid pairlist)
# SIZE: grid cell size (or auto = 0.5 * RCUTP)
# TYPE: chargegoup(0) (chargegroup based cutoff)
# atomic(1) (atom based cutoff)
#
# algorithm NSNB RCUTP RCUTL SIZE TYPE
  0 5 0.8 1.4 0.4 0
END
```

Different algorithms can be selected for the generation of the pairlist (a list containing the atoms interacting with each other). Here, the grid based pairlist generation `ALGORITHM` (0) is selected. With this algorithm, the space is discretized into grid cells and only the neighboring cells are searched for interacting partners. The use of this algorithm results in a significant speed increase because the scaling of the algorithm is changed from $\mathcal{O}(N^2)$ to $\mathcal{O}(N)$. The pairlist is generated every 5th (`NSNB`) step. `RCUTP` and `RCUTL` are the cutoffs for the pairlist construction of the short-range and the long-range interactions.

```
**********************
The POSITIONRES block
**********************

POSITIONRES
# values for NTPOR
# 0: no position re(con)straining
# 1: use CPOR
# 2: use CPOR/ ATOMIC B-FACTORS
# 3: position constraining
# NTPOR NTPORB NTPORS CPOR
  1 1 0 2.5E4
END
```

The position restraining of the protein (solute) is handled here. `NTPOR` specifies the restraining by mean of a harmonic spring, the force constant being given by `CPOR` (overwritten). All the parameters indicated above as "overwritten" will be replaced by values specified in the job script. Have a look at the `equilibrium.jobs` job script file

```
cat equilibrium.jobs
```

The equilibration.jobs file lists the seven jobs that will be performed in a row. The switches of the first job are selected appropriately to generate initial random velocities (NTIVEL=1, TEMPI=60) appropriate for a temperature of 60 K. The job is then carried out at 60 K with strong position restraints. The following three jobs progressively increase the temperature by steps of 60 K and decrease the position restraint force constant by steps of one order of magnitude, down to zero. The next two jobs further raise the temperature to 298 K and the last one

to 348 K. The `subdir` column specifies the directory in which the job will run. The `run_after` column specifies which order the jobs are run.

Now you should in principle run `mk_script` (`mk_script @f eq_mkscript.arg`) and run the jobs (`./job_submit.sh`), but...

**OUT OF TIME RESTRICTION**
**The thermalisation/equilibration step has already been performed for you. The reason is that these 7 jobs would take around 14 h and we want to start the production runs this week.**

... so if you look at the content of the current directory, you already see all the files that would be produced in these 14 hours. Now you have to decide together with your MD simulation partner which of the two temperatures you will start your production MD with. Please choose one of the two following equilibrated files

- 298 K: `protein_6.cnf`
- 348 K: `protein_7.cnf`

And in the following, replace the "TEM" in the directory/file names by the temperature (298 K or 348 K) at which you are going to run your simulation

**(Q16)** Based upon the files generated by the 7 equilibration jobs, the two plots in Fig. 3 show the system temperature as a function of time and total energy, total kinetic energy, and total potential energy as a function of time. Briefly discuss them.

**(Q17)** What do you expect to happen during a longer MD simulation of both proteins at the different temperatures?

Let's continue with the start of the actual production runs

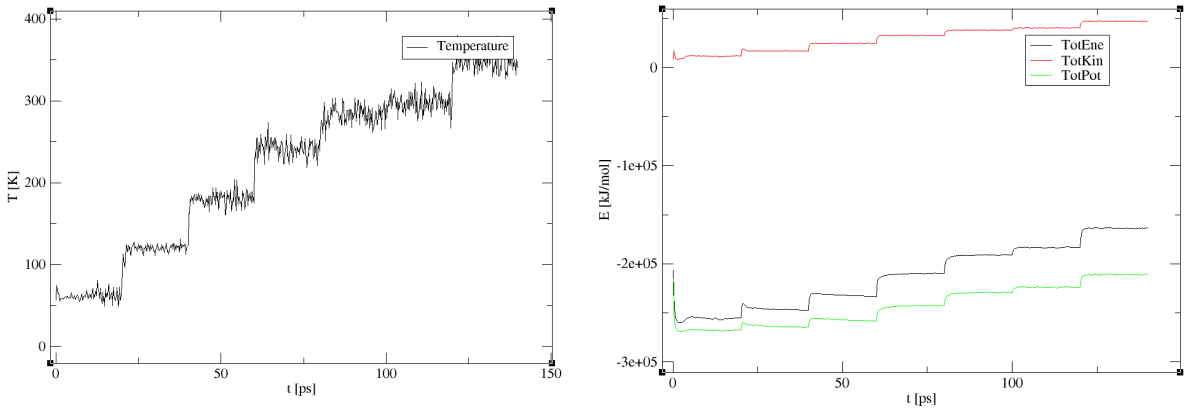


Figure 3: **Left** The temperature curve (black) is shown for the protein during the thermalisation of 140 ps. Each 7 steps represent the increase of the temperature till 348 K and the coupling of the temperature during this step. **Right** The overall energy (black line), Potential energy (green) and kinetic energy (red) are shown for the protein during the thermalisation time of 140 ps. The 7 steps reflect the increase in temperature till 348 K.

```
cd ../md
```

### 2.4.7 Molecular Dynamic Sampling Simulation

GROMOS++/MD++ PROGRAM NEEDED: `mk_script md`

INPUT FILES THERE: `md_mkscript.arg md.imd`

OUTPUT FILES THAT WILL BE CREATED:
`md_protein_*.imd md_protein_*.run md_protein.cnf md_protein_*.trc.gz md_protein_*.tre.gz`

TO DO: complete the `md_mkscript.arg` and `md.imd`

Your protein is now ready to enter the production MD simulation! We will now change from constant volume to constant pressure conditions. And again, the GROMOS++ `mk_script` is used to prepare the job scripts.

⇒ A more detailed explanation regarding the procedure can be found in Exercise 2.

Edit the `mk_script` argument file

```
vim md_mkscript.arg
```

Instead of the `@files`, the `@script` argument is now used. Also, the position restraints and joblist files are not needed anymore. Check and complement the file (missing fields which are marked "`TO_DO`"). The `@sys` flag should be set to `protein_298K` or `protein_348K` depending on the temperature you chose.

**WATCH OUT:** if you are going to run your simulation at 348 K, change the `coord` line from `protein_6.cnf` to `protein_7.cnf`. If you are going to run at 298 K, no such change is needed.

Then edit the md input file

```
vim md.imd
```

Check and complement the file (missing fields which are marked "`TO_DO`"). Check in particular the `MULTIBATH` block. Now you can create the 2 consecutive job scripts with the command

```
mk_script @f md_mkscript.arg
```

Now 2 `protein*.run` files are created. Have a look if all is ready for submission

```
vim job_submit.sh
```

Now submit the job scripts :-)

```
./job_submit_TEM.sh
```

**(Q18)** What are the main differences between the `md.imd` file and the `equilibration.imd` (previous step) file?

**(Q19)** How many nanoseconds does one of your jobs lasts and how long do you think it will take to run your simulations? (Hint: remember we told you above that the 140 ps thermalisation took 14 hours)

# 3 Week Two (Analysis)

For the analysis, *REMEMBER to replace "`TEM`" by the simulation temperature, that is 298 K or 348 K in the following section!!!* Now let's first go to the simulation folder to check your simulations

```
cd md
```

## 3.1 State of the Simulations

After a week of simulation, you will find some more files in your simulation folder. They include coordinate trajectory files (`*.trc.gz`), final configuration files (`*.cnf`), energy trajectory files (`*.tre.gz`), and GROMOS standard output files (`*.omd`). Have a look at these files and answer the following questions

**(Q20)** What is the length of each job? What is the total length of your simulation? What is the time interval between your stored trajectory frames?

## 3.2 Thermodynamic Parameters

PROGRAMS NEEDED: `ene_ana xmgrace`

INPUT FILES THERE: `ene_ana.arg`

OUTPUT FILES THAT WILL BE CREATED: `ene_ana_TEM.out solutemp.dat solvtemp.dat totene.dat totpot.dat totkin.dat pressu.dat`

```
cd ../ana_TEM/ene_ana
```

As a first step in the analysis, it is always good to check the convergence/fluctuation properties of some basic thermodynamic parameters. Here, we will consider

- Temperature (solute and solvent separately)
- Pressure
- Total energy
- Potential energy
- Kinetic energy

The values of these properties were calculated during simulations and written out by GROMOS in the energy trajectories files (`*.tre.gz`).They can be extracted by the program `ene_ana` (as usual, have a quick look at `ene_ana.arg` before!).

```
ene_ana @f ene_ana.arg > ene_ana_TEM.out &
```

If you get warnings at this point about the topology and the Boltzmann constant, you can ignore them and let `ene_ana` run. The topology would be needed for masses and molecule numbers, which we do not need, and the hardcoded value of the Boltzmann constant is used if no other is specified.

The output file `ene_ana_TEM.out` (have a look at it in your editor!) includes the averages, fluctuations, and estimated statistical uncertainties (by block averaging) of the thermodynamic properties monitored. Furthermore, separate files are written in which the time series of each property can be found: `solutemp.dat`, `solvtemp.dat`, `pressu.dat`, `totene.dat`, `totpot.dat` and `totkin.dat` for solute and solvent temperature, pressure, total energy, total potential energy, and total kinetic energy of the system, respectively.

**(Q21)** Look at the time series of the six quantities. Do they hint towards a proper equilibrium situation (no systematic drifts)?

**(Q22)** Plot the solute and solvent temperatures of the system. Which of the two has the larger fluctuations? Why?

**(Q23)** Compare your results with the ones of your colleague having the other temperature choice (298 K vs. 348 K). What are the main differences?

*A LITTLE PRESENT FROM YOUR ASSISTANT:* At this point, we are going to offer you a little present - 9 more ns of simulation! Then you can carry out all subsequent analyses considering 10 ns trajectories instead of 1 ns one. Also, everyone can now have the trajectories at the two temperatures if they wish (but you can also keep working with your colleague and one temperature each).

To make a link to the 10 ns trajectory at 298 K, type

```
cd
cd ex3
ln -s /usr/local/CSCBP/ex3/md_298K_10ns md_298K_10ns
```

To make a link to the 10 ns trajectory at 348 K, go to the appropriate directory ( `/ex3`) and type

```
ln -s /usr/local/CSCBP/ex3/md_348K_10ns md_348K_10ns
```

Now, you can go on working with the trajectory files of the assistant as if they were in your own directory (note, however, that you can only read and not write into these linked directories).

### 3.3 RMSD

PROGRAMS NEEDED: `rmsd xmgrace`

INPUT FILES THERE: `rmsd.arg`

OUTPUT FILES THAT WILL BE CREATED: `rmsd_TEM.dat`

```
cd ana_TEM/rmsd
```

The atomic positional root mean square deviation (RMSD) with respect to a given reference structure tells you how dissimilar the structure sampled in your simulation is to the reference (after least-squares-fit superimposition to remove the effect of overall translation and rotation). This property can be monitored as a function of time using the GROMOS++ program `rmsd` (as usual, have a quick look at `rmsd.arg` before!).

```
rmsd @f rmsd.arg > rmsd_TEM.dat &
```

**(Q24)** What do we use here as a reference structure?

**(Q25)** What is the purpose of the @pbc argument?

**(Q26)** Plot the RMSD time series at 298 K and 348 K, and comment on the difference

### 3.4 RMSF

PROGRAMS NEEDED: `rmsf xmgrace`

INPUT FILES THERE: `rmsf.arg`

OUTPUT FILES THAT WILL BE CREATED: `rmsf_TEM.dat`

```
cd ../rmsf
```

The atom-positional root mean square fluctuation (RMSF) gives us information about how locally flexible the protein is, i.e. how the different atoms of the protein fluctuate around their average positions. The results are typically averaged on a per residue basis. A reference structure is also required, but it is only used for the least-squares-fit superimposition to remove the effect of overall translation and rotation. This property can be monitored as a function of time using the GROMOS++ program rmsf (as usual, have a quick look at `rmsf.arg` before!).

```
rmsf @f rmsf.arg > rmsf_TEM.dat &
```

**(Q27)** Plot the RMSF over 10 ns as a function of the residue sequence number at 298 K and 348 K, and comment on the difference.

### 3.5 Hydrogen Bonds

PROGRAMS NEEDED: `hbond`

INPUT FILES THERE: `hbond.arg`

OUTPUT FILES THAT WILL BE CREATED:
`hbond_TEM.dat Hbts.out Hbnumts.out`

```
cd ../hbond
```

Hydrogen bonds are very important for the structure of the protein. The program `hbond` analyses the coordinate trajectories, and gives the information on the hydrogen bonds formed during your simulation. A hydrogen bond is considered to be present if the distance between a hydrogen atom (H) connected to a donor atom (D) is within 0.25 nm from an acceptor atom (A) and the D-H-A angle is larger than 135 degrees. The program `hbond` calculates average angles, distances and occurrences for all observed hydrogen bonds over the trajectories and prints out a time series of the observed hydrogen bonds (as usual, have a quick look at `hbond.arg` before!)

```
hbond @f hbond.arg > hbond_TEM.dat &
```

Now have a look at `hbond_TEM.dat` in your editor!

**(Q28)** Which hydrogen bond is the most populated one. Which protein secondary structure does this hydrogen bond reflect?

### 3.6 Secondary Structure

PROGRAMS NEEDED: `dssp xmgrace`

INPUT FILES THERE: `dssp.arg`

OUTPUT FILES THAT WILL BE CREATED: `dssp_TEM.dat 4-Helix.out Beta-Strand.out...`

```
cd ../dssp
```

The program `dssp` can monitor secondary structure elements of proteins over a simulated trajectory. The amino acids are allocated to a secondary structure according to the Define Secondary Structure of Proteins (DSSP) rules in Ref. [7]. The program summarizes the observed occurrences of the secondary structure elements including $\beta$-sheet/bridge, $\alpha$-helix, $\pi$-helix, 310-helix, turn and bend, and averages the different properties over the protein. In addition time series for every type of secondary structure element are written to file (as usual, have a quick look at `dssp.arg` before!).

```
dssp @f dssp.arg > dssp_TEM.dat &
```

Now have a look at `dssp_TEM.dat` in your editor!

**(Q29)** Have a look at `dssp.dat` file, how many $\alpha$-helix and $\beta$-strand does your protein contain during the simulation? Are they consistent with the experimental data?

To make a nice plot of the time series of the secondary structure element

```
xmgrace -p dssp_grace.prm *.out
```

**(Q30)** Insert this graph in your report and comment on it.

### 3.7 Ramachandran Map

PROGRAMS NEEDED: `tser xmgrace`

INPUT FILES THERE: `tser_ramach.arg`

OUTPUT FILES THAT WILL BE CREATED: `phipsi.dat`

```
cd ../ramachandran
```

A Ramachandran plot is a way to visualize the amino acids backbone dihedral angles $\psi$ and $\phi$ against each other. Backbone dihedral angles $\psi$ is defined through the backbone C-N- C$\alpha$-C atoms and $\phi$ is defined through the backbone N-C$\alpha$-C-N atoms. The Ramachandran plot (Figure 4) of the native protein structure is shown in Figure 4:

For our analysis, the GRMOS++ program `tser` can be used to calculate the backbone dihedral $\psi$ and $\phi$ angles from the coordinate trajectories (as usual, have a quick look at `tser_ramach.sh` before!).

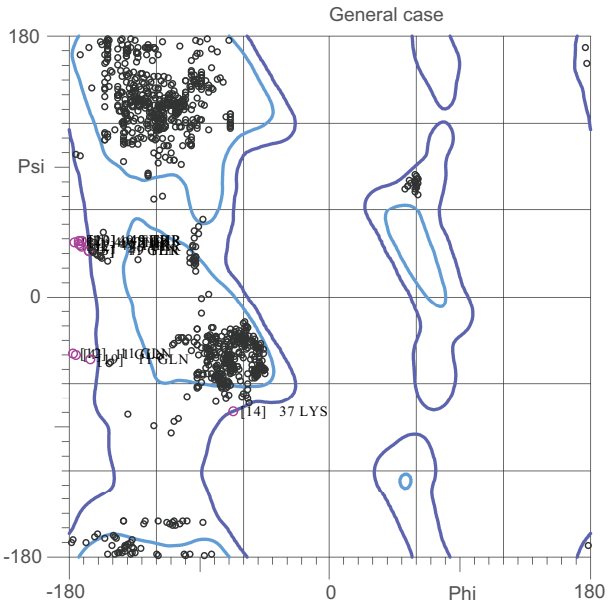


Figure 4: Ramachandran plot of the model 1 NMR structure of the protein.

```
./tser_ramach.sh &
```

This script calculates the backbone dihedral angles for all the residues except for the two terminal ones. Because this analysis is very time-consuming, only the last coordinate trajectory (0.5 ns) was analyzed. A file `phipsi.dat` is generated with $\psi$ as the first column and $\phi$ as the second column.

**(Q31)** Plot the Ramachandran plot. Do your simulations cover more or less the same region of the Ramachandran plot compared with the native protein structure?

**(Q32)** From the plot, can you tell what the main secondary structures of your protein are? Is there any difference between the plots from the simulations at two different temperatures?

## 3.8 Visualization

PROGRAMS NEEDED: `pymol`

INPUT FILES THERE: `protein_TEM_movie.pdb`

Beside the quite dry graphs analysis, visualizing the dynamics of the protein (molecular movie) is also important and fun! It will often give you ideas on what properties are interesting to monitor later in a numerical way. For this reason, you would normally do it right after finishing the simulations. For this exercise, however, we left it for the end (cherry on top of the pie!) because we can skip it if there is not enough time in the exercise session. Also, to save time, the movie has already been prepared for you (10 ns trajectory at the two temperatures, solvent removed). This was done using the GROMOS++ program `frameout`. To view it, just do

```
pymol /user/local/CSCBP/ex3/movies/protein_TEM_movie.pdb
```

To play the movie

```
mplay
```

To adjust the speed of the movie, go into the movie menu and subsequently you can adjust the frame rate to *e.g.* 5 FPS. You should now see your protein wiggling, diffusing and tumbling. What is more interesting to see is the internal movement of the protein. To be able to see this, all the time frames can be fit on the first frame by using the command

```
intra_fit protein
```

To center your protein use the command

```
orient
```

To get again the cartoon

```
hide all
show cartoon
```

However, this will not result in the nice secondary structure representation, instead it gives thick tubes. This is due to the fact that there is no secondary structure information in the pdb file. PyMOL can calculate for one frame the secondary structural elements and project this one on all the other frames by using the command

```
dss
```

**(Q33)** Can you see any differences in behavior of the protein at the different temperatures?

**(Q34)** Which secondary structure elements have a higher fluctuation? (Alpha helix / beta-sheets

# 4 Report

## 4.1 General Information

Just as for experimental approaches, mastering the technique is only one component in the scientific investigation of a given problem. Equally important components - in experiment as well as in simulation - are to:

- Formulate the question clearly
- Design an appropriate experiment to answer the question
- Interpret the results in terms of the question
- Be aware of the shortcomings and approximations of the employed method

To train these components (at least to some extent), we expect you to hand in a **short report** after each exercise series. This report should be a bit like the "results and discussion" section of a scientific article. No need to repeat all what you did. Just quote your main results and observations, possibly using tables or/and graphs, and discuss what scientific message can be extracted from them. To help you with this, at almost all the sections of the exercise one or more questions are asked. Please keep you report short and precise. If possible:

- Keep the length to **2 pages** but if you need more (max. 4 pages) this is fine (excluding the space taken by possible graphs or tables).
- Please use **Times New Roman** and **font size 11**.
- The **deadline** to hand in your report is the end of the week following the second week of the exercise!
- Hand in your report to the responsible assistant, either by e-mail (one single printable PDF document!) or on paper.
- See front page of this document for the exact date and assistant's contact details.
- Any suggestions/feedback for improving the exercise would be appreciated! (Likes, dislikes or improvements/changes). Thanks a lot!

## 4.2 Simulation Results

Give your answers to the questions/tasks given throughout the document (week 1 and week 2). Feel free to add any further material you consider useful/relevant.

## 4.3 Thinking Questions

(A) We start our simulations from an experimentally determined structure. But we could in principle start from any arbitrary structure of the protein (random coil or entirely extended chain) and equilibrate long enough. Why don't we do that?

(B) In Section 2.4.2 we stressed that the PDB structure is inferred based on NMR data and not determined by the NMR data. The NMR experiment [4] is able to determine 918 average proton-proton distances in the protein. These are the observables. The protein Gb88 has $N$ atoms (by now, you know $N$ from the PDB file), so that we want to determine 3N-6 Cartesian coordinates (the minus 6 is because we don't care about the position and orientation of the protein). These are the parameters. What is the corresponding observable-to-parameter ratio?

(C) The aliphatic-group geometry, the bond lengths, the bond angles and the improper dihedrals are fairly unambiguous. Assuming that we can take "standard" values for these, the number of parameters to be determined would actually be $3N' - M' - 6$, where $N'$ is the number (united) atoms and $M'$ the total number of bonds+angles+dihedrals+impropers. You can get both from the GROMOS molecular topology file. How does the observable-to-parameter ratio look like in this case?

(D) In Section 2.4.2, we said that PDB files for X-ray structures typically lack hydrogen atom coordinates, because these atoms have a too low electron density to be detected using X-rays. Sometimes, however, X-ray scattering experiments are complemented by neutron scattering experiments, and the PDB file contains then hydrogen-atom coordinates too. Can you explain how neutrons help? And can you guess why far fewer structures include such a neutron scattering determination of the hydrogen-atom coordinates, compared to those which only involve X-rays and exclude these coordinates?

(E) In Sections 2.4.3 and 2.4.5 we performed EM steps, once for the protein in vacuum and once for the computational box containing the protein in water (following the protocol of Ref. [4] and Ref. [5]). But one might argue that (1) the first EM is actually not needed, and even that (2) it might be better to skip it. Still, someone else might reply that (3) it does not really matter much. Can you formulate arguments in favor of (1), (2) and (3) for this virtual discussion?

(F) In Section 2.4.6, we have set NDFMIN = 3 and NSCM = -1000, the minus sign in the latter meaning that we remove both the overall (center-of-mass) translation and rotation of the computational box. Can you explain why this combination is in fact inconsistent? And why it is actually not very wise to remove the box rotation every 1000 steps in a simulation under PBC? What would then be the appropriate combination?

(G) In Section 2.4.6, we performed the thermalisation at constant volume, and in Section 2.4.7, we immediately switched to constant pressure. It might have made sense to already perform the end of the thermalisation at constant pressure. Describe briefly the changes you would need to make in `eq_mkscript.arg` and `eq.imd` in Section 2.4.6 so that the last three jobs are at constant pressure instead of constant volume.

(H) The following three graphs show the time evolutions of the temperature, total potential energy and RMSD considering two situations: your 140 ps thermalisation (in black) and a 140 ps simulation that was carried out by directly assigning random velocities appropriate to 298 K and not using any position restraints (in red). Based on these three curves, explain why the careful thermalisation is worth the effort.

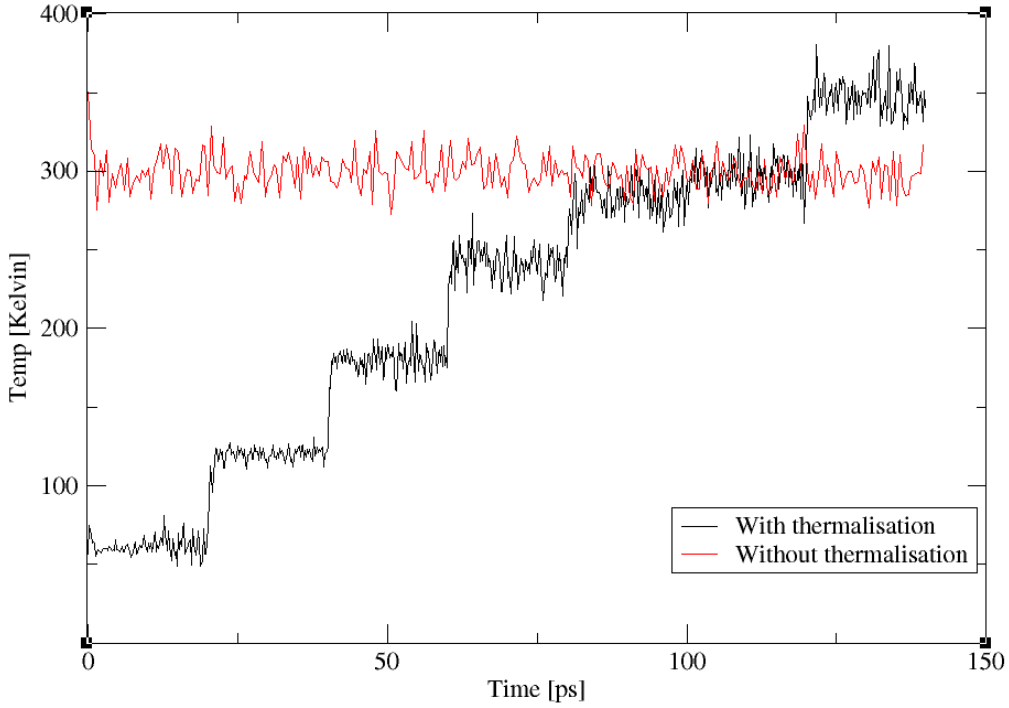


Figure 5: Equilibration step (150 ps) of the protein; protein with thermalisation (black), protein without thermalisation (red).

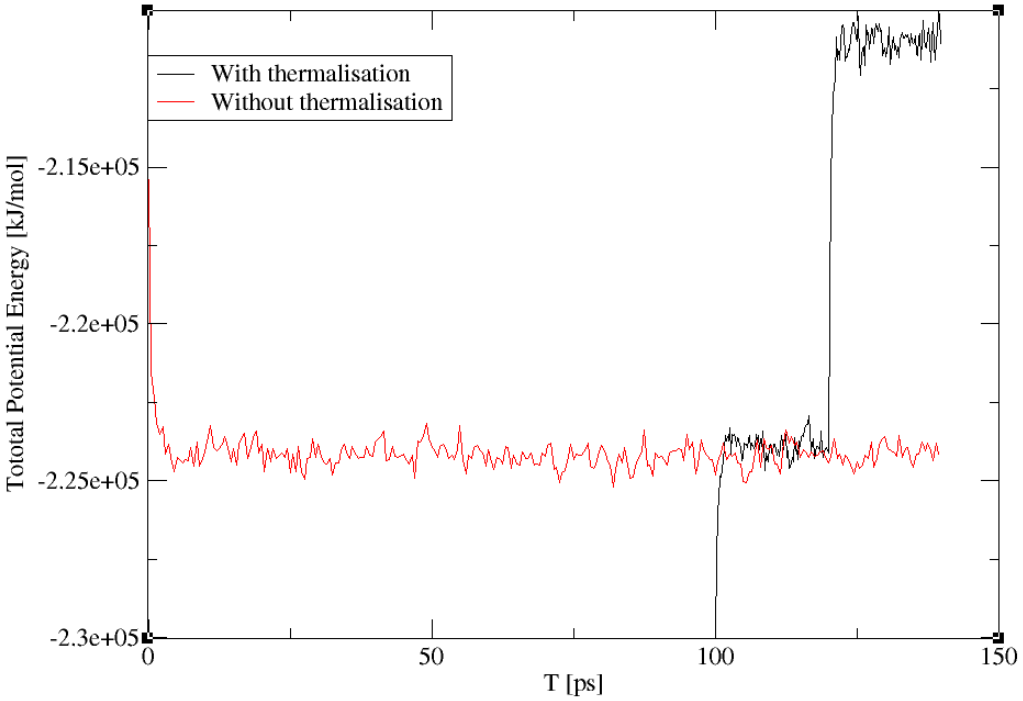


Figure 6: Equilibration step (150 ps) of the protein; protein with thermalisation (black), protein without thermalisation (red).

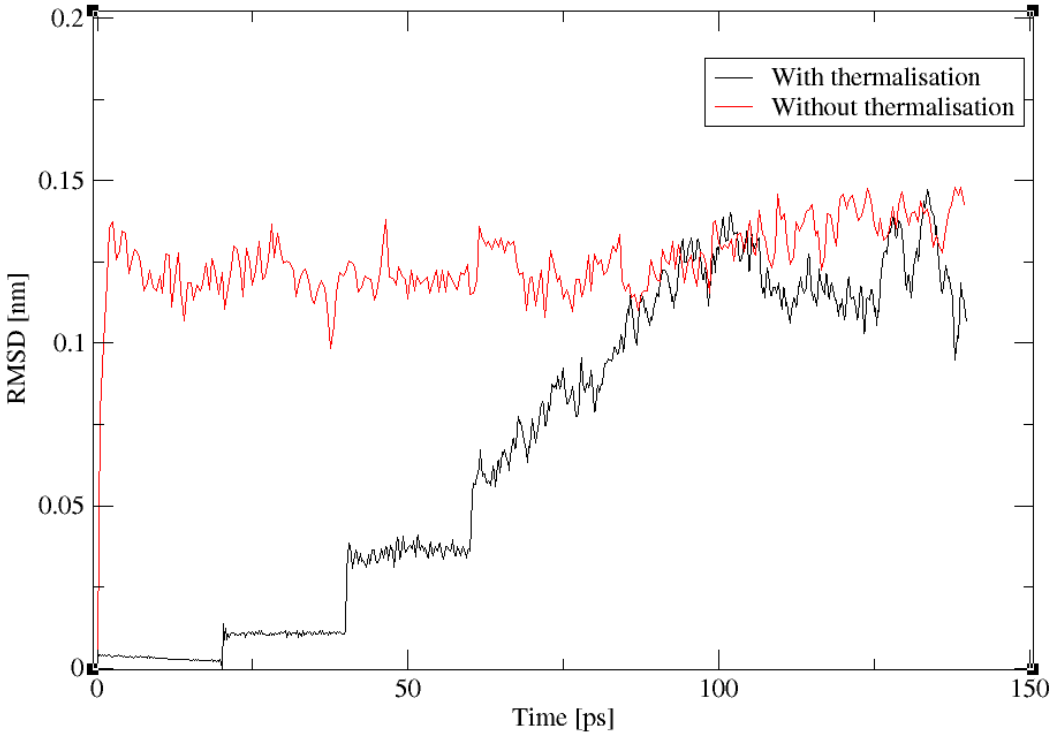


Figure 7: Equilibration step (150 ps) of the protein; protein with thermalisation (black), protein without thermalisation (red).

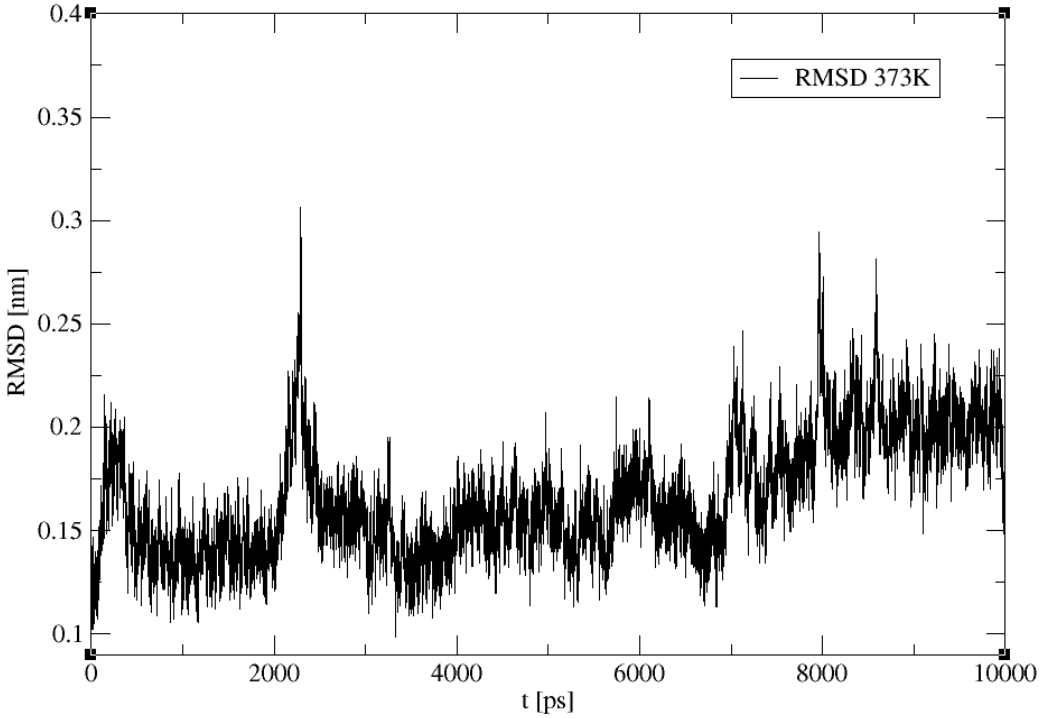


Figure 8: Additional RMSD plot - MD run at 373 K - protein does not unfold.

## Appendix A: Available files

In the main exercise directory, you will find `ex3.pdf`, the digital version of the document you are reading and eight sub-directories:

- `pdb/`
  Contains `protein_model1.pdb` the pdb coordinate file of the protein (model 1 from the NMR based conformations)
- `topo/`
  Contains the required file to build the proteins topology; `make_top_protein.arg` and the force field building blocks and parameters: `54a7.mtb`, `54a7.ifp`
- `coord/`
  Contains the required files to convert the protein starting coordinates to GROMOS format; `pdb2g96.arg, pdb2g96.lib`
- `min/`
  Contains the required files to perform an energy minimization of the configuration; `em_protein.imd`, `em_protein.run`
- `box/`
  Contains the required files to get the protein solvated in a box and a `frameout` input file for visualization;
  `sim_box_protein.arg`, `h2o.cnf`, `frameout_box.arg`
- `min_h2o/`
  Contains the required files to perform an energy minimization of the configuration in SPC water;
  `em_protein_box.run`, `em_protein_box.imd`
- `eq/`
  Contains the files for the thermalisation and equilibration of the protein;
  `protein_*.run, protein_*.imd, protein_*.omd, protein_*.trc.gz, protein_*.tre.gz, equilibration.jobs, eq_mkscript.arg, eq.imd`
- `md/`
  Contains `md_mkscript.arg, md.imd, job_submit_TEM.sh`

## Appendix B: Brief Overview of pH and pKa for non-Chemists

The pKa determines if a molecule keeps or gives its protons ($H^+$) away. The pKa is in turn dependent on the equilibrium acid dissociation constant, Ka, which expresses the acid/base concentration ratio of the reaction

$$HA + H_2O \rightarrow H_3O^+ + A^- \tag{1}$$

where HA represents the weak acid and $A^-$ the anion and the dissociation constant subsequently describes the equilibrium in dilute solution (mol/L)

$$K_a = \frac{[H_3O^+][A^-]}{[HA]} \tag{2}$$

The scale of acidity is often expressed as negative logarithm of Ka

$$\mathrm{pK_a} = -\log(\mathrm{K_a}) \tag{3}$$

The smaller the pKa value, the stronger the acid, and vice versa. The pKa values for the termini and side chains of amino acids are thus determined by several factors, such as their extend of hydrogen bonding, nature of their neighbours, etc. The pKa value can be determined by experimental methods *e.g.* NMR. You find in Table 1 the pKa values of several amino acids, which you can use in answering the questions.

## References


[1] GROMOS Tutorial Vol. 7, `www.gromos.net`.

[2] Huang, W., Lin, Z., van Gunsteren, W.F. *J. Chem. Theory Comput.* **2011** 7, 1237-1243.

[3] Alexander, A.A., He, Y., Chen, Y., Orban, J., Bryan, P.N. *Proc. Natl. Acad. Sci. U.S.A.* **2007** 29, 11963-11968.

[4] He, Y., Chen, Y., Alexander, A.A., Bryan, P.N., Orban, J. *Proc. Natl. Acad. Sci. U.S.A.* **2008** 105, 14412-14417.

[5] Allison, J.R., Bergeler, M., Hansen, N., van Gunsteren, W.F. *Biochemistry* **2011** 50, 10965-10973.

[6] Doig, A., Baldwin, R.L. *Protein Sci.* **1995** 4, 1325-1336.

[7] Kabsch, W., Sander, C. *Biopolymers,* **1983** 22, 2577-2637.

[8] Berendsen, H.J.C, Postma, J.P.M., van Gunsteren, W.F., Hermans, J. In: *Intermolecular Forces* B. Pullman ed., Reidel, Dordrecht, **1981**, 331-342