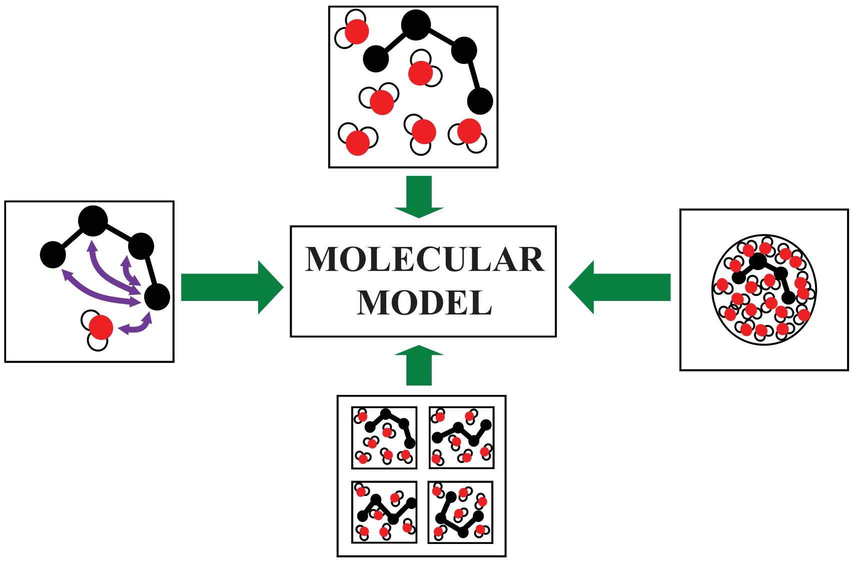
CSBMS

Classical Simulation of (Bio)Molecular Systems

Prof. P.H. Hünenberger



STUDENT EXERCISE SCRIPT HS19

(version August 2019)

http://www.csms.ethz.ch/education/CSBMS
(downloads: use your nethz-password)

CSBMS Computational Setup

CSBMS - Team

August 27, 2019

Contents

1 Before You Start

As is the case in most academic and industrial computational chemistry groups, the CSBMS exercises will require using computers at two different locations[1]:

- The computer you log into in the exercise room D267.4 will be referred to as your *workstation*. The workstation is the computer on your desk, where you do interactive and computationally inexpensive work (editing files, structure visualisation, web browsing, e-mail, ...).
- The computer you actually use for the heavy calculations, which is more powerful and kept largely free of any other processes besides number-crunching jobs. For CSBMS, this computer is actually a cluster of 10 computers (nodes) belonging to our research group and called realbeaver,[2] further referred to simply as *beaver*.

The beaver cluster has one special node, the *login node*, which you can access remotely from your workstation in an interactive session (*e.g.* a window on your workstation behaves like a window directly on the beaver login node). The nine other nodes of beaver, the *execution nodes*, cannot be accessed interactively. They are strictly reserved for calculations (jobs), *e.g.* GROMOS simulations, which you submit from the login node using a *queueing system*. Finally, you should realize that your workstation and the beaver cluster have distinct *storage spaces*, so that you may have to transfer files explicitly between your workstation and beaver to have them where you want. Figure 1 provides a schematic description of the computational setup of the course CSBMS.

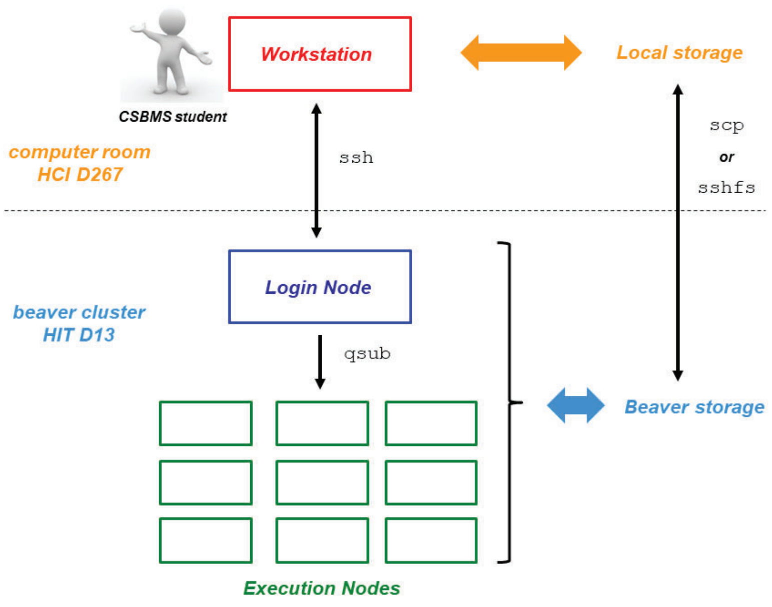


Figure 1: Description of the computional setup of the CSBMS course.

The main goals of this document are to:

- Make sure the workstation/beaver setup works properly for you (Section 2)
- Explain how to use and access the beaver cluster (Sections 3-4)
- Explain how to open interactive sessions with and transfer files from/to beaver (Section 5)

[1] Our group does not do it differently. We use workstations in our offices the daily interactive work, and the cluster "euler" of the ETH high performance computing center for running simulations.

[2]It is located in HIT D13 - feel free to ask an assistant if you want to visit it once!

- Provide a list of programs/files needed for the CSBMS exercises[3] (Sections 6-8)
- Explain how the queueing system works on beaver (Section 9)

2 Getting Started

Follow the series of steps below to make sure every component of the computational setup is working properly for you and to learn how to do the basic operations. More details on each step are provided in the corresponding sections of this document as indicated between parentheses.

1. Login into a workstation of D267.4 using your ETH (nethz) username and password.
2. Open a terminal session - this one will be accessing your workstation and we'll call it the *workstation terminal*.
3. Open another terminal session - this one will be accessing the beaver login node and we'll call it the *beaver terminal*.
4. *Beaver terminal*: access the beaver login node through `ssh` (Section 4).
5. *Beaver terminal*: change your password on beaver (Section 4.3).
6. *Workstation terminal*: try copying files from/to your workstation home to/from your beaver home by means of the command `scp` (Section 5.1).
7. *Workstation terminal*: mount your beaver home under your workstation home using `sshfs` (Section 5.2).
8. *Workstation terminal*: try again copying files from/to your workstation home to/from your beaver home, now by using the mounted directory, by means of the command `cp`.
9. *Workstation terminal*: unmount your beaver home using `fusermount` (Section 5.2)
10. *Workstation terminal*: try to start `firefox` (Section 6).
11. *Workstation or beaver terminal*: try to start your favorite editor like `gedit`, `vim` or `emacs` (Section 6).
12. *Workstation or beaver terminal*: try to start the plotting program `xmgrace` (Section 6).
13. *Workstation terminal*: try to start the visualisation program `vmd` (Section 6).
14. *Workstation terminal*: try to start the visualisation program `pymol` (Section 6).
15. *Beaver terminal*: check that you find the course-material directory (Section 7).
16. *Beaver terminal*: try to run the GROMOS `md_mpi` program (Section 8).
17. *Beaver terminal*: check the jobs in the queueing system (Section 9.1).
18. *Beaver terminal*: submit a test job to the queueing system (Section 9.2).
19. *Beaver terminal*: check again the jobs in the queueing system (Section 9.1).
20. *Beaver terminal*: remove your test job from the queue (Section 9.3).
21. Read about the beaver cluster (Section 3).
22. Read about the optimal usage of beaver (Section 10).

[3] In CSBMS, the login node of beaver will also be used for many interactive manipulations. It is OK because we are only a few students - and it was more convenient for us to install a number of programs on beaver than on the D267.4 workstations. In a more realistic setup, the login node of a cluster is really *only* for babysitting jobs and file transfers, and *not* for any other interactive manipulations (unless you want to be kicked out by angry sysadmins!)

23. Read about troubleshooting (Section 11).

24. If at this point you have headache, just take an aspirin.

If you completed all steps above then you know the basis of the CSBMS computational setup and operations - and can now turn to the fun part: the simulation exercises.

3 The Cluster

The beaver (realbeaver) cluster belongs to the Informatikgestützte Chemie (IGC) group at ETH Zürich. It is installed in HIT D13 and consists of ten physically distinct computers (nodes), nine execution nodes and one login node, dedicated to user login, filesystem operations and queueing system management. Each node comprises two AMD 6220 CPUs with 8 processors each (total of 144 processors in the cluster) and 32 GB memory[4]. This cluster is used by IGC and invited members for research and educational work.

3.1 Restrictions

The cluster is used simultaneously by many different people, so that some *policies* (restrictions) must be respected in the usage of the login node, of the storage space, and of the queueing system. The execution nodes are only accessible interactively to the system administrators.

3.1.1 Number of Simultaneous Logins

The maximum number of simultaneous login sessions per user is set to eight.

3.1.2 Memory Allocation

The maximum total memory per user at the login node is set to 512 MB.

3.1.3 Processes

The maximal duration of any process run at the login node is set to 30 minutes, and the maximal number of simultaneous processes is set to 20. The login node should not be used to run big calculations. Calculations that take more than 5-10 minutes should be submitted to the queueing system instead.

3.1.4 Disk Space and Usage

No disk quota is enforced at the moment, but users should not have more than 50 Gb in their beaver home (data in excess of this may be removed by the system administrators without warning in case of system malfunction). This disk space must be treated as scratch (temporary) space and only be used to store CSBMS-related data. Note that it is not backed up.

3.1.5 Queueing System

The following restrictions apply to the queueing system (see section 9).

The maximum number of simultaneous jobs per user is set to 16[5] and the default maximum job duration is 10 hours.

[4] This is more than enough memory for MD. For example, a MD simulation of 512 water molecules requires about 20 MB memory

[5] This implies a maximum of 4 simulateous jobs if you request 4 processors for each job.

4 Account and Login

4.1 Account

Each student will be provided with a beaver account and an initial password, that must be changed at the first exercise session for security reasons. Your username and password should be sent to you per e-mail already, otherwise ask the assistants. Never share your password with anyone else. Never use anyone's account even with his/her permission.

4.2 Login

The beaver cluster can be accessed within the ETH network through `ssh`. You can even run x-windows programs like `emacs` and `xmgrace` remotely if you enable window-tunneling (X11 forwarding) within.`ssh` [6]

The beaver cluster can be accessed using simply

```
ssh <your-beaver-username>@realbeaver.ethz.ch
```

In order to use graphical programs on beaver one should add the `'-X'` option after the `ssh` command. If you are lucky, you have the same login name on your workstation and on beaver, and then you can also use

```
ssh realbeaver.ethz.ch
```

4.3 Changing Password at First Login

In order to change your password after the first login type

```
passwd
```

Then follow the instructions.

5 Accessing your Files

There are two ways of accessing files in your beaver home. The first one is through a *network copy* and this is the preferred way when one has a slow internet connection to beaver, which is not the case for the computer room D267.4.

The other one is through a *local mount* of your beaver home into your workstation home. Mounting means that your beaver home will look exactly like a subdirectory in your workstation home (although the beaver disks are physically located in another room!). This is the preferred way when you have a fast connection to beaver. However it requires that you manually do the mount and, after usage, the unmount.

The following sections explain how to use either methods.

5.1 Network Copy

In order to copy files from/to beaver one can use the `rsync` or the `scp` commands[7]. They work as

```
 scp/rsync <options> <source> <destination>
```

where `source` and `destination` can be a directory or file.

For example, if you want to copy a directory named `folder_on_workstation` from your workstation current directory to your beaver home, you can type (in the workstation window)

```
 scp -r folder_on_workstation <your-beaver-username>@realbeaver.ethz.ch:
```

[6]For windows users: `http://realprogrammers.com/how_to/set_up_an_ssh_tunnel_with_putty.html`. For Linux/Mac please use the option `'-X'` within `ssh`

[7] Roughly speaking, `scp` copies whereas `rsync` updates. For a file, this does not make any difference. For a directory which does not yet exist in the destination folder, there is no difference either. But for a directory that exists in both source and destination folders, `rsync` will copy from the source only what is new or differs from the destination content (`scp`, in contrast, will copy the source directory inside the destination one, which is probably not what you want).

And to copy a directory called `folder_on_beaver` from your beaver home to your workstation current directory, you can type (in the workstation window)

```
scp -r <your-beaver-username>@realbeaver.ethz.ch:folder_on_beaver .
```

Do not forget the option '`-r`' for the `scp` command, which is needed to copy entire directories with their content. If you have the same login name on your workstation and on beaver, then you can omit the "`<your-beaver-username>@`". Finally, you can use most of unix ways to specify files/directories, *e.g.*

```
scp -r ../doc/folders[0-9]* <your-beaver-username>@realbeaver.ethz.ch:doc/my_folders/
```

should work.

The same actions using the program `rsync` look like[8]

```
rsync -avz folder_on_workstation <your-beaver-username>@realbeaver.ethz.ch:
```

and

```
rsync -avz <your-beaver-username>@realbeaver.ethz.ch:folder_on_beaver .
```

5.2 Mounting Directories Through `sshfs`

Sometimes, repeatedly copying files/directories from one machine to another using `scp` or `rsync` can be fairly annoying. An alternative is to mount the directory that contains the required files on beaver into your workstation home directory, so that you can use it as if it was a subdirectory of your workstation home.

One way of doing so is through `sshfs`. This program uses the `ssh` protocol to share files through a secure channel. The mounted folders are visible through the `ssh` protocol, which means in particular that the read/write/execute rules of file access are still enforced. In order to avoid security issues with your data and decrease the network usage of the cluster, one must not forget to unmount the folder when the mount point is no longer required. In order to properly use the `sshfs` system you must go through the following steps:

1. Create an empty mount-point directory.
2. Mount the external folder to the mount point directory.
3. Use the mount point.
4. Unmount the folder, so as to release the network connection.

These are described below.

5.2.1 Creating the Mount Point Folder

You can create the mount point folder wherever you have write permission, using the command `mkdir`

```
mkdir <mount-folder>
```

5.2.2 Mounting the External Folder to the Mount Point Folder

For simplicity, we will assume that you want to mount your beaver home (you could as well mount a subdirectory of this home). In order to mount your home directory just type

```
sshfs <your-beaver-username>@realbeaver.ethz.ch:/home/<your-beaver-username> <mount-folder>
```

[8]The option '`-a`' forces the files to be transferred in "archive" mode, which ensures that symbolic links, devices, attributes, permissions, ownerships, etc. are preserved in the transfer. The option '`-z`' forces compression of files to reduce the size of data portions of the transfer. And the option '`-v`' increases program verbosity.

5.2.3 Unmounting the Mount Point Folder

In order to unmount the mount point folder just type:

```
fusermount -u <mount-folder>
```

6 List of Programs

In this section there is a list of programs that might be necessary during the CSBMS exercises. Some programs are available on the workstation, some on the beaver login node, and some on both, as indicated.

Table 1: List of programs that might be necessary during the CSBMS exercises.

Program	Available	Type	Notes
`firefox`	workstation	web browser	
`vim`	workstation/beaver	editor	
`emacs`	workstation/beaver	editor	
`xmgrace`	workstation/beaver	2D Plotting tool	
`python`	workstation/beaver	python interpreter	
`vmd`	workstation	molecular visualisation	located at /opt
`pymol`	workstation	molecular visualisation	
GROMOS programs	beaver	molecular simulation	most recent release

All graphical programs should be used on the workstations.

7 Course-Material Directory

The CSBMS course material can be found on beaver at the following folder

```
/usr/local/CSBMS
```

There are subdirectories for each of the exercises (plus one for this document)

8 Using GROMOS

The GROMOS program compiled with MPI (multiprocessor) is available on beaver (login as well as execution nodes). Its most recent release can be found in the folder

```
/opt/progs/gromos
```

The GROMOS simulation program `md_mpi` can be found at

```
/opt/progs/gromos/bin/md_mpi
```

8.1 GROMOS Manual and Doxygen Files

The 9 volumes of the GROMOS manual can be accessed in the folder

```
/usr/local/gromos/GROMOS_pdf
```

For further details of GROMOS program implememtation you can access the doxygen documentation through a browser pointed at the following addresses `https://dstar.ethz.ch/gromos/md++-1.3.1/` (for md++ documentation) or `https://dstar.ethz.ch/gromos/gromos++-1.3.1/` (for gromos++ documentation).

8.2 GROMOS mk_script Library File

The GROMOS `mk_script` program relies on a template file in order to create a sequence of calculations. These files define among other things, the number of processors and the type of parallelisation strategy that will be used by the GROMOS `md_mpi` program. As discussed above the beaver cluster only contains the MPI version of GROMOS. Therefore parallel calculations using the GROMOS program on this cluster must use MPI. GROMOS does not scale well above 8 processors, thus a smaller number of processors should be used. The mk_script library files defining the MPI environment and three different number of processors (1, 2 and 4) are available at the folder:

```
/usr/local/CSBMS/lib
```

9 Queueing System

This section describes the basic commands to get started with the Sun Grid Engine (SGE) queueing system installed on beaver.

9.1 Queue Status

To check the status of your jobs in the queue use

```
qstat
```

For further information you can add the flags `'-f'` and/or `'-u \*'`

```
qstat -f -u \*
```

The `'-f'` option specifies a "full" format display of information, and causes summary information on all queues to be displayed along with the queued job list.

The `'-u user, ...'` displays information only on those jobs and queues being associated with the users from the given user list. Queue status information is displayed if the `'-f'` or `'-F'` options are specified additionally and if the user runs jobs in those queues. An asterisk `'\*'` can be used as username wildcard to request any users' jobs be displayed.

On beaver, a shortcut for the command above is

```
queue
```

9.2 Submitting Jobs to the Queue

In order to submit jobs to the queue use the following command:

```
qsub -N <job-name> -cwd -pe mpi <NSLOTS> ./<script-name>
```

where `<job-name>` is how you want to name your job. Note that job names cannot start with numbers. `<script-name>` is the name of the script/program you want to run. `<NSLOTS>` defines the number of processors to run the calculation. For better performance you should match the `<NSLOTS>` variable to be equal to the number of processors defined by the `mk_script` library file.

The jobs should be started from the user home directory or a subdirectory within this home.

9.3 Removing Jobs from the Queue

In order to remove jobs from the queue use the following command:

```
qdel <ID>
```

where `<ID>` is job number given by the queue system to your job. To find the job `ID` use the command `qstat`.

9.4 Further Information

For further information please refer to the full SGE user guide on beaver (`http://realbeaver/sgedoc/820-0699.pdf`).

10 Optimal Usage of beaver

The optimal way of using beaver is the combination of the following conditions:

- User passwords are more than 8 characters long and include numbers, capitalized and non-capitalized letters as well as special characters.
- Files are accessed through the `sshfs` protocol and the mount point folders are mounted only as long as they are needed.
- The required (side)programs are used on the workstations and not on beaver.
- GROMOS parallelisation is achieved through MPI.
- GROMOS mk_script program uses the provided mk_script library files.
- No job with more than 8 processors is submitted to the queueing system.
- User folders are kept below 50 GB and store only CSBMS-related files.
- Calculations and analysis are submitted to the queue system, which should guarantee the fair share of resources.

11 Troubleshooting

If you have any questions or experience any problem please contact the assistant.

Getting Started Really Quickly

1. Login into a workstation of D267.4 using your ETH (nethz) username and password;
2. Open a terminal window - this one will be accessing your workstation and we'll refer to it as *workstation window*;
3. Open another terminal window - this one will be accessing the beaver login node and we'll refer to it as *beaver window*;
4. *Beaver window*: access the beaver login node using `ssh`:

   ```
   ssh -X <your-beaver-username>@realbeaver
   The password is given by the assistant.
   Please change your password (use command passwd).
   ```

5. *Workstation window*: mount your username folder from your workstation's home using `sshfs`;

   ```
   mkdir ~/beaver-home
   sshfs <your-beaver-username>@realbeaver: beaver-home
   ```

6. *Beaver window*: Copy the `/usr/local/CSBMS/ex1` folder to your home;

   ```
   cp -r /usr/local/CSBMS/ex1 ~
   ```

7. *Workstation window*: Open the file EGM.top from the folder on `ex1/topo` on your home directory.

   ```
   cd ~/beaver-home/ex1
   <your-prefered-text-editor> topo/EGM.top
   ```

   where `<your-prefered-text-editor>` can be `gedit`, `emacs` or `vi`

Please before you leave:

1. *Workstation window*: Do not forget to unmount the `sshfs` folder mounted on the Workstation;

   ```
   cd
   fusermount -u beaver-home
   ```

2. *Beaver window*: logout

   ```
   exit
   ```

3. Please logout from your workstation.

Thanks for your cooperation!

CSBMS: Exercise 1

Topology Creation & Parameter Transferability

Document version: 27.08.2019

Exercises week 1:	24.09. or 26.09.
Exercises week 2:	01.10. or 03.10.
Deadline for the report:	13.10.
Contact:	`linker@phys.chem.ethz.ch`
	Stephanie Linker - HCI G239

Summary

In this first exercise, you will create a model for ethylene glycol monoacetate, a molecule for which no parameters exist within the GROMOS force field, by combining well-tested GROMOS parameters from similar molecules, ethyl acetate and propanol. This means that you will write most of the corresponding molecular topology file by hand, thereby learning to know the different topological parameters. To assess the quality of your model, you will run simulations of the compound in the liquid state, calculate its density (ρ) and heat of vaporization (ΔH_{vap}), and compare these with experiment. You will also qualitatively investigate the effect of the environment on the conformational properties of the molecule, by running simulations of the compound in vacuum and in water.

1 Introduction

Any GROMOS simulation using the program `md` (or `md_mpi`) requires at least three files to be provided, namely

- an *input file* (flag `@input`), containing all switches and parameters specifying the desired simulation run (*e.g.* number of steps, timestep, writeout frequency, temperature and pressure coupling, ...),
- a *topology file* (flag `@topo`), containing the specification of the atom content, force-field terms and force-field parameters for the system to be simulated (*e.g.* number and types of atoms, bond, angle, improper and dihedral angle definitions, charges and Lennard-Jones interaction parameters, ...),
- a *starting configuration file* (flag `@conf`), containing the starting coordinates of all atoms and, for a continuation run, the corresponding starting velocities (possibly along with other relevant configurational information).

For this first exercise, all the input and configuration files will be provided to you (but you can still have a look at them for the sake of curiosity!), and the focus will be on the *topology file*. In many situations, creating a GROMOS molecular topology file for your system is relatively easy. This is the case when the GROMOS force field already includes parameters for the molecule of interest or, considering a polymer, for the corresponding monomers. These molecules or monomers are referred to as *building blocks*, and can be assembled using standard GROMOS tools. This will be the situation in Exercise 2. But today, you are not that lucky. You will have to construct your topology file by hand for a new molecule (and perform an initial validation of the resulting model against experimental data).

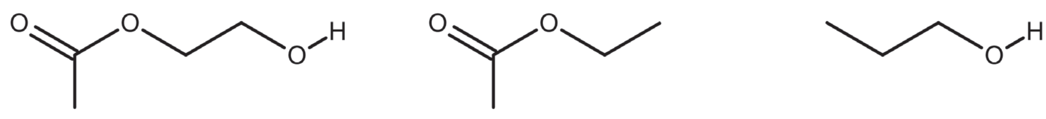
2 Week 1 - Creating the Topology / Starting the Simulations

The goal of this exercise is to create a model for ethylene glycol monoacetate (EGM, see Figure 1), an organic molecule that is in the liquid state at room temperature and ambient pressure. A quick check (*e.g.* using the list of building blocks in the GROMOS manual volume 3) reveals that this specific compound is not available as a ready-made building block in the currently available force fields. On second look, it is not so bad, since two similar compounds, ethyl acetate and propanol (EAE and PPL, see Figure 1), have been parametrised within the GROMOS $53A6_{OXY}$ force field [1], which should be a good source for the parameters needed using a *transferability assumption.* Nothing guarantees that the resulting model will be good, *i.e.* it will still need to be validated (and further refined if not sufficiently good). So, we are going to proceed in three steps:

- Construct the topology of EGM by analogy with EAE and PPL, first in terms of "drawings" (Section 2.1).
- Write this topology into a topology file, with a format recognised by GROMOS (Section 2.2).
- Perform an initial characterisation/validation of this model using simulations (Section 2.3).

2.1 Defining the Topology

Your first task is to create the topology for a single EGM molecule. For that, let us start by looking at Figure 1 and determining the different atom types found in the molecule. An *atom type* corresponds to a given atom in a specific chemical environment. This is a relatively fuzzy definition that arises from the observation that one can make "better" force fields by assigning different parameters to the same atom in distinct contexts, as suggested by chemical intuition. For example, the $53A6_{OXY}$ force field [1] could only achieve a sufficiently accurate description of (uncharged) oxygen-containing organic compounds by distinguishing three types of oxygen atoms (labelled O, OE and OA, see below). Also remember that for efficiency (and historical) reasons, GROMOS represents an aliphatic carbon atom and the attached non-polar hydrogen atoms as a single "atom" called a *united-atom.*



(a) EGM (b) EAE (c) PPL

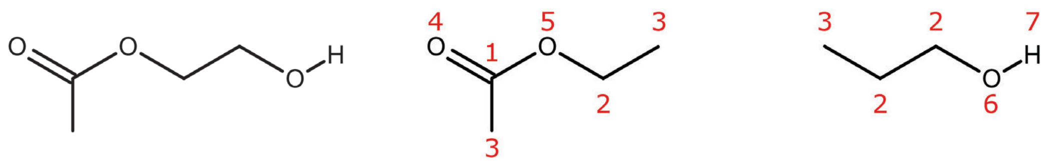
Figure 1: Ethylene glycol monoacetate, ethyl acetate and propanol

Given these considerations, EGM is built from the following atom types:

1. C, a plain carbon atom;

2. CH2, a carbon united-atom with two implicitly attached hydrogen atoms;
3. CH3, a carbon united-atom with three implicitly attached hydrogen atoms;
4. O, a carbonyl oxygen atom;
5. OE, an ether or ester oxygen atom;
6. OA, an alcohol oxygen atom;
7. H, an explicit hydrogen atom.

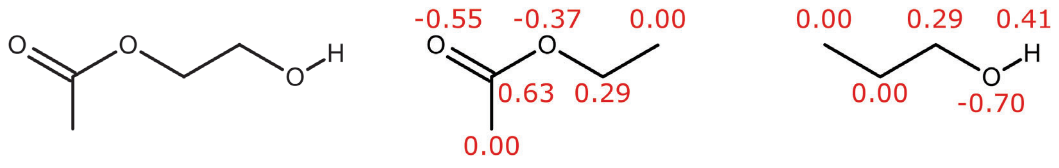
In the following, we will use the above numbering for the integer atom code (`IAC`) of a given atom type[1] (*e.g.* the `IAC` of atom type OE will be 5). The `IAC`s of the atoms of EAE and PPL are shown in Figures 2b and 2c. Now you can (and should!) add the `IAC`s for EGM in Figure 2a.



(a) EGM (b) EAE (c) PPL

Figure 2: Atom Types (`IAC`)

By choosing an `IAC` for each atom in the molecule, you have actually decided the form of the van der Waals interaction between any pair of atoms (short-range repulsion when the atoms overlap + longer-range attraction due to dispersion). This is because GROMOS determines the parameters of this interaction based on the `IAC` of the two interacting atoms[2]. The electrostatic (Coulombic) interaction, however, is not determined by the `IAC`. For this one, you have to assign partial charges to all atoms in your molecule. Because we have partial charges for EAE and PPL from the 53A6$_{\mathrm{OXY}}$ force field, we will simply assume here that these charges can be directly transferred to EGM by analogy. The partial charges of the atoms of EAE and PPL are shown in Figure 3b and 3c. Now you can (and should!) add the partial charges for EGM in Figure 3a.



(a) EGM (b) EAE (c) PPL

Figure 3: Atomic Partial Charges

[1]The 53A6$_{\mathrm{OXY}}$ force field considers many more molecules than just EAE and PPL, and has thus as many as 53 atom types. However, for simplicity, we have taken the types relevant for EGM and renumbered them from 1 to 7.

[2]This is not entirely true. Within a molecule, you still need some additional information on the topological relationship between the pair, *i.e.* we need to know the `IAC`s of the two interacting atoms and whether they are in a normal, third-neighbour or excluded relationship.

In a next step, the bond stretching, bond-angle bending, improper dihedral-angle distortion and proper dihedral-angle torsion potential energy terms defining the covalent flexibility of the molecule have to be defined. The first three types of terms are defined by a reference value (length or angle) and a force constant regulating the strength (the energetic cost of a deviation away from the reference value). The torsional terms are periodic and characterised by a multiplicity and a phase shift instead of a single reference value.

These parameters can be obtained from experimental structure-determination (*e.g.* X-ray or NMR) and spectroscopic (*e.g.* IR) measurements, or from quantum mechanical (QM) calculations. Fortunately, the parameter optimisation was already performed for EAE and PPL by the authors of the $53A6_{OXY}$ force field, so that we will simply assume that they are transferable by analogy to EGM. The covalent parameters of the $53A6_{OXY}$ force field relevant for EGM are listed in Table 1. For each of the four types of term, each set of parameter is assigned a type code for the ease of further reference.[3,4] The type codes of the bond stretching, bond-angle bending, improper dihedral distortion and proper dihedral torsion terms for EAE and PPL are shown in Figures 4b - 7b and Figures 4c - 7c, respectively[5]. Now you can (and should!) add the corresponding type codes for EGM in Figures 4a - 7a.

2.2 Writing the Topology File

Congratulations! You have now determined all potential energy terms present in the EGM molecule, *i.e.* the form of the van der Waals, electrostatic and covalent interactions. Time to let GROMOS know about this success. And for this, you have to encode your freshly acquired knowledge into a *topology file.*

If you have not done so yet, copy the main exercise directory `ex1/`[6] to your home directory. See Appendix A for an overview of the files found in the directory.

Open the file `topo/EGM.top` in your favourite text editor. In general, a GROMOS file consists of a series of *blocks*. Each block starts with the block name in capital letters and ends with the capitalised keyword END, both required to be on separate lines. Each block must contain a well-defined number of entries (character strings, integer numbers or real values, depending on the specific block) in a defined order. It is important to prepare GROMOS files with great care, as input errors can only be detected in some cases. In the file `EGM.top`, you will find a sketch for the topology file of EGM based on the model you constructed in Section 2.1. All the required blocks are listed, but some have been left empty, with the line `# TODO` instead of a content.[7] Your task is to fill these blocks yourself.

The subsections 2.2.1-2.2.4 below describe in more details the block content of the file (and

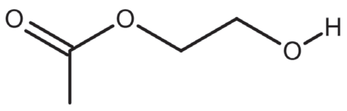
[3] Here again, the $53A6_{OXY}$ force field considers many more molecules than just EAE and PPL. For simplicity, we have only taken the types relevant for EGM and renumbered them from 1.

[4] Note that GROMOS can use two different forms of potential-energy terms for both bond stretching and bond-angle bending (the form employed in a given simulation is selected in the input file). For the bond stretching, the quartic and harmonic force constants are listed with a "q" and a "h" superscript, respectively. For the bond-angle bending, the cosine-harmonic and angle-harmonic force constants are listed with a "c" and a "h" superscript, respectively.

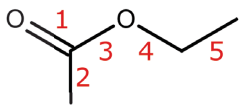
[5] The proper dihedral torsion potential number 1 is defined using the atoms (CH3)-(C)-(OE)-(CH2).

[6] See the document "CSBMS computational setup" for the exact location of the directory

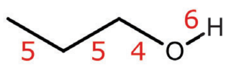
[7] Note that in a GROMOS file, any line starting with a hash (`#`) is a comment, *i.e.* the entire line ignored upon reading. Feel free to write your own explanatory comments when making your topology file, it is actually a good practise.



(a) EGM

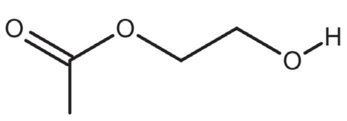


(b) EAE

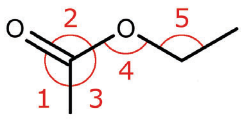


(c) PPL

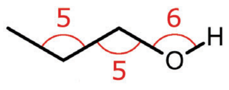
Figure 4: Bond Stretching Potentials



(a) EGM

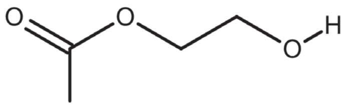


(b) EAE

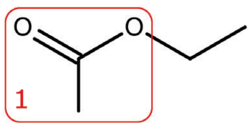


(c) PPL

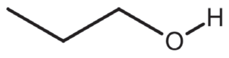
Figure 5: Bond-Angle Bending Potentials



(a) EGM

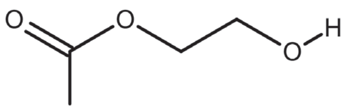


(b) EAE

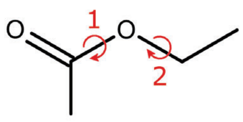


(c) PPL

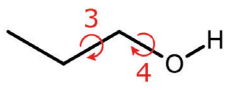
Figure 6: Improper Dihedral Distortion Potentials



(a) EGM



(b) EAE



(c) PPL

Figure 7: Proper Dihedral Torsion Potentials

[ex 1]

bond type code	K_b^q /[kJ mol^{-1} nm^{-4}]	K_b^h /[kJ mol^{-1} nm^{-2}]	$b_\circ$ /[nm]
1	1.66×10^7	5.02×10^5	0.123
2	7.15×10^6	3.35×10^5	0.153
3	1.02×10^7	3.77×10^5	0.136
4	8.18×10^6	3.35×10^5	0.143
5	7.15×10^6	3.35×10^5	0.153
6	1.57×10^7	3.14×10^5	0.100

(a) Bond Stretching Potential

angle type code	K_θ^c /[kJ mol^{-1}]	K_θ^h /[kJ mol^{-1} deg^{-2}]	$\theta_\circ$ /[deg]
1	750	0.153	125.0
2	700	0.153	122.0
3	545	0.140	113.0
4	635	0.153	117.0
5	530	0.140	111.0
6	450	0.122	109.5

(b) Bond-Angle Bending Potential

improper type code	K_ξ /[kJ mol^{-1} deg^{-2}]	$\xi_\circ$ /[deg]
1	0.0510	0.0

(c) Improper Dihedral Distortion Potential

dihedral type code	K_ϕ /[kJmol^{-1}]	δ /[deg]	m
1	16.70	180.0	2
2	3.77	0.0	3
3	5.92	0.0	3
4	1.26	0.0	3

(d) Proper Dihedral Torsion Potential

Table 1: Potential Energy Terms in the 53A6$_{\mathrm{OXY}}$ force field

[ex 1]

what you have to place into the blocks), from top to bottom. Subsection 2.2.5 tell you how to do a first consistency check of the file you created using the GROMOS program `check_top`.

2.2.1 Title, Constants, Atom and Residue Names

First comes the `TITLE` block. Here, you can give a title consisting of any number of lines. This is merely a help for yourself, to remember what kind of topology you are doing, very useful in case you are working with dozens of topologies at a time or have to find out half a year later what exactly you were doing in this file (the topology title will also be printed in the `md` output, so you know what topology was used for a specific run). Type a descriptive title replacing the `# TODO` tag, so that your block looks something like

```
TITLE
     My first GROMOS topology
     Compound: Ethylene glycol monoacetate (EGM)
     Author: Sir Isaac Newton
     Note: Adapted from ethyl acetate and propanol in 53A6_OXY
     Date: September 22, 2015
END
```

The next two blocks are already filled and need no change. The first one, `PHYSICALCONSTANTS`, is rather self-explanatory and sets a number of physical constants. The second one, `TOPVERSION`, is used internally by the GROMOS program package to keep track of different topology versions.

The following block `ATOMTYPENAME` contains the number of distinct atom types in your topology[8] (first line), then lists their name in order of ascending `IAC` (each on a separate line). For simplicity, we already filled out this block for you. You will recognise the 7 atom types of Section 2.1, plus an additional oxygen atom called `OW` and given the `IAC` of 8. It will be used for the representation of water.

The `RESNAME` block contains the number of residues (monomers) in the molecule, then lists their name in order of ascending residue number. For simplicity, we already filled out this block for you. Your molecule is not a polymer, so it has only one "residue" which we called `EGM`. This string will be used to refer to the residue in the analysis programs, but has no influence on the parameter selection.

2.2.2 The `SOLUTEATOM` Block

The `SOLUTEATOM` block contains the number of atoms in the molecule (first line), then lists the atoms in sequence and provides information on a per-atom basis (two lines per atom[9]). For simplicity, we already filled out this block, but you should definitely go through it very carefully and understand what is there. For each atom, the two lines must list in sequence:

[8]In general, this block will contain not only the types you need for your specific molecule, but the entire set for a given force field, *e.g.* 53 types for the $53A6_{OXY}$ force field. This is because we do not want to renumber the atom types for every new molecule. We better use one numbering for all atom types in a force field, take the types we need for the given molecule, and ignore the others. For the present exercise, however, we did the renumbering and only include the 7 useful atom types plus one needed for water.

[9]It would also work to make these two lines a single one (GROMOS moves to the next atom whenever it has all records for one atom, irrespective of the number of line breaks), but the two-line format is more readable.

- `ATNM`: The atom number.
  This number will be used to define the covalent potential-energy terms (*e.g.* in the list specifying the bonded atom pairs). The numbers of the different atoms should start from one and be incremental in the list[10]. For your molecule, the numbering we chose is shown in Figure 8a.[11]

- `MRES`: The residue number.
  This number specifies the residue to which the atom belongs. In our case, having only one residue, this will always be `1`.

- `PANM`: The atom name.
  This string will be used to refer to the atom in the analysis programs, but has no influence on the parameter selection[12]. For your molecule, the naming we chose is shown in Figure 8b.

- `IAC`: The integer atom code.
  The `IAC` is used to define the (Lennard-Jones) interaction parameters associated with this atom. These parameters are given later in the form of a two-dimensional matrix, the lines and columns of which are the `IAC`s of the two interacting atoms (see `LJPARAMETERS` block later). The `IAC`s listed in the block should match those you have in Figure 2a.

- `MASS`: The atomic mass, given in u.
  Watch out that the masses of the united-atoms include those of the attached hydrogen atoms.

- `CG`: The atomic partial charge of the atom, given in e.
  The charges listed in the block should match those you have in Figure 3a.

- `CGC`: The charge group code (boolean, `0` or `1`).
  For different reasons[13], sets of successive atoms are grouped into so-called charge groups (CG). A `0` indicates that the current CG starts or continues. A `1` indicates the last atom of the CG. So, your molecule has 3 CG. Try to draw them in Figure 3a. What are their net charges?

- `INE`: The excluded-atom list.
  Excluded-atom pairs are pairs of close covalent neighbours in a molecule (*i.e.* normally those separated by one or two bonds) which are exempted from mutual van der Waals and electrostatic interaction. For a given atom, the `INE` record specifies the number of atoms with higher `ATNM` sequence numbers that belong to the exclusion list of this atom, then lists their `ATNM` sequence numbers in ascending order. Are the excluded pairs listed in the file those you would expect considering the structure of your molecule?

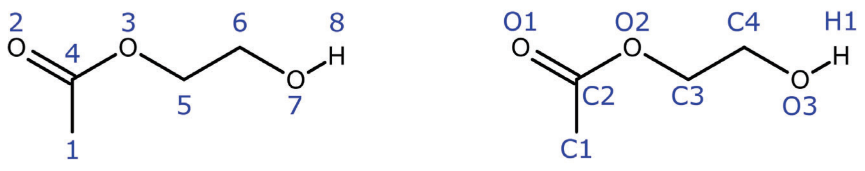
[10]Note that although the list in the file must be sequential, there are still many different ways to number the atoms of a molecule (some choices make your life easier than others, though).

[11]Warning: Do not confuse the atom number `ATNM` in Figure 8a with the atom type `IAC` in Figure 2a. The first one is just a "label", the second one determines the van der Waals interactions.

[12] Here also, there are many different ways to name the atoms of a molecule (some choices make your life easier than others, though).

[13] These are explained in the lecture: computational speed-up (making a list of CG pairs is faster than making a list of atom pairs) and reduced cutoff noise (cutting off interactions between neutral CG induces less noise than cutting off interactions between charged atoms). The CGs should be reasonably small and as much as possible overall neutral.

- `INE14`: The third-neighbour list.
  Third-neighbour atom pairs are pairs of third covalent neighbours in a molecule (*i.e.* those separated by three bonds), that are subject to special (reduced) mutual van der Waals interaction. The electrostatic interaction is unchanged. For a given atom, the `INE14` record specifies the number of atoms with higher `ATNM` sequence numbers that belong to the third-neighbour list of this atom, then lists their `ATNM` sequence numbers in ascending order. For readability, one generally aligns the `INE14` record just below the `INE` record. Are the third-neighbour pairs listed in the file those you would expect considering the structure of your molecule?



(a) `ATNM` (b) `PANM`

Figure 8: The unique atom numbers (`ATNM`) and the atom names (`PANM`), as defined in the `SOLUTEATOM` block. Not to be confused with the atom type (`IAC`), given in Figure 2a.

2.2.3 Covalent Potential Energy Terms

The `BONDSTRETCHTYPE` block defines all possible bond stretching potential energy terms in the molecule. The first entry defines the number of distinct bond potential energy terms, the following values, in groups of three, the actual potential energy terms. Each potential definition consists of the quartic (`CB`) and the harmonic (`CHB`) force constants and the bond length at the energy minimum (`B0`). The successive bond potentials are referred to by a bond-type code (in ascending order starting from 1), which will be used later to assign a given bond-stretching potential to a given bond in the molecule.
To complete the blocks defining the bond stretching potentials, do the following:

- Carefully compare the values from Table 1 with the entries in the `BONDSTRETCHTYPE` block to make sure that the order and the definitions are identical.

- Bring together the bond definitions (`BONDSTRETCHTYPE` block and Figure 4a) and the atom numbers (`SOLUTEATOM` block and Figure 8a) to fill in the `BONDH` (all bonds containing at least one hydrogen atom) and `BOND` (all other bonds) blocks.[14] The first entry denotes the number of bonds in the block, the entries afterwards define the bonds. Every bond definition consists of three integer numbers, the first two referring to the atom numbers defined in the `SOLUTEATOM` block (`ATNM`, *not* `IAC`), the third defining the bond type as implicitly defined by the order in the `BONDSTRETCHTYPE` block.

For a three-atomic molecule containing no explicit hydrogen atoms, the three blocks could look like

```
BONDSTRETCHTYPE
#  NBTY: number of covalent bond types
```

[14]The differentiation has purely historical reasons.

```
    2
#   CB:  quartic force constant
#   CHB: harmonic force constant
#   B0:  bond length at minimum energy
#          CB          CHB          B0
        1.00e7      5.21e5      0.110
        2.32e7      7.42e5      0.155
END
BONDH
#   NBONH: number of bonds involving H atoms in solute
    0
#   IBH, JBH: atom sequence numbers of atoms forming a bond
#   ICBH: bond type code
#    IBH    JBH ICBH
END
BOND
#   NBON: number of bonds NOT involving H atoms in solute
    2
#   IB, JB: atom sequence numbers of atoms forming a bond
#   ICB: bond type code
#     IB      JB  ICB
      1       2     1
      2       3     2
END
```

In analogy to the bond-stretching potential, fill out the bond-angle bending potential blocks (BONDANGLEBENDTYPE, BONDANGLEH, BONDANGLE), the improper dihedral distorsion blocks (IMPDIHEDRALTYPE, IMPDIHEDRALH, IMPDIHEDRAL) and the proper dihedral torsion blocks (TORSDIHEDRALTYPE, DIHEDRALH, DIHEDRAL). Note that for each kind of potential, it is again distinguished whether the potential includes a hydrogen atom or not. In contrast to the bond-stretching terms, which involve two atoms, the bond-angle terms involve three atoms and the (proper and improper) dihedral terms involve four atoms. The order of the bond-angle terms (first atom - central atom - last atom) and of the proper dihedral terms (first atom - central atom 1 - central atom 2 - last atom) follows the bonds between the atoms. The bond-angle and proper dihedral terms are identical under inversion of the definition order. For the (planar or tetrahedral) improper dihedral terms, the central atom is defined first, followed by the three outer atoms in arbitrary order.

The next (already filled) blocks in the file are the CROSSDIHEDRALH and CROSSDIHEDRAL, defining cross dihedral potential energy terms not used in our molecule. This completes the covalent interaction part of the topology.

2.2.4 Lennard-Jones Parameters, Temperature and Pressure Groups, Exceptions, Solvent

The LJPARAMETERS block spans the (triangular) two-dimensional matrix defining the Lennard-Jones interaction between all defined atom types, both for the normal as well as for the third-neighbour interactions. In case of nonbonded interactions, the simulation program will resort to this matrix to determine the correct potential parameters[15].

The SOLUTEMOLECULES block is used to subdivide the solute into separate molecules or fragments, if necessary. Here, we have only one molecule defined. The last atom of each molecule or fragment (or of the only molecule) must be given.

The next blocks, TEMPERATUREGROUPS and PRESSUREGROUPS allow to treat certain atoms differently in terms of the coupling to a thermostat or a barostat - you will hear about these techniques later in the lecture. For now, we will simply treat all atoms equivalently, which means defining only one temperature and pressure group and give the last atom of the molecule as last atom of the group, just as with the SOLUTEMOLECULES block above.

The LJEXCEPTIONS block is another special block allowing to tweak certain pairwise Lennard-Jones interactions specifically. We do not need this for our molecule.

The last two blocks of the topology do not concern the solute anymore, but the solvent it is (or might be) solvated in. The use of solvent can still be turned off in the input file, even though it needs to be present here. Here we chose to have water as a solvent, represented by the SPC (simple point charge) model [2], a very simple but surprisingly accurate water model represented by three point charges. The SOLVENTATOM block defines the properties of the solvent atoms, starting by a unique identifier (I), followed by a name (ANMS), which in analogy to the solute is only used for the identification during analysis. Then, the Lennard-Jones parametrization is given *via* the IACS code, as well as the mass (MASS, in u) and the (point) charge (CGS, given in e). No exclusion or exception list is provided, as all intramolecular non-bonded interactions in solvents are excluded by definition.

The last block, SOLVENTCONSTR, gives the solvent constraints, defining the geometry of the solvent molecules.[16]

2.2.5 Final Check

To help finding mistakes in hand-written topologies, GROMOS++ offers a tool called check_top. After you saved your topology written in the Sections 2.2.1 to 2.2.4, navigate to your ex1 directory, and invoke check_top by calling

```
check_top @topo topo/EGM.top @coord crd/EGM.g96 @pbc r gbond
```

topo/EGM.top is the topology file you just wrote, crd/EGM.g96 a single EGM molecule coordinate file provided, and @pbc r gbond tells the program about the boundary conditions and the gathering. Please refer to the doxygen for further information on the options.

In a first phase, check_top performs a number of simple consistency checks on your topology and writes out errors in case it finds something dubious. Read the output carefully and try to find possible mistakes in your topology. A correct topology passes the test without errors or warnings!

[15]Generally, first and second covalent neighbours (connected by one or two bonds) are excluded and will have no van der Waals interaction at all. Second covalent neighbours (connected by three bonds), *i.e.* third-neighbours, will have reduced van der Waals interactions. Only beyond is the interaction "normal" and specified by the IACs of the two interacting atoms.

[16]Solvents in GROMOS do not have potential energy terms for bonds, bond angles and dihedrals, but are kept completely rigid. You can imagine a water molecule as a triangle with three bonds, each of them always kept at the same distance. This is computationally much more efficient (as constraints are more efficient than bond potentials, and there are a lot more solvent molecules than solute molecules in a typical system), and the approximation done by this treatment is justifiable, since we are in general interested in an exact treatment of the solute under the effect of a solvent, not in the exact treatment of the solvent. If a flexible solvent is needed, it must be defined as a set of molecules formally belonging to the solute part of the topology.

In a second phase, `check_top` calculates the potential energies of all bonded potentials defined in the topology using the coordinates provided with the `@coord` flag. An unusually high energy suggests that the coordinates are far from the reference position given by the potential assigned and indicates a probable error in the topology, given the coordinates are correct. Use this second indication to further check your topology. As a guideline, the covalent potential energies corresponding to the given coordinates should be around $0.46\,\mathrm{kJ\,mol^{-1}}$.

2.3 Initial Validation

Up to now, we have created a model for EGM inspired by analogy with existing parameters for EAE and PPL, and managed to encode this information into a GROMOS topology file. This is expected to be a reasonable way of proceeding if the EAE-like and PPL-like fragments of EGM do not present a dramatic influence on each other in terms of electron density distribution.[17] Even if we believe the transferability assumption to work well in the present case, we cannot trust our model without a minimal amount of *validation* against experimental data.

The following three subsections describe in turn: (*i*) the principle of the validation; (*ii*) the setup of the required simulations; (*iii*) the execution of these simulations on the beaver cluster.

2.3.1 Principle of the Validation

For the purpose of initial validation, we are going to simulate EGM under standard conditions, *i.e.* at 298.15 K and 1 bar, considering three distinct situations:

- GAS: In the gas phase (ideal gas limit[18])
- LIQ: In the pure-liquid phase
- WAT: In water (infinite dilution limit)

For the GAS, we consider a set of N EGM molecules placed at very large distances from each other (initial positions at intermolecular distances of at least 500 nm) under periodic boundary conditions in a box large enough to mimic an ideal (non-interacting) gas. For the LIQ, we consider a set of N (the same number, for simplicity) EGM molecules within a cubic box simulated under periodic boundary conditions and kept at a size yielding the appropriate pressure for the chosen (standard) conditions. For the WAT, we consider a single EGM molecule surrounded by N_{wat} water molecules within a cubic box simulated under periodic boundary conditions again kept at an appropriate size.

The LIQ simulation will give access to the pure-liquid density (ρ) using the equation

$$\langle \rho \rangle = \frac{MN}{\langle V \rangle}\,, \tag{1}$$

[17]A lot of organic chemistry is about substituent effects, *i.e.* cases where this assumption breaks down! For example, the properties of a substituted benzene ring are seldom the "sum" of those of a free benzene and a separate substituent, due to resonance effects.

[18]Thermodynamically, an ideal gas can exist at any temperature and pressure, unlike a liquid or a solid, which are real phases bound to a specific part of the phase diagram. In practice, we will mimic an ideal gas by placing molecules so far from each others that there is no interaction, keeping the (very large) box volume unchanged during the simulation. The pressure during the simulation will not be properly defined, but is not relevant for the value of the (intramolecular) gas phase energy that we are interested in.

where M is the molecular weight of EGM, N the number of molecules in the computational box, V the volume of the computational box, and $\langle\cdot\rangle$ denotes an average over the simulated trajectory.

The LIQ and GAS simulations will give access to the enthalpy of vaporization (ΔH_{vap}) of the liquid using the equation

$$\Delta H_{\text{vap}} = \frac{\langle \mathcal{U} \rangle_{\text{gas}}}{N} - \frac{\langle \mathcal{U} \rangle_{\text{liq}}}{N} + RT\,, \tag{2}$$

where $\mathcal{U}$ is the total potential energy, R the ideal gas constant, T the absolute temperature, and $\langle\cdot\rangle_{\text{gas}}$ and $\langle\cdot\rangle_{\text{liq}}$ denote averages of the GAS and LIQ trajectories, respectively.

A third quantity of great interest for validation is the hydration free energy ΔG_{wat} of the molecule. Calculating this quantity is, however, a bit more complicated (as you will learn in Exercise 5), and in the WAT simulation, we are merely going to simulate the compound in water to examine its conformational properties.

The experimental data for ρ and ΔH_{vap} is reported in Table 2, along with that for the liquids EAE and PPL. There, you can see in particular that the 53A6$_{\text{OXY}}$ force field does a good job at reproducing experimental data for EAE and PPL. The question is: does our newly constructed model for EGM do comparably well?

comp	$\rho^{(e)}(298.15\,\text{K})$	$\rho^{(s)}(298.15\,\text{K})$	$\Delta H^{(e)}_{\text{vap}}(298.15\,\text{K})$	$\Delta H^{(s)}_{\text{vap}}(298.15\,\text{K})$	T_b	$\Delta H^{(e)}_{\text{vap}}(T_b)$
	$/[\text{kg}\,\text{m}^{-3}]$	$/[\text{kg}\,\text{m}^{-3}]$	$/[\text{kJ}\,\text{mol}^{-1}]$	$/[\text{kJ}\,\text{mol}^{-1}]$	$/[\text{K}]$	$/[\text{kJ}\,\text{mol}^{-1}]$
PPL	800 [4]	781.0±0.15 [1]	47.5 [4]	49.5±0.02 [1]	370 [3]	41.44 [3]
EAE	895 [4]	881.8±0.27 [1]	35.6 [4]	36.0±0.02 [1]	350 [3]	31.94 [3]
EMG	1108 [3]		63.9 [5]		461 [3]	55.1 [5]*

Table 2: Experimental data ($^{(e)}$ superscript) and simulation results using the 53A6$_{\text{OXY}}$ force field ($^{(s)}$ superscript) for the compounds considered. The boiling temperature T_b and the enthalpy of vaporization at the boiling point $\Delta H^{(e)}_{\text{vap}}(T_b)$ are given as an additional information. They are not needed for our simulations but listed in view of Question 5 in Section 4.3.
*No experimental value at boiling point could be found. The value reported here is measured at 378 K.

2.3.2 System Size, Combined Topologies

The topology we have created contains a single solute molecule. As explained in Section 2.3.1, we need to simulate a larger number N of solute molecules for the GAS and LIQ calculations. For the WAT calculation, we need only one solute molecule, but surrounded by N_{wat} water (solvent) molecules. To chose the number of molecules, we go for a simple rule: The system should be big enough to be simulated under periodic boundary conditions with a cutoff of 1.4 nm[19] without ever encountering self-interactions with a molecules periodic copy, but otherwise as small as possible to keep simulations short. Values which have proved to work well for similar systems are $N = 512$ for the GAS and LIQ simulations and $N_{\text{wat}} = 1024$ for the WAT simulations.

[19] 1.4 nm is a value used very commonly in recent atomistic simulations. Its justifications comes from water simulations - at 1.4 nm, the coulombic interactions between two water molecules drop below 1% of their magnitude in the first molecule shell. For general systems possibly having much lower electrostatic shielding, this value is somewhat arbitrary, but it can be seen as a part of the model's parametrization.

To be able to use 512 solute molecules in a GROMOS simulation, we need to combine 512 single topologies in a single one, creating one solvent consisting of 512 molecules.[20] No worries, you do not have to redo the work you did before 512 times - for this, there is a GROMOS++ program:

```
com_top @topo 512:topo/EGM.top @param 1 @solv 1 > topo/EGM_512.top
```

`EGM.top` is the single molecule topology you just wrote, and we told the program to use it 512 times. The `@param 1` and `@solv 1` flags tell to use the parameters and solvent definitions of the first topology.[21] This writes a new topology file called `EGM_512.top`, which now contains the 512 solute molecules mentioned before. Go ahead and have a look at the file, checking the differences to the single topology you just wrote by hand.

2.3.3 Simulation Setup

As stated in Section 1, the generation of input files and starting configuration files are beyond the scope of this exercise. For this reason, they are provided to you ready-for-use. Except for the `topo` directory you have worked in until now, you will see four further directories in the main exercise directory: `crd`, `GAS`, `LIQ` and `WAT`. `crd` contains the starting configurations for the different simulations, go have a look if you are curious. The other directories contain everything else needed to run the calculations mentioned in Section 2.3.1, namely *input files* and *running scripts*. The latter are automatically generated scripts that prepare the GROMOS simulations, run the calculations, clean up the files, and, if needed, start the next job. The input files, as mentioned in the very beginning, contain all switches and parameters specifying the desired simulation run. You will look at them in more details in the next exercises. The most important characteristics of the three simulations are summarised in Table 3.

	GAS	LIQ	WAT
number of independent jobs	3	6	3
simulation time per job	100 ps	500 ps	500 ps
total simulation time	300 ps	3 ns	1.5 ns
simulation temperature	298.15 K	298.15 K	298.15 K
simulation pressure	–	1 bar	1 bar
number of solute molecules	512	512	1
number of solvent molecules	0	0	1024

Table 3: Characteristics of the Simulations

Note that there are several sequentially numbered input files and scripts in each directory. In general, we are used to subdivide longer jobs in independent, sequentially starting subjobs, connected only via the end configuration of the first job serving as an input configuration to the

[20]When looking at the input file (already this week if you are curious, otherwise in the next exercise for sure), you will see that there is a switch to chose the number of solute and solvent molecules. While this is nicely working for the solvent (we will use that to have N_{wat} water molecules in our solvated simulation), this functionality has not been implemented to date for the solute, despite the switch. This is the reason for the workaround using `com_top`.

[21]For this simple example, this does not sound very relevant - but `com_top` does also allow to combine various different topologies to a single one - in which case the user has to decide which parameters shall be valid in the combined topology. Obviously, this implies that the topologies to be combined are compatible with each other - at least one topology needs to contain the parameters relevant to all molecules to be combined.

next. Besides a number of general advantages[22], in our case a number of independent energy trajectories will make the analysis in terms of convergence much more convenient.

2.3.4 Submitting and Checking Your Simulations

Once that everything is ready, start your first run by navigating to the folder `LIQ` and sending the first job to the queue

```
qsub -N liq_1 -cwd -j y -o liq_1.o ./liq_1.run
```

Check that your simulation actually starts to run[23] - if your job is finished after a few second, you most probably had a crash during initialisation. In this case, go to the folder, and have a look if you can understand the reason by looking at the file `liq_1.omd`. Ask an assistant for help if necessary. If everything seems fine, repeat the above step correspondingly for the GAS and the WAT simulations in the respective folders.

The next part of doing simulations should be waiting while the computer is doing its part of the labour. Unfortunately, this is not strictly true in practise - it can feel much more like baby-sitting, especially when working with little known systems or new methods or programs. It is important to check from time to time whether the simulations are still running and the system is behaving in the desired way. As you will only learn to do analysis on the systems in the coming week, you do not have to worry about it this time - your assistants will have a backup solution for the worst case.[24] If your runs are still up a few minutes after you started them, you are done for this week!

3 Week 2 - Analysis

Last week, we created a model for ethylene glycol monoacetate (EGM) and set up a number of simulations to assess the quality of the model. This assessment is the task this week.

3.1 Status

Navigate to your simulation directory. You should find the following additional files compared to last week:

- compressed coordinate trajectories `*.trc.gz`
- compressed energy trajectories `*.tre.gz`
- GROMOS output files `*.omd`
- queue output files `*.o`

Start by checking that all simulations finished successfully. To do this, open one of the GROMOS output file. You can check the first part, everything before the lines

[22]Circumvention of queue time, memory and file size limitations as well as easier restarting and less loss in the event of crashes, just to name a few.

[23]Use the command `queue` to check the current status of the queue - see the Document “CSBMS Computational Setup” for further information.

[24]In case you anyway want to check, do not forget that from home, you need a running VPN connection to access to `beaver`. If you want to use any program having a graphic user interface, use `ssh -X` when logging in to `beaver`. Check the “CSBMS Computational Setup” document for further information.

```
==================================================
 MAIN MD LOOP
==================================================
```

to see how the simulation was set up. Then move to the very end of the file. You will find a summary of the simulation and, hopefully, the line

```
MD++ finished successfully
```

Check that all simulations finished successfully.[25] If you notice any problem with your simulations, try to find out what the problem could be by checking the messages in the output file. Then talk to an assistant to check how to solve the problem.

3.2 Visualisation

In a first step, we would like to visualise our simulations, thereby getting a first feeling of what is happening. Let us start with the gas-phase simulations by moving to the `GAS` directory. The coordinates of the system during the simulation run are buried within the coordinate trajectory files. In order to visualise them in an external program like `vmd`, we must extract them. The `GROMOS++` program for this task is called `frameout`. Create a file called `frameout.arg` within the `GAS` folder with the following content:

```
@topo          ../topo/EGM_512.top
@pbc           r gbond
@spec          ALL
@outformat     pdb
@include       ALL
@time          0  2
@single
@traj          gas_2.trc.gz gas_3.trc.gz
```

Please check the `GROMOS++` doxygen (navigate to `available`, then find `frameout`) for further documentation on the options chosen above. You can then call `frameout` as[26]

```
frameout @f frameout.arg
```

Then, open `vmd` on your local machine and load the file just created under `File`→`New Molecule`. It should be named `FRAME_00001.pdb`. In the gas phase, you will not see a lot initially, as the box is extremely large compared to the molecules. We should therefore choose one molecule to focus on - you can do that *via* `Graphics`→`Representations`. Under `Selected Atoms`, replace `all` by `resid 25`[27] and press `Enter` on your keyboard. Then, click on the "Display" window, and press `=` on your keyboard. Your view should now be centred on the chosen molecule. You can now play around with the different options in the "Graphical Representations" window to change the

[25] *Useful trick:* To rapidly check a larger number of files, try a command like
`tail -n5 *.omd`
or
`for f in *.omd; do echo $f; grep 'MD++ finished successfully' $f; done`

[26] Note that you could also type the arguments directly, without the use of an argument file, just as we did using `check_top`.

[27] As you might have guessed, any number between 1 and 512 (for the GAS and LIQ simulations) after `resid` will give you access to a specific molecule.

appearance of your molecule, and view different viewing angles and zooms in the “Display” window. To look at the other frames of the simulation, the camera needs to follow the chosen molecule. This is done using `Extensions→Analysis→RMSD Trajectory Tool`. In the top left corner of the new window, replace `protein` by `resid 25`, then click `ALIGN`. Go back to the “Display” window, hit `=` once more, then go to the “VMD Main” window and click the small “Play” icon in the lower right corner of the window. Feel free to play around a bit with VMD, then repeat the steps just done for the LIQ and WAT simulations.
You now have the tools to answer Question 4.2.1

3.3 Energy Trajectory Analysis

Now it is time to look at other properties of the system along the course of the simulation. A large number of observables gets calculated during the simulation run and saved in the energy trajectories. The GROMOS++ program `ene_ana` extracts values from the trajectory, saves them in time series and calculates the average, standard deviation and error[28] over the simulation run. The properties that `ene_ana` is aware of are defined in a library. You can create your own (given a certain knowledge of the structure of the energy trajectories) or modify the one standardly distributed with GROMOS. For our purposes, however, the standard library is more than enough. Create a file called `ene_ana.arg` within the folder `LIQ` with the following content:

```
@topo         ../topo/EGM_512.top
@library      /usr/local/CSBMS/lib/ene_ana.md++.lib
@prop         densit
              totpot
@time         0  2
@en_files     liq_2.tre.gz liq_3.tre.gz liq_4.tre.gz
              liq_5.tre.gz liq_6.tre.gz
```

`densit` and `totpot` are two of these properties defined in the library. Open the one we are using here, and have a look at the properties defined. You will need at least one other property to answer the questions at the end of this exercise, and many more in the exercises to come! You can then call `ene_ana` as

```
ene_ana @f ene_ana.arg > ene_liq.out
```

As mentioned, `ene_ana` returns the average values of the chosen properties, with standard deviation and errors (check the file `ene_liq.out`). Further on, it also writes a timeseries for each property separately, see for example `totpot.dat`. A very simple way to have a quick look at this timeseries is by using `xmgrace totpot.dat`. `xmgrace` has a lot of options to customise your plot, just explore a bit! Note that `ene_ana` always chooses the same file names for the timeseries, make sure that you do not overwrite important data. Repeat the steps just done for the GAS simulations using an `ene_ana.arg` file in the `GAS` folder containing

```
@topo         ../topo/EGM_512.top
@library      /usr/local/gromos-1.3.2/share/gromos++/ene_ana.md++.lib
@prop         totpot
```

[28]The errors are calculated using block averages of varying sizes. Thereby averages taken over different intervals of the total simulation time are compared to get an estimate of the error. More details can be found in Ref [6].

```
@time         0  2
@en_files     gas_2.tre.gz gas_3.tre.gz
```

You now have the tools to answer Questions 4.2.2 and 4.2.3

4 Report

4.1 Format

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 also (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, at the end of each exercise series, you will find a "report" section with two subsections (here, these are the next two subsections): **simulation results** and **thinking questions**. The first one provides a hint on how you could structure the discussion of the results of the specific exercise in your report. The second one asks "outlook" questions related to the theme of the exercise (and the corresponding lecture material), which should also be answered in your report.

Try to keep you report **short** and **precise**. If possible, keep its length to a maximum[29] of **two A4 pages text** (*i.e.* excluding the space taken by possible graphs or tables). The **deadline** to hand in your report is the end of the week following second week of the exercise (see front page of this document for the exact date). Please hand in your report **to the responsible assistant** either *via* e-mail (**one single printable PDF document!**) or on paper. You can find the contact details of the assistants on the course web page.

4.2 Simulation Results

4.2.1 Visualisation

(a) Give a qualitative description of the conformational behaviour of one EGM molecule in the gas phase, in the liquid, and in water. What are the most striking differences? Can you explain the reason for these differences?

[29]But we are not going to hang anyone if it is three (or even four) pages, so no need to decrease the character font to 6 points so as to squeeze all on two pages!

(b) Repeat this analysis considering a few different single molecules in the liquid and the gas phase. Are the conformational behaviours similar for all these molecules, *i.e.* are your observations noted in (a) representative?

(c) Direct visualisation is a very crude method to monitor the conformational behaviour of a molecule (but it is not a bad start to get a "feeling"!). Can you suggest observables we could calculate along the trajectories (*i.e.* quantities that can be given a value for each trajectory frame), so as to characterise the conformational behaviour on a quantitative basis?

4.2.2 Convergence

(a) The `ene_ana` input file given for the LIQ phase (see Section 3.3) omits the first trajectory file. Repeat the analysis including also the first trajectory file and plot the timeseries of the total potential energy and the density. What do you observe? What do you conclude concerning the starting configuration that was provided to you? When is it sensible to discard an initial piece of simulation when doing the analysis?

(b) Again omitting the first trajectory, redo the analysis using 1, 2, 3, 4 and all 5 trajectories. From the output of `ene_ana`, read out the average and the error estimate. Plot the density and the heat of vaporization[30] (including error bars) as a function of the number of trajectories considered in the calculation. Was the total simulation length chosen long enough?

(c) The WAT simulation alone is not sufficient to calculate the hydration free energy ΔG_{wat} of EGM (you will see this in Exercise 5). But we can use it to estimate the hydration enthalpy ΔH_{wat} based on the solute-solvent non-bonded energy. Write the corresponding equation and use `ene_ana` to estimate the value based on your simulations.

(d) The approach in (c) is not rigorously correct. Why? How could we make it right?

4.2.3 Quality of the Model / Parameter Transferability

(a) Report the values of ρ and ΔH_{vap} you calculated for EGM (including an error bar estimate) and compare these with the experimental data in Table 2. Is the agreement as good as for EAE and PPL?

(b) From this comparison, is the transfer of EAE and PPL parameters to construct a model for EGM appropriate? If not, what could be the reason for the breakdown of the transferability assumption?

(c) If you wanted to refine your initial EGM model to improve agreement with experiment by slightly adjusting parameters, what kind of parameters would have the largest influence on ρ and on ΔH_{vap}?

4.3 Thinking Questions

Answer the following questions:

[30]You can keep the calculation for the GAS potential energy unchanged.

1. Do the atomic partial charges of your EGM molecule (Figure 3a) follow more or less what you expect based on chemical intuition, *e.g.* considering the electronegativities of the different atoms?

2. Use the `LJPARAMETERS` block of your topology file to calculate the pairwise C_6 and C_{12} Lennard-Jones interaction coefficients for the (normal) interactions between the CH2-CH2, CH2-CH3 and CH3-CH3 atom-type pairs. Convert these values[31] to well depths (ϵ) and distances at the minimum (R_{min}) of the Lennard-Jones curve. Comment on the differences between the three pairs of values.

3. In the gas-phase simulations, we started with molecules being separated by at least 500 nm, simulated for 300 ps at 298.15 K, used a cutoff of 1.4 nm, and hoped that the molecules would never collide (or even just interact). Do a back-of-the-envelope calculation showing that this is a reasonable assumption. For this, you need to estimate the average translational velocity of a gas-phase EGM molecule at the selected temperature.

4. Have a look at the formula to calculate the enthalpy of vaporization given in Equation 2 (Section 2.3.1). Why did we use the average total potential energy $\langle \mathcal{U} \rangle$ and not the average total energy $\langle \mathcal{H} \rangle = \langle \mathcal{K} + \mathcal{U} \rangle$? Why did we have to add RT?

5. When you look up for experimental data on the enthalpy of vaporization of liquids, you typically find two types of values, both corresponding to a standard pressure of 1 bar: a value at the boiling point T_b of the liquid and a value the standard temperature of 298.15 K. What are the two different types of experimental measurements leading to these two values? In force-field parametrization, we preferably use the standard value to be compared with a simulation at 298.15 K (as we did for EGM). Why is it so?

A Available Files

In the main exercise directory `ex1/`, you will find `ex1.pdf`, the digital version of the document you are reading, and five additional directories, namely

- `topo/`
  Contains `EGM.top`, a sketch for the topology file to be created during this exercise.

- `crd/`
  Contains the starting coordinates for the systems you will simulate to validation the topology (`EGM_gas.g96`, `EGM_liq.g96` and `EGM_h2o.g96`) as well as the coordinates of a single EGM molecule (`EGM.g96`) used in a first consistency check in Section 2.2.5.

- `GAS/`
  Three input files (`*.imd`) and three script files (`*.run`) ready to be used to run the gas phase simulation, once the topology is created.

- `LIQ/`
  Six input files (`*.imd`) and six script files (`*.run`) ready to be used to run the liquid phase simulation, once the topology is created.

[31] The equations are in the lecture notes

- `WAT/`
  Three input files (`*.imd`) and three script files (`*.run`) ready to be used to run the solvated simulation, once the topology is created.

References

[1] Horta, B.A.C.; Fuchs, P.F.J.; van Gunsteren, W.F.; Hunenberger, P.H.; *J. Chem. Theory Comput.* **2011**, 7, 10161031.

[2] Berendsen, H.J.C.; Postma, J.P.M.; van Gunsteren, W.F.; Hermans, J.; In *Intermolecular Forces*, edited by B. Pullman; Reidel, Dordrecht, 1981, p. 331.

[3] http://www.hbcpnetbase.com

[4] Riddick, J.A.; Bunger, W.B.; Sakano,T.K.; *Organic solvents,physical properties and methods of purification*; John Wiley & Sons, New York, 1986.

[5] Acree Jr, W. & Chickos, J.S.; *J. Phys. Chem. Ref. Data*, **2010**, 39, 043101.

[6] Allen, M.P.; and Tildesley, D.J.; *Computer Simulation of Liquids*; Clarendon Press, Oxford, 1987.

CSBMS: Exercise 2

MD Simulation of a β-peptide

Document version: 27.08.2019

Exercises week 1:	08.10. or 10.10.
Exercises week 2:	15.10. or 17.10.
Deadline for the report:	27.10.
Contact:	`thomas.stadelmann@phys.chem.ethz.ch`
	Thomas Stadelmann - HCI G238 / HCI E314

Summary

In this second exercise, we will perform molecular dynamics (MD) simulations of a β-hexapeptide in methanol. The peptide considered is known experimentally to adopt either a hairpin (folded) or more disordered (unfolded) conformation in this solvent, depending on the environmental conditions. We will thus consider different conditions of ionic strength and temperature in the simulations, in order to investigate the influence of these parameters on the folding-unfolding equilibrium of the peptide. In terms of setup, you will learn how to: (*i*) construct a molecular topology file by assembling building blocks already available in GROMOS; (*ii*) generate an initial configuration including solvent and ions based on a file providing coordinates for the peptide alone; (*iii*) modify/complement an existing input file for carrying out an energy minimization (EM) or a MD simulation. In terms of analysis, you will use the following properties to investigate the folding-unfolding equilibrium: (*i*) visualization of the trajectory; (*ii*) atomic positional root-mean-square deviation (RMSD) from a canonical hairpin structure; (*iii*) end-to-end distance of the peptide; (*iv*) solute-solute and solute-solvent electrostatic interaction energies.

1 Introduction

A β-peptide is an oligomer of so-called β-amino acids, *i.e.* amino acids which, compared to the α-amino acids constituting natural peptides and proteins, possess an additional carbon atom between the amino and carboxylic acid groups[1]. For these peptides, the two aliphatic carbon atoms C_α and C_β of the chain may or may not be functionalized and, when functionalized, may be of R or S chirality[2].

In this exercise, we consider the β-hexapeptide shown in Fig 1. The residue sequence is given in the caption, and the peptide is shown in a form that corresponds to a pH where the two amino groups (one terminus and one sidechain) are protonated and the carboxylic acid group (other terminus) deprotonated, resulting in a net peptide charge of $+1e$. The conformation displayed corresponds to a hairpin, inferred from NMR experiments as being a dominant folded conformer

[1] These peptides are of great interest as peptidomimetic drug candidates, because they "look" like natural peptides but cannot be degraded by proteases. They have been studied extensively, both experimentally and theoretically, see in particular the work of the Seebach and van Gunsteren groups at ETHZ [1, 2, 3]

[2] The sequence of atoms is H_2N-C_β-C_α-COOH and the chirality is specified as (R/S,R/S) where the first specifier refers to C_β and the second to C_α.

in methanol under “usual” conditions[3]. The hairpin[4] consists of two anti-parallel strands, so that the N- and C-termini, which bear net charges of opposite signs, are in close proximity. The strands are also connected by a number of hydrogen-bonds between backbone amide NH and carbonyl CO groups.

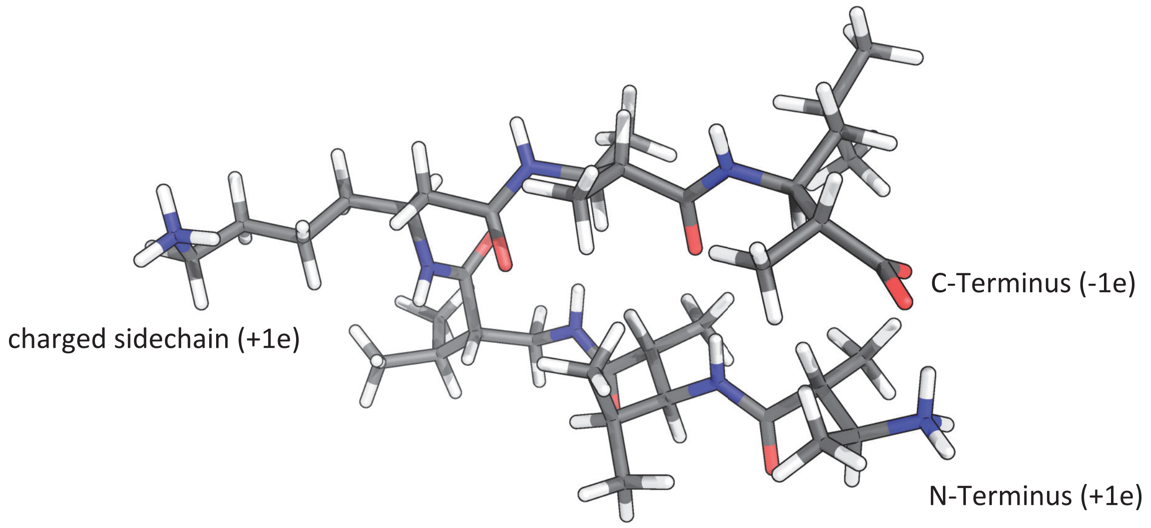


Figure 1: The β-hexapeptide studied in this exercise. The peptide is shown with two protonated amino groups and a deprotonated carboxylic acid group, and in a model hairpin conformation. The sequence of the peptide is H_2^+ -(S,R)-β^3-HAla(αMe) –(S,R)-β^3-HVal(αMe) –(−,R)-β^2-HVal –(S,−)-β^3-HLys –(S,R)-β^3-HAla(αMe) –(S,R)-β^3-HLeu(αMe) -OH. For more information, please refer to Ref. [4] (note that this study considered a different peptide protonation state, with the carboxylic acid protonated).

We are going to perform molecular dynamics (MD) simulations of the peptide (in the specified protonation state) in methanol considering a set of different environmental conditions of ionic strength and temperature, summarized in Table 1. Because it would be silly to ask everyone to run the 9 combinations, each participant will be asked to run only one set of conditions. Now is a good time to ask your assistant to distribute the various conditions among participants, if she/he did not do it already! [5]

[3] One should be very careful about such statements. It would be tempting to say simply that the represented structure is “the structure of the peptide in methanol”. The truth is that: (*i*) it is a single-structure model for an ensemble of conformations; (*ii*) even under “usual” conditions this “folded” ensemble still only accounts for a fraction of the conformers at equilibrium, the one that is detected by NMR precisely because it is structured; (*iii*) the model structure is inferred from NMR on the basis of NOE-derived average proton-proton distances, which always underdetermine an ensemble, *i.e.* must be complemented by some modeling; (*iv*) what is referred to as “usual” conditions must be specified very precisely before any further statement (*e.g.* concerning the population of this “folded” ensemble) makes sense (pH, counter-ion content, residual water, pressure, temperature, ...). Considering all these complicating factors, you should already see why MD studies (characterizing the detailed content of conformational ensembles under different environmental conditions) can help a lot in the correct interpretation of experimental data.

[4] More precisely called a β-hairpin, where the β refers to the specific structural motif, and has nothing to do with the β in “β-peptide”.

[5] Note that “none” corresponds to a non-neutral system and “380K” is above the boiling point of methanol

ions	300K	340K	380K
none	A	B	C
1 Cl^-	D	E	F
$11Cl^- + 10Na^+$	G	H	I

Table 1: Set of 9 environmental conditions (labels A-I) to be considered in the simulations. The simulations are to be performed for the β-peptide of Fig. 1 in methanol and at 1 bar, with the selected peptide protonation state (which implicitly determines the pH and an overall peptide charge of $+1e$), at the indicated temperature and with the indicated counter-ion content (which implicitly determines the ionic strength). Each participant will be assigned one of the 9 cases A-I by the assistant. Please do not continue this exercise before you are assigned one case.

As stated in Exercise 1 (see Introduction therein), any GROMOS simulation using the program `md` (or `md_mpi`) requires at least three files to be provided, namely

- an *input file* (flag `@input`)
- a *topology file* (flag `@topo`)
- a *starting configuration file* (flag `@conf`)

In Exercise 1, you made the *topology file* the hard way, namely from scratch. In this exercise, your life will be easier because the current GROMOS force field [6, 7] (version 54A7) includes parameters (in the form of building blocks) for all residues involved in your β-peptide, for the Cl^- and Na^+ ions, and for the methanol solvent. So, you'll just have to learn the way to assemble these elements into a topology for your specific peptide. And since the topology making is far easier, we can now invest a bit more time on the two other files...

For the *starting configuration file*, we will provide you with a GROMOS configuration file that contains atomic coordinates for the peptide alone in the model hairpin conformation of Fig. 1. You will then learn how to: (*i*) solvate the peptide with methanol so as to fill a rectangular computational box that will be simulated under periodic boundary conditions; (*ii*) add the appropriate number of counter-ions into your box; (*iii*) perform the initial energy minimization (EM) of the system, required to avoid having a catastrophically high potential energy (clashes and gaps between atoms) when you start your MD simulation.

And for the *input file*, we will provide you with a partially filled template, leaving it up to you to complement the most relevant blocks (and understand what they mean). Because the EM and subsequent MD steps use largely similar input files, you will actually create the EM input file first, and later adjust it into the MD input file.

This will be it for Week 1. Then you will run one system of Table 1 (again, ask the assistant which one). And in Week 2, we will gather the trajectories generated by all participants in a common directory, so that you can analyse *all* runs of Table 1. For the *analysis* of the simulations, we will focus on observables that give us a hint on the course and energetics of the folding-unfolding equilibrium (RMSD from the model hairpin structure, end-to-end distance, solute-solute and solute-solvent electrostatic interaction energy).

(338 K). We consider these "unphysical" conditions to underline trends: in MD, everything is possible - not only real situations!

2 Week 1 - Setting Up and Starting the Simulations

The work for Week 1 is described in the five following subsections and consists of

- Creating the topology file (Section 2.1)
- Preparing the initial coordinate file (Section 2.2)
- Performing EM to relax the system (Section 2.3)
- Setting up the MD simulations (Section 2.4)
- Setting up the simulation jobs (Section 2.5)
- Starting the simulation jobs (Section 2.6)

The files provided to you for these operations can be found in the directory `/usr/local/CSBMS/ex2`. Make a `ex2` subdirectory in your `home` directory for all files of this exercise and copy the `lib`, `md`, `min` and `topo` directories from `/usr/local/CSBMS/ex2` to your home.

```
mkdir ~/ex2

cp -r /usr/local/CSBMS/ex2/lib  ~/ex2/

cp -r /usr/local/CSBMS/ex2/md   ~/ex2/

cp -r /usr/local/CSBMS/ex2/min  ~/ex2/

cp -r /usr/local/CSBMS/ex2/topo ~/ex2/
```

For simplicity, this document will only describe how to generate the situation labelled E in Table 1, *i.e.* a simulation at 340 K with a single Cl^- counter-ion. To keep track of the system considered in a particular set of files, the filenames should *always contain a string* like "`hexapepE`", where "`E`" refers to the *situation considered*. If the assistant asked you to run the simulation for another situation of Table 1, do first the variant for system E and then extrapolate a bit to make the setup for the other system. In particular *always use the corresponding string in filenames*, *e.g.* "`hexapepA`" for situation A and so on.

2.1 Creating the Topology

The GROMOS community develops not only a simulation package (MD engine plus setup and analysis tools), but also a force field. The current version of the force field is called[6] 54A7. The force-field information for 54A7 is contained in two kinds of files[7].

- A set of *molecular topology building block* (`mtb`) *files*, which you can find under `/usr/local/CSBMS/ex2/topo/54a7.mtb` (main file) and `/usr/local/CSBMS/ex2/topo/54a7_*.mtb` (specialized files).

[6] 54 is the number of atom types, A stands for a force field for condensed-phase simulations (*e.g.* pure liquids or hydrated molecules) and 7 is the version number

[7] These can be directly found in the standard gromos distribution under `/usr/local/gromos/forcefields/official`, but we copied the relevant ones to `/usr/local/CSBMS/ex2` for simplicity.

- An *interaction function parameter* (`ifp`) *file*, which you can find under `/usr/local/CSBMS/ex2/topo/54a7.ifp`.

The `mtb` *files* contain records for a series of *building blocks*. These can be either entire molecules or monomers (residues) to be linked together within a polymer. For each building block, the atoms and their properties (name, mass, charge, IAC, exclusions, third neighbors) are listed, followed by a list of covalent terms (bonds, angles, impropers, dihedrals). For residues, there is also information on how to do the linkage of a residue with the preceding and following residues. Special building blocks are also available for the "capping" of polymers, *i.e.* to terminate a chain before the first residue or after the last residue. Finally, building blocks are also available for various solvents. For your β-peptide, you will need the main `mtb` file `54a7.mtb`, which contains the solvent methanol and the Cl^- and Na^+ ions, and the specialized `mtb` file `54a7_beta.mtb`, which contains the β-amino acid residues and associated capping groups. Most parameters are not directly provided within the `mtb` file, but referred to by a code (IAC, bond-type, angle-type, ...). The `ifp` *file* in turn binds these codes to specific values of the force-field parameters. GROMOS provides an automatic tool called `make_top` that can assemble residue building blocks into a chain according to a specified *residue sequence* (which must also include the two capping groups).

So our first task is to define the sequence corresponding to the β-hexapeptide of Fig. 1 (with the specified protonation states). For that, you would have to compare the sequence given in the caption of this figure and find the corresponding building-block names, *e.g.* by looking in the GROMOS manual Volume 7 [5] or in the `54a7_beta.mtb` file. For simplicity, we did this search for you and the sequence of your peptide in the GROMOS nomenclature is:
`NH3+ SRAM SRVM SAV SBKH SRAM SRLM COO-`
We selected the charged capping groups `NH3`$^+$ and `COO`$^-$ and the protonated form `SBKH` of β^3-HLys to match the required protonation states[8]. Possible counter-ions must also be specified in the topology, and for this, you can simply add them after the peptide sequence[9], here as `CL-` or `NA+` building blocks. For our reference example of system E in Table 1, we need to add one single `CL-` at the rear of the peptide sequence.

At this point, it is not a bad idea to open `54a7_beta.mtb` in your favorite editor, and take a glimpse at what the records look like for the 8 building blocks above. Although we are not going to explain this in more details, you will see that a lot of the data looks very similar to what you wrote by hand in your topology file for Exercise 1. You can also look at the solvent molecule methanol (building block name `CH3OH`) and the ions (`CL-` and `NA+`), which are located in `54a7.mtb`. Now: let's build!

Go to the subdirectory `topo` in your `ex2` directory.

```
cd ~/ex2/topo
```

Create a file `make_top_hexapepE.arg` with the arguments required for running `make_top`. These arguments are:

[8] Here, we chose the protonation states for you. In general, you will have to pay special attention to this by yourself, selecting deprotonated or protonated forms depending on pKa's and pH. Watch out: this decision can have a very large impact on the simulation results! Whenever you have a bit of time for additional exploration: can you find in the `mtb` file the uncharged variants of the two capping groups and of the β^3-HLys residue?

[9] Another is to set up the topologies of the peptide and the ions separately and use the `com_top` program in order to combine these topologies [5].

- `@build` followed by the `mtb` files to be used (we need two of them, see above)
- `@param` followed by the `ifp` file to be used (one file, see above)
- `@seq` followed by the sequence of building blocks (we need eight of them plus one ion for situation E, see above)
- `@solv` followed by the name of the solvent building block (see above)

Now, you can run `make_top` to generate `hexapepE_54a7.top`

```
make_top @f make_top_hexapepE.arg > hexapepE_54a7.top
```

The generated file `hexapepE_54a7.top` contains the complete molecular topology of the peptide plus one Cl^- counter-ion, along with methanol as a solvent. The first thing to do is definitely to *look at it*. You should recognize the exact same structure (block content) you had in Exercise 1 for your organic molecule[10]. The second reflex you should now have is to check it using `check_top`. If you don't remember how to set up the argument (`.arg`) file for this program, refer to the script of Exercise 1 or, like a real GROMOS pro, try to find the arguments needed using the doxygen documentation.[11] For the `@coord` argument of `check_top`, you can use the coordinate file `/usr/local/CSBMS/ex2/coord/hexapep_vac.cnf`, `/usr/local/CSBMS/ex2/coord/hexapep_Cl_vac.cnf` or `/usr/local/CSBMS/ex2/coord/hexapep_11Cl_10Na_vac.cnf` depending on your system according to Table 1.[12]

At this point, you may also want to generate the topology file for the specific variant of Table 1 that has been assigned to you (with appropriate filenaming containing the corresponding letter A-I). In addition, create the topology file for system A since we need this file in later sections (when using the `sim_box` and `ion` modules).

2.2 Preparing the Initial Coordinates

In general, preparing an initial (solute) coordinate file for a given biomolecule is not a trivial task, because, in the best case[13], you may have to convert from non-GROMOS formats (*e.g.* PDB) and possibly reorder, add or delete atoms[14]. All these issues will be discussed in Exercise 3.

For the present exercise, we decided to be nice and to provide you with a file in GROMOS format listing the coordinates of all atoms for the model hairpin structure of your peptide as displayed in Fig. 1, already in the correct order[15]. This file can be found in `/usr/local/CSBMS/ex2/coord/hexapep_vac.cnf`. All you have to do is to add the solvent and the counter-ions.

[10] The main differences are that : (*i*) the lists of atoms and force-field terms (bonds, angles, ...) are much longer, because your peptide is much larger; (*ii*) the lists of types (atom types, bond types, angle types, ...) and the matrix of Lennard-Jones parameters are also much longer, because they correspond to the entire 54A7 force field, not just a subset of those you need for your specific molecules; (*iii*) the solvent is now methanol and not water.

[11] see in the "CSBMS Computer Setup" document if you don't know how to access it

[12] The pH is implicitly determined by the protonation state and the ionic strength is implicitly determined by the counter-ion content.

[13] The worst case is when a piece (*e.g.* loop) is missing in your experimental coordinate file or, worse, if there is no experimental structure. Then the only option is to try to model the structure.

[14] Particularly hydrogen atoms, which may be present in the source structure but absent in the simulation (united atoms) or, in the opposite, absent in the source structure (X-ray structure) but present in the simulation.

[15] The structure is taken from Ref [2] where it has been derived based on 20-NOE-derived proton-proton distances in combination with MD simulations.

We are going to simulate the peptide in methanol under periodic boundary conditions based on a rectangular computational box. This means that we will only consider atomic coordinates for the particles within this rectangular box, but implicitly assume that the box is surrounded by an infinite number of periodic copies of itself. Initial coordinates for the methanol molecules surrounding the peptide can be generated using the GROMOS++ program `sim_box`. The size of the box (three edge lengths) is typically defined by imposing a minimum distance between any solute atom and the closest box wall, with the solute at the box center (and possibly optimally rotated). We will use a value of 1.5 nm for this minimal distance[16].

The program `sim_box` will use a reference file containing the coordinates of methanol molecules in the pure liquid[17].

This file is provided[18] in `/usr/local/CSBMS/ex2/coord/ch3oh.cnf`, and `sim_box` will pave our box with methanol using these coordinates. Obviously some methanol molecules will nastily overlap with the peptide and must be deleted. Any solvent molecule with an atom closer than some minimal distance from any atom of the peptide will be removed by `sim_box`. We will use a value of 0.23 nm for this minimal distance.

Go to the subdirectory `box` in your directory `ex2`.

```
cd ~/ex2/box
```

```
sim_box @f sim_box_hexapep.arg > hexapepE_met.cnf
```

Create a file `sim_box_hexapepE.arg` with the arguments required for `sim_box`. These arguments are:

- `@topo` followed by the molecular topology file to be used[19]
- `@pbc` followed by the desired box shape (see above; r: rectangular, t: truncated octahedron, c: triclinic)
- `@pos` followed by the coordinate file of the solute in vacuum (see above)
- `@solvent` followed by the coordinate file of the reference pure-solvent box (see above)
- `@minwall` followed by the minimum solute-to-wall distance (see above)
- `@thresh` followed by the minimum solute-to-solvent distance (see above)
- `@rotate` to optimally rotate the solute prior to filling the box which does not need any additional arguments (see above; rotation will lead to the minimal possible box size)

Now you can solvate the peptide using

[16] This distance must be larger than the electrostatic long-range cutoff of 1.4 nm, but the choice of 1.5 nm is a bit minimalistic! We do this because we want the simulations to run fast. For serious work, we should consider using a larger distance.

[17] Well equilibrated by someone else in previous simulations at 300 K and 1 bar, and provided in the GROMOS distribution.

[18] These types of files can be directly found in the standard gromos distribution under `gromos/forcefields/official`, but we copied the relevant one to `/usr/local/CSBMS/ex2` for simplicity.

[19] Remember that at present, we have no ions in the coordinate file. So, you need a topology without ions. You can generate it as described in the previous section. To save time, the resulting file is also provided in `/usr/local/CSBMS/ex2/topo/hexapepA.top`.

```
sim_box @f sim_box_hexapep.arg > hexapepE_met.cnf
```

Counter-ion atomic coordinates can then be generated using the GROMOS++ program `ion`, which replaces a given number of solvent molecules by ions. The program proceeds by successive replacements, substituting every time the molecule at the location of highest or lowest electrostatic Coulomb potential (depending on the ion charge - we use 0.8 kJ/mol) by an ion. Ions can only be generated beyond a given distance from any solute atom. We will use a value of 0.35 nm for this minimal distance.

Go to the subdirectory `ion` in your `ex2` directory.

```
cd ~/ex2/ion
```

Create a file `ion_hexapepE.arg` with the arguments required for `ion`. These arguments are:

- `@topo` followed by the molecular topology file to be used (without any counter-ions)
- `@pbc` followed by the desired box shape
- `@negative` followed by the number and the name of the negative ion(s) to be added
- `@positive` followed by the number and the name of the positive ion(s) to be added
- `@potential` followed by the value for the electrostatic potential (see above)
- `@mindist` followed by the minimum distance to the counter-ions (see above)
- `@pos` followed by the coordinate file of the peptide in methanol (the one we just generated with `sim_box`)

You can now add the ions by typing

```
ion @f ion_hexapepE.arg > hexapepE_box.cnf
```

Done. You have prepared your computational box! It is now a good idea to open the file `hexapepE_box.cnf` to see how it looks like. What are the dimensions of the box? Is it rather isotropic (close to a cube) or anisotropic (elongated)? How many methanol molecules have been generated? Have the ions been inserted correctly? You may also want to visualize the box using `vmd` which you already know from Exercise 1, it should look like shown in Fig. 2.

At this point, you may also want to generate the coordinate file for the specific variant of Table 1 that has been assigned to you (with appropriate filenaming containing the corresponding letter A-I).

2.3 Relaxing the System

Let's face it: throwing the peptide into a box of solvent using the `sim_box` program and swapping solvent molecules for ions using the `ion` program is a rather barbarian procedure to generate an initial configuration. Not unexpectedly, the system will not be very happy in this patched-together configuration, *i.e.* the potential energy is likely to be quite high. There will be high-energy atom-atom contacts as well as unfavorable vacuum gaps, prominently at the solute-solvent interface. In

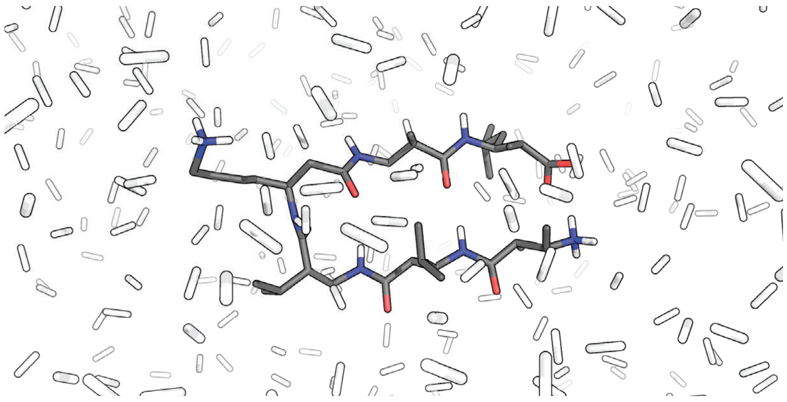


Figure 2: Neighborhood of the β-hexapeptide in a hairpin configuration solvated by methanol (white sticks) within the computational box. The ions are not visible in this picture.

order to relax this initial configuration, the system should undergo energy minimization (EM), which can be done using the program[20] `md` (or `md_mpi`).

Go to the subdirectory `min` in your `ex2` directory.

```
cd ~/ex2/min
```

Copy the file `hexapep_em_template.imd` to the name `hexapepE_em.imd`. This file already contains the appropriate block structure for running an EM. Some of the blocks are only partially filled with the string `#TO_DO` replacing specific entries. You guessed right: what is filled is what we don't want to discuss in details today (often, these are values that you generally don't change), and the `#TO_DO` you must replace by appropriate values.

In the `TITLE` block

```
TITLE
    Exercise 2: My first GROMOS MD simulation
    Compound: hexapeptide in methanol and counter-ions
    Author: Sir Isaac Newton
    Date: October 6, 2017
END
```

you provide as usual basic info on origin/nature of the system/job.

If an `ENERGYMIN` block

```
ENERGYMIN
#     NTEM    NCYC    DELE    DX0     DXM   NMIN   FLIM
      #TO_DO  1       #TO_DO  0.01    0.05  2000   0.0
END
```

is present, we tell the MD++ program to perform an EM run (default is a MD run). The `NTEM` switch selects the EM algorithm, where 1 is steepest descent (SD) and 2 is conjugate gradient (CG). The integer number `NCYC`, used in CG only, sets the interval (in steps) at which we reset the search

[20]Yes, in GROMOS, the same program is used to run EM and MD.

direction to the gradient.[21] The real number DELE is the termination criterion, *i.e.* we will stop the EM when the potential energy difference over a step becomes smaller than this. [22] The real number DX0 sets the initial step size. The real number DXM sets the maximal step size. The real number FLIM can be used to limit the absolute value of the forces to a maximum value before the algorithm is applied. With NMIN, we set the minimum number of iteration steps for our algorithm. We will use SD until the potential energy difference over a step becomes smaller than 0.1 kJ mol$^-1$.
Reasonable values for the other parameters are already filled in.

The SYSTEM block

```
SYSTEM
#       NPM       NSM
        1         #TO_DO
END
```

specifies the number of solute (NPM) and solvent (NSM) molecules in the system. You only have one solute[23]. The number of solvent molecules has been determined by sim_box (solvation) and ion (replacement by ions). Have a look in the file ../ion/hexapepE_box.cnf (and keep in mind that it might not be the same for the other systems of Table 1).

The INITIALISE block

```
INITIALISE
#     NTIVEL    NTISHK    NTINHT     NTINHB     NTISHI   NTIRTC      NTICOM
       NTISTI    IG        TEMPI
      #TO_DO    0         0          0          1        0           0
                 0         210185    #TO_DO
END
```

specifies how a run should be initialized. It will be explained later in Section 2.4. The values NTIVEL and TEMPI are irrelevant in EM. So, just set them to 0 and 1.0 for now.

The STEP block

```
STEP
#   NSTLIM     T          DT
    #TO_DO     #TO_DO     #TO_DO
END
```

specifies the maximum number of steps (NSTLIM) for the energy minimisation[24]. Let's set this maximum to 5000 steps. The values T and DT are irrelevant in EM and will be explained later in Section 2.4 for MD. So, just set them to 0.0 and 1.0 for now.

[21] This is because CG may get "trapped" in a subspace of the conformational space, failing to achieve full minimization.

[22] This is because CG may get "trapped" in a subspace of the conformational space, failing to achieve full minimization.

[23] By "solute", it is meant whatever is not solvent in the topology file. If you have counter-ions along with the peptide in the solute topology file, they are considered to belong to the solute and you still have only one solute "molecule". In fact, NPM > 1 is no longer supported by MD++ (it was a feature of the 1987 and 1996 codes), *i.e.* will result in an error message. To simulate *e.g.* a box containing two peptide molecules, you have to use com_top to "multiply" the topology of the peptide (so that there is still one "solute" consisting of two peptide units).

[24] The EM should ideally stop when convergence is reached as specified by DELE. But if convergence is hard to reach, it will stop ion any case after NSTLIM steps. It is a good idea to check the EM output file for the kind of termination, since in the latter case, you may not have reached the minimum!

The BOUNDCOND block

```
BOUNDCOND
#       NTB      NDFMIN
        #TO_DO   3
END
```

specifies the spatial boundary conditions. The NTB switch selects the boundary conditions employed, where 0 is vacuum, -1 is periodic based on a truncated octahedron box, 1 is periodic based on a rectangular box, and 2 is periodic based on a triclinic box. The value NDFMIN is irrelevant in EM and will be explained later in Section 2.4 for MD.

At this point of the file, you will see that three blocks are commented out: MULTIBATH, PRESSURESCALE and COMTRANSROT. Leave them untouched, they will be uncommented and used for MD in Section 2.4 (they are not used for EM).

The PRINTOUT block

```
PRINTOUT
#      NTPR           NTPP
       #TO_DO         0
END
```

specifies how often (every NTPR$^{\text{th}}$ step) you are printing out the energies to the output file. Let's print them every 10th step. NTPP controls dihedral angle transition monitoring which we are not interested in.

MD++ produces a massive amount of data and it is impossible to store all the data it produces. The WRITETRAJ block meets this demand:

The WRITETRAJ block

```
WRITETRAJ
#      NTWX      NTWSE       NTWV       NTWF     NTWE       NTWG       NTWB
       #TO_DO    0           0          0        #TO_DO     0          0
END
```

specifies how often information about the trajectory frames is written to file. It will be explained later in Section 2.4. We don't want to generate trajectories in EM, so just set NTWX and NTWE to 0 for now.

The CONSTRAINT block

```
CONSTRAINT
#       NTC         NTCP    NTCP0(1)      NTCS        NTCS0(1)
        #TO_DO      #TO_DO  0.0001        #TO_DO      0.0001
END
```

regulates the application of bond-length constraints. Here, as is usual practice in biomolecular simulation (for reasons explained in the lecture), we will constrain all bond lengths and impose the full rigidity of the solvent molecules (NTC=3). In this example, the constraints are imposed by the SHAKE algorithm for both solute (NTCP=1) and solvent (NTCS=1) with a relative geometric tolerance of 0.0001.

The FORCE block

```
FORCE
# NTF array
# bonds              angles    imp.     dihe     elec.  vdW
  #TO_DO               1        1         1        1      1
# NEGR    NRE(1)    NRE(2)    ...      NRE(NEGR)
  #TO_DO  #TO_DO    #TO_DO    ...      #?TO_DO
END
```

specifies which of the various types of force-field terms are included into the potential energy, and serves to define energy groups. For bond angles, improper dihedrals, torsional dihedrals, non-bonded electrostatic and non-bonded van der Waals, the standard terms of the GROMOS force field are switched on (1). We let you guess what is to be done with the bond-stretching terms, given that we apply constraints on all bond lengths (see above). In the second line of this block, so-called energy groups are defined. This definition will not affect the simulated trajectory, but what you get in the energy trajectory file. There, a partitioning of the energy into intra-group (N values for N groups) and inter-group ($N(N-1)/2$ values for N groups) contributions will be reported, which can be very useful for analysis purposes.[25] In general, we define one or more energy groups for the solute, and another one comprising all the solvent molecules. The first integer NEGR is the number of energy groups we want to use. The following numbers NRE(1)...NRE(NEGR) are the atom sequence numbers of the last atom of each energy group. Here, define your energy groups so as to have hexapeptide, all counter-ions (all Na+ and all Cl-), and all solvent molecules separately as a group. To find the last atom of each energy group have a look in the file ../ion/hexapepE_box.cnf (and keep in mind that the number of groups and their terminal atoms might not be the same for the other systems of Table 1).

The COVALENTFORM block

```
COVALENTFORM
#   NTBBH    NTBAH    NTBDN
        0        0        0
END
```

specifies which functional form we will use for bond-stretching (NTBBH), bond-angle bending (NTBAH) and torsional dihedral (NTBDN) interactions. We just use the default options for all functional forms.

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                #TO_DO   #TO_DO   #TO_DO   0.4      #TO_DO
```

[25] Think very carefully about the definition of energy groups before running the simulation. Group-based energy partitioning cannot be recalculated from the trajectories in an efficient way. So, if you realize after the simulation that you did not choose a clever partitioning for the analysis you have to make, you will likely have to rerun the simulation.

```
END
```

specifies which algorithm you will use for the pairlist generation. The cut-off used in the short-range pairlist construction is given by `RCUTP` and for GROMOS it is usually 0.8 nm. The cut-off used in the long-range interactions is given by `RCUTL` and for GROMOS it is usually 1.4 nm. The pairlist is recalculated every `NSNB` steps, and we typically use 5. Finally `TYPE` specifies the type of the cut-off, whether it is based on the distance between charge-groups (0) or on the distance between atoms (1). Here we choose charge-groups based.

The `NONBONDED` block

```
NONBONDED
# NLRELE    APPAK      RCRF      EPSRF   NSLFEXCL
      #TO_DO      0.0       #TO_DO      #TO_DO          1
# NSHAPE   ASHAPE    NA2CLC   TOLA2   EPSLS
      -1        1.4        2   1e-10       0
# NKX     NKY   NKZ    KCUT
   10      10    10     100
# NGX   NGY   NGZ  NASORD  NFDORD   NALIAS  NSPORD
   32    32    32       3       2        3       4
# NQEVAL   FACCUR   NRDGRD   NWRGRD   NLRLJ    SLVDNS
  100000      1.6        0        0       0      33.3
END
```

specifies how to treat the electrostatic interactions. Since you will use the reaction-field method, the value of `NLRELE` should be equal to 1. The long-range electrostatic interactions are truncated beyond a certain cutoff (`RCUTL` in the `PAIRLIST` block). Beyond the reaction-field cut-off radius (`RCRF`) the electrostatic interactions are replaced by a static reaction field with a dielectric permittivity `EPSRF`. `RCRF` and `RCUTL` should be identical. Because we are doing the EM in methanol `EPSRF` is set to the experimental permittivity of this solvent (32.3 [26]). With `NSLFEXCL` equal to 1, you include the contributions of excluded atoms to the electrostatic energy. The ionic strength of the continuum is set to 0 (`APPAK`). All other switches are not used for the reaction-field method.

In order to run the EM, a shell script has been created for you (`hexapepE_em.run`). You only have to adapt the paths and the names of the files according to your setup. Then run the script

```
./hexapepE_em.run
```

The EM should run relatively fast, which is why you are allowed to run it without sending the calculation to the cluster.

When the process is finished, you should find in your current directory a file `hexapepE_min.cnf` with the optimized coordinates and the EM output file `hexapepE_em.omd`. Have a look at the output file to verify that the potential energy was indeed going down and that the EM has completed successfully. You may again want to visualize the box using `vmd`.

At this point, you may also want to generate the energy minimized coordinate file for the specific variant of Table 1 that has been assigned to you (with appropriate filenaming containing the corresponding letter A-I).

[26] Note that in prinicple it would be better to use the permittivity of the methanol model, which would be 27.8 [8].

2.4 Setting Up the Simulations

Now let's start with the MD. Go to the subdirectory `md` in your `ex2`.

```
cd ~/ex2/md
```

In the previous section, we have already made an input file for the EM, so it would be silly to write the MD input file entirely from scratch. Copy the file `../min/hexapepE_em.imd` to your current directory under the name `hexapepE_md.imd`. Now we just have to adjust this file so that it does a MD run instead of an EM run.

The first thing is to delete (or comment out) the entire `ENERGYMIN` block (including the terminal `END`). Also, update the `TITLE` block accordingly.

Then we have to reconsider the `INITIALISE` block

```
INITIALISE
#    NTIVEL   NTISHK   NTINHT    NTINHB    NTISHI  NTIRTC     NTICOM
      NTISTI   IG       TEMPI
     #TO_DO   0        0         0         1       0          0
               0        210185   #TO_DO
END
```

which specifies how a run should be initialized (remember that we had inserted dummy numbers 0 and 1.0 for `NTIVEL` and `TEMPI`, which we have to reconsider). The `NTIVEL` switch determines the source of the atomic velocities, where 0 means they should be read from the starting configuration file and 1 means they should be generated pseudo-randomly from a Maxwell-Boltzmann distribution. The `NTISHK` switch enforces the application of bond-length constraints in the initial configuration. The `NTINHT` and `NTINHB` switches are only relevant for Nose-Hoover thermo- and barostats and can be ignored in our case. The `NTISHI` switch determines the source of the lattice-shift vectors[27], where 0 means they should be read from the starting configuration file and 1 means they should be set to zero. The `NTIRTC` is only relevant if we use roto-translational constraints[28] and can be ignored in our case. The `NTICOM` switch enforces the removal of the center-of-mass component in the initial velocities. The `NTISTI` switch is used to reset the stochastic integrals in stochastic dynamics (SD) simulations. The integer number `IG` is the pseudo-random number generator seed[29] (used only if `NTIVEL` is set[30]). The real number `TEMPI` provides the initial temperature for generating pseudo-random initial velocities (used only if `NTIVEL` is set). The energy minimized configuration obtained in the last section does not contain velocities. So, we have to generate them at random, by setting `NTIVEL` to 1 and `TEMPI` to 340 K (watch out that the value can be 300 K or 380 K for the other systems of Table 1). Let's not change `IG` so we can compare later with the results of the assistants (who also use this specific `IG`).

We also have to look again at the `STEP` block

```
STEP
#   NSTLIM          T         DT
```

[27] The lattice-shift vectors indicate where an atom is located in the infinite periodic system. For each atom, a specific Cartesian component is changed by ± 1 every time the atom crosses the corresponding box wall.

[28] Such constraints can be used to block the overall translation and rotation of the solute in the box.

[29] This seed is included in the input file for reproducibility purposes. Even if the velocities are generated at random, the same seed will generate the same velocity distribution.

[30] Or if we perform SD

```
    #TO_DO          #TO_DO    #TO_DO
END
```

which now specifies the number of steps (NSTLIM) for the simulation (remember that we had inserted 5000 for NSTLIM along with dummy numbers 0.0 and 1.0 for T and DT, which we have to reconsider). The real number DT is the simulation timestep. Given that we have bond-length constraints, a value of 2 fs is appropriate[31]. For each MD job (we will run 50 of them in sequence), we are going to perform 500'000 steps (yes, half a million!). So, what will be the duration of each job in ns? And the total duration of the simulation? (find out and then check[32]!). The real number T specifies the time origin of each job. For now, insert 0.0, as appropriate for the first job.

Remember that in the file, we have three blocks that are commented out: MULTIBATH, PRESSURESCALE and COMTRANSROT. Now, we need them, so just uncomment them (remove the first hash of each line).

The MULTIBATH block

```
MULTIBATH
#   ALGORITHM
    #TO_DO
#   NBATHS
    #TO_DO
#   TEMPO(1 ... NBATHS)  TAU(1 ... NBATHS)
    #TO_DO      0.1
    #TO_DO      0.1
#   DOFSET: number of distiguishable sets of d.o.f.
    2
#   LAST(1 ... DOFSET)  COMBATH(1 ... DOFSET)  IRBATH(1 ... DOFSET)
    #TO_DO         1         1
    #TO_DO        2         2
END
```

specifies if and how we want to apply thermostating. First we specify which algorithm we will use. Here, we will use the weak-coupling scheme (ALGORITHM=0). How many temperature baths we want to couple to the system is specified by NBATHS. We want two baths, one for the hexapeptide and one for the ions and the solvent. For each bath, you must specify the temperature using the TEMPO parameter. They should both be 340 K (watch out that the value can be 300 K or 380 K for the other systems of Table 1). And for each bath, you must insert the coupling time TAU used in the weak-coupling method for this bath, which we take to be 0.1 ps. DOFSET specifies the number of distiguishable sets of degrees of freedom. LAST is pointing to the last atom for the set of degrees of freedom, *e.g.* to the last atom of the hexapeptide and the last solvent atom. To find these last atoms, have a look at the file ../min/hexapepE_min.cnf (and keep in mind that the number of groups and their terminal atoms might not be the same for the other systems of Table 1). COMBATH is the temperature bath to which we want to couple the centre of mass motion 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.

The PRESSURESCALE block

[31] Watch out the units. GROMOS wants times in ps, not fs!
[32] Answer: 1 ns per job, 50 ns in total.

```
PRESSURESCALE
# COUPLE   SCALE       COMP        TAUP  VIRIAL
  #TO_DO   #TO_DO      0.0004575   0.5   #TO_DO
# SEMI
       1        1        1
# PRES0(1...3,1...3)
 0.06102        0        0
       0  0.06102        0
       0        0  0.06102
END
```

specifies if and how we want to calculate the pressure and apply barostating. We tell GROMOS to calculate and scale the pressure by setting COUPLE to 2. As the box should be isotropically scaled we set SCALE equal to 1. The weak-coupling method uses two additional parameters: COMP is the isothermal compressibility and TAUP is the coupling time. We are calculating the molecular virial (VIRIAL is equal to 2), so intramolecular forces do not contribute to the pressure. The next line is only used for semi-anisotropic pressure coupling and can be ignored in our case. Finally, we have to specify the reference pressure[33] (1 bar) in a tensor form.

The COMTRANSROT block

```
COMTRANSROT
# NSCM
  #TO_DO
END
```

is needed to remove the centre of mass motion (translation and rotation). Without this block it can happen that all the kinetic energy is converted to centre of mass translation (the so called ”flying ice cube problem”). With NSCM we specify how often the center-of-mass (COM) motion is removed. If NSCM is < 0 translational and rotational motion are removed every $|\texttt{NSCM}|^{\text{th}}$ step. If NSCM is > 0 only translational motion is removed every $\texttt{NSCM}^{\text{th}}$ step. Set NSCM to 1000. Also recall that the BOUNDCOND block has a parameter NDFMIN, which specifies the number of degrees of freedom subtracted from the total number of degrees of freedom for the calculation of the temperature. If we suppress center-of-mass translation, we should remove 3 degrees of freedom (the value should already be 3 in your file).

Then we have to reconsider the PRINTOUT block

```
PRINTOUT
#      NTPR      NTPP
       #TO_DO    0
END
```

which specifies how often you are printing out the energies to the output file (remember that we had inserted 10 for EM, which we have to reconsider). Change this parameter to 1000 in order to save space on the disk.

Finally, we have to reconsider the WRITETRAJ block

```
WRITETRAJ
#      NTWX      NTWSE       NTWV       NTWF     NTWE       NTWG       NTWB
```

[33] Recall that the GROMOS units of pressure are $\mathrm{kJ\,mol^{-1}\,nm^{-3}}$.

```
       #TO_DO    0           0          0        #TO_DO     0          0
END
```

which specifies how often information about the trajectory frames are written to file (remember that we had inserted 0 and 0 for `NTWX` and `NTWE`, which we have to reconsider). More specifically, we specify the frequency at which we write 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`) to specific files[34]. In the present case, we are only interested in the coordinates (`NTWX`) and energies (`NTWE`), and will write them at every 500th step[35]. So, how much time in ps will separate records be written to file? And how many records will we write in total for the 50 jobs (50 ns trajectory)? (find out and then check[36]!).

At this point we are done. You may notice that the blocks `SYSTEM`, `BOUNDCOND`, `CONSTRAINT`, `FORCE`, `COVALENTFORM`, `PAILIST` and `NONBONDED` were not modified relative to the EM run. The blocks `INITIALISE`, `PRINTOUT` and `WRITETRAJ` were adjusted. And the new blocks `MULTIBATH`, `PRESSURESCALE` and `COMTRANSROT` were added.

At this point, you may also want to generate the MD input file for the specific variant of Table 1 that has been assigned to you (with appropriate filenaming containing the corresponding letter A-I).

2.5 Generating the Simulation Jobs

As already mentioned, we are not going to run one job of 1 ns as set up in the previous section, but 50 successive jobs of 1 ns each[37]. Because the manual set-up of 50 job scripts can be a painful and error-prone procedure, there is a little but powerful helper called `mk_script`. This GROMOS++ program is able to automatically generate a job script from a given model input file and a series of arguments.

At this point, you should only do the specific variant of Table 1 that has been assigned to you - since it is the system you are going to actually run. We'll keep using the letter E below, but you should replace it by your own letter A-I. Still in your subdirectory `md` of your `ex2` directory, create the input file `hexapepE_make_script.arg` required for running `mk_script` by using the following arguments:

- `@sys` followed by a string referencing your simulation (`hexapepE` with E replaced by A-I).
- `@bin` followed by the executable name, which in our case is `/opt/progs/gromos/bin/md_mpi`
- `@dir` followed by the directory path your MD is going to run in, *i.e.* your current directory.
- `@files`

[34]The second switch (`NTWSE`) defines selection criterion for trajectories: if `NTWSE = 0` the normal coordinate trajectory will be written, or if `NTWSE > 0` a minimum energy trajectory will be written.

[35] *Warning:* It makes no sense to write out configurations too often. First, it needs a lot of disk space. Second, the data is highly correlated and so no additional information is gained from it.

[36] Answer: 1 ps between records, 1000 records per job, so 50000 records in total. The energy records are reasonably small, but the coordinate records (including solvent) will take about 1'000'000 kilobytes (so we are shooting for 1 Gb of trajectory in total!).

[37] The advantages of splitting the 50 ns across 50 jobs are that (*i*) you can fit the time-limit of the cluster queue for each job and (*ii*) if a job crashes, you loose only a small piece of data, not the whole. You will submit the first job to the queue and each job will submit the next one when done.

  - `topo` followed by the path to your topology file (`../topo/hexapepE.top` with E replaced by A-I)
  - `input` followed by the path to your input file (`hexapepE_md.imd` with E replaced by A-I)
  - `coord` followed by the path to your initial coordinate file (`../min/hexapepE_min.cnf` with E replaced by A-I)

- `@template` followed by the standard `mk_script` configuration file (`mk_script.lib`)
- `@version` followed by the program of interest (`md++`)
- `@script` followed by two numbers, the first one indicating the starting number of the first job script (1), the second one indicating the number of job scripts to generate (50)

Now run `mk_script` to generate the job scripts

```
mk_script @f hexapepE_make_script.arg
```

This will create 50 `hexapepE_*.run` job scripts numbered from 1 to 50, along with the corresponding 50 input files (`hexapepE_*.imd`). Now there is one last important thing to take care of. Remember that in the last section, we asked you to set `NTIVEL` to 1? This means in the beginning of each job velocities will be created from Maxwell-Boltzmann distribution. This isn't appropriate for a continuation job, where we want to read the velocities from the initial configuration (they are there, since the previous job wrote them out), but not for the first job, where we need to generate them. To fix this, just edit the file `hexapepE_md.imd` and change `NTIVEL` back to 0 and type again

```
mk_script @f hexapepE_make_script.arg
```

You can see that you got one error and that first job hasn't been written out. This is fine since the first job has been written before with `NTIVEL` equal to 1 so it creates its own velocities. Other jobs created previously were overwritten so now they read velocities from the previous job.

2.6 Starting the Simulations

Now, you are ready to start your 50 ns MD simulation (split across 50 jobs that will each submit the next one) on the beaver cluster. Submitting your calculation is similar as in Exercise 1, so if you don't remember, please consult the script of Exercise 1. The simulation of 50 ns should run for about 5 days on 4 cores. Just type (remember that E should be replaced by your letter A-I !)

```
qsub -N hexapepE_1 -cwd -j y -o hexapepE_1.o -pe mpi 4 ./hexapepE_1.run
```

3 Week 2 - Analyzing the Simulations

When getting close to the session of Week 2, your jobs should be finished. Then you should check that the simulation finished successfully as you did in Exercise 1, and leave the trajectories where they are in `ex2/md`. On the afternoon preceding your session of Week 2, the responsible assistant will create a directory `/usr/local/CSBMS/ex2/all_runs`, collect the coordinate and energy trajectories of all participants from their home directories and copy them into `/usr/local/CSBMS /ex2/all_runs/md`. In this way, everyone will have access to the 50 ns trajectories corresponding to the 9 situations A-I of Table 1. For simplicity, the topology files and model hairpin structures in vacuum (including possible ions at large distance) corresponding to systems A-I will also be available under /usr/local/CSBMS/ex2/all_runs/topo and /usr/local/CSBMS/ex2/all_runs/coord, respectively.

For your analysis, it is important that you *do not copy* the trajectories from /usr/local/CSBMS/ex2/all_runs/md to your home directory, otherwise we would run out of space! Just use the files by pointing to them using the appropriate path. Also, pay attention to the ordering of the files. We have reordered the output of the whole simulation with new labels going from 101 to 150 which will make scripting easier (see below).

Since we are mostly interested in the folding-unfolding behavior of the peptide under different conditions, the following analysis seems highly relevant and is described in the next four subsections:

- Visualization of the trajectory (Section 3.1)
- Atomic positional root-mean-square deviation (RMSD) from a canonical hairpin structure (Section 3.2)
- End-to-end distance of the peptide (Section 3.3)
- Solute-solute and solute-solvent electrostatic interaction energies (Section 3.4)

Just a little comment to simplify your life. At one point or another, you will have to repeat the same operation for the 9 systems A-I. When you reach this point, it may not be silly to think about *scripting* your analysis. For example, in BASH

```
#!/bin/bash
dir="/usr/local/CSBMS/ex2/all_runs"
GROMOS="/opt/progs/gromos/bin/some_gromos_analysis"
for sys in A B C D E F G H I; do
  $GROMOS @topo ${dir}/topo/hexapep${sys}_54a7.top \
                       @ref ${dir}/coord/hexapep_${sys}.cnf \
                       @traj $( ls ${dir}/md/${sys}/md${sys}_1
                          [0-4][0-9].trc.gz )
                          > analysis_results_${sys}.out
done
```

will run the same analysis of the full trajectories (excluding job 1) of the 9 systems in one go. Don't forget that to run a script you have to change its access permissions with the chmod command. To give yourself permission to execute, just type

```
chmod u+x your_script_name
```

For this bash script, you need to be aware of the following: The line-continuations '\' will fail if you have whitespace after the backslash and before the newline. However, we need those line continuations as 'some_gromos_analysis' expects the argument string on one line. Make sure you do not have any additional empty spaces in your script as bash is very sensitive for this!

It takes some time to calculate 49 ns for 9 systems, therefore you have to submit your script into a queueing system. In that case you have to specify the full path to the analysis program so *e.g.* instead of writing frameout write /opt/progs/gromos/bin/frameout. To find the full path of your program type which your_program. As a reminder, in order to submit a job, type

```
qsub -N arbitrary_job_name ./your_script_name
```

If you prefer to run everything on the command line instead, you can also use the wildcard '∗' or a range such as $\{a..b\}$, so that your command would look something like this (this, however, does NOT work within an .arg file):

```
some_gromos_analysis @topo all_runs/topo/hexapepE_54a7.top
                      @ref all_runs/coord/hexapep_E.cnf
                      @traj all_runs/md/E/mdE_*.trc.gz
                         > analysis_results_E.out
```

or

```
some_gromos_analysis @topo hexapep/topo/hexapepE_54a7.top
                      @ref hexapep/coord/hexapep_E.cnf
                      @traj hexapep/md/mdE_{110..120}.trc.gz
                         > analysis_results_E.out
```

As was the case in Week 1, the subsections below will use system E as an example. And you will have to adjust things accordingly to process other systems A-I, possibly taking advantage of scripting. In the following, we show the example arguments which you should use in your scripts.

3.1 Visualization

In Exercise 1, you have seen how to visualize a trajectory as a movie using the GROMOS program `frameout` and the graphic program `vmd`. It would take too much time to visualize the 9 trajectories, so maybe focus on systems A, C and E only to get an idea of typical behaviors. Include the ions into your visualized trajectories to also see how they behave. For the visualization, make a new subdirectory `ana` and also a new subdirectory `vis` in your `ex2` directory and work in there.

You can generate PDB snapshots from your trajectory (or any other) coordinate files using the program `frameout`, which you already know from Exercise 1. Go to the directory `ana/vis` and have a look at the example arguments:

```
@topo        /usr/local/CSBMS/ex2/all_runs/topo/hexapepE_54a7.top
@pbc         r cog
@spec       EVERY
@frames     100
@outformat  pdb
@include    ALL
@time       0 0.001
@single
@traj       /usr/local/CSBMS/ex2/all_runs/md/E/mdE_102.trc.gz
```

Because now you have long trajectories you don't want to write out every frame. With `@spec EVERY` and `@frames 100` you specify that every 100th frame will be written out. If you would like to get specific frames (let's say 36 and 408) you should write `@spec SPEC` and `@frames 36 408` instead. `@single` tells the program to put frames in one file. We advise you not to take the whole MD trajectory into account for visualization as this would take too long for this exercise. Instead, take a look at the output for the first ns, then another output in the middle of the whole run, and then another output at the end of the simulation. You will then get a feeling of what has happened during the simulation overall.

You now have the tools to answer Question 4.2.1

3.2 Monitoring the RMSD

Atom positional root-mean-square deviations (RMSD) is a measurement of structural difference between two given conformations. For two conformations of N atoms with the $3N$-dimensional coordinates vectors $\mathbf{r}$ and $\mathbf{r}_{ref}$, the RMSD is given by

$$RMSD(\mathbf{r}, \mathbf{r}^{ref}) = \left(\frac{1}{N} \sum_{i=1}^{N} (\mathbf{r}_i - \mathbf{r}_i^{ref})^2 \right)^{\frac{1}{2}} \tag{1}$$

where $\mathbf{r}_i^{ref}$ is the position vector of the i^{th} atom in the reference conformation and $\mathbf{r}_i$ is the corresponding position in the test conformation, after least-squares translational and rotational superimposition of the two structures.

For the RMSD calculations, make a new subdirectory `rmsd` in your `ex2/ana` directory and work in there.

Create a file `rmsd_hexapepE.arg` with the arguments required for the program `rmsd`. Required arguments for the program `rmsd` are:

```
@topo       /usr/local/CSBMS/ex2/all_runs/topo/hexapepE_54a7.top
@pbc        r cog
@time       0 0.001
@atomsrmsd  1:C,CA,CB,N
@ref        /usr/local/CSBMS/ex2/all_runs/coord/hexapep_E.cnf
@traj       /usr/local/CSBMS/ex2/all_runs/md/E/mdE_102.trc.gz
```

Again the topology is given by `@topo` and `@pbc` defines the periodic boundary condition and gathers the frames. In `@atomsrmsd` one gives the atom specifier of the atoms of which one wants to calculate the RMSD compared to a reference structure `@ref`. Here we want to analyse the backbone of the peptide (C, C_α, C_β, N) as a function of time. With `@traj` the coordinate trajectories are specified. Now calculate the RMSD using the GROMOS++ program[38] `rmsd`

```
% ~/ex2/ana/rmsd>
rmsd @f rmsd_hexapepE.arg > rmsd_hexapepE.out
```

You now have the tools to answer Question 4.2.2

3.3 Monitoring the end-to-end distance

Often you are interested in the time change of a certain geometric property. You can monitor the properties of your system using time series. In addition, you may want to compare a property of your simulation with an experimental value. In this case a time-average is calculated which can be compared to experimental data. Such kind of analysis is carried out with the `tser` GROMOS++ program. `tser` is a very powerful program and only its basic function is explained here.

[38] *Hint:* The atom specifier for the whole protein is 1:a (all atoms of the first molecule).

Make a new subdirectory `tser` in your `ex2/ana` directory and work in there. Create a file `tser_hexapepE.arg` with the arguments required for the program `tser`. The example arguments are:

```
@topo    /usr/local/CSBMS/ex2/all_runs/topo/hexapepE_54a7.top
@pbc     r cog
@time    0 0.001
@prop    d%1:3,61
@traj    /usr/local/CSBMS/ex2/all_runs/md/E/mdE_102.trc.gz
```

First, you have to tell `tser` were the topology (`@topo`) resides and which boundary conditions (`@pbc`) you are using. Second, tell `tser` using `@prop` which properties it should calculate and print out. Here `d` stands for distance and `1:3,61` stands for atom number 3 and 61 in the first molecule. Have a look at the atom type of those two atoms. With the `@traj` argument, tell `tser` where it can find the trajectory coordinates files. In our hairpin system an interesting property is the head to tail distance. Its fluctuations over time give you an indication on the stiffness of the secondary structure.

Now call `tser` and redirect its output

```
tser @f tser_hexapepE.arg > tser_hexapepE.out
```

Now you can open the file in `xmgrace` and plot the time series[39] [40]

```
xmgrace -block tser_hexapepE.out
```

You now have the tools to answer Question 4.2.3

3.4 Monitoring the energy components

From Exercise 1, you should know how the program `ene_ana` works.

Make a new subdirectory `ene_ana` in your `ex2/ana` directory and work in there. Create a file `ene_ana_hexapepE.arg` with the arguments required for the program `ene_ana`. The example arguments are:

```
@topo     /usr/local/CSBMS/ex2/all_runs/topo/hexapepE_54a7.top
@time     0 0.001
@prop     ele_solusolu ele_solusolvE
@library  ../../lib/ene_ana.md++.lib
@en_files /usr/local/CSBMS/ex2/all_runs/md/E/mdE_102.tre.gz
```

With `@en_files` you tell `ene_ana` which energy trajectories should be read in. The `@prop` argument specifies for which properties the time series should be extracted from the energy trajectory. Properties are defined in `ene_ana.md++.lib` which you can copy from `/usr/local/CSBMS/ex2/lib` to your current directory. `ele_solusolu` and `ele_solusolvE` stand for electrostatic interaction within the solute itself and between solute and solvent with ions, respectively. Note that for the second property you have to specify a code of the system. One specifies an `ene_ana` library with the `@library` argument. Now you can run `ene_ana`

[39] *Hint*: The last line of the output file contains the averages of the properties.

[40] *Hint*: Have a look at the doxygen documentation of Property Specifier. There you will find that you can specify many more properties (`@prop`).

```
ene_ana @f ene_ana_hexapepE.arg > ene_ana_hexapepE.out
```

The program calculates the average of the specified properties as well as the root-mean-square deviations (`rmsd`) and a statistical error estimate (`error est.`). The error estimate is calculated from block averages of growing sizes extrapolating to infinite block size. [41]

The program `ene_ana` also produced a time series of calculated properties. Have a look at them using `xmgrace` as above.

You now have the tools to answer Question 4.2.4

4 Report

4.1 Format

Please refer to the corresponding section of Exercise 1 for information on the goal and expected structure of the report.

4.2 Simulation Results

4.2.1 Visualization

(a) Visualize the three trajectories of the systems A, C and E. Describe qualitatively the conformational behavior of the peptide in the three cases. Also describe qualitatively the behaviour of the counter-ions.

(b) In systems C, F and I, we simulated methanol at 380 K, *i.e.* far above its boiling point (338 K) - to extract a trend considering an "unphysical" situation. Verify that methanol actually remains liquid even in this case (you can do this by monitoring the average box volume in the different simulations: partial or complete vaporization would lead to a very large increase).

4.2.2 RMSD

(a) Calculate and display graphically the RMSD time series for the 9 simulations A-I (think about scripting!)

(b) Make a simplified 3×3 table similar to Table 1 where you report as entries a "o" if the peptide stays close to the hairpin structure throughout the simulation, a "+" if the peptide alternates between hairpin and non-hairpin structures, and a "*" if the peptide drifts away from the hairpin structure and never comes back.

(c) Discuss your simplified table. Is there a clear trend? What seem to be the effects of the temperature and ionic strength on the conformatonial behavior?

[41] *Warning:* Sometimes the error estimates are NaN (not a number), which is due to the fact that we do not have enough values to calculate a meaningful error estimate.

4.2.3 End-to-end distance

(a) Calculate and display graphically the time series of the end-to-end distance for the 9 simulations A-I (think about scripting!)

(b) Compare your observations to those based on the RMSD. Do the RMSD and end-to-end distance correlate? If yes, why do you think it is so?

4.2.4 Energy components

(a) Calculate and display graphically the time series of the solute-solute and solute-solvent electrostatic energy for the 9 simulations A-I (think about scripting!)

(b) How do these time series correlate with each other, and with the RMSD and end-to-end distance time series?

(c) Can you now propose a molecular interpretation for the effect of ionic strength on the conformational behavior of the peptide?

(d) Also explain the effect of the temperature.

4.3 Thinking questions

Answer the following questions:

(a) A rigorous way of characterizing the folding-unfolding equilibrium of the peptide would be to monitor the relative populations of folded and unfolded conformers. The population ratio is the equilibrium constant, giving in turn access to the relative free energy. In this exercise, instead, we started from a folded conformation and examined whether it is maintained or disrupted under different environmental conditions on the 50 ns timescale. Why did we take this second approach rather than the first one? What are the shortcomings of the second approach?

(b) Related to the previous question. Imagine you would run 50 ns in situation E, where the peptide unfolds, save the final coordinates, then revert all the atomic (solute, ions and solvent) velocities ($v \rightarrow -v$). We would then give this configuration to the CSBMS students next year, telling them to run 50 ns in the same situation E. What would they observe (assuming infinite numerical precision in the simulations)? What (incorrect!) conclusion would they be tempted to draw based on this simulation?

(c) Calculate the "effective" molar (mol solute per liter solution) concentration of the peptide in methanol for your simulated system. For this, you need to consider the volume of the box containing one peptide molecule. Calculate similarly the ionic strength of the solution when the box contains either the peptide plus one Cl^- or the peptide plus 11 Cl^- plus 10 Na^+. What do you think about the resulting values (in comparison to a typical experimental situation)? Would it be correct to say that the simulations are representative for an experimental setup at the concentration and ionic strength you calculated? If not, what is the main difference between the "effective" concentration in a simulation and the "real" concentration in an experiment?

(d) The autoionization constant of methanol is 16.70 at 298 K [9]. The pK_a-values of propanoic acid and ethylamine in methanol are 9.71 and 11.0, respectively, at 298 K [10]. Using the two latter compounds as models for the C- and N-termini, respectively, of your peptide, what would be the expected ionization state of the peptide at neutral pH in methanol? And considering the ionization states you used in the simulations, what is the pH range your simulations were actually probing?

(e) What type of force-field interaction (bonds, angles, dihedrals, impropers, Lennard-Jones or electrostatics) dominates the folding-unfolding equilibrium of the peptide? What crucial approximation in the treatment of this interaction could lead to a very large bias in the simulated results?

(f) In systems C, F and I, we simulated methanol at 380 K, *i.e.* far above its boiling point (338 K), and observed that methanol actually remains liquid even in this case. Can you suggest reasons why methanol evaporation does not occur in these simulations?

(g) When generating the methanol box, we imposed a minimal distance of 1.5 nm between any peptide atom and the nearest box wall. Do you think this minimal distance, enforced in the initial configuration, is maintained throughout the simulations? If not, why?

(h) To relax the system, we have applied EM to the box containing the peptide, the counter-ions and the solvent. Then, we directly started the MD. Do you think such a set up protocol is sufficiently careful? It clearly overlooks a relatively slow equilibration process, do you see which one?

A Available Files

In the main exercise directory `/usr/local/CSBMS/ex2/`, you will find `ex2.pdf`, the digital version of the document you are reading, and five additional directories, namely

- `topo/`
  Contains `54a7_beta.mtb` and `54a7.mtb` which are molecular topology building block files and the `54a7.ifp` interaction function parameter file.
- `coord/`
  Contains the starting coordinates for the hexapeptide (`hexapep_vac.cnf`) and the coordinates of the solvent (`ch3oh.cnf`). The last two coordinates of the peptide containing ions (`hexapep_11Cl_10Na_vac.cnf` and `hexapep_Cl_vac.cnf`) are supposed to be used for `check_top`.
- `min/`
  Contains the input file (`hexapep_em_template.imd`) and `hexapepE_em.run` to be modified and to run the energy minimization of the compound after solvation.
- `md/`
  Contains the library file for `mk_script` (`mk_script.lib`).
- `all_runs`
  Contains the finished exercise. You will get read-permission in the second week of Exercise 2.

- `lib`
  Contains the library `ene_ana.md++.lib` for the program `ene_ana`.

References

[1] Bonvin, A. M. J. J. van Gunsteren, W. F.; *J. Mol. Biol.* **2000**, 296, 255–268.

[2] Daura, X., Gademann, K., Schaefer, H., Jaun, B., Seebach, D. and van Gunsteren, W. F. *J. Am. Chem. Soc.*, **2001**, 123, 2393–2404.

[3] Santiveri, C. M., Jimnez, M. A., Rico, M., van Gunsteren, W. F., Daura, X.; *J. Peptide Sci.*, **2004**, 10, 546–565.

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

[5] GROMOS tutorial Vol. 7, www.gromos.net

[6] Poger, D., van Gunsteren, W. F., Mark, A. E.; *J. Comput. Chem.*, **2010**, 31, 1117–1125.

[7] Schmid, N., Eichenberger, A. P., Choutko, A., Riniker, S., Winger, M., Mark, A. E., van Gunsteren, W. F.; *Eur. Biophys. J.*, **2011**, 40, 843–856.

[8] Riniker, S., Kunz, A.-P. E., van Gunsteren, W. F.; *J. Chem. Theroy Comput.*, **2011**, 7, 1469 – 1475.

[9] Parsons, G. H., Rochester, C. H. *J. Chem. Soc. Faraday Trans. 1*, **1972**, 68, 523 – 532.

[10] Rived, F., Roses, M., Bosch, E.; *Anal. Chim. Acta*, **1998**, 374, 309 – 324.

CSBMS: Exercise 3

Protein Simulations & Properties

Document version: 27.08.2019

Exercises week 1:	22.10. or 24.10.
Exercises week 2:	29.10. or 31.11.
Deadline for the report:	10.11.
Contact:	`sadra.gheta@phys.chem.ethz.ch`
	Sadra Gheta - HCI G243

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 β-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.* α-helix or β-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.

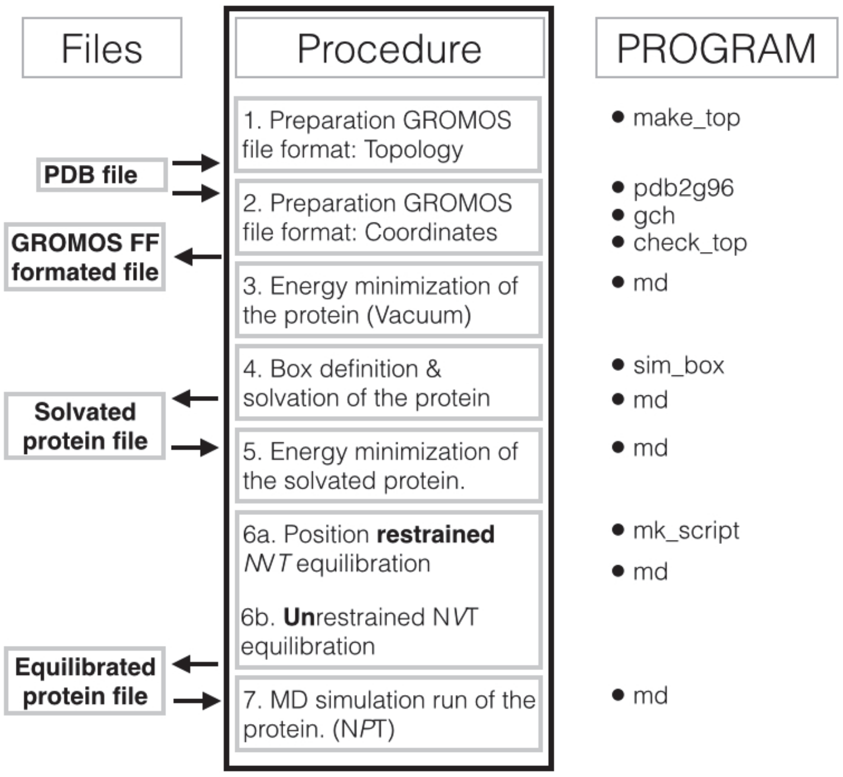

Files
Procedure
PROGRAM
1. Preparation GROMOS file format: Topology
make_top
PDB file
2. Preparation GROMOS file format: Coordinates
pdb2g96
gch
check_top
GROMOS FF formated file
3. Energy minimization of the protein (Vacuum)
md
4. Box definition & solvation of the protein
sim_box
md
Solvated protein file
5. Energy minimization of the solvated protein.
md
6a. Position restrained NVT equilibration
mk_script
md
6b. Unrestrained NVT equilibration
Equilibrated protein file
7. MD simulation run of the protein. (NPT)
md

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 in your local machine 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), β-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/CSBMS/ex3/dir_for_students/* ./
```

There should now be **10 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.
- the **ana_298K** dir: will be used for the analysis of the MD simulations at 298 K.
- the **ana_348K** dir: will be used for the analysis of the MD simulations at 348 K.

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 CSBMS 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 for the experimental pH (hint: besides replacing the NH (charged?) and CO (charged?) by something meaningful,

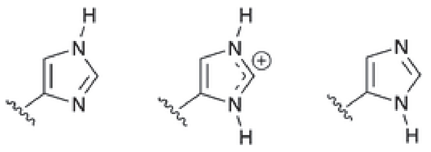


Figure 2: Histidine protonation states labelled HISA (H on ND1), HISH (protonated) and HISB (H on NE2).

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 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
list of 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 of solute atoms to be positionally restrained
END
REFPOSITION
# first 24 chars ignored
1 THR H1 1 5.144434523 5.236928112 4.193546264
......
......
5181 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)

Reminder: you can find the doxygen documentation here:
For GROMOSXX: `http://realbeaver/gromos/md++/`
For GROMOS++: `http://realbeaver/gromos/gromos++/`

(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

```
cd ../md
```

2.4.7 Molecular Dynamic Sampling Simulation

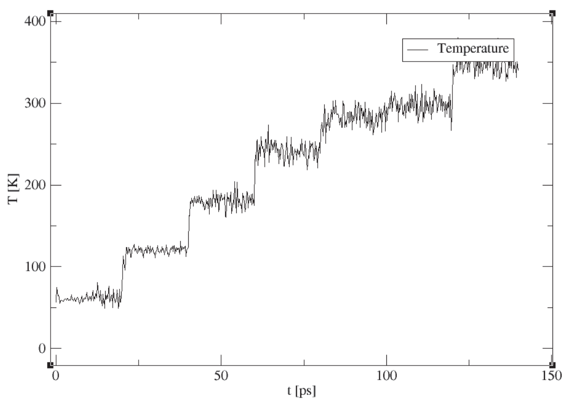
GROMOS++/MD++ PROGRAM NEEDED: `mk_script md`

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.



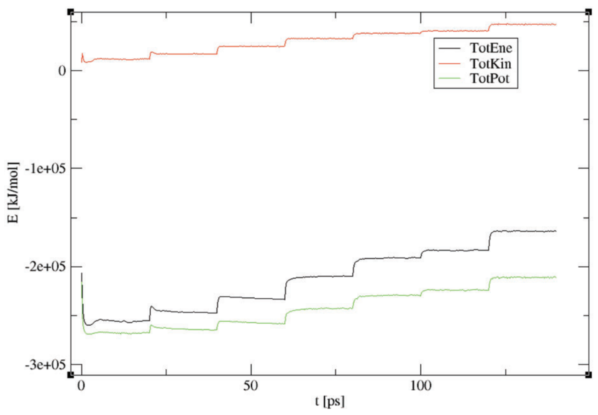


Figure 3: **Left** The temperature curve (black) is shown for the protein during the thermalisation of 140 ps. Each 7 steps represent the increase of the temperature till 348 K and the coupling of the temperature during this step. **Right** The overall energy (black line), Potential energy (green) and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

⇒ 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 `@joblist`, 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/CSBMS/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/CSBMS/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 β-sheet/bridge, α-helix, π-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 α-helix and β-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 ψ and ϕ against each other. Backbone dihedral angles ψ is defined through the backbone C-N- Cα-C atoms and ϕ is defined through the backbone N-Cα-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 ψ and ϕ angles from the coordinate trajectories (as usual, have a quick look at `tser_ramach.sh` before!).

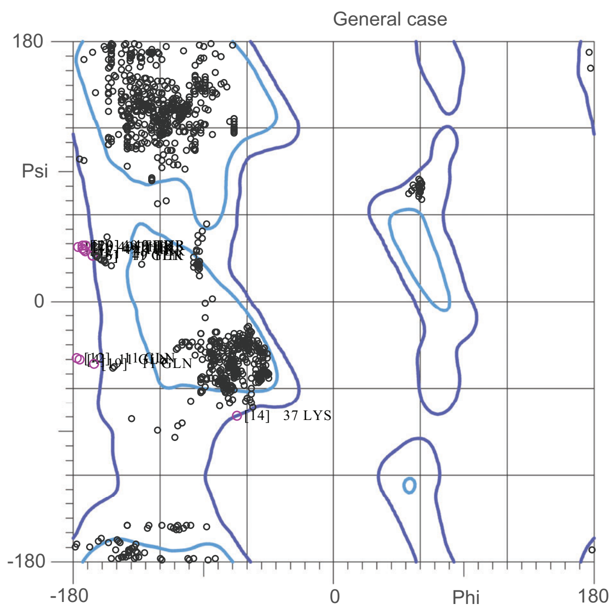


Figure 4: Ramachandran plot of the model 1 NMR structure of the protein.

```
./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 ψ as the first column and ϕ 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 /usr/local/CSBMS/ex3/movies/protein_TEMK_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.

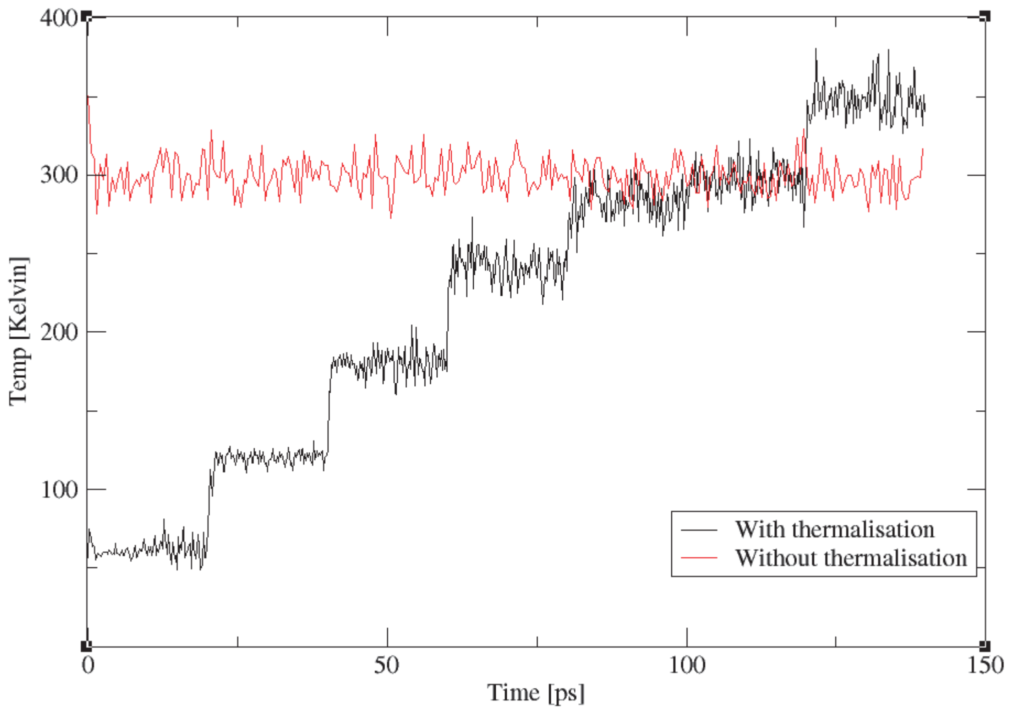


Figure 5: Equilibration step (150 ps) of the protein; protein with thermalisation (black), protein without thermalisation (red).

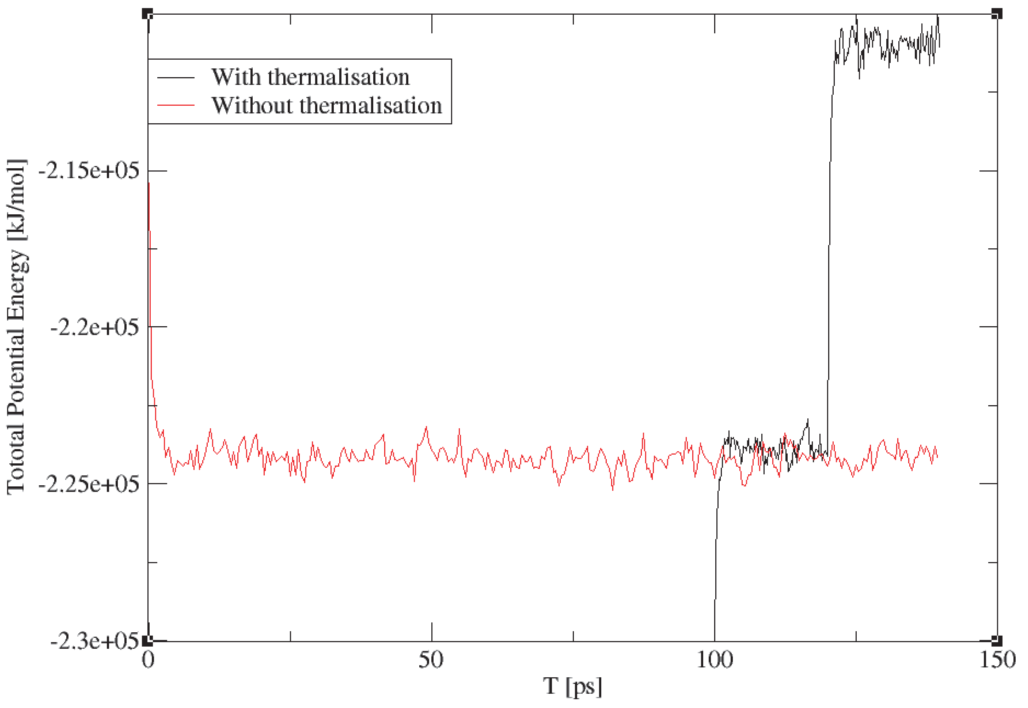


Figure 6: Equilibration step (150 ps) of the protein; protein with thermalisation (black), protein without thermalisation (red).

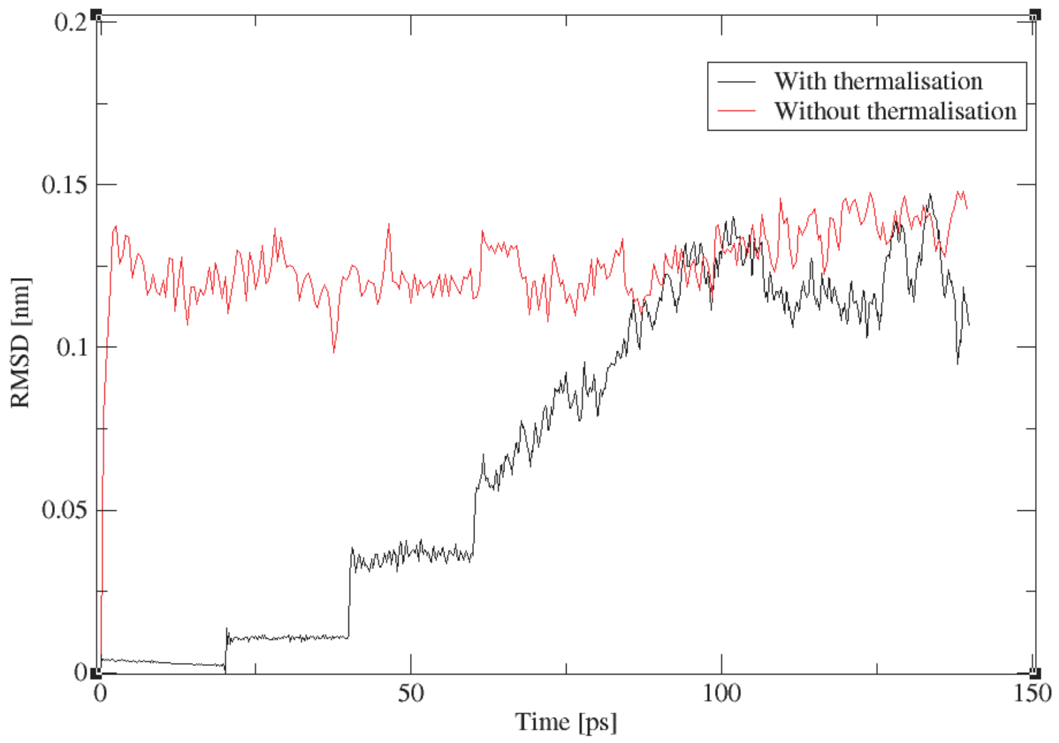


Figure 7: Equilibration step (150 ps) of the protein; protein with thermalisation (black), protein without thermalisation (red).

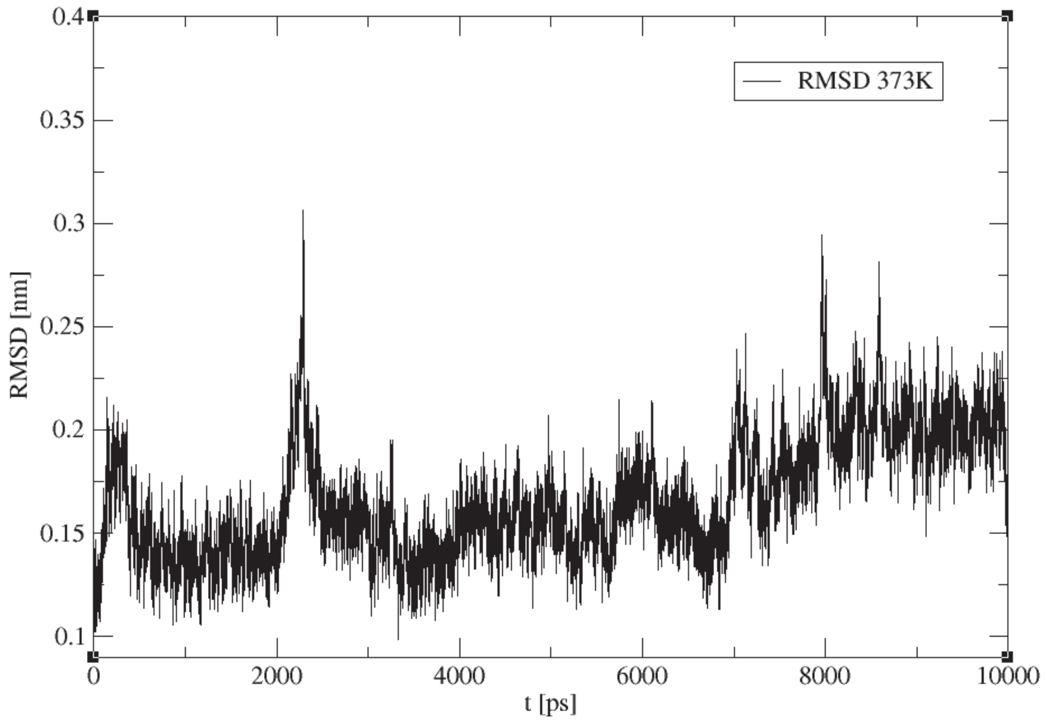


Figure 8: Additional RMSD plot - MD run at 373 K - protein does not unfold.

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

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

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

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

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

CSBMS: Exercise 4

Liquid Simulations & Properties

Document version: 27.08.2019

Exercises week 1:	05.11. or 07.11.
Exercises week 2:	12.11. or 14.11.
Deadline for the report:	24.11.
Contact:	`carmen.esposito@phys.chem.ethz.ch`
	Carmen Esposito - HCI G235

Summary

In this fourth exercise, we will focus on the calculation of a number of properties specific to liquids using molecular dynamics (MD) simulations employing GROMOS. Four different organic liquids (ketones) will be taken as examples. Six different types of liquid properties will be monitored and, whenever possible, compared with experimental data. The main focus of this exercise is on the new types of analyses (week 2). Considering the experience you already acquired in Exercises 1-3, the setup of the simulations (week 1) should be comparatively rapid and painless.

1 Introduction

Among the three common phases of matter (solid, liquid and gas), the liquid state (pure liquids and solutions) plays a particularly important role in classical molecular simulations. There are many reasons for this, but one may mention in particular the facts that:

- Most of the experimentally relevant (bio)chemical processes happen in the liquid phase.
- The intermolecular interactions in the gas and solid states are usually well described in terms of approximate analytical theories (ideal gas, harmonic crystal), which is not the case for the liquids.
- Many pure liquid properties (structural, thermodynamic, dielectric, transport and kinetic) can be calculated accurately based on relatively short simulations (10 ns or less), and can often be compared to equally accurate experimental values.

The last point is of particular interest in the context of force-field parameterization. Force-field parameters can be optimized against experimental data in the context of liquids of simple organic molecules, and then ported to more complicated systems using a transferability assumption in terms of molecular fragments.

In this exercise, we will focus on the calculation of a number of properties specific to liquids using MD simulations employing GROMOS. Four different organic liquids, the ketones listed in Table 1, will be taken as examples. Building blocks (*i.e.* full molecular topologies including force-field parameters) for these four compounds are available in the GROMOS $53A6_{\text{OXY}}$ force field,[1] which we will use in this exercise. Six different types of liquid properties will be monitored and, whenever possible, compared with experimental data. These properties, referring to atmospheric pressure and ambient temperature, include: (*i*) the liquid density ρ; (*ii*) the molar enthalpy of vaporization ΔH_{vap}; (*iii*) the molar isochoric heat capacity c_V; (*iv*) the pairwise radial distribution function (RDF) and coordination number (CN); (*v*) the static relative dielectric permittivity ϵ; and (*vi*) the self-diffusion coefficient D.

The main focus of this exercise is thus on the new types of analyses (week 2). And in this context, we will also show you a few "tricks" how to do some simple data transformations using `xmgrace`. Considering the experience you already acquired in Exercises 1-3, the setup of the simulations (week 1) should be a piece of cake for you now (or almost). It is thus a good opportunity to consolidate your knowledge concerning the GROMOS setup procedure, by looking in some more details into the files and asking your assistants all sorts of smart, advanced or/and unsettling questions! Along these lines, since you now followed the CSBMS lecture on "electrostatic interactions", we will also discuss a bit more the non-bonded interaction blocks of the GROMOS input file. And we will briefly discuss things you can do when you are idle because an equilibration job takes a few minutes to complete.

Note, finally, that each student will only carry out the simulation for one of the four liquids of Table 1 (week 1), but the four trajectories will be made available to all students at the time of analysis (week 2).

2 Week 1

2.1 Getting Started

Before we start the exercise we need to define which system you will run (Propanone, Butanone, 3-Pentanone or 3-Hexanone; see Table 1). In order to make sure that at least one student will run each of the four systems, we will choose it according to a predefined rule. After that, you need to copy the proper files to your home directory and you can start the exercise.

2.1.1 Choosing which System to Run

Please count the number of students sitting at the same row and located on your left . Calculate the remainder of the division of this number by 4, *e.g.* 5 remainder 4 = 1. Then pick the system related to this number according to Table 1. From there on, you should always replace `<mol-code>` when written in this script by the molecule three-letter code corresponding to your system according to Table 1. And `<capitalized-mol-code>` for the capitalized version of the `<mol-code>`, e.g., `PPN` if your mol-code is `ppn`.

Table 1: Four liquids considered.

Index	Compound	Molecule Code
0	Propanone	ppn
1	Butanone	btn
2	3-Pentanone	3pn
3	3-Hexanone	3hn

2.1.2 Setting up your Directory

In this exercise, we will assume that you always work in the folder

```
~/ex4/<mol-code>
```

within your home directory (∼), where `<mol-code>` is the molecule code of your system (Table 1).

To get this folder set-up with appropriate initial content, just do

```
%mkdir ~/csbms
mkdir ~/ex4
cp -r /usr/local/CSBMS/ex4/<mol-code> ~/ex4/
```

2.2 Creating the Molecular Topology File

As usual, before we start any simulation, we need to make a molecular topology file for our system. We will simulate a pure liquid system consisting of 512 molecules of your selected ketone. GROMOS formally offers three options to simulate N replicas of the same molecule in a system:

1. Set up your topology so that it contains N replicas of the molecule as *solute* and use NPM=1 (one solute copy) in the input files for the simulations.

2. Set up your topology so that it contains 1 replica of the molecule as *solute* and use NPM=N (N solute copies) in the input files for the simulations (in practice not possible, see below).

3. Set up your topology so that it contains 1 replica of the molecule as *solvent* and use NSM=N (N solvent copies) in the input files for the simulations (not possible for all types of molecules, see below).

We are going to use the option (1) above. Option (2) above is no longer supported in the GROMOS code.[1] Option (3) would not work for us, because what is declared as a *solvent* in GROMOS must obey two important constraints:[2] (*i*) be an entirely rigid molecule; (*ii*) consist of a single charge group. Our ketones do not satisfy these constraints, so they must belong to the *solute*. Note, finally, that the solvent can only consist of one type of molecule, whereas the solute can be anything.

Following option (1), we will proceed as described on Figure 1.

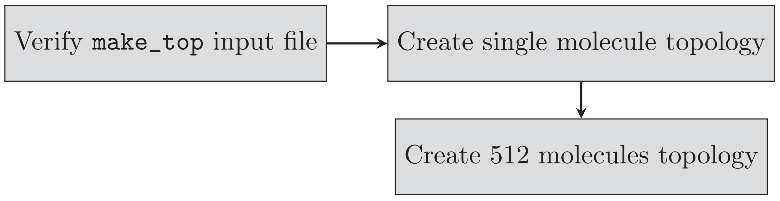

Verify make_top input file
Create single molecule topology
Create 512 molecules topology

Figure 1: Scheme of Topology Creation Steps.

Here, the process of creating your molecular topology file for a *single* molecule using `make_top` is quite trivial. Go to the folder `~/ex4/<mol-code>/topo`. There, we provide the required molecular topology building block `mtb` file and the required force field parameter `ifp` files (taken directly from the GROMOS distribution). In this exercise we will use the files `53a6_oxy.mtb` and `53a6_oxy.ifp` corresponding to the GROMOS 53A6$_{\text{OXY}}$ force field,[1] specifically optimized for alcohols, ethers, aldehydes, ketones, carboxylic acids, and esters. Have a look in `53a6_oxy.mtb`.

[1] It was available in the 1996 version of GROMOS in Fortran77 and was never ported to the C++ version; if you try NPM≠1 in the current code, you will get an error message "currently only NPM=1 allowed".

[2] The fullfilment of these constraints enables GROMOS to handle far more efficiently these molecules. Since water is the most common solvent and most water models satisfy these constraints, you save a lot of time in simulating aqueous systems by declaring water as a solvent rather than part of the solute. If you are picky there is another small difference in the treatment of solute and solvent molecules in GROMOS: the charge-group centers are defined as the center of geometry of the charge-group in the solute, but as the first atom in the solvent.

It already contains the four molecules of Table 1 in the form of building blocks. Can you find yours? (The name of the building blocks are in the form of `<capitalized-mol-code>`)

Type `make_top` (without arguments) to remind yourself of the available arguments for `make_top`. We already made the required `make_top_<mol-code>.arg` file for you. It is quite simple, and should look like

```
@build  53a6_oxy.mtb
@param  53a6_oxy.ifp
@seq    <capitalized-mol-code>
@solv   H2O
```

Now, run the program `make_top` using

```
make_top @f make_top_<mol-code>.arg > <mol-code>.top
```

Have a look in the generated `<mol-code>.top` file to make sure all looks healthy (in principle, you should also consider checking it with `check_top`, but we'll forget about this here).

Now, you need to create the topology for your entire system, which consists of *512 identical molecules* as a solute. For replicating the single molecule topology 512 times, we will use the program `com_top`.[3] Type `com_top` (without arguments) to remind yourself of the available arguments for `com_top`. We will use the arguments `@topo`, `@param` and `@solv`, as

```
com_top @topo 512:<mol-code>.top @param 1 @solv 1 > <mol-code>_512.top
```

The arguments `@param 1` and `@solv 1` tells `com_top` to use the force field and solvent defined on the the first topology, respectively. In our case, the 512 topologies are the same therefore any value below 512 produces the same results.

Have a look in the generated `<mol-code>_512.top` file to make sure all looks healthy (in principle, you could again check it with `check_top`). Maybe also consider updating the TITLE block of this file so that it is more informative.

2.3 Creating the Initial Coordinate File

The process of create your initial coordinates, 512 molecules of your ketone in a simulation box at a reasonable density, is also quite trivial. Because liquids, unlike biomolecular systems such as proteins (see Exercise 3) loose very quickly the "memory" of the initial configuration, typically within 10-100 ps or so, the initial coordinates do not matter much as long as we equilibrate the system decently afterwards. We will use the program `ran_box`, which creates a computational box with a specified number of randomly located and oriented molecules, avoiding the most nasty overlaps and enforcing a specified density. It does so by rotating and translating reference coordinates for one single molecule, that has to be provided to the program. Go to the folder ∼`/ex4/<mol-code>/coord`, and type `ran_box` (without arguments) to check the available arguments. We will use the arguments `@topo`, `@pbc`, `@pos`, `@nsm` and `@dens`. To choose these artuments, note we that: (*i*) the topology file to be used here should be that of a single molecule (not of the 512 molecule system); (*ii*) we want to generate a rectangular box (use `@pbc r`); (*iii*) reference single-molecule coordinates are provided to you with the name `<mol-code>.g96`; (*iv*) we want to generate 512 molecules in the box; (*v*) Each of the four liquids has its own density,[4] and you can find the corresponding experimental value for your specific ketone in Table 2 (see Appendix A of this document). Now, run the program `ran_box` to create the `<mol-code>_512.g96` file, *i.e.* as (obviously, you have some replacement to do here!)

[3] Note that we could alternatively bypass `com_top` by directly listing 512 times `<mol-code>` in the `@seq` argument of `make_top`.

[4] Note that `ran_box` expects a density in units of kg m^{-3}.

```
ran_box @topo <fill in the location of the topology file>
        @pbc  <fill in the right letter>
        @pos  <fill in the right file>
        @nsm  <fill in the right number>
        @dens <fill in the correct target density> > <mol-code>_512.g96
```

Have a look in the generated `<mol-code>_512.g96` file to make sure all looks healthy. In particular, it is a good idea to check out the dimensions of the generated box (be it only to make sure it is larger than twice the cutoff distance we will use in the subsequent simulations!) Maybe also consider updating the TITLE block of this file so that it is more informative.

2.4 Running the Simulations

The plan for the simulations is illustrated in Figure 2.

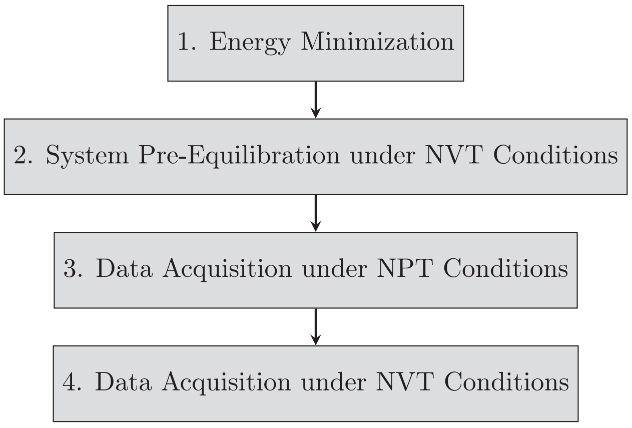

1. Energy Minimization
2. System Pre-Equilibration under NVT Conditions
3. Data Acquisition under NPT Conditions
4. Data Acquisition under NVT Conditions

Figure 2: Scheme of our Simulation Steps.

All steps displayed in the figure will be performed using GROMOS `md` program. To start the chain, we will need the molecular topology file created in Section 2.2 and the initial coordinate file created in Section 2.3. In addition, for each step, we will need a default `md` input file and a series of job files (including small variations from the default input file) as generated automatically by the program `mk_script`.

Steps 1 and 2 will be carried out during this session, as described in Sections 2.4.1 and 2.4.2. Then, we will prepare the files and submit the simulations for Step 3, as described in Section 2.4.3. These will serve as a basis for the analysis of 5 liquid properties characteristic of the liquid at 300 K and 1 bar. Finally, we will prepare the files for the simulations for Step 4, as described in Section 2.4.4. These will serve as a basis for the analysis of the 6^{th} liquid property, the molar isochoric heat capacity, at 300 K and a volume corresponding to the equilibrium density of the liquid considered. Since these simulations rely on the final configuration of Step 3, we will arrange that the job of Step 3 automatically submits the calculations of Step 4.

2.4.1 Energy Minimization

As usual, we do not want to start MD with a high-energy initial configuration. To relax strain and bad atom overlaps (which `ran_box` tries to avoid but not always entirely successfully), we first need to perform an energy minimization (EM).

Go to the folder ∼`/ex4/<mol-code>/min`. We already provided you with a script `min.bash` for the EM. Please open it and take a look. The input coordinate file of the minimization

```
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
                ?         ?        ?       ?      0.4     ?
END
```

Figure 3: The PARILIST block of the GROMOS input file

procedure, the one we generated with `ran_box`, is defined by the option `@conf`. The output coordinate file of the minimization procedure, the one we will use for subsequent work, is defined by the option `@fin`. The GROMOS input file for `md` is defined by the option `@input`.

Now, please open this input file `min.imd` and take a look. Obviously, we will use the steep descent algorithm (in which block do you find this information?). Many of the blocks we have already discussed in the previous exercises, so you should know what they contain and we won't repeat it here. But since you now followed the CSBMS lecture on "electrostatic interactions", we will discuss a bit more the non-bonded interaction blocks. These are the `PAIRLIST` and the `NONBONDED` blocks.

Irrespective whether we use the reaction-field or a lattice-sum method to calculate electrostatic interactions, we need to determine a set of atom pairs in close proximity, called a pairlist, as determined by a threshold distance, called a cutoff. For the *reaction-field method*, atoms in close proximity interact *via* a modified Coulombic interaction, whereas atoms at larger distances do not interact. The modification of the Coulombic interaction is designed so as to encompass the mean effect of the neglected interactions beyond the cutoff distance. For *lattice-sum methods*, atoms in close proximity interact *via* a so-called real-space interaction that is finite-ranged (this range should be shorter than the cutoff). The rest of the interaction, including the effect of periodic cells assumed to surround the reference cell, is calculated separately using Fourier series (Ewald method) or fast Fourier transforms (P^3M method), and referred to as the reciprocal-space interaction. For the *Lennard-Jones interactions*, only pairs within the cutoff distance interact.

The `PAIRLIST` block (Figure 3) controls how the pairlist is generated.

The `algorithm` switch determines whether the pairlist is generated using a standard (slow) double-loop algorithm or using a (fast) grid-based pairlisting method, where `SIZE` determines the grid-cell size.[2] The `NSNB` integer determines the frequency (number of steps) a new pairlist is constructed. If atoms diffuse relatively slowly, it makes sense and saves time to keep the same pairlist over a few timesteps. The `RCUTP` and `RCUTL` reals determine the short- and long-range cutoffs for the twin-range scheme. If they are equal, there is only one cutoff. Otherwise, the pairlist is evaluated every `NSNB` steps for atom pairs up to `RCUTP`. But at the same time, the intermediate-range interaction (energy force) for atom pairs between `RCUTP` and `RCUTL` is calculated. Between pairlist updates, both the pairlist and the intermediate-range interactions are assumed constant, and only the short-range interaction is recalculated based on the current pairlist. The effect is to effectively increase the cutoff to `RCUTL` at low computational cost, by neglecting the high-frequency fluctuations in the intermediate-range interaction. Finally, the `TYPE` switch determines if the pairlist is made based on distances between charge-groups

```
# Longrange reaction field correction
NONBONDED
# NLRELE    APPAK      RCRF   EPSRF  NSLFEXCL
       1        ?         ?       ?         ?
# NSHAPE   ASHAPE    NA2CLC   TOLA2     EPSLS
       3      1.4         2   1e-10         0
#    NKX      NKY       NKZ    KCUT
      10       10        10     100
# NGX   NGY   NGZ  NASORD  NFDORD   NALIAS  NSPORD
   32    32    32       3       2        3       4
# NQEVAL   FACCUR   NRDGRD   NWRGRD
  100000      1.6        0        0
# NLRLJ    SLVDNS
      0      33.3
END
```

Figure 4: The `NONBONDED` block of the GROMOS input file

or atoms. The standard setup for GROMOS simulations, which we will use in the present simulations, involves short- and long-range cutoffs of 0.8 and 1.4 nm, respectively, an update frequency of 5 timesteps for the pairlist, and a charge-group cutoff. One should be very careful about changing the pairlist parameters since some of these choices, especially regarding the cutoff distances, are correlated with the choice of optimal force-field parameters! And because both pairlist-generation algorithm provide the same pairlist we are going to use the fastest one.

The `NONBONDED` block (Figure 4), controls how the non-bonded interacions are calculated.

The `NLRELE` switch determines whether we use the reaction-field (value 1) or a lattice-sum (values >1) method for the electrostatic interactions. The GROMOS community generally prefers the reaction-field method. For this reason, the material specific to lattice-sum methods (all parameters of the block except the first and last lines) is not discussed here but in Appendix B at the end of this document. The `RCRF` real specifies where the dielectric continuum starts. Obviously, this value should be close to `RCUTL` and we always set it this way. The `EPSRF` real specifies the relative permittivity of the continuum surrounding the cutoff sphere. For self-consistency, this value should be set equal to the relative permittivity of the liquid (or solvent) model considered, which is in general the same as (or close to) the experimental value (Table 2). The `APPAK` real the inverse-Debye screening length of the continuum if it is meant to include ions at a certain ionic strength.[3] This is zero for a pure liquid, and is even generally not used when there are ions in the system. Finally, the GROMOS force field requires the exclusion of first and second covalent neighbors from electrostatic interactions. But the exclusion should only involve the direct Coulombic interaction, not the indirect reaction-field component. These interactions are included when the switch `NSLFEXCL` is set to 1.

You probably noticed that we have (intentionally!) inserted some question marks in Figures 3 and 4, which are also there in the `PAIRLIST` and `NONBONDED` blocks of your file `min.imd`. Obviously, they won't make GROMOS happy... So you should definitely replace the `'?'` by appropriate values in the input file before proceeding further! (you should find all the required information in the text above).

Now you can submit this script to the queueing system by typing

```
qsub -N min -j y -cwd ./min.bash
```

The minimization should not take more than a few seconds. If it takes longer please check the job status using the command `qstat`. Have a look in the output file `min.omd` to make sure all looks healthy (*e.g.* that the potential energy went down and that the program terminated normally).

2.4.2 Pre-Equilibration

Again as usual, before we start the production MD, we need to generate initial pseudo-random velocities and to equilibrate the system so that is looses the "memory" of the initial random configuration (`ran_box` generation) and velocities (pseudo-random Maxwell distribution). We will do this using 5 simulations of 50 ps each, generating pseudo-random velocities at the start of the first one. Although it would not be really necessary for such a pure-liquid system,[5] we will start at 60 K and increase the temperature by 60 K at every step.

Go to the folder ∼`/ex4/<mol-code>/eq`. In order to create the simulation chain corresponding to the 5 jobs we will use the program `mk_script` with the argument file `mk_script.arg`. Open this argument file and have a look. The argument `@input` specifies the GROMOS input file containing the default parameters. The argument `@joblist` specifies a file `eq.jobs` defining the parameter alterations for each of the successive jobs. Now open the `eq.jobs` file and check that it indeed corresponds to the temperature ramp we want to use. Finally, open the `eq.imd` file and check that the `PAIRLIST` and `NONBONDED` blocks are appropriately filled (this time, we were nice and directly filled them for you). Also determine from the file whether we run the equilibration at constant volume (NVT) or at constant pressure (NPT). Now run `mk_script` by typing

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

And submit the first calculation by typing

```
qsub -N run_eq_1 -cwd -j y -o run_eq_1.o -pe mpi 4 ./run_eq_1.run
```

The equilibration should take about 20 minutes to complete.

Which brings us to an important topic: what can you do when you are idle because an equilibration job takes a few minutes to complete? There are many options like reading the newspaper, going for a coffee, or flirting with the guy/girl at the neighbor computer.[6] Our recommendation is to stand up and sing the Swiss National Anthem. It will pass the time in a pleasant fashion and reinforce your patriotic feelings. So: **let's all stand up and sing the Swiss National Anthem together!** If you do not remember the lyrics, you will find them in Appendix C.

2.4.3 NPT Simulations

We can now start with the production runs. If you checked the input file in the previous section (as it was asked!) you will have noticed that we simulated at constant volume (NVT). Here, we will simulate first at constant pressure NPT.[7] We will use a temperature of 300 K and a

[5] We might as well start directly at 300 K with little harm.

[6] This is not guaranteed to succeed, as it requires a very specific combination between your gender and gender-inclination, and those of the person at the next computer. Tip: start with the person on one side, and if it fails, you may have more luck with the person on the other side. Second tip: synchronize your equilibration jobs with those of the person in question to increase the likelihood of simultaneous idleness.

[7] If we are picky, it would have been smarter to run the last pre-equilibration job already at constant pressure, because now, the first NPT job will include a small volume relaxation - which would better have belonged to the equilibration than to the production. But since the density was already chosen in a reasonable fashion when running `ran_box`, this relaxation is minimal and it will be very fast, so it does not really make a problem here.

pressure of 1 bar to match the thermodynamic standard conditions,[8] so as to later compare to experimental data under these conditions.

Go to the folder ∼`/ex4/<mol-code>/npt` and check the default GROMOS input file `npt.imd`. Finally, open the `npt.imd` file and check that the `PAIRLIST` and `NONBONDED` blocks are appropriately filled (here also, we were nice and directly filled them for you). Also check in the `STEP` block that the `NSTLIM` variable is 100000 and the `DT` variable to 0.002, and calculate what will be the length of a job in ns. And finally check the `PRESSURESCALE` block to make sure we run at a constane pressure of 1 bar.

Now open the file `mk_script.arg` and check the `@script` variable, which should indicate that we will be runing 50 successive jobs. Now you can guess again what will be the total simulation length. Note that the first job will receive the number 11.[9]

Now we can generate the job files by typing

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

and submit the first calculation by typing

```
qsub -N npt_11 -cwd -j y -o npt_11 -pe mpi 4 ./npt_11.run
```

These calculations should take about 3 days to complete. This is less than the full week separating us from the next exercise session, so we will make sure beaver has some more work to do until then. This is the goal of the next section.

2.4.4 NVT Simulations

The 50 jobs that are just starting are NPT simulations. But for next week, we will also need an additional set of NVT calculations. More precisely, we are going to calculate the molar isochoric heat capacity c_V of the liquid using finite-difference in temperature, namely, by comparing the average total potential energy of the system at 290, 300 and 310 K. Who says isochoric says NVT, thus the need for NVT simulations. And because we want to calculate c_V at a volume corresponding to the equilibrium density of the liquid model at 300 K and 1 bar, we are going to branch these calculations right after the 50th job of the previous NPT calculations, where the box volume should long have reached the appropriate equilibrium value.

Go to the folder ∼`/ex4/<mol-code>/nvt-300` and open the `nvt.imd` file. Here also, the `PAIRLIST` and `NONBONDED` blocks are already appropriately filled. Still check whether the `EPSRF` (`NONBONDED` block) and the `TEMPO` (`MULTIBATH` block) variables are set to the correct values (the last one should be 300 K for this directory), and that we will indeed run at constant volume as planned. Now have a quick look at `mk_script.arg`. It should be a chain of 50 jobs of 200 ps each, initiated from the final coordinates of the last NPT job, numbered 60 in the directory `../npt`.

Now do the very same operations in the folders ∼`/ex4/<mol-code>/nvt-290` and ∼`/ex4/<mol-code>/` `nvt-31` noting that the `TEMPO` variable should now be 290 K and 300 K, respectively.

Go to the folder ∼`/ex4/<mol-code>/npt`. The last job file for the NPT calculation has been generated by `mk_script` (see previous section) and is named `npt_60.run`. We need to include instructions to submit the 3 NVT job chains as soon as this last NPT job is completed. For this, append at the end of the file

[8] In fact, the standard temperature is 298.15 K, but it does not make much difference in practice.

[9] This is a little trick we use on the group. This way, when you use the UNIX `ls` command, the jobs will be listed in the right order. Starting from 1, *e.g.* 10 would be listed before 2. This works provided you have no more than 89 jobs. If you have between 90 and 899 jobs, you can start from 101.

```
cd ~/ex4/<mol-code>/nvt-290
/usr/local/gromos-1.3.2/bin/mk_script @f mk_script.arg
qsub -N nvt_11 -cwd -j y -o nvt_11.o -pe mpi 4 ./nvt_11.run

cd ~/ex4/<mol-code>/nvt-300
/usr/local/gromos-1.3.2/bin/mk_script @f mk_script.arg
qsub -N nvt_11 -cwd -j y -o nvt_11.o -pe mpi 4 ./nvt_11.run

cd ~/ex4/<mol-code>/nvt-310
/usr/local/gromos-1.3.2/bin/mk_script @f mk_script.arg
qsub -N nvt_11 -cwd -j y -o nvt_11.o -pe mpi 4 ./nvt_11.run
```

Note that the NPT chain will run the 50 jobs one after the other, whereas the 3 NVT chains will start and run *simultaneously* as soon as `npt_60.run` reaches the above statements.

2.5 Checking the Calculations

We are now all done for week 1. But please, over the coming days, do not forget to **check your jobs from time to time**. Since you run long job chains over a long time period, the likelihood that a minor problem (short power outage, temporary network interruption between a node and the filesystem, machine reboot for maintenance, etc...) causes one job to crash is non-negligible. In this case, restarting the chain is normally very easy. But the time between the stop and the restart is lost. If you check every day, this lost time is at most a day. If you check every week, obviously, this lost time can be up to a week.

To check your jobs, login on realbeaver (you may need `vpn` if you do this from outside ETH) and use the command `qstat`. If you want to check the directory in which a particular job is running use the option `'-j'`. This is very useful here because the NPT and NVT chains have identical job names, but run in different directories.

If you don't see the jobs you expect to be running, check the presence (and possibly the length) of the already generated trajectory files, `.trc` (usually the last one is partially complete if a job crashed while running) or `.trc.gz` (if the last one has a reasonable length, this may indicate a failure at submission of the next job). Then remove the corrupted/incomplete files that have been generated by the job that crashed, resubmit the corresponding job script to the queue with `qsub`, and recheck that all is fine with `qstat`.

So far, so good. We are done for this week... If you still have difficulties with the singing of the Swiss National Anthem, we recommend you to set aside a couple of hours this weekend for practicing, because we might need it again at the next session (and for the following exercises). See you next week!

3 Week 2

Welcome to Week 2. In this week we are analyzing our simulations.

3.1 Analyses

All the analyses performed here will depend on different programs from the GROMOS++ package and on `xmgrace`. The programs used and the properties calculated are shown in gray and white boxes, respectively, in Figure 5.

These analyses are explained in turn in Sections 3.1.1-3.1.6.

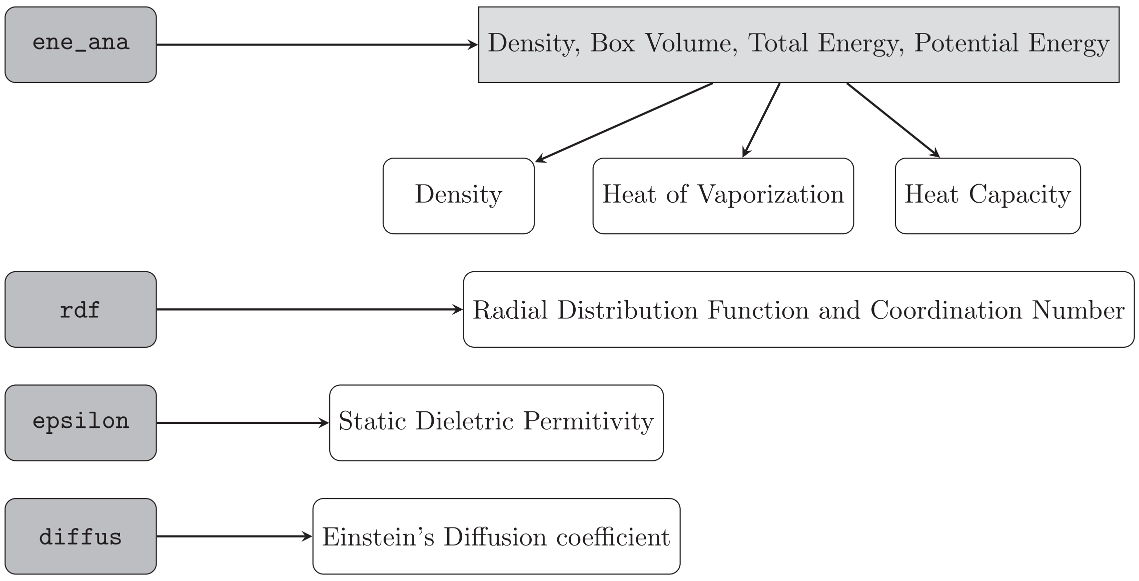

ene_ana
Density, Box Volume, Total Energy, Potential Energy
Density
Heat of Vaporization
Heat Capacity
rdf
Radial Distribution Function and Coordination Number
epsilon
Static Dieletric Permitivity
diffus
Einstein's Diffusion coefficient

Figure 5: GROMOS analysis program and possible generated data.

3.1.1 Density

The equilibrium box volume, density, total energy and total potential energy (among other quantities) can be calculated using the `ene_ana` program. The arguments are

```
%# Usage:
%#
%# ene_ana
    @en_files    <one or more energy files>
    @fr_files    <one or more free energy files>
    @prop        <properties to monitor>
    [@topo       <molecular topology file> (for MASS and NUMMOL)]
    [@time       <t and dt> (overwrites TIME in the trajectory files)]
    [@library    <library for property names> [print] ]
```

Note that one of `@en_files` or `@fr_files` is required, but not both (free energy calculations will be discussed in Exercise 5). First, we will use `ene_ana` on the NPT trajectory in order to extract all the average quantities listed above at 300 K and 1 bar.

Go to the folder ~`/ex4/<mol-code>/ana/ene/npt-300` and check whether the `ene.arg` file is similar to

```
@prop           boxvol densit totene totpot
@topo           ../../../topo/<mol-code>_512.top
@library        ../../ene_ana.md++.lib
@en_files
../../../npt/npt_21.tre.gz
../../../npt/npt_22.tre.gz
../../../npt/npt_23.tre.gz
...
(continues with the tre.gz files)
...
```

```
../../../npt/npt_58.tre.gz
../../../npt/npt_59.tre.gz
../../../npt/npt_60.tre.gz
```

Then run the analysis program by typing

```
ene_ana @f ene.arg > result.log
```

Have a look at the output file `result.log`. It should contain the average value of our properties, along with standard deviations and an error estimate based on block averaging.

You now have the tools to answer Question A in Section 4.2

3.1.2 Heat of Vaporization

The molar heat of vaporization is the enthalpy change corresponding to the transfer of one mole of the compound from the (pure) liquid phase to the (ideal) gas phase at a given temperature and pressure. We can calculate it based on our liquid simulations at 300 K and 1 bar using

$$\Delta H_{\mathrm{vap}} = U_{\mathrm{gas}} - U_{\mathrm{liq}} + RT + \Delta_{\mathrm{QM}} H_{\mathrm{vap}} \tag{1}$$

Here, U_{liq} is the molar total potential energy (total potential energy divided by the number of molecules) in the liquid state, U_{gas} the corresponding value in the gas phase, and R the ideal gas constant ($8.314\ \mathrm{J\,mol^{-1}\,K^{-1}}$). The third term RT accounts for the pressure-volume contribution to the enthalpy, and assumes that the molar volume of a liquid is negligible compared to that of an ideal gas. The last term $\Delta_{\mathrm{QM}} H_{\mathrm{vap}}$ accounts for quantum corrections to the values calculated using a classical model, predominantly correcting for the fact that the intramolecular degrees of freedom are not treated very accurately by the GROMOS force field (bond constraints, approximate force constants for the angles, torsional dihedrals and improper dihedrals). The nature of this term is further discussed in Appendix D. This correction can be calculated on the basis of quantum-mechanical (QM) calculations,[4] but it is typically very small and this term will be neglected in the present exercise.

The total potential energy of the system of 512 molecules in the liquid state was calculated in Section 3.1.1 (see file `result.log`), so that you know u_{liq}. The calculation of u_{gas} would require an additional simulation of the isolated molecule in vacuum (or of a set of molecules at very large [non-interacting] distances). Since we already did the corresponding calculation in Exercise 1, we will not repeat it, and merely provide you with the resulting values for the four ketones. These are $u_{\mathrm{gas}} = 0.04$, 3.54, 6.60 and 7.06 $\mathrm{kJ\,mol^{-1}}$ for ppn, btn, 3pn and 3hn, respectively.

You now have the tools to answer Question B in Section 4.2

3.1.3 Molar Isochoric Heat Capacity

The molar isochoric heat capacity is the quantity of heat that must be transfered to one mole of the substance for increasing its temperature by 1 K under constant-volume conditions. For the liquids considered, we calculate it here, for a temperature of 300 K and at a volume determined by the equilibrium density of the liquid for a pressure of 1 bar, based on three NVT simulations performed at 290, 300 and 310 K. The equation used is

$$c_V = \frac{1}{N}\frac{\partial E}{\partial T} \approx \frac{1}{N}\frac{U(T_1) - U(T_0)}{T_1 - T_0} + \alpha R + \Delta_{\mathrm{QM}} c_V \tag{2}$$

where E is the total energy of the system, $U(T)$ is the total potential energy of the system at temperature T, N the number of molecules in the system. The term αR accounts for the temperature derivative of the kinetic energy. Based on the equipartition principle, α is equal to one-half the number of unconstrained degrees of freedom of the molecule. If the molecule has N_{at} atoms and N_c constraints (here, we employ rigid bonds), one has $\alpha = (3N_{at} - N_c)/2$. You can find out the value of α by drawing structure of your ketone, counting the number of atoms and counting the number of bonds. Alternatively, you can have a look in the corresponding molecular topology file.[10] The last term $\Delta_{\mathrm{QM}} c_V$ accounts for quantum corrections to the values calculated using a classical model, predominantly correcting for the fact that the intramolecular degrees of freedom are not treated very accurately by the GROMOS force field (bond constraints, approximate force constants for the angles, torsional dihedrals and improper dihedrals) and the fact that a number of the unconstrained intramolecular degrees of freedom are still very "stiff" (*i.e.* their approximation by a classical harmonic oscillator with a c_V contribution of k_B is not appropriate at room temperature). The nature of this term is further discussed in Appendix D. This correction can be calculated on the basis of experimental IR and Raman spectra of the molecule, or of quantum-mechanical (QM) calculations.[5] Unlike for $\Delta_{\mathrm{QM}} H_{\mathrm{vap}}$ (previous section) the correction term is sizeable and cannot be neglected. The calculations are given in Appendix D and we simply provide you with the resulting estimates, namely 16.8 for ppn, 21.5 for btn, 26.3 for 3pn and 31.1 for 3hn, in units of J mol^{-1} K^{-1}.

Equation 2 is written in a simple finite-difference form. But since we have results at three tempertures, we can do better, by fitting a line to the potential energy as a function of temperature curve, and extract the corresponding slope. And that's exacly the way we are going to do it here.

Go to the folder ~/ex4/<mol-code>/ana/ene/nvt-290 and check whether the ene.arg file looks like

```
@prop           totpot
@topo           ../../../topo/<mol-code>_512.top
@library        ../../ene_ana.md++.lib
@en_files
../../../nvt-290/nvt_21.tre.gz
../../../nvt-290/nvt_22.tre.gz
../../../nvt-290/nvt_23.tre.gz
...
(continues with the tre.gz files)
...
../../../nvt-290/nvt_58.tre.gz
../../../nvt-290/nvt_59.tre.gz
../../../nvt-290/nvt_60.tre.gz
```

If not please modify it accordingly. Then run the ene_ana program by typing

```
ene_ana @f ene.arg > result.log
```

Do the very same procedure in the folders ~/ex4/<mol-code>/ana/ene/nvt-300 and ~/ex4/<mol-code>/ana/ene/nvt-310.

Now copy the potential energy values from each result.log file and create a file with the temperature in the first column and the total potential energy as second column like this (of course, your potential energy values will differ from those listed below)

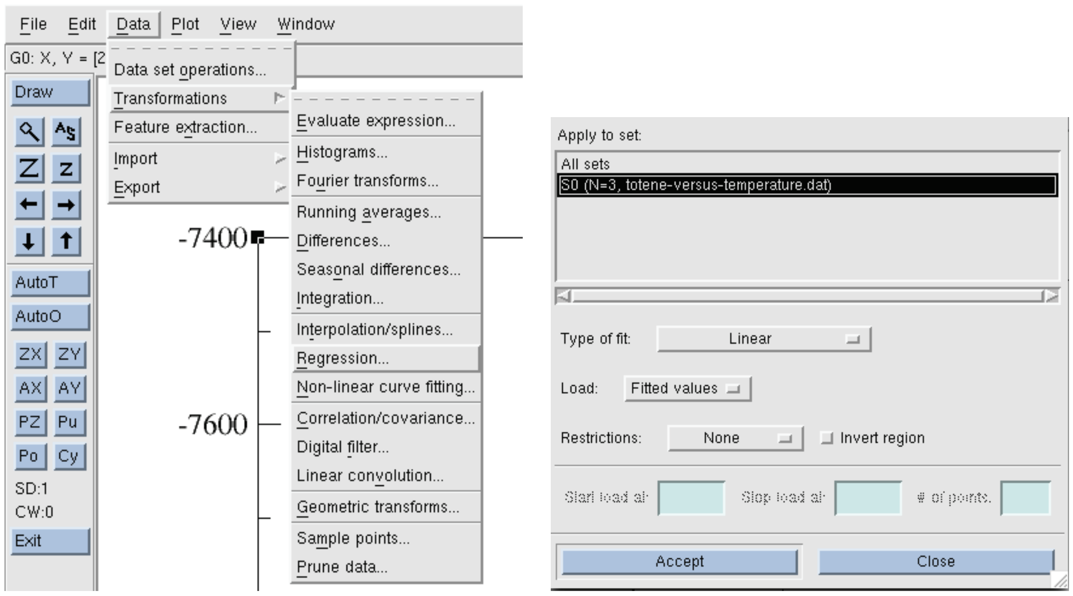
[10] An even simpler alternative would be to use the total energy instead of the total potential energy in the equation, and omit the term αR. But this would add noise in the calculated results (kinetic energy fluctuations) and the use of the αR variant is also more didactical.

```
# temperature  totpot
290    -19744.1234
300    -19571.5678
310    -19400.9876
```

Open this file with `xmgrace`.

```
xmgrace <your-file-here>
```

Perform the Linear Regression, by choosing Data→Transformations→Regression... as shown on Figure 6a. And select the total energy set (probably S0) and click on Accept (Figure 6b) and copy the slope of the least-squares-fit line.



S0 (N=3, totene-versus-temperature.dat)

Type of fit: Linear

Load: Fitted values

Restrictions: None

(a) Regression analysis menu `xmgrace`. (b) Regression analysis dialog box `xmgrace`.

Figure 6: Regression analysis in `xmgrace`.

The slope you obtain is given in GROMOS units of kJ mol^{-1} K^{-1} and pertains to the full system of 512 molecules. Use Equation 2 with the calculated α and the provided $\Delta_{\mathrm{QM}} c_V$ values. Do not forget the $\frac{1}{N}$ term, which converts the energy to a per molecule basis.

You now have the tools to answer Question C in Section 4.2

3.1.4 Radial Distribution Function and Coordination Number

The radial distribution function (RDF) $g_{IJ}(r)$ represents the local density of atoms of type I at a certain distance r from atoms of type I, relative to the bulk density of atoms of type J. This function will be zero at $r = 0$ (atoms cannot overlap) and tend towards one at large distances (bulk behavior). In the liquid state, it will generally evidence a sudden rise from zero followed by successive peaks of decreasing magnitudes corresponding to shells of excess density at favorable interatomic distances (the first one typically corresponding to contact distance), and then level off to one.

If N_I and N_J are the total numbers of atoms of types I and J in your computational box of volume V, the RDF for pairs IJ when $J \neq I$ is given by

$$g_{IJ}(r) = \frac{V}{N_I\,(N_J - \delta_{IJ})} \frac{1}{4\pi r^2 dr} \sum_{i \in I}^{N_I} \sum_{j \in J, j \neq i}^{N_J} dn(r_{ij}, r, dr) \tag{3}$$

where dr is the bin size for the histogram approximation of the RDF, $dn(r_{ij}, r, dr)$ is one if the (minimum-image) distance r_{ij} between atoms i and j and between r and $r + dr$, and δ_{IJ} is one if $I = J$ (RDF over the same atom types in different molecules, excluding an atom with itself [zero distance]) and zero otherwise (RDF over different atom types in the same or different molecules). The factor $4\pi r^2 dr$ accounts for the growth of the accessible volume at a distance r as a function of r (volume of a shell). It is factored out in the RDF in such a way that this function provides information on the excess density of atoms, rather than on the excess number of particles (*i.e.* we remove the baseline corresponding the case of randomly distributed [non-interacting] atoms). It is easily seen that a RDF obeys the normalization

$$\int_0^\infty dr\, 4\pi r^2\, g_{IJ}(r) = V \tag{4}$$

The RDF can be converted into a potential of mean force (PMF), providing information on the mean intermolecular interactions. Experimentally RDF's for liquids can be inferred (albeit with a significant uncertainty) from X-ray or neutron scattering experiments.

The coordination number (CN) function $G_{IJ}(R)$ represents the average number of atoms of type J that can be found within a certain distance R from an atom of type I (*i.e.* at any distance equal to R or less). Obviously, it can be derived from the RDF by integration, as

$$G_{IJ}(R) = \rho_J \int_0^R dr\, 4\pi r^2\, g_{IJ}(r) \tag{5}$$

where $\rho_J = N_J/V$ (case $I \neq J$) or $\rho_J = (N_I - 1)/V$ (case $I = J$). In particular, due to the normalization, $G(R)$ evaluates to N_J (case $I \neq J$) or $(N_I - 1)$ (case $I = J$) in the limit $R \to \infty$. In practice, the value of $G_{IJ}(R)$ at the location of the first (second) minimum in $g_{IJ}(r)$ is of most interest, representing the number of atoms of type J in the first (first+second) cordination shell of an atom of type I. For NVT simulations the number density is constant and for NPT simulations where the fluctuation of the box volume is small the ρ_J can be taken to be it's average value along the simulation.

In order to compute the $g_{IJ}(r)$ we will use the GROMOS program `rdf`, whose arguments are:

```
# Usage:
#
# rdf
    @topo     <molecular topology file>
    @pbc      <boundary type> [<gather method>]
    @centre   <atoms to take as centre>
    @with     <atoms to calculate distances for>
    @cut      <maximum distance>
    @grid     <number of points>
    [@nointra <skip intramolecular atoms>]
    @traj     <trajectory files>
```

The `centre` option defines the first atom type I. We will take the carbonyl oxygen atom of the ketone. The `with` option defines the second atom type J. We will take the carbonyl carbon atom of the ketone.

Now let's compute the rdf. Go to the folder ∼`/ex4/<mol-code>/ana/rdf` and check whether the `rdf.arg` file looks like

```
@topo          ../../topo/<mol-code>_512.top
@pbc           r
@centre        a:O
@with          a:C
@cut           1.4
@grid          100
@nointra
@traj
../../npt/npt_21.trc.gz
../../npt/npt_22.trc.gz
../../npt/npt_23.trc.gz
...
(continues with the tre.gz files)
...
../../npt/npt_58.trc.gz
../../npt/npt_59.trc.gz
../../npt/npt_60.trc.gz
```

If not please modify it accordingly. Then run the `rdf` program by typing

```
rdf @f rdf.arg > rdf_o_c.log
```

Plot the `rdf_o_c.log` file containing the RDF $g_{IJ}(r)$ using `xmgrace`. The CN $G_{IJ}(R)$ we will compute using `xmgrace`. But `xmgrace` cannot directly integrate $4\pi r^2 \rho_J g_{IJ}(r)$. We must first multiply our $g_{IJ}(r)$ by the function $4\pi r^2 \rho_J$ ourselves. In order to do that, first get the box volume value computed on Section 3.1.1.

Then, open the `rdf_o_c.log` file using `xmgrace` and perform the multiplication of $g_{IJ}(r)$ by $4\pi\rho_J r^2$ using Evaluate expression... tool (Data→Transformations→Evaluate expression...) as shown on Figure 7a. Notice that the last two values of the multiplication are the number of molecules in the system and the box volume. In our case, you just have to change the box volume to the value reported by `ene_ana`. After modifing the box volume accordingly, click on Apply.

The last step is the integration of this new function. Select the integration tool by clicking on Data→Transformations→Integration..., then select the new function data set (probably S1) and click on Accept (Figure 7b). Save the graph with these modifications for you will need it to write the report.

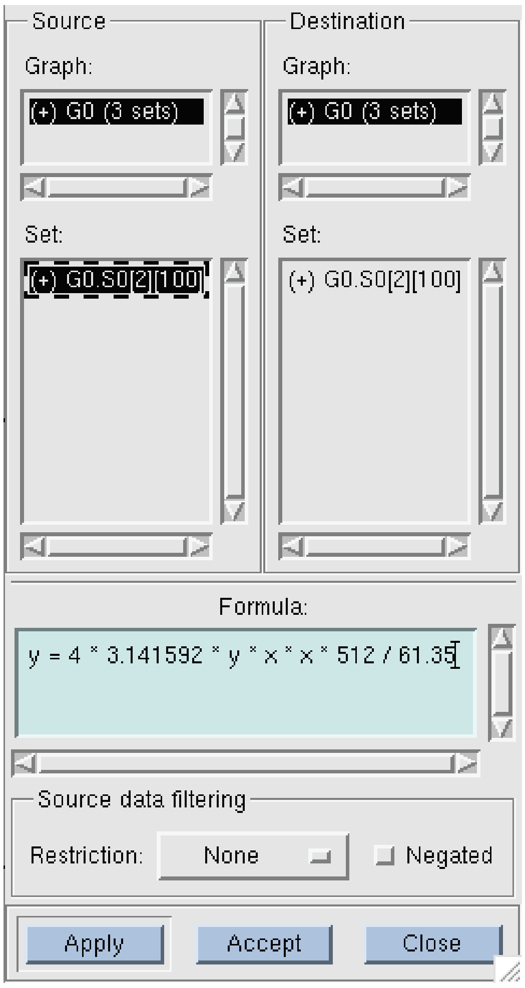
The last thing is to analyse the generated $G_{IJ}(R)$ curve to get the relevant first- and second-shell CN's. Move the mouse over the graph around the points shown on Figure 8 and mark the corresponding r positions. Use this x positions now on the integrated curve and get the number of carbonyl-carbons around the carbonyl-oxygens for each position.

Repeat this analysis now considering the carbonyl oxygen atom of the ketone as both atom type I and J, *i.e.* replacing the `@with` argument of `rdf` also by `a:O`.

You now have the tools to answer Question D in Section 4.2

3.1.5 Static Dieletric Permitivity

The static relative dieletric permitivity ε of a liquid characterizes its ability to screen electrostatic interactions (*e.g.* when placed between condensator plates or between ions embedded in it). This quantity can be evaluated from a simulation of the pure liquid based on the fluctuations of the total dipole moment of the computational box. When the reaction-field method is employed


Source

Graph:

(+) G0 (3 sets)

Set:

(+) G0.S0[2][100]

Destination

Graph:

(+) G0 (3 sets)

Set:

(+) G0.S0[2][100]

Formula:

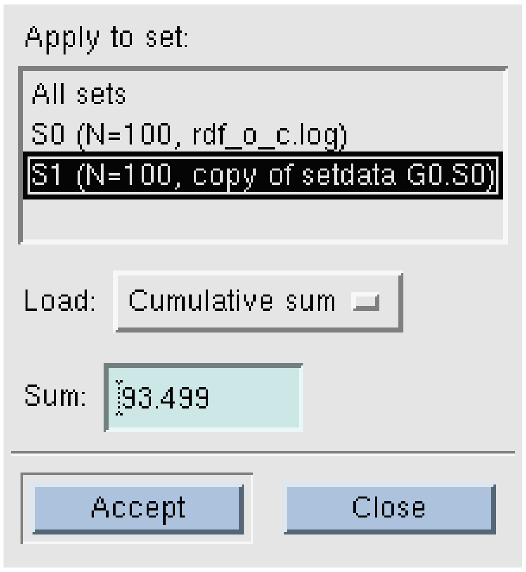
y = 4 * 3.141592 * y * x * x * 512 / 61.35

Source data filtering

Restriction: None

Negated

(a) Evaluate expression dialog xmgrace.


Apply to set:

All sets

S0 (N=100, rdf_o_c.log)

S1 (N=100, copy of setdata G0.S0)

Load: Cumulative sum

Sum: 93.499

(b) Integrate function dialog xmgrace.

Figure 7: Regression analysis in xmgrace.

to treat the electrostatic interactions (with a reaction-field permittivity $\varepsilon_{\mathrm{RF}}$) the appropriate Kirkwood-Fröhlich type of equation reads

$$(\varepsilon - 1)\left(\frac{2\varepsilon_{\mathrm{RF}} + 1}{2\varepsilon_{\mathrm{RF}} + \varepsilon}\right) = \frac{\langle \mathbf{M}^2 \rangle - \langle \mathbf{M} \rangle^2}{3\varepsilon_0 V k_{\mathrm{B}} T}, \tag{6}$$

where ε_0 is the vacuum permitivity, $\mathbf{M}$ the box dipole-moment vector, V is the box volume, k_{B} is the Boltzmann constant and T is the temperature. One can isolate ε to get

$$\varepsilon = \frac{3\left(2\varepsilon_{\mathrm{RF}} + 1\right)\varepsilon_0 V k_{\mathrm{B}} T + 2\varepsilon_{\mathrm{RF}}\left(\langle \mathbf{M}^2 \rangle - \langle \mathbf{M} \rangle^2\right)}{3\left(2\varepsilon_{\mathrm{RF}} + 1\right)\varepsilon_0 V k_{\mathrm{B}} T - \left(\langle \mathbf{M}^2 \rangle - \langle \mathbf{M} \rangle^2\right)}. \tag{7}$$

Note that for a sufficiently long simulation, $< \mathbf{M} >= \mathbf{0}$.

Equation 7 is used by GROMOS program epsilon to compute the model static permitivity. The program accepts as arguments

```
%# Usage:
%#
```

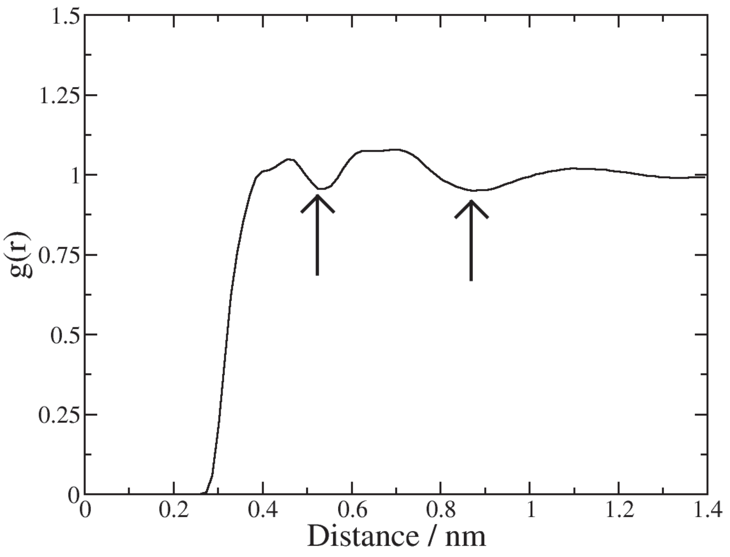


Figure 8: Position to compute the coordination number.

```
%# epsilon
    @topo <molecular topology file>
    @pbc    <boundary type> [<gather method>]
    [@time   <time and dt>]
    [@e_rf   <reaction field epsilon>]
    @temp   <temperature>
    @traj   <trajectory files>
```

Now let's compute the static dieletric permitivity of the target liquid. Go to the folder ~/ex4/<mol-code>/ana/eps and check that the eps.arg file looks like

```
@topo           ../../topo/<mol-code>_512.top
@pbc            r nog
@e_rf           <your system reaction field epsilon>
@temp           300
@traj
../../npt/npt_21.trc.gz
../../npt/npt_22.trc.gz
../../npt/npt_23.trc.gz
...
(continues with the tre.gz files)
...
../../npt/npt_58.trc.gz
../../npt/npt_59.trc.gz
../../npt/npt_60.trc.gz
```

If not please modify it accordingly. The static dieletric permitivity analysis can take sometime, so it is a good practice to submit them to the queue system by typing

```
qsub -cwd -b y /usr/local/gromos-1.3.2/bin/epsilon @f eps.arg > result.log
```

The `result.log` file should contain the box dipole moment and the cumulative static dieletric permitivity ε. Open this file with your prefered text editor and go to the last line and take note on the ε value. Plot this file using `xmgrace` using the option `'-nxy'` and check whether the ε value converged. The cumulative static dieletric permitivity should be the red curve.

You now have the tools to answer Question E in Section 4.2

3.1.6 Self-Diffusion Coefficient

The self-diffusion coefficient can be computed by Einstein relation as

$$D = \lim_{t\to\infty} \frac{\left\langle [\mathbf{r}(t) - \mathbf{r}(0)]^2 \right\rangle}{6t}, \tag{8}$$

where the $\mathbf{r}(t)$ is the position of the particule of interest in a given time t and $< ... >$ denotes averaging over particles and time origins.

In order to compute the diffusion coefficient using Einstein's formula we will use the GROMOS program `diffus`, whose arguments are

```
@topo      <molecular topology file>
@pbc       <boundary type> [<gather method>]
[@time     <time and dt>]
@dim       <dimensions to consider>
@atoms     <atoms to follow>
@traj      <trajectory files>
```

Now let's compute the model Einstein's diffusion coefficient. Go to the analysis folder `~/ex4/<mol-code>/ana/diffus` and check whether the `dif.arg` file looks like

```
@topo          ../../topo/<mol-code>_512.top
@pbc           r
@dim           x y z
@atoms         a:a
@traj
../../npt/npt_21.trc.gz
../../npt/npt_22.trc.gz
../../npt/npt_23.trc.gz
...
(continues with the tre.gz files)
...
../../npt/npt_58.trc.gz
../../npt/npt_59.trc.gz
../../npt/npt_60.trc.gz
```

If not please modify it accordingly. The diffus analysis can take sometime, so it is a good practice to submit them to the queue system by typing

```
qsub -cwd -b y /usr/local/gromos-1.3.2/bin/diffus @f dif.arg > result.log
```

The self-diffusion coefficient is given in GROMOS units which is nm^2 ps^{-1}, thus multiply the number by 1000 to get the units in usual SI units of 10^{-9} m^2 s^{-1}.

The `diffus` program fits $\left\langle [\mathbf{r}(t) - \mathbf{r}(0)]^2 \right\rangle$ using the curve contained in the `diffusdp.out` file. The Einstein's relation is only valid in the limit of linear dependency. So check whether the curve is linear (using `xmgrace`) and if not choose only the linear range and redo the fitting.

You now have the tools to answer Question F in Section 4.2

4 Report

4.1 Format

Please refer to the corresponding section of Exercise 1 for information on the goal and expected structure of the report.

4.2 Simulation Results

Answer the following questions:

A. Report the liquid density ρ calculated for your ketone at 300 K and 1 bar (including error bar) and compare the result with the experimental value in Table 2.

B. Report the molar enthalpy of vaporization ΔH_{vap} calculated for your ketone at 300 K and 1 bar (including error bar) and compare the result with the experimental value in Table 2.

C. Report the molar isochoric heat capacity c_V calculated for your ketone at 300 K and a liquid density appropriate for 1 bar (including error bar) and compare the result with the experimental value in Table 2.

D. Report the graphs of the RDF and CN for the cases considered (I =carbonyl oxygen, J =carbonyl carbon or $I = J$ =carbonyl oxygen), and report the corresponding first- and second-shell coordination numbers.

E. Report the graph of the cummulative estimate of static relative dielectric permittivity ϵ for your ketone at 300 K and 1 bar (including error bar), report the final estimate (including error bar), and compare the result with the experimental value in Table 2.

F. Report the graph of mean-square displacement as a function of time (including your linear least-squares fit) for your ketone at 300 K and 1 bar, report your corresponding estimate for the diffusion constant D (including error bar and regression coefficient).

4.3 Thinking Questions

Answer the following questions:

1. In Section 2.2, we explained the difference between the solute and solvent parts of the GROMOS topology. Can you explain why a box of SPC water declared as solute or as solvent have (very slightly) different simulated properties?

2. Considering again the solute-solvent distinction in GROMOS, how would you set up a systems consisting of a protein dimer (two copies of the same protein) in a 1:10 or a 10:1 methanol-water mixture so that the simulation is most efficient? (both the water and methanol models are assumed fully rigid and consisting of a single charge group).

3. In Section 3.1.2 we provided an equation for ΔH_{vap} that involves an approximation, namely that the molar volume of a liquid (v_{liq}) is negligible compared to that of an ideal gas ($v_{\mathrm{gas}} = RT/P$). At 300 K and 1 bar, and using the value of the liquid density from Table 2, calculate the corresponding error Pv_{liq} for the compound ppn and show that it is indeed negligible.

4. In Section 3.1.3, we did the setup and analysis, respectively, of a series of MD simulations to extract the molar isochoric heat capacity c_V of a liquid. How would you set up corresponding series of simulations to extract the molar isobaric heat capacity c_P, the thermal expansion coefficient γ_P, and the compressibility κ_T?

5. Would you expect the molar isobaric heat capacity c_P to differ significantly from the molar isochoric heat capacity c_V?

6. We calculated c_V by finite difference in temperature and chose an interval of 10 K between the simulations considered. This is a good compromise. What could happen if we choose the temperature interval too small or too large? (for too small, think about the effect of the error on the two calculated potential energies; for too large, think about the nature of the approxiation made in a finite-difference estimate)

7. Could we use classical MD simulations to extract the molar heat of formation ΔH_f of the liquid, *i.e.* the heat corresponding to the process of combining elements in their standard states (most stable compound and phase of the element at 298.15 K and 1 bar) for forming one mol of the liquid?

8. Assume I calibrate very carefully a force field for ketones and another force-field for alcohols. Clearly, I have no experimental information on the interaction between ketone and alcohol molecules in my training set. Yet, I can do simulations of the mixtures with GROMOS. What assumption is used within GROMOS to compensate for the absence of information on these the cross-interactions?

9. Assume I want to test if the GROMOS implicit assumptions for the cross-interactions are valid or not in the context of my ketone-alcohol mixtures (and maybe refine them if they are not good). What kind of systems / properties should I look at and compare with experiment?

Appendix A: Experimental Data on the Liquids Considered

Table 2: Experimental properties of the four liquids considered.

Compound	Molecule Code	ρ [kg m^{-3}]	ε_{RF}	ΔH_{vap} [kJ mol^{-1}]	c_V [†] [J mol^{-1} K^{-1}]
Propanone	ppn	784	19.1	31.3	125.2[a]
Butanone	btn	800	17.7	34.5	127.2[b]
3-Pentanone	3pn	809	16.6	38.6	196.4[c]
3-Hexanone	3hn	815	14.5	42.5	217.4[d]

[a] Ref. [6]
[b] Ref. [7]
[c] Ref. [8]
[d] Ref. [9]
[†] c_V is approximated here by c_p (for liquids, the difference should be very small).

Appendix B: NONBONDED block

In the main text (Section 2.4.1), we only discussed the reaction-field method and the corresponding input switches and parameters. The remaining parameters in Figure 4 are used for the Ewald sum and P^{3}M methods. The `NSHAPE` switch controls the charge-shaping function, where -1 is Guassian and 3 is a harmonic polynomial. The `ASHAPE` real controls the width of the charge-shaping function. The function should be zero beyond the cutoff, thus an appropiate `ASHAPE` value must be chosen. Since a Gaussian only vanishes in the limit of larger (∞) exponents, we tend to use other type of functions with better convergence.[11] The more complex the function the worse it is to compute it in the real-space. Therefore a compromise has to be made. In general, we set the `NSHAPE` to 3 and the `ASHAPE` equal to `RCUTL`. The `NA2CLC` switch controls how we compute the self interaction terms in the real-space A_2 and in the reciprocal-space $\hat{A}_2$. There is not analytical solution for this term, except for a quasi-analytical solution for the cubic-box case. So programs tend to solve this term numerically in both real- and reciprocal-spaces (switch value 2). The `TOLA2` real controls the tolerance in the numerical solution of the previous option and it should be a very small number but greater than the machine precision. The `EPSLS` real defines the lattice-sum permitivity used by the lattice-sum and emulated reaction-field method. Zero means infinity or no emulated reaction-field method. The last four integers `NKX`, `NKY`, `NKZ` and `KCUT` control the fast-fourier transformation in the Ewald's sum. The former three define maximum number k-vectors components in each direction and the latter controls the Ewald k-space cutoff. The values listed in Figure 4 tend be the best trade-off of accuracy and speed on GROMOS for most condensed phase systems.

Ewald's sum scales as the square of the number of particles (O(N^2)), which is not computationally efficient. Another way to solve Ewald's sum is to perform the actual sum using the descrite three-dimensional fast Fourier transform (3D-FFT) algorithm, which scales as O($N \ln N$). The FFT requires the point charges to be represented as grid points (discrete coordinates). The redistribution of this charge density in a grid and some more technical implementation details,

[11] If you use Gaussian type function good approximative values for `ASHAPE` would be $^1/_3$ or $^1/_4$ of the cutoff, in order to account for at least 95-99% of the Guassian area.

which are far away from the scope of this exercise, define the particle-particle-particle mesh or P^3M method.

The number of grid points around each point charge is controlled by the `NGX`, `NGY`, `NGZ` options. More points implies in better accuracy but slower calculations. So a compromise has to be made, and 32 points in each dimension is far better than the original 9 points in each dimension of the particle-mesh Ewald (PME) method. The `NASORD` flag controls the order of the mesh charge-assingnment function, `NFDORD` defines the order of the finite-difference operator and `NALIAS` defines the number of mesh alias vectors. The `NSPORD` determines the order of the B-spline function used to interpolate the grid points.

GROMOS allows for the reevaluation of the P^3M method accuracy every `NQEVAL` steps and it can impose a threshold on the force influence function through the option `FACCUR`. It also allows for the read and write of the influence function to a file through the `NRDGRD` (read) and `NWRGRD` (write) flags, where 1 turn on these options.

The last two options (`NLRLJ` and `SLVDNS`) are not related to long-range electrostatic corrections, but to long-range Lennard-Jones correction. This correction can be turned on by setting the `NLRLJ` flag to one and the `SLVDNS` flag defines the solvent density in molecules per nm^3. The value 33.3 molecules nm^{-3} corresponds to water with density of ≈ 1 g mL^{-1}.

Appendix C: Swiss National Anthem

1. Trittst im Morgenrot daher,
Seh' ich dich im Strahlenmeer,
Dich, du Hocherhabener, Herrlicher!
Wenn der Alpen Firn sich rötet,
Betet, freie Schweizer, betet,
Eure fromme Seele ahnt,
Eure fromme Seele ahnt,
Gott im hehren Vaterland!
Gott, den Herrn, im hehren Vaterland!

2. Kommst im Abendglühn daher,
Find' ich dich im Sternenheer,
Dich, du Menschenfreundlicher, Liebender!
In des Himmels lichten Räumen
Kann ich froh und selig träumen;
Denn die fromme Seele ahnt
Denn die fromme Seele ahnt
Gott im hehren Vaterland!
Gott, den Herrn, im hehren Vaterland!

3. Ziehst im Nebelflor daher,
Such' ich dich im Wolkenmeer,
Dich, du Unergründlicher, Ewiger!
Aus dem grauen Luftgebilde
Bricht die Sonne klar und milde,
Und die fromme Seele ahnt
Und die fromme Seele ahnt
Gott im hehren Vaterland!
Gott, den Herrn, im hehren Vaterland!

4. Fährst im wilden Sturm daher,
Bist du selbst uns Hort und Wehr,
Du, allmächtig Waltender, Rettender!
In Gewitternacht und Grauen
Lasst uns kindlich ihm vertrauen!
Ja, die fromme Seele ahnt
Ja, die fromme Seele ahnt
Gott im hehren Vaterland!
Gott, den Herrn, im hehren Vaterland!

1. Sur nos monts, quand le soleil
Annonce un brillant réveil,
Et prédit d'un plus beau jour Le retour,
Les beautés de la patrie
Parlent à l'âme attendrie;
Au ciel montent plus joyeux
Au ciel montent plus joyeux
Les accents d'un coeur pieux,
Les accents émus d'un coeur pieux.

2. Lorsqu'un doux rayon du soir
Joue encore dans le bois noir,
Le coeur se sent plus heureux près de Dieu
Loin des vain bruits de la plaine
L'âme en paix est plus sereine;
Au ciel montent plus joyeux,
Au ciel montent plus joyeux,
Les accents d'un coeur pieux,
Les accents émus d'un coeur pieux.

3. Lorsque dans la sombre nuit
La foudre éclate avec bruit,
Notre coeur pressent encore le Dieu fort.
Dans l'orage et la détresse,
Il est notre forteresse.
Offrons-Lui de coeurs pieux
Offrons-Lui de coeurs pieux
Dieu nous bénira des cieux,
Dieu nous bénira du hauts des cieux.

4. Des grand monts vient le secours,
Suisse! espère en Dieu toujours!
Garde la foi des aëux, vis comme eux!
Sur l'autel de la partrie
Met tes biens, ton coeurs, ta vie!
C'est le trésor précieux
C'est le trésor précieux
Que Dieu nous bénira des cieux,
Que Dieu nous bénira du hauts des cieux.

Appendix D: Vibrational Normal Modes for the Heat Capacity Correction

In Sections 3.1.2 and 3.1.3, we introduced correction terms $\Delta_{\text{QM}} H_{\text{vap}}$ and $\Delta_{\text{QM}} c_V$, respectively, associated with quantum-mechanical (QM) corrections to the classically calculated values.

For $\Delta_{\text{QM}} H_{\text{vap}}$ we set

$$\Delta_{\text{QM}} H_{\text{vap}} = Q_{\text{int}} + Q_{\text{ext}}$$

where Q_{int} and Q_{ext} are defined as

$$Q_{\text{int}} = \Delta H_{\text{QM}}^{\text{intra,vib}} - \Delta H_{\text{classical}}^{\text{intra,vib}} \tag{9}$$

$$Q_{\text{ext}} = \Delta H_{\text{QM}}^{\text{inter,vib}} - \Delta H_{\text{classical}}^{\text{inter,vib}}, \tag{10}$$

where $\Delta H_{\text{QM}}^{\text{intra,vib}}$ and $\Delta H_{\text{classical}}^{\text{intra,vib}}$ are the difference of the intramolecular vibrational energies of the molecule in the gas and liquid phases for the quantum-mechanical (QM) and classical (classical) models, respectively (Figure 9). And the $\Delta H_{\text{QM}}^{\text{inter,vib}}$ and $\Delta H_{\text{classical}}^{\text{inter,vib}}$ are the difference of the intermolecular vibrational energies of the molecule in the liquid phase for the quantum-mechanical (QM) and classical (classical) models, respectively. Both values have opposite sign and the sum of then results, in general, in a very small correction, which can be (and will be) neglected.[4][12]

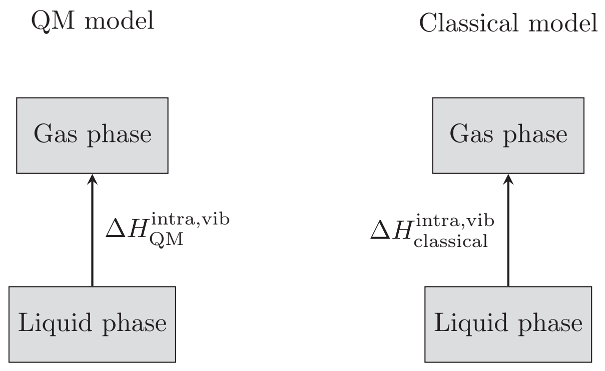

QM model
Classical model
Gas phase
Gas phase
$\Delta H_{\text{QM}}^{\text{intra,vib}}$
$\Delta H_{\text{classical}}^{\text{intra,vib}}$
Liquid phase
Liquid phase

Figure 9: Difference of the intramolecular vibrational energies of the molecule in the gas and liquid phases for the quantum-mechanical (QM) and classical (classical) models.

For $\Delta_{\text{QM}} c_V$ we set $\Delta_{\text{QM}} c_V = c_{V,\text{QM}} - c_{V,\text{classical}}$ where $c_{V,\text{QM}}$ and $c_{V,\text{classical}}$ refer to QM and classical estimates of c_V for the isolated molecule in the gas phase.

$c_{V,\text{QM}}$ is obtained by calculating the partition function for a QM harmonic oscillator representing the molecular vibrations. The harmonic oscillator vibrational energies can be computed as

$$E_{\text{vib}} = \sum_i \left(\frac{1}{2} h\nu_i + \frac{h\nu_i}{e^{h\nu_i / k_{\text{B}} T} - 1} \right), \tag{11}$$

where the h is Planck's constant, ν is the vibrational frequency, k_{B} is Boltzmann constant and

[12]Often this term is omitted due to the very small contribution of $\Delta_{\text{QM}} H_{\text{vap}}$ to the heat of vaporization and the high cost to compute $\Delta H_{\text{QM}}^{\text{intra,vib}}$ and $\Delta H_{\text{QM}}^{\text{inter,vib}}$ terms. And it also assumes that the quantum mechanical calculation is highly precise, which is not always true.

T is the temperature. Thus its derivative relative to the temperature is

$$c_{V,\mathrm{QM}} = \frac{\partial E_{\mathrm{vib}}}{\partial T} = k_{\mathrm{B}} \sum_i \left(\left(\frac{h\nu_i}{k_{\mathrm{B}}T} \right)^2 \frac{e^{h\nu_i/k_{\mathrm{B}}T}}{\left(e^{h\nu_i/k_{\mathrm{B}}T} - 1 \right)^2} \right). \tag{12}$$

If we use experimental data for the vibrational frequencies, we have to include both the experimental infrared (IR) and Raman frequencies.[13] However, sometimes the experimental frequencies are difficult to assing due to overlap between peaks or spectrum poor resolution. More rarely for some molecules the IR and Raman spectra are not available. For these cases, quantum calculations are employed in order to obtain the approximative IR and Raman spectra.[14] The IR and Raman spectra of all liquids in this exercise have superpositioned peaks and thus we will apply quantum calculations. It is not within the scope of this exercise to compute the vibrational modes. So, Table 3 shows the wave number associated to each normal mode for the different molecules.

$c_{V,\mathrm{classical}}$ is obtained from gas-phase simulations of the classical model and a temperature finite-difference analysis like the one we did for the liquid. This is not difficult, and the results are 0.01 for ppn, 0.03 for btn, 0.04 for 3pn and 0.06 for 3hn, in units of J mol^{-1} K^{-1}.

Table 3: Vibrational wave number of different compounds computed using B3LYP/6-31G(d) level of theory.

System	ν / cm^{-1}	System	ν / cm^{-1}	System	ν / cm^{-1}	System	ν / cm^{-1}
ppn	36.3148	btn	7.0040	3pn	33.4469	3hn	20.4023
ppn	134.7888	btn	109.9742	3pn	67.4993	3hn	57.6768
ppn	375.5500	btn	205.1310	3pn	180.3786	3hn	93.8056
ppn	486.7620	btn	248.3861	3pn	194.8510	3hn	142.5249
ppn	531.4720	btn	401.0956	3pn	203.9486	3hn	200.5033
ppn	786.3179	btn	473.8448	3pn	310.0432	3hn	247.9813
ppn	894.6618	btn	588.8532	3pn	407.6554	3hn	278.5554
ppn	897.7198	btn	762.8954	3pn	467.0511	3hn	331.0470
ppn	1095.1964	btn	764.6922	3pn	624.0100	3hn	409.2050
ppn	1131.2039	btn	949.3396	3pn	721.8086	3hn	469.0414
ppn	1245.4742	btn	962.1255	3pn	784.6312	3hn	650.1399
ppn	1411.0683	btn	1006.8751	3pn	829.6853	3hn	710.9060
ppn	1413.0129	btn	1117.4596	3pn	971.3150	3hn	781.6880
ppn	1490.4339	btn	1144.0941	3pn	1013.5106	3hn	832.5015
ppn	1494.8987	btn	1199.7642	3pn	1020.9420	3hn	880.7439
ppn	1498.7802	btn	1295.5536	3pn	1026.3933	3hn	907.0739
ppn	1516.3611	btn	1389.3695	3pn	1124.8935	3hn	1005.7251
ppn	1823.7870	btn	1412.6907	3pn	1147.4339	3hn	1031.9319
ppn	3044.8902	btn	1440.2818	3pn	1152.5530	3hn	1032.4265
ppn	3051.8904	btn	1483.3869	3pn	1277.3906	3hn	1059.5684
ppn	3101.1524	btn	1497.1376	3pn	1319.7742	3hn	1134.1219

Continues on next page

[13]None of the IR and Raman spectra should be ignored. Due to selection rules some vibrational modes can be not present in the IR or in the Raman spectra.

[14]Due to computational cost in general the frequencies are computed without anharmonic effects which should (in general) shift the frequency by $\approx 10\%$.

Table 3: continued from previous page

System	ν / cm^{-1}	System	ν / cm^{-1}	System	ν / cm^{-1}	System	ν / cm^{-1}
ppn	3108.5057	btn	1506.4167	3pn	1373.4029	3hn	1153.0349
ppn	3166.1486	btn	1524.0536	3pn	1403.8334	3hn	1159.5872
ppn	3166.9248	btn	1528.7107	3pn	1442.0674	3hn	1260.2971
ppn	—	btn	1817.5627	3pn	1442.9708	3hn	1303.7980
ppn	—	btn	3023.4873	3pn	1481.7052	3hn	1332.3249
ppn	—	btn	3048.0762	3pn	1493.5344	3hn	1342.0023
ppn	—	btn	3049.3907	3pn	1522.9916	3hn	1390.2569
ppn	—	btn	3065.2231	3pn	1523.1481	3hn	1419.9336
ppn	—	btn	3104.0560	3pn	1528.3113	3hn	1442.0931
ppn	—	btn	3132.9678	3pn	1528.5433	3hn	1443.2571
ppn	—	btn	3141.0795	3pn	1810.7233	3hn	1479.9670
ppn	—	btn	3163.7851	3pn	3021.9220	3hn	1492.4147
ppn	—	btn	—	3pn	3032.0060	3hn	1517.4382
ppn	—	btn	—	3pn	3047.1732	3hn	1523.6329
ppn	—	btn	—	3pn	3058.9059	3hn	1528.9590
ppn	—	btn	—	3pn	3065.3616	3hn	1530.7801
ppn	—	btn	—	3pn	3065.8772	3hn	1537.3364
ppn	—	btn	—	3pn	3133.1201	3hn	1808.7807
ppn	—	btn	—	3pn	3133.4380	3hn	3013.7884
ppn	—	btn	—	3pn	3140.9059	3hn	3028.4250
ppn	—	btn	—	3pn	3141.2870	3hn	3038.4885
ppn	—	btn	—	3pn	—	3hn	3040.4170
ppn	—	btn	—	3pn	—	3hn	3055.0191
ppn	—	btn	—	3pn	—	3hn	3062.3216
ppn	—	btn	—	3pn	—	3hn	3065.6382
ppn	—	btn	—	3pn	—	3hn	3086.9050
ppn	—	btn	—	3pn	—	3hn	3111.3842
ppn	—	btn	—	3pn	—	3hn	3113.7612
ppn	—	btn	—	3pn	—	3hn	3133.3079
ppn	—	btn	—	3pn	—	3hn	3141.1393

Appendix E: List of files

```
ex4/ppn/npt/npt.imd
ex4/ppn/npt/mk_script.arg
ex4/ppn/npt/mk_script.4.procs.lib
ex4/ppn/coord/ppn.g96
ex4/ppn/eq/eq.jobs
ex4/ppn/eq/mk_script.arg
ex4/ppn/eq/eq.imd
ex4/ppn/eq/mk_script.4.procs.lib
ex4/ppn/nvt-300/mk_script.arg
ex4/ppn/nvt-300/nvt.imd
ex4/ppn/nvt-300/mk_script.4.procs.lib
ex4/ppn/topo/make_top_ppn.arg
ex4/ppn/topo/53a6_oxy.ifp
ex4/ppn/topo/53a6_oxy.mtb
ex4/ppn/nvt-290/mk_script.arg
ex4/ppn/nvt-290/nvt.imd
ex4/ppn/nvt-290/mk_script.4.procs.lib
ex4/ppn/ana/ene_ana.md++.lib
ex4/ppn/ana/diffusion/dif.arg
ex4/ppn/ana/eps/eps.arg
ex4/ppn/ana/ene/nvt-300/ene.arg
ex4/ppn/ana/ene/nvt-290/ene.arg
ex4/ppn/ana/ene/npt-300/ene.arg
ex4/ppn/ana/ene/nvt-310/ene.arg
```

```
ex4/ppn/nvt-310/mk_script.arg
ex4/ppn/nvt-310/nvt.imd
ex4/ppn/nvt-310/mk_script.4.procs.lib
ex4/ppn/min/min.imd
ex4/ppn/min/min.bash
ex4/3pn/npt/npt.imd
ex4/3pn/npt/mk_script.arg
ex4/3pn/npt/mk_script.4.procs.lib
ex4/3pn/coord/3pn.g96
ex4/3pn/eq/eq.jobs
ex4/3pn/eq/mk_script.arg
ex4/3pn/eq/eq.imd
ex4/3pn/eq/mk_script.4.procs.lib
ex4/3pn/nvt-300/mk_script.arg
ex4/3pn/nvt-300/nvt.imd
ex4/3pn/nvt-300/mk_script.4.procs.lib
ex4/3pn/topo/make_top_3pn.arg
ex4/3pn/topo/53a6_oxy.ifp
ex4/3pn/topo/53a6_oxy.mtb
ex4/3pn/nvt-290/mk_script.arg
ex4/3pn/nvt-290/nvt.imd
ex4/3pn/nvt-290/mk_script.4.procs.lib
ex4/3pn/ana/ene_ana.md++.lib
ex4/3pn/ana/diffusion/dif.arg
ex4/3pn/ana/eps/eps.arg
ex4/3pn/ana/ene/nvt-300/ene.arg
ex4/3pn/ana/ene/nvt-290/ene.arg
ex4/3pn/ana/ene/npt-300/ene.arg
ex4/3pn/ana/ene/nvt-310/ene.arg
ex4/3pn/nvt-310/mk_script.arg
ex4/3pn/nvt-310/nvt.imd
ex4/3pn/nvt-310/mk_script.4.procs.lib
ex4/3pn/min/min.imd
ex4/3pn/min/min.bash
ex4/3hn/npt/npt.imd
ex4/3hn/npt/mk_script.arg
ex4/3hn/npt/mk_script.4.procs.lib
ex4/3hn/coord/3hn.g96
ex4/3hn/eq/eq.jobs
ex4/3hn/eq/mk_script.arg
ex4/3hn/eq/eq.imd
ex4/3hn/eq/mk_script.4.procs.lib
ex4/3hn/nvt-300/mk_script.arg
ex4/3hn/nvt-300/nvt.imd
ex4/3hn/nvt-300/mk_script.4.procs.lib
ex4/3hn/topo/make_top_3hn.arg
ex4/3hn/topo/53a6_oxy.ifp
ex4/3hn/topo/53a6_oxy.mtb
ex4/3hn/nvt-290/mk_script.arg
ex4/3hn/nvt-290/nvt.imd
ex4/3hn/nvt-290/mk_script.4.procs.lib
ex4/3hn/ana/ene_ana.md++.lib
ex4/3hn/ana/diffusion/dif.arg
ex4/3hn/ana/eps/eps.arg
ex4/3hn/ana/ene/nvt-300/ene.arg
ex4/3hn/ana/ene/nvt-290/ene.arg
ex4/3hn/ana/ene/npt-300/ene.arg
ex4/3hn/ana/ene/nvt-310/ene.arg
ex4/3hn/nvt-310/mk_script.arg
ex4/3hn/nvt-310/nvt.imd
ex4/3hn/nvt-310/mk_script.4.procs.lib
ex4/3hn/min/min.imd
ex4/3hn/min/min.bash
ex4/btn/npt/npt.imd
ex4/btn/npt/mk_script.arg
ex4/btn/npt/mk_script.4.procs.lib
ex4/btn/coord/btn.g96
ex4/btn/eq/eq.jobs
ex4/btn/eq/mk_script.arg
ex4/btn/eq/eq.imd
ex4/btn/eq/mk_script.4.procs.lib
ex4/btn/nvt-300/mk_script.arg
ex4/btn/nvt-300/nvt.imd
ex4/btn/nvt-300/mk_script.4.procs.lib
ex4/btn/topo/make_top_btn.arg
ex4/btn/topo/53a6_oxy.ifp
ex4/btn/topo/53a6_oxy.mtb
ex4/btn/nvt-290/mk_script.arg
ex4/btn/nvt-290/nvt.imd
ex4/btn/nvt-290/mk_script.4.procs.lib
ex4/btn/ana/ene_ana.md++.lib
ex4/btn/ana/diffusion/dif.arg
ex4/btn/ana/eps/eps.arg
ex4/btn/ana/ene/nvt-300/ene.arg
ex4/btn/ana/ene/nvt-290/ene.arg
ex4/btn/ana/ene/npt-300/ene.arg
ex4/btn/ana/ene/nvt-310/ene.arg
ex4/btn/nvt-310/mk_script.arg
ex4/btn/nvt-310/nvt.imd
ex4/btn/nvt-310/mk_script.4.procs.lib
ex4/btn/min/min.imd
ex4/btn/min/min.bash
```

References

[1] Horta, B.A.C., Fuchs, P.F.J., van Gunsteren, W.F. and Hünenberger, P.H. *J. Chem. Theory Comput.* **2011** 7, 1016-1031

[2] Schmid N., Christ C.D., Christen M., Eichenberger A.P. and van Gunsteren W.F. *Comp. Phys. Comm.* **2012** 183, 890-903

[3] Tironi I.G., Sperb R., Smith P.E. and van Gunsteren W.F. *J. Chem. Phys.* **1995** 102, 5451-5459

[4] Walser R., Mark A.E., van Gunsteren W.F., Lauterbach M. and Wipff G. *J. Chem. Phys.* **2000** 112, 10450-10459

[5] Gee P.J. and van Gunsteren W.F. *Molecular Physics* **2006** 104, 477-483

[6] Ibberson, R.M., David, W.I.F., Yamamuro, O., Miyoshi, Y., Matsuo, T. and Suga, H. *J. Phys. Chem.* **1995** 99, 14167-14173

[7] Tamura, K., Yamasawa, T. *J. Therm. Anal.* **2002** 69, 849-863

[8] Baglay, A.K., Gurariy, L.L. and Kuleshov, G.G. *J. Chem. Eng. Data* **1988** 33, 512-518

[9] Andon, R.J.L., Counsell, J.F., Lees, E.B., and Martin, J.F. *J. Chem. Soc. A* **1970** 833-837

CSBMS: Exercise 5

Free Energy Calculations

Document version: 27.08.2019

Exercises week 1:	19.11. or 21.11.
Exercises week 2:	26.11. or 28.11.
Deadline for the report:	08.12.
Contact:	`alzbetak@phys.chem.ethz.ch`
	Alzbeta Kubincová - HCI G227

Summary

In this fifth exercise, we will focus on free-energy calculations using molecular dynamics (MD) simulations employing GROMOS. You will move from chemists to alchemists, with the thermodynamic integration (TI) method by calculating the relative hydration free energies of benzene, toluene and phenol and comparing them with experimental data. The focus of this tutorial is on: (*i*) the preparation of the perturbation topologies; (*ii*) the set-up of the series of TI simulations; and (*iii*) the analysis of these simulations to obtain estimates for the free-energy differences.

1 Introduction

Under isothermal-isobaric conditions, free-energy changes control the spontaneous evolution of all macroscopic physical, chemical, and biological phenomena, as well as the maximal (reversible) work that can be collected during this evolution. Consequently, it has for a long time been one of the central tasks of molecular simulation to calculate free-energy differences associated with specific processes, and to understand the cause of these differences at the microscopic level. Free-energy simulations have been applied to numerous processes including phase transitions, mixing, solvation, ligand-receptor binding, macromolecular folding, and many others. In order to obtain accurate free-energy differences, two main challenges have to be met. First, a model for the system (in MD, a classical force field) has to be defined that correctly describes its thermodynamic properties. Second, an efficient scheme must be employed to sample relevant microscopic configurations for all the macroscopic states of interest, and include a sufficient number of transitions between these.

The processes for which free-energy differences can be calculated fall into three main categories: (*i*) thermodynamic processes, in which the states correspond to different values of a specific boundary parameter of the system (e.g. pressure or temperature); (*ii*) conformational processes, in which the states are defined as distinct regions of the configurational space of a system (e.g. folded and unfolded conformations of a macromolecule, free and bound state of a ligand-receptor complex); and (*iii*) alchemical processes, in which the states are defined by different molecular topologies (i.e. Hamiltonian functions) for the same set of particles (e.g. chlorobenzene vs. bromobenzene in water). The “traditional” free-energy calculation methods include pressure- or temperature-integration for the thermodynamic case, direct counting (DC) or umbrella sampling (US) for the conformational case, and thermodynamic integration (TI) or free-energy perturbation (FEP) for the alchemical case. Although it would be fun to try out one example for each category (of change as well as of method), for the sake of time the present tutorial will only consider alchemical changes and the TI method.

We will consider three closely-related molecules, displayed in Figure 1 (top) and very dear to the heart of organic chemists, namely benzene (BNZ), toluene (TOL) and phenol (PHE). Well…

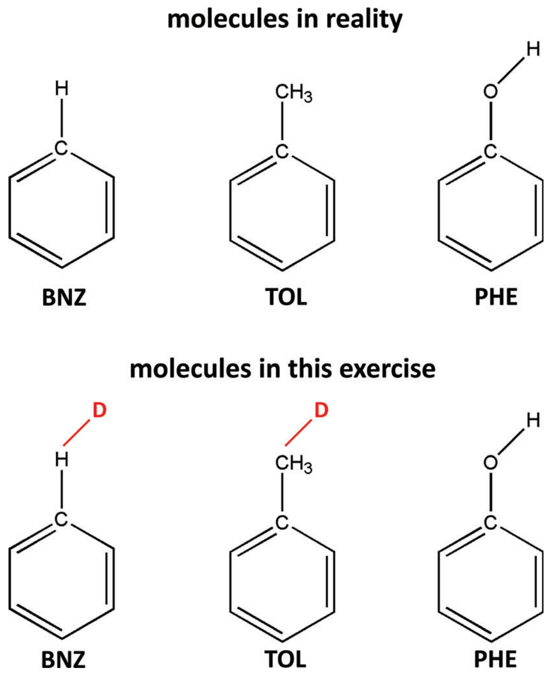


Figure 1: The molecules of benzene (BNZ), toluene (TOL), and phenol (PHE), in reality (top) and as considered in this tutorial (bottom; BNZ and TOL including a dummy atom D).

in fact not exactly. True, we will consider PHE. But this molecule has one more atom (in the GROMOS united-atom representation) compared to BNZ and TOL. So, we will actually consider instead of the true benzene/toluene molecules strange entities consisting of a benzene/toluene with a so-called dummy atom attached to the hydrogen/methyl atom, see Figure 1 (bottom). These weird species will still be labelled BNZ and TOL. The dummy atom is covalently attached to the rest of the molecule, but it has no non-bonded interactions whatsoever (neither intra- nor intermolecular). The choice of intramolecular covalent interactions involving the dummy atom is arbitrary, but must be the same throughout all calculations. Here, for BNZ, we will use everywhere a H-D bond and a C-H-D angle identical to the O-H bond and C-O-H bond-angle in PHE. For TOL, we will do the same for the CH_3-D bond and the C-CH_3-D bond-angle.

Using MD and TI, we will be able to calculate the relative free energies of the three compounds, and we will do this in vacuum (isolated molecules) and in water (hydrated molecules). On their own, the resulting numbers bear no meaning, for three reasons: (*i*) the GROMOS force field is not engineered to produce the correct quantum-mechanical intramolecular (electronic) energies of molecules; (*ii*) the mutations involve changes in the atom content of the molecule (alchemy) and have no real-world (chemistry) counter-part; and (*iii*) the results will be affected by the arbitrary choice of intramolecular interaction parameters for the dummy atoms. So, why should we bother calculating these irrelevant numbers? Simply because all these effects cancel out when we compare the free energies calculated in water to those calculated in vacuum, as illustrated in the thermodynamic cycle of Figure 2, considering the change BNZ$\rightarrow$PHE as an example.

The vertical legs of the cycle represent the alchemical mutations BNZ$\rightarrow$PHE in vacuum (left) and in water (right). The horizontal legs represent the hydration processes of BNZ (top) and PHE (bottom). Because free energy is a state function, we can write the difference in the

$$\begin{tikzcd}[column sep=huge, row sep=huge]
\text{BNZ in vacuum} \arrow[r, "\Delta G^{\text{vac}\to\text{wat}}_{\text{BNZ}}"] \arrow[d, "\Delta G^{\text{vac}}_{\text{BNZ}\to\text{PHE}}"'] & \text{BNZ in water} \arrow[d, "\Delta G^{\text{wat}}_{\text{BNZ}\to\text{PHE}}"] \\
\text{PHE in vacuum} \arrow[r, "\Delta G^{\text{vac}\to\text{wat}}_{\text{PHE}}"'] & \text{PHE in water}
\end{tikzcd}$$

Figure 2: Thermodynamic cycle used to calculate the relative hydration free energies of BNZ and PHE.

hydration free energies of BNZ and PHE as

$$\Delta\Delta G^{\text{vac}\to\text{wat}}_{\text{BNZ}\to\text{PHE}} = \Delta G^{\text{vac}\to\text{wat}}_{\text{PHE}} - \Delta G^{\text{vac}\to\text{wat}}_{\text{BNZ}} = \Delta G^{\text{wat}}_{\text{BNZ}\to\text{PHE}} - \Delta G^{\text{vac}}_{\text{BNZ}\to\text{PHE}} \quad . \tag{1}$$

This approach to calculate the relative free energies of two physical processes using simulations of two alchemical processes is called the thermodynamic-cycle approach. Unlike the alchemical free energies $\Delta G^{\text{vac}}_{\text{BNZ}\to\text{PHE}}$ and $\Delta G^{\text{wat}}_{\text{BNZ}\to\text{PHE}}$, the hydration free energies $\Delta G^{\text{vac}\to\text{wat}}_{\text{PHE}}$ and $\Delta G^{\text{vac}\to\text{wat}}_{\text{BNZ}}$ are physical quantities and could be compared with their experimental counterparts. In particular, the dummy atom in BNZ has no influence on the hydration free energy because it has no non-bonded interactions with the environment (so, in water, it does not "see" the water molecules). However, these physical quantities may be difficult to calculate directly with a high accuracy (we would have to "grow" full molecules from dummy-atom skeletons). The drawback of the thermodynamic-cycle approach is that it only gives access to the difference $\Delta\Delta G^{\text{vac}\to\text{wat}}_{\text{BNZ}\to\text{PHE}}$, not to the individual hydration free energies.

To calculate each of the alchemical free-energy changes, we will use TI, which relies on the statistical-mechanically exact expression

$$\Delta G_{0\to 1} = \int_0^1 \mathrm{d}\lambda \left\langle \frac{\partial \mathcal{H}\left(\mathbf{r},\mathbf{p},\mathrm{d}\lambda'\right)}{\mathrm{d}\lambda'} \right\rangle_\lambda \quad , \tag{2}$$

where $\mathcal{H}\left(\mathbf{r},\mathbf{p},\mathrm{d}\lambda\right)$ is the hybrid Hamiltonian, equal to that of the initial state when $\lambda = 0$ and to that of the final state when $\lambda = 1$, and $\langle\cdots\rangle_\lambda$ denotes Boltzmann (trajectory) averaging over configurations generated at given value of λ. We will use a soft-core coupling scheme for the hybrid Hamiltonian, and two approximations will be introduced in Equation 2: (*i*) the integral will be replaced by a quadrature sum (trapezoidal rule) based on simulations at 11 λ-points; and (*ii*) the infinite-ensemble averages in the integrand will be replaced by averages over 1 ns simulations (discarding 0.05 ns equilibration).

For the sake of time, you will only be asked to set-up and perform the simulations for the mutation BNZ→PHE in vacuum and in water. For the two other pairs of mutations, BNZ→TOL and TOL→PHE, the simulations have already been set-up and performed for you.

The work for now is described in Part 1. It involves looking at the set-up for the BNZ→TOL and TOL→PHE cases (final files already available), and setting-up the BNZ→PHE case (files to create or modify yourself). The analyses of the generated trajectories are described in Part 2.

2 Week 1

2.1 Simulation Set-up

Before we started, let's login to the beaver cluster and copy a directory that was already prepared for you.

```
cp -r /usr/local/CSBMS/ex5 ∼/
```

In the next section we will discuss the set-up of the simulations in water and in vacuum, respectively.

2.1.1 Simulations in Water

There are three mutations involved here. The most logical directions for the TI simulations would be BNZ→TOL, TOL→PHE and BNZ→PHE (left to right in Figure 1, i.e. every time from a "simpler" to a "more complicated" compound), as suggested in Introduction.[1]

When you do TI, you obviously need in some way to have two molecular topologies, one for the initial state (A) and one for the final state (B). There are two different ways to handle this. In the so-called dual-topology implementation, you will set up two full-blown molecular topology files with the same file structure, one for the state A and one for the state B topology. In the so-called single-topology implementation, you only set up a full-blown topology for the state A. And in a separate file, called a perturbation topology file, you will specify the topology changes getting you from topology A to topology B. The single-topology approach is advantageous when B is not very different from A because: (*i*) the perturbation topology file is small; and (*ii*) when you calculate the energies and forces, it is most efficient to calculate the potential energy of state A and then the terms contributing to the difference A-to-B (if you calculated the potential energies of A and B separately, you would calculate twice all the common force-field terms!). For these reasons and because GROMOS is smart, it relies on a single-topology implementation.[2]

As already stated in Introduction, the TOL→BNZ and PHE→TOL cases are already completely handled. All the files are in the directories: `TI_TOL_BNZ` and `TI_PHE_TOL`, respectively. The case BNZ→PHE is left for you to handle, in directory `TI_PHE_BNZ`.

Now let's have a look at the existing files, starting with the PHE→TOL case.

```
cd ∼/ex5/topo
```

Have a look at the topolgy file `phe.top` and convice yourself that this is indeed the appropriate topology file for a single PHE molecule (i.e. if you did not do a free-energy calculation, you could use this file directly for a standard GROMOS simulation of PHE). To help you in this check, have a look at Figure 3 (right column) where the topological information on PHE is reported graphically.

```
cd ∼/ex5/TI_PHE_TOL
```

Then have a look at the perturbation topology file `phe_to_tol.ptp` (displayed below), where the changes between the state A and the state B topologies are listed.

[1]We have actually set up everything for the reverse mutations, i.e. TOL→BNZ, PHE→TOL and PHE→BNZ.

[2]We have set-up the reverse mutations, simply because the A-state of the two latter mutations, PHE, is a normal molecule without dummy atoms, for which a topology building block is already available in GROMOS. In other words, we can set up the A-state topology for each of these two mutations with a single `make_top` command.

```
TITLE
perturbation topology, phenol to toluene
END
PERTATOMPARAM
# number of perturbed atoms
   3
#
#   NR RES NAME IAC(A)  MASS(A)  CHARGE(A) IAC(B) MASS(B) CHARGE(B)   ALJ  ACRF
   11   1    C6     12   12.011      0.203    12   12.011     0.00      0     0
   12   1    O6      3   15.9994    -0.611    16   15.035     0.00      1     1
   13   1    H6     21    1.008      0.408    22    1.008     0.00      1     1
END
PERTBONDSTRETCH
# number of perturbed bonds
    1
#   atom(i) atom(j) bond_type(A) bond_type(B)
         11      12           13           27
END
```

The PERTATOMPARAM block specifies the perturbed atoms. NR is the sequence of the atom, RES is the sequence of the residue, NAME is the name of the atom, IAC(A) is the integer atom code in state A, MASS(A) is the mass of the atom in state A, CHARGE(A) is the atomic partial charge of the atom in state A, IAC(B) is the integer atom code in state B, MASS(B) is the mass of the atom in state B, CHARGE(B) is the atomic partial charge of the atom in state B. Finally, ALJ and ACRF are scaling factors for the soft-core parameters involved in Lennard-Jones and Coulombic interactions, respectively (see further below). The PERTBONDSTRETCH block specifies the perturbed bonds, in which atom(i) and atom(j) are the sequences of the two atoms defining the bond, while bond_type(A) and bond_type(B) are the bond types in states A and B, respectively. Have a look again at Figure 3 (middle and right columns), and verify that the changes listed in the file reflect the expected topology changes when going from PHE to TOL. Note in particular that, as discussed in Introduction, the hydroxyl hydrogen atom of PHE has not been deleted but transformed into a dummy atom (atom type 22 in GROMOS).

Next, have a look at the GROMOS input file TI.inp (displayed below). By now, you certainly know the structure of these files by heart, but... there is a new type of block here, the PERTURBATION block

```
PERTURBATION
#    NTG: 0..1 controls use of free-energy calculation.
#         0: no free-energy calculation (default)
#         1: calculate dH/dRLAM
#  NRDGL: 0,1 controls reading of initial value for RLAM.
#         0: use initial RLAM parameter from PERTURBATION block
#         1: read from configuration
#   RLAM: 0.0..1.0 initial value for lambda
#  DLAMT: >= 0.0 rate of lambda increase in time.
# ALPHLJ: >= 0.0 Lennard-Jones soft-core parameter
#  ALPHC: >= 0.0 Coulomb-RF soft-core parameter
#   NLAM: > 0 power dependence of lambda coupling
# NSCALE: 0..2 controls use of interaction scaling
#         0: no interaction scaling
#         1: interaction scaling
#         2: perturbation for all atom pairs with scaled
#            interactions. No perturbation for others.
#
#     NTG   NRDGL    RLAM   DLAMT
        1       0     0.0     0.0
#  ALPHLJ   ALPHC    NLAM  NSCALE
      0.5     0.5       1       0
END
```

NTG set to one switches on the perturbation code in GROMOS for a TI simulation, RLAM is the value of the λ in the specific TI simulation (if NRDGL was set to one, the value would be read from the initial configuration file instead), DLAMT is set to zero (so λ will be kept constant at RLAM throughout the simulation), while ALPHLJ and ALPHC are the soft-core parameters for Lennard-Jones and Coulombic interactions, respectively. Soft-core interactions are used to avoid singularities when creating or deleting atoms (more precisely: when converting them from or to dummy atoms). The soft-core interaction will only be applied for Lennard-Jones and Coulombic interactions, respectively, to atoms for which the ALJ and ACRF parameters, respectively, differ from 0 in the perturbation topology (see above). Note that NRDGL and DLAMT are useful if you do slow growth (an alternative form of TI where you "sweep" λ from 0 to 1 over a run or a series of runs; but TI at a series of fixed λ-points is more accurate). NLAM and NSCALE allow for more variations in the form of the coupling scheme, which are not relevant for us here.

Finally, have a look at the mk_script job-list file joblist_TI.dat. Herein, we specify 11 different folders for the simulations at the 11 successive λ-points in water. For each λ-point, 25000 steps (0.05 ns) of equilibration and 500000 steps (1 ns) of sampling will be carried out.

Now let's have a look at the second set of existing files, corresponding to the TOL→BNZ case.

```
cd ∼/ex5/topo
```

And take a look at the state-A topology file tol.top to convince yourself that this is indeed the appropriate topology file for a single TOL molecule (including a dummy atom) by comparison with Figure 3 (middle column).

```
cd ∼/ex5/TI_TOL_BNZ
```

Then have a look at the perturbation topology file tol_to_bnz.ptp. In addition to the blocks you already saw in the previous perturbation, you will find a new block PERTBONDANGLE for the perturbed bond angles. Atom(i), atom(j), and atom(k) are the atoms defining the perturbed angle, while type(A) and type(B) are the bond angle types in state A and B, respectively.

```
PERTBONDANGLE
# number of perturbed bond angles
    2
#    atom(i) atom(j) atom(k) type(A) type(B)
           7      11      12      27      25
           9      11      12      27      25
END
```

By comparison with Figure 3 (left and middle columns), verify that the changes listed in the file (PERTATOMPARAM, PERTBONDSTRETCH and PERTBONDANGLE blocks) reflect the expected

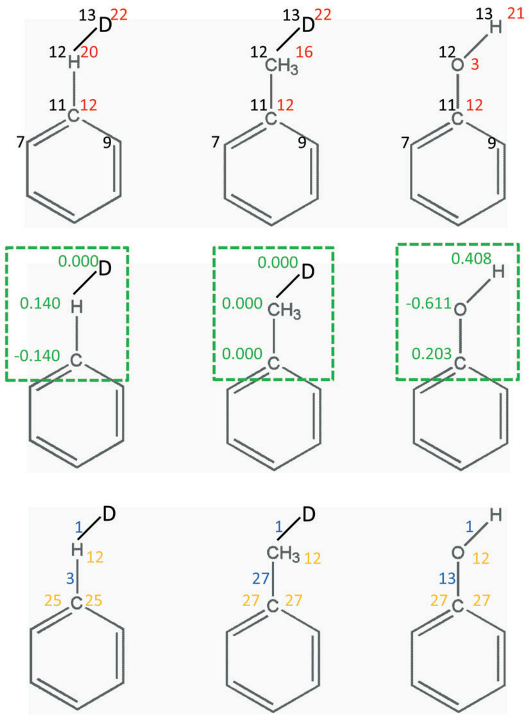


Figure 3: Relevant parameters in the topologies of BNZ, TOL, and PHE. Note that 5 explicit hydrogen atoms on the benzene rings are not indicated. Upper panel: sequence numbers of the atoms (black) and integer atom codes (red). Middle panel: atomic partial charges (green) and charge-group boundary (dashed line). Lower panel: bond types (blue) and bond-angle types (orange).

topology changes when going from TOL to BNZ. In particular can you figure out why the `PERTBONDANGLE` block is needed here, whereas it was not in the previous perturbation?

Note that there are other blocks which are not needed here, but could be used in other types of perturbations. For example, a `PERTIMPROPERDIH` block is used for perturbed improper dihedrals, a `PERTPROPERDIH` block is used for perturbed dihedrals, a `PERTPOLPARAM` block is used for perturbed polarizable atoms, and a `MPERTATOM` block is used for enveloping distribution sampling (EDS) simulations.

The GROMOS input file `TI.inp` and the `mk_script` job-list file `joblist_TI.dat` are exactly identical to the files used in the previous perturbation, so no need to look at them. Then, what is left for us is to set-up the PHE→BNZ mutation in water. So let's roll up our sleeves take a deep breath and start working.

When you feel mentally ready for this big step, just type

```
cd ~/ex5/TI_PHE_BNZ
```

Due to time constraints we will skip the molecular topology preparation steps today, but in a free-energy calculation, the procedure for state A is exactly the same as for a plain MD simulation. The topology and coordinates files are directly provided to you in the `topo` and `coord` directories, respectively. The `mk_script` job-list file `joblist_TI.dat` is also the same as in the two previous perturbations, so it is already prepared.

You first task is to complete the perturbation topology file `phe_to_bnz.ptp` (displayed below).

```
TITLE
TO_DO
END
PERTATOMPARAM
# number of perturbed atoms
    3
#
#   NR RES NAME IAC(A)  MASS(A)  CHARGE(A) IAC(B) MASS(B) CHARGE(B)   ALJ  ACRF
    TO_DO
END
PERTBONDSTRETCH
# number of perturbed bonds
    1
#   atom(i) atom(j) bond_type(A) bond_type(B)
    TO_DO
END
PERTBONDANGLE
# number of perturbed bond angles
    2
#   atom(i) atom(j) atom(k) type(A) type(B)
    TO_DO
END
```

The number of perturbed atoms, bonds, and bond angles are already given. Based on Figure 3, and what you have learned from the examples in the previous two TI simulations, finalize the perturbation topology file.

The second task is to complete the parameter input file `TI.inp` by replacing the `TO_DO` with the `PERTURBATION` block. In this case, you only need to insert in the following `PERTURBATION` block with the correct parameters.

```
PERTURBATION
#    NTG: 0..1 controls use of free-energy calculation.
#         0: no free-energy calculation (default)
#         1: calculate dH/dRLAM
#  NRDGL: 0,1 controls reading of initial value for RLAM.
#         0: use initial RLAM parameter from PERTURBATION block
#         1: read from configuration
#   RLAM: 0.0..1.0 initial value for lambda
#  DLAMT: >= 0.0 rate of lambda increase in time.
# ALPHLJ: >= 0.0 Lennard-Jones soft-core parameter
#  ALPHC: >= 0.0 Coulomb-RF soft-core parameter
#   NLAM: > 0 power dependence of lambda coupling
# NSCALE: 0..2 controls use of interaction scaling
#         0: no interaction scaling
#         1: interaction scaling
#         2: perturbation for all atom pairs with scaled
#            interactions. No perturbation for others.
#
#     NTG   NRDGL    RLAM   DLAMT
        ?       ?       ?       ?
```

```
#  ALPHLJ   ALPHC    NLAM  NSCALE
        ?       ?       ?       ?
END
```

The last step is to finalize the `mk_script_TI.arg` file. In the `@sys` argument, give a name to your first TI simulation (and keep it in mind, because you need to use this name several times later). And there are several other entries you need to take care of as well.

```
@sys     TO_DO
@joblist TO_DO
@bin     /usr/local/gromos-1.3.2/bin/md_mpi
@dir     $PWD
@files
        topo        ../topo/phe.top
        input       TO_DO
        coord       ../coord/phe.cnf
        pttopo      TO_DO
@template   ../lib/mk_script.4.procs.lib
@version md++
```

Once you set up everything correctly, you can use `mk_script` to generate the job files for TI simulations.

```
mk_script @f mk_script_TI.arg
```

Now you should have eleven folders called `L_*` in your directory. Look into these folders!!!

To submit your TI simulation, open the file `job_submit.sh`, and change `TO_DO` to the name of your simulation, which should be the same as what you put in as `@sys` argument in the `mk_script_TI.arg` file.

```
TI=TO_DO

cd L_0.0
qsub -N ${TI}_1 -cwd -j y -o ${TI}_1 -pe mpi 4 ./${TI}_1.run
```

Now you can submit your simulation by running

```
./job_submit.sh
```

Check whether your simulation is running properly on the cluster using

```
qstat
```

If so, you can move to the next step. If not, don't swear, keep the faith, and just find out what is wrong (in this tutorial, failure is not an option!).

2.1.2 Vacuum Simulations

In order to complete the thermodynamic cycle shown in Figure 2, three TI simulations involving the same perturbations need to be carried out in vacuum. As for the aqueous situation, the TOL$\rightarrow$BNZ and PHE$\rightarrow$TOL cases are already completely handled in directories `TI_vacuum_TOL_BNZ` and `TI_vacuum_PHE_TOL`, respectively. The case BNZ$\rightarrow$PHE is left for you to handle in directory `TI_vacuum_PHE_BNZ`. Look inside the completed folders, e.g.

```
cd ~/ex5/TI_vacuum_PHE_TOL
```

When you consider a single small molecule in vacuum, it is often unwise to use MD with a Berendsen thermostat. There are two reasons for this. First, the Berendsen thermostat maintains the correct average temperature, but does not lead to rigorously correct temperature fluctuations. This is generally no problem for systems with hundreds of degrees of freedom or more, but can lead to artifacts when considering a single molecule with very few atoms. Second, small systems tend to be closer to being "harmonic", and being trapped in long-timescale quasi-harmonic oscillations may impair the sampling. We can solve the problem of representing a molecule in vacuum by considering an "ideal gas" consisting of many copies of the molecules at very large distances from each other, and all coupled to the same Berendsen thermostat (as applied in the simulations in water). Here, we will use an alternative approach: consider a single molecule, but replace the MD/Berendsen simulations by stochastic dynamics (SD) simulations. In SD simulations, random kicks and frictional forces are introduced, which has the effect of both randomizing (no long-timescale quasi-harmonic motions) and thermostating (temperature determined by the balance between random and frictional forces) the system.

In the GROMOS input file `TI.inp`, have a look at the `STOCHDYN` block. `NTSD` set to one switches on the stochastic dynamics, `CFRIC` sets the friction coefficient, and `TEMPSD` is the temperature of the stochastic bath. The magnitude of the stochastic kicks will be automatically determined from `CFRIC` and `TEMPSD` in such a way that we generate a canonical ensemble at temperature `TEMPSD`. Note that the value of `CFRIC` influences the dynamics of the system but not its thermodynamics (and thus the free-energy results). Here we arbitrarily took a value mimicking friction in water (the only constraint is that the stochastic coupling should be strong enough to kill possible quasi-harmonic motions in the molecule).

```
STOCHDYN
# NTSD    0,1 controls stochastic dynamics mode
#         0: do not do stochastic dynamics (default)
#         1: do stochastic dynamics
# NTFR    0..3 defines atomic friction coefficients gamma
#         0: set gamma to 0.0 (default)
#         1: set gamma to CFRIC
#         2: set gamma to CFRIC*GAM0
#         3: calculate gamma using subroutine FRIC (based on CFRIC)
# NSFR    > 0 recalculate gamma every NSFR steps
# NBREF   > 0 threshold number of neighbour atoms for a buried atom
# RCUTF   >= 0.0 interatomic distance considered when calculating gamma
# CFRIC   >= 0.0 global weighting for gamma
# TEMPSD  >= 0.0 temperature of stochastic bath
#
#     NTSD     NTFR     NSFR   NBREF  RCUTF    CFRIC    TEMPSD
         1        1        1       6    0.3     91.0     300.0
END
```

As in the Section 2.1.1, you are asked to set up the TI simulations for the perturbation PHE→BNZ in vacuum by yourself. So, go to the folder

```
cd ~/ex5/TI_vacuum_PHE_BNZ
```

Here, you have to repeat what you did for the corresponding TI simulation in water.[3] Pay

[3] Be smart before doing hard work... E.g. for the perturbation topology, do you think there will be any difference between the file for a perturbation in vacuum and the one for the perturbation in water?

special attention to the STOCHDYN and PERTURBATION blocks in the parameter input file TI.inp.

```
STOCHDYN
# NTSD    0,1 controls stochastic dynamics mode
#         0: do not do stochastic dynamics (default)
#         1: do stochastic dynamics
# NTFR    0..3 defines atomic friction coefficients gamma
#         0: set gamma to 0.0 (default)
#         1: set gamma to CFRIC
#         2: set gamma to CFRIC*GAM0
#         3: calculate gamma using subroutine FRIC (based on CFRIC)
# NSFR    > 0 recalculate gamma every NSFR steps
# NBREF   > 0 threshold number of neighbour atoms for a buried atom
# RCUTF   >= 0.0 interatomic distance considered when calculating gamma
# CFRIC   >= 0.0 global weighting for gamma
# TEMPSD  >= 0.0 temperature of stochastic bath
#
#     NTSD     NTFR     NSFR   NBREF  RCUTF    CFRIC    TEMPSD
         ?        ?        ?       ?      ?        ?         ?
END
PERTURBATION
#    NTG: 0..1 controls use of free-energy calculation.
#         0: no free-energy calculation (default)
#         1: calculate dH/dRLAM
#  NRDGL: 0,1 controls reading of initial value for RLAM.
#         0: use initial RLAM parameter from PERTURBATION block
#         1: read from configuration
#   RLAM: 0.0..1.0 initial value for lambda
#  DLAMT: >= 0.0 rate of lambda increase in time.
# ALPHLJ: >= 0.0 Lennard-Jones soft-core parameter
#  ALPHC: >= 0.0 Coulomb-RF soft-core parameter
#   NLAM: > 0 power dependence of lambda coupling
# NSCALE: 0..2 controls use of interaction scaling
#         0: no interaction scaling
#         1: interaction scaling
#         2: perturbation for all atom pairs with scaled
#            interactions. No perturbation for others.
#
#     NTG   NRDGL    RLAM   DLAMT
        ?       ?       ?       ?
#  ALPHLJ   ALPHC    NLAM  NSCALE
        ?       ?       ?       ?
END
```

After submitting, check whether your simulation is running properly on the cluster by

```
qstat
```

3 Week 2

3.1 Analyses

First of all, let's check whether all went smoothly with your simulations. Go in turn to the folders `TI_PHE_BNZ` and `TI_vacuum_PHE_BNZ`, and try

```
ls *
```

Did your simulations finish successfully? If not, try to find out why or/and contact the assistants (committing seppuku might be considered as a very last resort!). If all is well, you can move on to the analysis part.

In the TI method, the alchemical free-energy differences are calculated using Equation 2. During the TI simulations, $^{\partial\mathcal{H}}/_{\partial\lambda}$ was calculated and written out to the energy trajectory files (*.tre.gz) as well as to the GROMOS output files (`*.omd`). So the analysis involves the following steps: (*i*) extract the values of $^{\partial\mathcal{H}}/_{\partial\lambda}$ from the energy trajectory files and calculate the corresponding averages and statistical uncertainties using the GROMOS++ program `ene_ana` for all the simulations of different λ values; (*ii*) carry out the integral of the averages over λ using numerical quadrature (trapezoidal rule), and estimate the associated overall uncertainty.

We have already prepared a script `analyse.sh` which does these tasks automatically for all six simulations. Have a look at it.

```
cd ∼/ex5/scripts/
vi analyse.sh
```

Try to understand how the script works. For every line in `dirfile`, which is given as an argument to the script, it changes to the specified directory (first column in `dirfile`) and newly creates a directory `ana` (if it already exists, it will be deleted first (!)). In this directory `ana`, two other scripts are executed: Firstly `jdhdl.sh`, which analyses the trajectories of the simulations and secondly `integrate_plot.sh`, which integrates the average $^{\partial\mathcal{H}}/_{\partial\lambda}$ values and creates plots afterwards.

First have a look at `jdhdl.sh`. You should recognize a part running `ene_ana` for the second trajectory of each λ value and therefore does not take into account of the equilibration period and a part gathering the averages and uncertainties which are then written to a file called `ave_dhdl.dat`. Then, script `integrate_plot.py` takes over, carries out the integration of both the averages and uncertainties (trapezoidal quadrature), interpolates the data with cubic splines and integrates these afterwards, too. The results are then appended to the file `ave_dhdl.dat`. A plot of the average $^{\partial\mathcal{H}}/_{\partial\lambda}$ values plus their statistical uncertainties, its splines and the two running integrals as a function of λ is generated. If you have any questions about these scripts, do not hesitate to ask the assistants. What you still need to do is to put the name of the directory, the name of the corresponding simulation (`@sys` argument in the `mk_script_TI.arg` file in Section 2.1.1) and the desired title of the plots (no blanks, only underscores to separate words) to the file `∼/ex5/dir_list.txt`. A first line is given as an example. Please fill out the `TO_DO`s.

Now run the script using

```
cd ∼/ex5
./scripts/analyse.sh dir_list.txt
```

In every `ana` directory, you will see some outputs from the `ene_ana` program, and the plot of $^{\partial\mathcal{H}}/_{\partial\lambda}$ as a function of λ (average and estimated statistical uncertainties). Check the `ave_dhdl.dat` file.

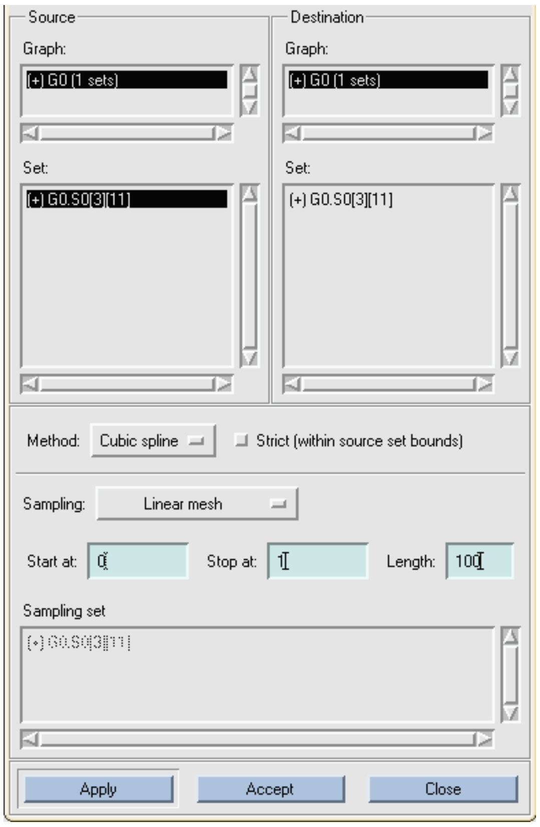
Here, you will find the final estimate for the free-energy difference, along with the estimated overall statistical uncertainty. What value do you get?

You could also do the integration using `xmgrace`, and it is actually a nice exercise to try this out.

```
xmgrace -settype xydy ave_dhdl.dat
```

On the `xmgrace` plot, go to the menu Data/Transformations/Integration, then click on Accept. What value do you get? Now you can see one red curve on the plot, which is the running integral.

We could also do a spline fitting of the curve before the integration. Click on the menu Data/Transformations/Interpolation/splines and set the parameters according to Figure 4.



Source
Graph: (+) G0 (1 sets)
Set: (+) G0.S0[3][11]

Destination
Graph: (+) G0 (1 sets)
Set: (+) G0.S0[3][11]

Method: Cubic spline
Strict (within source set bounds)

Sampling: Linear mesh
Start at: 0
Stop at: 1
Length: 100

Sampling set
(+) G0.S0[3][11]

Figure 4: `xmgrace` setting for spline fitting

Now, we can do the integration of the fitted (smooth) curve by clicking on the menu Data/-Transformations/Integration, selecting the set S1, and then clicking on Accept.What value do you get now?

Collect the results and statistical error estimates for the six free-energy changes considered. What about modifying the script `analyse.sh` so that it writes out the results of all six free-energy changes into one file? When you have all of them, you can compare the values against Table 1.

4 Report

4.1 Format

Please refer to the corresponding section of Exercise 1 for information on the goal and expected structure of the report.

4.2 Simulation Results

Answer the following questions:

A. Look at the GROMOS output files you generated (`*.omd` files). Which terms contribute to $^{\partial\mathcal{H}}/_{\partial\lambda}$? Are the contributing terms different in vacuum and in water?

B. Plot the average $^{\partial\mathcal{H}}/_{\partial\lambda}$ curves for the TI simulations, along with the running integral of the curve.

C. Compare the integration results obtained from two different approaches: (1) trapezoidal integration; (2) trapedzoidal integration after spline fitting. Comment on the differences.

   Note that from here on, use the free energy values obtained from trapezoidal integration.

D. Give all the six values of the alchemical free-energy differences (including estimated error bars) using the template shown in Figure 5.

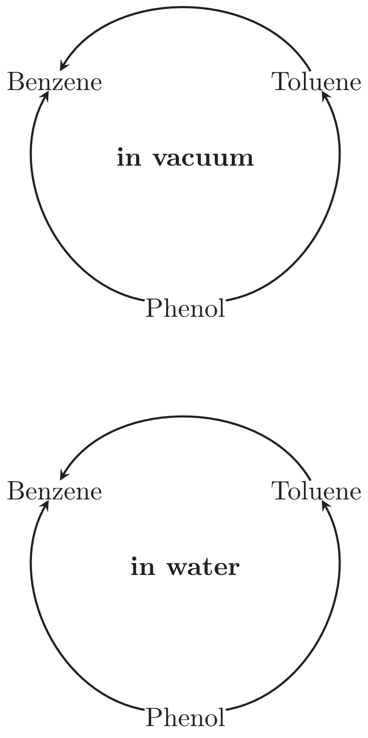


Figure 5: Template for free energy results.

E. Do the thermodynamic cycles in vacuum and in water close? (i.e. zero free-energy change around the cycle, within the error bars).

F. Calculate the hydration free energies of PHE and TOL relative to that of BNZ using the results of the PHE→BNZ and TOL→BNZ mutations and Eq. 1. Are the results compatible with the relative hydration free energy of PHE relative to TOL calculated based on the PHE→TOL mutations?

G. Compare these calculated relative hydration free energies with the experimental values provided in Table 1. Do the experimental differences make sense chemically? Are these differences reproduced in the calculated numbers?

4.3 Thinking Questions

Answer the following questions:

1. In this exercise, you calculated relative hydration free energies (e.g. of TOL relative to BNZ). What would you do if you needed to calculate an absolute hydration free energy (e.g. for BNZ)? Why would the corresponding calculation require more effort to get converged results (compared to that of a relative hydration free energy)?

2. Now imagine you want to calculate the absolute hydration free energies of 100 benzene derivatives in an efficient way (more efficient than 100 times the procedure of question!). How would you do it ?

3. When you calculate an absolute hydration free energy and want to compare with experiment, you have to worry about standard states. In simulations, the value characterizes the transfer of a molecule from a fixed point in vacuum to a fixed point in water. In experiment, standard data refers to the transfer from an ideal gas at 1 bar to an infinitely dilute solution extrapolated to 1 mol solute per kg solvent. Can you infer the form of the standard-state correction term you must apply to the simulated value before comparing with experiment? And can you explain why we don't need to worry about this when comparing relative hydration free energies to experiment?

4. For the simulations in vacuum, we used SD to overcome possible harmonic oscillation and improper thermostating problems. But in Exercise 1, we instead used a collection of molecules at large distances with MD/Berendsen. Could we also have applied a similar approach here?

5. In the three perturbations considered in this exercise, we never discussed the situation of improper dihedral angles and of torsional dihedral angles. In principle, the perturbations might affect the improper dihedral controlling the planarity at the C bearing the substituent (H, CH_3 or O). Why is it not the case? Similarly, PHE has a dihedral angle controlling the orientation of the OH group. What happens with this one when moving to BNZ or TOL?

6. We have carried out the perturbations in the reverse direction (TOL→BNZ, PHE→TOL and PHE→BNZ) because then, the A-state of the two latter mutations, PHE, is a "normal" molecule without dummy atom which is directly available as a building block in GROMOS. We could have benefitted from the same advantage by setting up the TOL→BNZ mutation as converting a "normal" TOL to a "normal" BNZ (i.e. no dummy atoms involved). (1) Can you explain why the final results in terms of hydration free energies would have been the same; (2) what about the thermodynamic cycles in vacuum and in water (question E above), would they still have closed exactly?

7. One might introduce a sort of “hierarchy” in the types of perturbations converting a molecule A into a molecule B: (1) the perturbation is chemically acceptable (i.e. it could have an experimental counterpart); (2) the perturbation is alchemical (i.e. it has no experimental counterpart) but involves no dummy atoms; (3) the perturbation is alchemical and requires using dummy atoms. What are the conditions on the molecules A and B for the perturbation to be in category (1), (2) or (3). Why is the result of a calculation using GROMOS still generally incorrect for category (1), i.e. only the difference of such results calculated in different environments (e.g. vacuum vs. water) is likely to be correct?

8. There would in principle be no need for the dummy atom to be attached to the same site as the atom to appear, or even to be attached to the molecule at all. Why is it still wise to covalently attach the dummy to the molecule, and to do this where the new atom will appear?

9. And here is a last one for the real hard-thinkers. In Section 1, we boldly stated that “the choice of intramolecular covalent interactions involving the dummy atom is arbitrary” and that it does not really matter “because all these effects cancel out when we compare the free energies calculated in water to those calculated in vacuum”. In fact, this is not entirely true. Can you explain why (think of the lecture discussion about free-energy components), think of a (pathological) counterexample, and formulate a more accurate condition of the allowed form of covalent interactions for the dummy atoms?

Appendix A. Experimental Data

Table 1: Experimental relative hydration free energies [1] of TOL and PHE relative to BNZ. .

Compound	Relative Hydration Free [kJ mol^{-1}]
BNZ	0.0
TOL	-0.1
PHE	-24.0

Appendix B. List of files

In the main exercise directory, you will find `ex5.pdf`, the digital version of the document you are reading and nine sub-directories:

topo/ Contains the topology files for phenol and methylbenzene.

coord/ Contains the coordinate files for phenol and methylbenzene in both vacuum and water box.

lib/ Contains the library file for the GROMOS++ program mk_script.

TI_TOL_BNZ/ Contains the TI simulation from toluene to benzene.

TI_PHE_TOL/ Contains the TI simulation from phenol to toluene.

TI_PHE_BNZ/ Contains some files required for the TI simulation from phenol to benzene.

TI_vacuum_TOL_BNZ/ Contains the TI simulation from toluene to benzene in vacuum.

TI_vacuum_PHE_TOL/ Contains the TI simulation from phenol to toluene in vacuum.

TI_vacuum_PHE_BNZ/ Contains some files required for the TI simulation from phenol to benzene in vacuum.

References

[1] Mobley, D.L., Bayly, C.I, Cooper, M.D., Shirts, M.R., and Dill, K.A. *J. Chem. Theory. Comput* **2009** 350-358

CSBMS: Exercise 6

Structure Refinement

Document version: 27.08.2019

Exercises week 1:	03.12. or 05.12.
Exercises week 2:	10.12. or 12.12.
Deadline for the report:	17.12 (shorter!).
Contact:	shuwang@phys.chem.ethz.ch
	Shuzhe Wang - HCI G238

Summary

In this sixth exercise, we will perform structure refinement of a peptide based on distance restraints derived from nuclear Overhauser enhancement (NOE) intensities observed in nuclear magnetic resonance (NMR) experiments. You will run three different molecular dynamics (MD) simulations of the peptide using the GROMOS program: a standard (unrestrained) simulation along with two simulations including NOE-derived distance restraints, applied either on an instantaneous or on a time-averaged basis. When analysing the simulation results, you will find out how the restraints and their application modus influence the simulation results, and the compatibility of the simulated ensemble with the experimental data. The main focus is on: (*i*) understanding the two different restraining methods, instantaneous restraints (IR) and time-averaged restraints (TAR); (*ii*) setting up a MD simulation with IR or TAR restraining; and (*iii*) analysing the simulation results in terms of agreement with the experimental NMR data, considering not only NOE-derived distances but also ^{3}J-coupling constants.

1 Introduction

Inspired by the beautiful structures displayed in biochemistry textbooks, we tend to think of biomolecules like proteins and nucleic acids as entities presenting a unique and well-defined three-dimensional structure. The truth is that biomolecules at equilibrium in solution exist as ensembles of many different conformers. In some cases, the ensemble is relatively “compact” around a given structure (at least in terms of backbone conformation), and the textbook picture may be reasonable. In many other cases, however, the ensemble is far less compact, and experimental observables represent averages over a given timescale and over many molecules that cannot be interpreted in terms of a unique structure. They must be explicitly related to a conformational ensemble.

As the number-one experimental technique to investigate the conformational properties of biomolecules in solution, nuclear magnetic resonance (NMR) spectroscopy does not escape this rule. The two most important observables extracted from NMR experiments in this context are nuclear Overhauser enhancement (NOE) intensities and three-bond scalar coupling constants (^{3}J-couplings). Given some interpretation model (e.g. the so-called model-free approach for NOEs [1] or a Karplus equation for ^{3}J-couplings [2]), NOE intensities can be related to the distance between specific hydrogen atom (proton) pairs in the molecule and ^{3}J-couplings to the torsional angle defined by four covalently-linked atoms. However, since the NMR signal accounts for an average over a sample of $\sim 10^{23}$ molecules and over a timescale on the order of the nanosecond, conformational averaging must be taken into account when interpreting this data. This means in practice that the NMR observables should be accounted for an ensemble of conformations at equilibrium rather than a single structure. And this observation has an important corollary: it is impossible to do structure determination (refinement) based on the NMR data alone, because the sole knowledge of a set of averages is never sufficient to determine an underlying distribution.

The solution to this problem is to introduce an assumption on the form of the conformational distribution. And a good way to formulate such an assumption is to say that the distribution should be Boltzmann-weighted in terms of an underlying potential energy function, i.e. to use a force field. In this case, molecular dynamics (MD) simulation based on the chosen force field represents the method of choice to generate the required conformational ensemble. This is why MD has become over the last three decades an essential tool in the area of NMR structure refinement.

In a dream world where force fields would be exact and sampling times infinite, we would merely need to run a simulation and calculate the averages corresponding to NMR observables. They would (let's dream on!) exactly match the experimental values and we would be done, i.e. able to provide the NMR spectroscopist with a full conformational ensemble accounting for her/his data. And indeed, in exceptional cases, it can be like this. But generally (now wake up!) things are not that simple, and the ensemble generated by a standard (so-called unrestrained) simulation only reproduces the NMR data with a limited accuracy. Because NMR spectroscopists tend to trust their data more than they trust their colleagues, they will usually think that you failed, and it won't do any good to your reputation. The remedy is to try to get the best from the two worlds: enforce the satisfaction of the average NMR observables in the simulated ensemble, but let the force field decide for all the rest, i.e. the form of the conformational distribution as well as the conformation of segments that are underdetermined by the NMR data. In practice, this is typically done by adding a biasing (restraining) potential-energy term V^{restr} to the physical force field V^{phys} during the simulation

$$V\left(\mathbf{r}(t)\right) = V^{\mathrm{phys}}\left(\mathbf{r}(t)\right) + V^{\mathrm{restr}}\left(\mathbf{r}(t)\right) \quad . \tag{1}$$

This term penalizes conformations for which the NMR observables deviate from their experimental reference values. A common way to define V^{restr} is to use a sum of harmonic restraining potentials

$$V^{\mathrm{restr}}\left(\mathbf{r}(t)\right) = \frac{1}{2}\sum_{i=1}^{N_{\mathrm{restr}}} k_i^{\mathrm{qr}}\left(q_i\left(\mathbf{r}(t)\right) - q_i^{\circ}\right)^2 \quad , \tag{2}$$

where k_i^{qr} is the restraint force constant, q_i is the current value of the observable in the simulation and q_i° its experimentally-inferred reference value. For NOEs, q_i is usually taken to be the distance between two protons, and q_i° is the corresponding value inferred from the experimental NOE intensity (typically precalculated using the so-called model-free approach [1]). In this case, one often uses a half-harmonic restraint instead, i.e. the restraint is only active when $q_i > q_i^{\circ}$, possibly also linearizing the dependence above a certain deviation q_i^1 (see Figure 1). For ^{3}J-couplings, q_i is usually taken to be the J-value itself (calculated on the flight using a given Karplus equation [2]), and q_i° is its corresponding experimental value.

One refers to the case where Equation 2 or its half-harmonic long-distance linearized analog (Figure 1) is applied strictly as instantaneous restraining (IR). The problem with IR is that it will enforce (when using sufficiently high force constants) the satisfaction of the experimentally-derived restraints in every single configuration along the simulation. But in fact, the restraints should rather apply to an average of the observable over the timescale of the NMR experiment, i.e. ideally on the order of the nanosecond. For example, it may happen that two NOE-derived distance restraints are incompatible within a single structure, but can still be satisfied along a

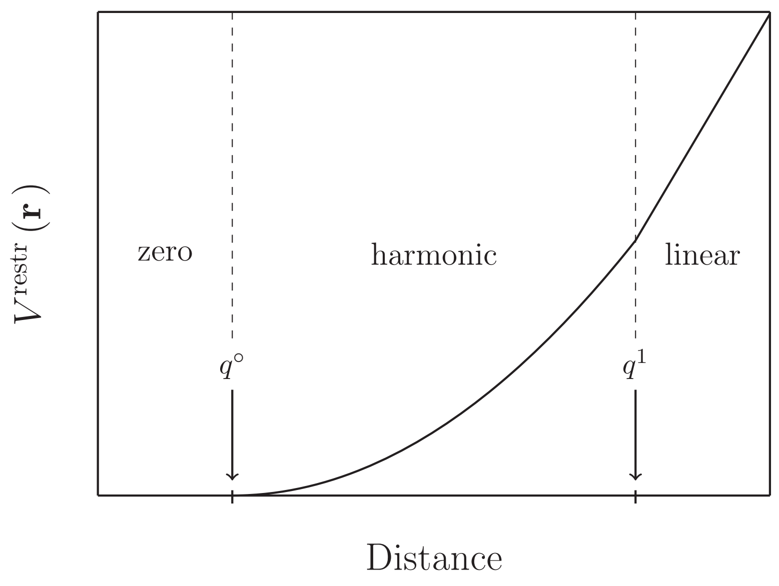


Figure 1: Potential energy term for atom-atom distance restraining.

trajectory where structures visited over a certain time period fulfil the first one while structures visited over another time period fulfil the second one.

To remedy this problem, Equation 2 or its half-harmonic long-distance linearized analog (Figure 1) can be applied with replacement of q_i by its (running) time average over a timescale. One refers to this approach as time-averaged restraining (TAR).[3, 4, 5, 6, 7, 8] The averaging over time is commonly achieved in practice using an exponential memory filter

$$\overline{q_i\left(\mathbf{r}(t)\right)} = \left\{ \frac{1}{\tau_{\mathrm{qr}}\left[1-\exp\left(^{-t}/_{\tau_{\mathrm{qr}}}\right)\right]} \int_0^t \exp\left(\frac{t'-t}{\tau_{\mathrm{qr}}}\right) q\left(\mathbf{r}(t')\right)^{-n} \mathrm{d}t' \right\}^{-1/n} , \qquad (3)$$

where τ_{qr} is the averaging time, typically set to 10-100 ps in ns simulations (better would be to set it to ns in multi-ns simulations if you can reach this timescale) and n is an exponent defining the form of the averaging. Based on the model-free approach, [1] the NOE intensity is (approximately) proportional to the inverse third power of the inter-proton distance for molecules in a slow tumbling regime, and to the inverse sixth power for molecules in a fast tumbling regime (where fast and slow refer to a comparison with the timescale of internal motions in the molecule). So, for example, Equation 3 will typically be applied with $n = 3$ for a large protein and $n = 6$ for a small peptide (with a gray zone in intermediate situations!).

In this exercise, we apply and compare the three refinement procedures (unrestrained MD, restrained MD with IR, restrained MD with TAR) in the context of a peptide in water, using GROMOS and the 54A7 force field version. The peptide considered is the C-terminal coiled-coil trigger sequence of the yeast transcriptional activator GCN4, denoted GCN4p16-31. [9] If you are not a biochemistry freak, you might just content yourself with the information that it is a hexadecapeptide with the sequence Asn-Tyr-His-Leu-Glu-Asn-Glu-Val-Ala-Arg-Leu-Lys-Lys-Leu-Val-Gly. This peptide has been studied experimentally [9] by NMR in aqueous solution at 278 K, 1 bar and pH 7.5. The investigation resulted in the determination (and full assignment in terms of the involved atoms) of 179 NOE intensities and 15 ^{3}J-coupling constants. The model-free approach [1] was used to convert the 172 (of 179) NOE intensities into corresponding estimated inter-proton distances. Using their own non-GROMOS modelling tools,[1] the exper-

[1]XPLOR refinement with simulated annealing, also including some information from measured chemical shifts.

imentalists used this dataset to refine a bundle of 20 structures that they deposited into the PDB under the entry code 2OVN. This peptide was already investigated in the group in terms of NMR-based MD refinement with GROMOS, see Reference [10]. The present exercise follows more or less the line of this study. Just as the authors of this work, you will use one of the 2OVN structures as starting configuration, and be provided with the files containing the 179 NOE-derived distances and 15 ^{3}J-coupling constants in the usual XPLOR format employed by spectroscopists.

This exercise is divided in two parts. In the first one, we prepare and understand the NOE files and submit the calculations. The second part, though is destined to the analyses of the restraints.

2 Simulation Set-up

Since you are already familiar with the set-up of GROMOS simulations (making the topology file, generating the initial coordinates, solvating the solute into a solvent box, energy minimizing and equilibrating the system) from the previous exercises, these operations have already been carried out for you. From there on, you need to set up and run three MD simulations: (*i*) a standard unrestrained MD simulation; (*ii*) a MD simulation with instantaneous restraints (IR); and (*iii*) a MD simulation with time-averaged restraints (TAR).

2.1 Getting Started

To get started, you have to login to realbeaver and copy a directory that was already prepared for you by typing

```
cp -r /usr/local/CSBMS/ex6 ∼/ex6
```

There should be 9 subdirectories.

`forcefield/` here, the GROMOS 54A7 force-field files are stored; two special terminal building blocks named `ACE` and `CONH2` have been added to the `mtb` file for representing the termini of your peptide;

`topo/` here you can find the topology file of the peptide, `peptide.top`, which is already prepared for you;

`coord/` here, you can find the coordinate file of the peptide in vacuum, `peptide_gch.cnf`, which is already prepared for you;

`min/` here, the energy minimization (EM) of the initial coordinates in vacuum has been performed, resulting in the coordinate file `peptide_min.cnf`;

`box/` here, the peptide was solvated into a box of water, and the EM of the box was performed, resulting in the coordinate file `peptide_H2O.cnf`;

`eq/` here, the equilibration/thermalization of the coordinates (solute+solvent) using MD has been performed (to 278 K and 1 bar within 100 ps), resulting in the equilibrated initial configuration file `eq_peptide_5.cnf`;

In addition, the structure was refined reling on MC calculations performed at very high temperature with a simplified force field and without explicit consideration of the solvent degrees of freedom.

`prep_noe/` here, you will prepare the NOE distance-restraints file, to be used for carrying out the restrained MD simulations and for performing the analysis in terms of NOE-derived distances;

`md/` here are 3 subdirectories that you will use to perform your own simulations:

- `unrestrained/` will be used for the MD simulation without restraints;
- `NOE_IR/` will be used for the MD simulation with IR; and
- `NOE_TAR/` will be used for the MD simulation TAR.

`ana/` here, you will perform the analysis of the simulation results; there are 4 subdirectories:

- `rmsd/` will be used for root-mean-square-deviation (RMSD) calculation.
- `rmsf/` will be used for root-mean-square fluctuations (RMSF) calculation.
- `noe/` will be used for comparing the simulation results with experimental NOE-derived distances.
- `jval/` will be used for comparing the simulation results with experimental ^{3}J-coupling values.

The work for the procedures corresponding to directories 1-6 has already been performed for you, and all the files therein have their final form. The setup was performed exactly as described in Reference [10], and we can just quote the article:

"The GCN4p16-31 peptide comprises the sequence: Ac-16Asn-17Tyr-18His-19Leu-20Glu-21Asn-22Glu-23Val-24Ala-25Arg-26Leu-27Lys-28Lys-29Leu-30Val-31Gly-NH2. The His residue is protonated at NE2, the Arg and Lys side chains are protonated with charge +e.[2] Coordinates of the first model structure of the NMR set of structures (PDB entry 2OVN) were taken as the starting coordinates for MD simulations. The last residue (32Glu) of the model structure was removed because it was not present in the NMR experiment.[3] After steepest descent energy minimisation, the structure was solvated in a rectangular box of approximately 3000 pre-equilibrated simple point charge (SPC) water molecules with a minimal solute-to-wall distance of 1.0 nm. The system was relaxed by performing a steepest-descent energy minimisation with harmonic positional restraints on all solute atoms (force constant $2.5 \cdot 10^4$ kJ mol^{-1} nm^{-2}) followed by a 100 ps long equilibration, in which the positional restraints were gradually released reducing the force constant to 0.0 kJ mol^{-1} nm^{-2} and the temperature was raised from 60 to 278 K. The initial atomic velocities were taken from a Maxwell distribution at 60 K."

Here are a few additional questions that you can easily answer by reading the text above and looking further into the files if needed (if time is short, save these questions for later and first move on to the setup of your own simulations)

A. (1) Why are the residues numbered 16-31 (rather than 1-16) in the above piece of text? (2) What do the "Ac-" and the "-NH2" at the start and end of the sequence mean exactly, and are the termini charged or neutral? (3) What is the overall charge of the peptide in the simulations? (4) What pH range does this correspond to? (5) Were counter-ions added? (6) Was the initial configuration equilibrated at constant volume (if yes, what volume) or at constant pressure (if yes, what pressure)?

[2]Although the text does not say it explicitly, the Glu residues were taken to be deprotonated with charge -1e.

[3]There are 17 residues (Asn16 - Glu32) in pdb files of 2OVN, but only 16 residues (Asn16 - Gly31) were present in Reference [9]. Obviously, the 2OVN structures refer to an extended peptide, but the experimental reference is unclear about this.

2.2 Preparing NOE-derived distance restraints file

To set-up an MD simulation with NOE-derived distance restraints (either IR or TAR), you need a GROMOS NOE distance-restraints file which can be derived from the file provided by the experimentalists, `NOElist.exp`.

So, please go into the folder ∼`/ex6/prep_noe` and have a look at the `NOElist.exp` file

```
TITLE
GCN4p16-31:
NOElist_02_02_2009
END
NOESPEC
   1       16  HN  15  HA  1.8 0.0 1.7
   2       16  HN  15  HN  1.8 0.0 3.2
   3       15  HN  14  HA  1.8 0.0 1.7
   4       15  HN  14  HN  1.8 0.0 3.2
   5       15  HN  13  HN  1.8 0.0 1.7
   6       14  HN  13  HA  1.8 0.0 1.7
...
(continues)
...
 174       1  HA   4  HD@@ 1.8 0.0 3.2
 175       8  HA   11 HB@  1.8 0.0 3.2
 176       8  HA   11 HG   1.8 0.0 3.2
 177       8  HA   7  HG@  1.8 0.0 3.2
 178       8  HG@  7  HG@  1.8 0.0 3.2
 179       8  HG@@ 5  HA   1.8 0.0 3.2
END
```

This file results from the processing of the NOE intensities using the program XPLOR, slightly altered by adding a `TITLE` block and relocating the distance information within a `NOESPEC` block. In the latter block, each line contains the sequential NOE number, the information on the first proton I involved in this NOE (residue number from 1 to 16 and XPLOR atom name) and the corresponding information on the second proton J. The last three entries stand for the distance upper bounds derived from the NOE intensities. There are three rules to parse the last three columns in the XPLOR file: (*i*) Upper bound = first column; (*ii*) Upper bound = first + third column; and (*iii*) Upper bound = first - second column. This file uses the second rule. The atoms names involving an "@" refer to pseudo-sites (see below).

Converting the XPLOR list into a corresponding GROMOS file is not entirely trivial, which is why GROMOS++ includes a program `prep_noe` to do this automatically for you. In the normal case, you just have to map the XPLOR atom sequence information on the protons I and J (residue number and XPLOR atom name) into a GROMOS atom sequence information (atom number in the GROMOS topology file). But there are two types of special cases.

The first special case arises because the GROMOS force field uses a united-atom representation, so that aliphatic hydrogen atoms are not present in the GROMOS topology file. In this case, one uses a so-called virtual atom, i.e. the position of the hydrogen "where it would be" is defined relative to the positions of 3-4 other atoms as illustrated in Table 1 (cases `ICDR`=1, 2 or 4). In the MD simulation, the distance restraint involving this virtual site will generate a force that will be redistributed onto the defining atoms. In effect, this is equivalent to reconstructing the coordinates of the virtual site at each time step, and assuming that this site is rigidly fixed to the defining atoms (constrained geometry). Because the distance calculation involves the position of the hydrogen "where it would be", there is no distance correction to apply in this case.

The second special case arises from the fact that multiple protons can be associated with the same NOE intensity, because they are topologically indistinguishable (e.g. three hydrogens of a methyl group, two hydrogens of a methylene group in a non-chiral molecule) or quasi-indistinguishable (e.g. pro-R and pro-S hydrogens of a prochiral methylene group; these could in principle be distinguished, in which case one speaks of a stereospecific assignment, but often cannot be resolved in practice). Rapid conformational equilibria may also cause two protons to become indistinguishable on the NMR timescale. In the XPLOR file, this is indicated by an "@" (equivalent protons on one carbon, e.g. methyl or methylene group), "@@" (equivalent proton on two carbons, e.g. isopropyl group) or "@@@" (equivalent proton on three carbons, e.g. tert-butyl group). In this case, GROMOS uses the so-called center-average approach. One again considers a virtual site, called in this case a pseudo atom. Its position is also defined relative to the positions of 2-4 other atoms as illustrated in Table 1 (cases ICDR=3, 5, 6 or 7), and corresponds to the mean position of all the equivalent protons.

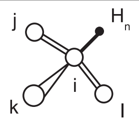
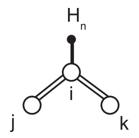
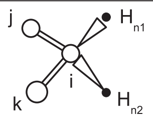
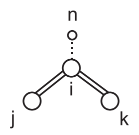
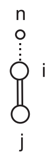
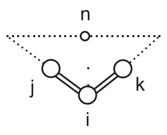
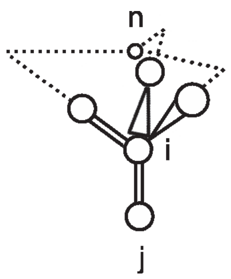
In this case, however, the position of the site is not that of the individual hydrogens, and a correction must be applied to the NOE-derived reference distance (an increase called pseudo-atom correction; because the distance from a given proton to the most remote proton of the group may actually be longer than the distance to the pseudo atom)

Now what you know what `prep_noe` does, let's use the program. You will need the input files `noecor.gromosXX`, `noelib.54a7` and `prepnoe.arg`. And the procedure will create the following output files `prep_noe.out`, `noe.filter` and `noe.dsr`. First have a look at the `prep_noe.arg` file

```
@topo ../topo/peptide.top
@title GCN4-p1-32GLUU 54A7
@filter 10000
@factor 10
@noe ./NOElist.exp
@lib ./noelib.54a7
@parsetype 2
@action
@correction ./noecor.gromosXX
```

The topology file is given by the argument `@topo`. With `@title` one specifies the title in the `prep_noe` output. The `@filter` argument determines the upper limit of the NOE distance bounds which should be considered. By setting `@filter` to very large value (10000 nm), we make sure that all distances will be taken into account. The NMR experimental data from XPLOR is reported in Angstrom (Å) (e.g. 1.8 Å), whereas using the GROMOS program the standard length unit is nanometer (nm) (e.g. 0.18 nm). Thus we have to give the program a `@factor` by which it has to scale (divide) all the NOE distance bounds (e.g. 1.8/10 = 0.18). The argument `@noe` specifies the input file `NOElist.exp`. The program will assign GROMOS atom sequence numbers based on the XPLOR information for residue numbers and XPLOR atom names. Thus we have to specify a library file which recognizes atom names specific to XPLOR. This is done with the argument `@lib`. There are three ways to parse the last three columns in the XPLOR file. This is selected using the `@parsetype` argument:

Table 1: Virtual and pseudo hydrogen atoms, correction term on distance restraint and geometric codes.

Group	Configuration	Atom type	Correction term on distance restraint	Geometric code ICDR1 or ICDR2
CH1 (aliphatic)		virtual	0.00	1
CH1 (aromatic)		virtual	0.00	2
CH2 (stereospecific)		virtual	0.00	4
CH2 (non-stereospecific)		pseudo	0.09	3
CH3		pseudo	0.10	5
two CH3 (non-stereospecific, Val, Leu)		pseudo	0.22	6
three CH3 (non-stereospecific, t-butyl)		pseudo	0.23	7

@parsetype=1: Upper bound = first number
@parsetype=2: Upper bound = first + third number (most common, default)
@parsetype=3: Upper bound = first - second number (commonly the lower bound)

If you set `@parsetype` to either 2 or 3, you have to also use an argument called `@action`, which applies a correction to the reference distance, the correction is either added (default) for `@parsetype` 2, or subtracted for `@parsetype` 3. Here, we use `@parsetype` 2. The argument `@correction` specifies a file containing the information about types of correction that should be taken into account by `prep_noe`.

Now you can finalize the `prep_noe.arg` for missing entries, and run the program

```
prep_noe @f prep_noe.arg > prep_noe.out
```

This produces three files, namely `prep_noe.out` (the redirected standard output), `noe.filter` and `noe.dsr`. Have a look at `noe.dsr` file, which will be our GROMOS NOE-derived distance restraints file for MD simulations in the next step.

```
TITLE
NOE distance restraints file for: GCN4-p1-32GLUU 54A7
END
DISTANCERESSPEC
#     DISH      DISC
       0.1     0.153
#IDR1 JDR1 KDR1 LDR1 ICDR IDR2 JDR2 KDR2 LDR2 ICDR        R0         W0      NRAH
#
# 1  1    16  GLY    H   HN   1    15  VAL   CA   HA
  171    0    0    0    0  164  162  165  168    1      0.35          1         1
# 2  1    16  GLY    H   HN   1    15  VAL    H   HN
  171    0    0    0    0  163    0    0    0    0       0.5          1         1
# 3  1    15  VAL    H   HN   1    14  LEU   CA   HA
  163    0    0    0    0  155  153  156  160    1      0.35          1         1
...
(continues)
...
# 178  1     8  VAL   CB  HG@   1     7  GLU   CG  HG@
   90   91   92    0    6   81   80   82    0    3      0.81          1         1
# 179  1     8  VAL   CB HG@@   1     5  GLU   CA   HA
   90   91   92    0    6   58   56   59   64    1      0.72          1         1
END
```

`DISH` carbon-hydrogen distance, used for virtual and pseudo atom geometries `ICDR` = 1-6;

`DISC` carbon-carbon distance, used for pseudo atom geometry ICDR = 6;

`IDR1`, `JDR1`, `KDR1`, `LDR1` atom sequence numbers of the real atoms defining the geometric position of the first atom of a distance restraint pair;

`ICDR1` geometric code defining the position of the first atom of a distance restraint pair [-2, -1, ..., 7]

`IDR2`, `JDR2`, `KDR2`, `LDR2` atom sequence numbers of the real atoms defining the geometric position of the second atom of a distance restraint pair;

`ICDR2` geometric code defining the position of the second atom of a distance restraint pair [-2, -1, ..., 7];

R0 in case of a full-harmonic distance restraint (NRAH = 0), R0 is the minimum-energy distance; in case of an attractive or repulsive half-harmonic restraint (NRAH = ±1), R0 is the upper or lower bound, respectively, beyond which the restraining forces become non-zero. When using distance restraints for NMR-NOE distance restraining, pseudo-atom corrections should already be included in R0;

W0 individual distance restraint weight factor, by which the distance restraint interaction term may be multiplied.

NRAH type of distance restraint; if NRAH = −1, a half-harmonic repulsive distance restraint is applied; if NRAH = 0, a full harmonic distance restraint is applied; if NRAH = 1, a half-harmonic attractive distance restraint is applied.

In this file, W0 is set to 1, but you can also set it to other value. When W0 is not 1, NTDIR and CDIR should be set differently (see Sections 2.3.2 and 2.3.3).

Here are a few additional questions that you can easily answer by looking at the files and having well understood the above procedure (if time is short, save this question for later and first move on to the setup of your own simulations)

B. Compare specific lines in the XPLOR file (NOElist.exp) and the GROMOS file (noe.dsr), and explain how the GROMOS line can be deduced from the XPLOR line, including the calculation of the reference distance R0 (for this, you also need to look at a GROMOS coordinate file e.g. coord/peptide_gch.cnf to relate the GROMOS atom numbers to specific atoms). Do this for the restraints Nr. 2, 46, 64 and 179.

2.3 Setting up MD simulations

In the ~/ex6/md directory, there are 3 subdirectories: unrestrained, NOE_IR and NOE_TAR, for three different MD simulations, which will each be 10 ns long.

2.3.1 Unrestrained MD Simulation

Go into the unrestrained directory and have a look at the md input file md_unrestrained.imd. You will find that some parameters are set to #TO_DO for you to fill. In the MULTIBATH block, the temperature should be set to 278 K for both solute and solvent. Because we don't need coordinate trajectory of the solvent, we will set NTWX to a *negative* value in the WRITETRAJ block to save disk space.

After finalizing the md_unrestrained.imd file, you can make the scripts for the MD simulation. Have a look at jmk_script.sh. You will find one #TO_DO for you to fill. You need to specify the starting configuration file there. The file is the *fifth* .cnf in the equilibration directory. Have a look for it! While filling the name in try to use the NAME variable defined in the top of the script.

```
./jmk_script.sh
```

To run the calculation use the jsubmit.sh script. But first, have a look so you know what it does! Then type

```
./jsubmit.sh
```

Don't forget to check with qstat to see if the simulation is running happily.

2.3.2 MD Simulation with Instantaneous Restraints (IR)

Go into the ∼/ex6/md/NOE_IR folder and have a look at the md input file md_NOE_IR.imd. Here, you will find a new DISTANCERES block which (obviously) did not appear in the md_unrestrained.imd file. Some parameters in this block are set to #TO_DO for you to fill in. The block specifies

```
DISTANCERES
#   NTDIR -2..2 controls distance restraining
#         0: no distrance restraining (default)
#         1: instantaneous, using force constant CDIR
#         2: instantaneous, using force constant CDIR x W0
#        -1: time-averaged, using force constant CDIR
#        -2: time-averaged, using force constant CDIR x W0
#  NTDIRA 0,1 controls values for initial distance averages
#         0: generate initial averages
#         1: read from configuration
#    CDIR >= 0.0 force constant for distance restraining
#    DIR0 >= 0.0 distance offset in restraining function
#  TAUDIR >  0.0 coupling time for time averaging
# FORCESCALE 0..2 controls approximation of force scaling
#         0: approximate d<r>/dr = 1
#         1: approximate d<r>/dr = (1.0 - exp(-Dt/tau))
#         2: use d<r>/dr = (1.0 - exp(-Dt/tau))*(<r>/r)^4
#    VDIR 0,1 controls contribution to virial
#         0: no contribution
#         1: distance restraints contribute to virial
#  NTWDIR >= 0 write every NTWDIRth step dist. restr. information to external file
#    NTDIR    NTDIRA      CDIR      DIR0    TAUDIR  FORCESCALE  VDIR   NTWDIR
         1         0      2000         1         0           0     0        0
END
```

Here, we want to use instantaneous restraining (IR) using a force constant of 2000 kJ mol^{-1} nm^{-2}. The distance offset is the reference distance beyond which we will linearize the restraining potential (Figure 1). It is 1 nm. Because we don't use time-averaged restraints, NTDIRA and TAUDIR are irrelevant (we can set them to 0). Since we don't need the detailed dist.restr information in an external file, we can set NTWDIR to 0. VDIR controls addition of distance restraining to virial in a case of intermolecular restraining and FORCESCALE controls approximation of force scaling, we can set both parameters to 0.

After finalizing the md_NOE_IR.imd file, you can make the scripts for the MD simulation. Have a look at jmk_script.sh You will find one blank (#TO_DO) for you to fill. You need to specify the distance restraint file will be used in this MD simulation. This distance restraints file noe.dsr you have already generated in Section 2.2. After finalizing the file jmk_script.sh, you can now make the GROMOS scripts by typing

```
./jmk_script.sh
```

If there is no error message, you can submit the IR-restrained MD simulation again with jsubmit.sh script. First, compare the script with the ../unrestrained/jsubmit.sh script. What is different? Then just type

```
./jsubmit.sh
```

Don't forget to check with qstat to see if the simulation is running happily.

2.3.3 MD Simulation with Time-Averaged Restrains (TAR)

Go into the folder ~`/ex6/md/NOE_TAR` and have a look at the md input file `md_NOE_TAR.imd`. In the `DISTANCERES` block you will again find some parameters are set to `#TO_DO` for you to fill. Here, we want to use time-averaged restraining (TAR) using a force constant of 6000 kJ mol^{-1} nm^{-2}. The coupling time for time averaging will be set to 20 ps. Distance offset in the restraining function will be 1 nm as in IR. Because the TAR method relies on distance averages calculated over the entire past of the simulation (Equation 3),[4] we should not reset them to zero at every new simulation job. Instead, current averages will be written at the end of each job to the final configuration file, and they should be read from this file as initial values. For this reason, we set `NTDIRA` to 1 (and will have to watch out that for the first job, we should reset the value to 0 so that the averages are initialized to zero; see below).

After completing the `md_NOE_TAR.imd` file, you can make the scripts for the MD simulation. Have a look at `jmk_script.sh`. The initial coordinates file is not same as we used in unrestrained and IR-restrained MD simulations. There, we started the simulations from the structure after equilibration. Here, we will start from the structure after 1 ns IR-restrained MD simulation, because TAR-restrained MD simulation requests more reasonable (less violations) starting structure. Since 1 ns IR-restrained MD simulation may take 10 hours, the structure after 1 ns IR-restrained MD simulation `md_NOE_IR_10.cnf` is provided to you. You just need to specify it on the place of `#TO_DO`. You can now make the GROMOS scripts by typing

```
./jmk_script.sh
```

But don't run the simulation yet!

In the `md_NOE_TAR.imd` file, we set the parameter `NTDIRA` to 1 so that initial distance averages are read from the starting configuration file at each job. But for the first job, there is no `DISTESEXPAVE` block in the initial configuration file `md_NOE_IR_10.cnf`. So, we need to reset `NTDIRA` to 0 for this first job in the corresponding input file `md_NOE_TAR_1.imd`.

After adapting file `md_NOE_TAR_1.imd`, you can now run your simulation jobs again with `jsubmit.sh` script. Now, you should know what is going to be the difference in comparison to the `../unrestrained/jsubmit.sh` and `../NOE_IR/jsubmit.sh` scripts. Check if you are right. Then just type

```
./jsubmit.sh
```

Don't forget to check with `qstat` to see if the simulation is running happily.

3 Analyses

3.1 Analyses

For each of the three MD simulations, four analyses will be performed: (*i*) atom-positional root-mean-square-deviation (RMSD); (*ii*) atom-positional root-mean-square-fluctuations (RMSF); (*iii*) NOE distance upper-bound violations (relative to the experimental data); and (*iv*) ^{3}J-coupling constant deviations (relative to the experimental data).

Because reaching the full 10 ns may take more than one week of computer time depending on the load of the beaver cluster, we provide you the whole 10 ns trajectory for analysis in case your own simulations did not complete yet.

[4]We should use ideally use $\langle r^{-6}\rangle^{-1/6}$ averaging for peptide, but $\langle r^{-3}\rangle^{-1/3}$ is hard coded in GROMOS program of distance restrained MD simulation. It does not matter much.

If your runs did not reach 10 ns, please, perform all analyses on the 10 ns simulation provided to you. In addition, do also the first (RMSD) analysis for your own three unfinished simulations. By comparing the curves, you can check that our 10 ns simulations are really the extension of what you would have obtained with yours (i.e. that you did no mistake in your setup!).

To make a link to the 10 ns trajectory, please go to the appropriate directory (∼`/ex6`) and type

```
ln -s /usr/local/CSBMS_assist/ex6/md ~/ex6/md_10ns
```

3.1.1 RMSD

Go to the directory for RMSD analysis ∼`/ex6/ana/rmsd`. The trajectories of all MD simulations should be analysed in terms of the time series of the RMSD relative to the energy-minimized initial structure. The RMSD values are to be calculated for the heavy atoms of the backbone (C_α, N, C) of non-terminal residues using the same atoms to perform the roto-translational least-squares fit superposition of the successive structures onto the reference one. A script `jrmsd_unrestrained.sh` has been provided to calculate RMSD time series for the unrestrained MD simulation. Have a look at it and then execute it

```
./jrmsd_unrestrained.sh
```

Then you have to make two copies of this script and modify them to calculate RMSD time series for the MD simulations with IR-restraining or TAR-restraining, i.e. do

```
cp jrmsd_unrestrained.sh jrmsd_NOE_IR.sh
cp jrmsd_unrestrained.sh jrmsd_NOE_TAR.sh
```

Then edit and adjust the two files, and then execute them

```
./jrmsd_NOE_IR.sh
./jrmsd_NOE_TAR.sh
```

You have to make three more copies of the script to analyse your own simulations if they were incomplete (<10ns). The template script `jrmsd_unrestrained_my.sh` is provided to you.

After you get the output files of the three RMSD calculations, you can plot them in one figure using `xmgrace` and compare them. You can also perform this using the script `jxmgrace.sh`

If the RMSD curves for your runs (<10ns) match the start of those for the assistant's runs (10ns), you set everything correctly last week, congratulation! If it doesn't match try to find the mistake in the setup and include it in your report. In both cases you can continue with the assistant's runs.

C. Print the figure with the RMSD time series and comment on the differences.

3.1.2 RMSF

The trajectories of all MD simulations should be analysed in terms of RMSF values as a function of the atom sequence number for the full 10 ns time period. The RMSF values should be calculated for the heavy atoms of the backbone (C_α, N, C) of all residues using the same atoms to perform the roto-translational least-squares fit superposition of the successive structures onto the reference one. A script `jrmsf_unrestrained.sh` has been provided to calculate RMSF values for the unrestrained MD simulation. Have a look at it and then execute it

```
./jrmsf_unrestrained.sh
```

Then you have to make two copies of this script and modify them to calculate RMSD time series for the MD simulations with IR-restraining or TAR-restraining, i.e. do

```
cp jrmsf_unrestrained.sh jrmsf_NOE_IR.sh
cp jrmsf_unrestrained.sh jrmsf_NOE_TAR.sh
```

Then edit and adjust the two files, and then execute them

```
./jrmsf_NOE_IR.sh
./jrmsf_NOE_TAR.sh
```

After you get the output files of the three RMSF calculations, you can plot them in one figure using `xmgrace` and compare them. You can also perform this using the script `jxmgrace.sh`

D. Print the figure of RMSF as a function of the atom sequence number and comment on the differences.

3.1.3 NOE Distance Upper-Bound Violations

To compare simulation results with the experimental NOE data, we will calculate violations relative to the NOE-derived distances, for each of the 172 proton pairs monitored experimentally and based on average distances calculated over the full 10 ns trajectories. The NOE-derived reference distances are already listed in the output file `prep_noe.out` we generated earlier with `prep_noe` (see Section 2.2). We will use the program `noe` to calculate the violations based on this information and the coordinate trajectories. The program will calculate the average distance in the 10 ns simulations according to $\langle r^{-p}\rangle^{-1/p}$ for values of $p = 1, 3, 6$. It will then calculate the deviations of these distances from the NOE-derived reference distances, q°. Note that a positive violation (too long average distance in the simulation) is considered to be a discrepancy relative to experiment, but a negative violation (too short average distance in the simulation) is not, because experimental factors (e.g. spin diffusion) may be invoked to explain a NOE intensity that is lower than "expected". If we want to characterize the level of discrepancy between simulation and experiment by a single number, we can use the average positive violation, i.e. the sum of all positive violations divided by the total number of NOE distances considered in the analysis. In practice, however, it is more informative to plot the individual violations as a function of the NOE sequence number. The output of the program `noe` is very detailed and not highly "human readable", so that it will also be post-processed using the `post_noe` program. Go to the NOE violation analysis directory

```
~/ex6/ana/noe
```

To run `noe` and `post_noe` you will need the input files `noe.arg`, `post_noe.arg`, `prep_noe.out` and `noe.filter`. And the procedure will create the following output files `noe.out` and `post_noe.out`. For the unrestrained simulation, the scripts `jnoe_unrestrained.sh` and `jpost_noe_unrestrained.sh` are already provided to you. First, have a look at the `jnoe_unrestrained.sh` file

```
...
@topo ../../topo/peptide.top
@pbc r
@noe #TO_DO
@traj
...
```

Topology and periodic boundary conditions are given by @topo and @pbc, respectively. The @noe argument points to the output file prep_noe.out we generated earlier with prep_noe (see Section 2.2). With the argument @traj we specify the trajectories which will be analyzed. Now, execute the script jnoe_unrestrained.sh

```
./jnoe_unrestrained.sh
```

It produces the file noe_unrestrained.out that contains the NOE violation information in a "computer readable" format. To make it more "human readable" you have to process it. Have a look at the jpost_noe_unrestrained.sh file

```
...
@topo ../../topo/peptide.top
@noe ../../prep_noe/prep_noe.out
@noeoutput #TO_DO
@filter ../../prep_noe/noe.filter
@averaging 6
...
```

The topology is given by @topo. The three following arguments @noe, @noeoutput, and @filter are output files of the prep_noe (see Section 2.2) and noe (see above) programs. The @noe and @noeoutput arguments serve here as input files, whereas the @filter filters the NOE distances which we will not use. When you specify @noeoutput use a variable which is defined in the beginning of the script. It will save you some time later. With @averaging 6 one specifies which averaging should be used. Here we are using the r^{-6} averaging, as appropriate for a peptide where tumbling is fast relative to internal motions.

Now, execute the script jpost_noe_unrestrained.sh

```
./jpost_noe_unrestrained.sh
```

It produces the file post_noe_unrestrained.out. Try to read it and understand each column.

Now, you have to repeat above steps to calculate the violations for the MD simulations with IR-restraining or TAR-restraining, i.e. do

```
cp jnoe_unrestrained.sh jnoe_NOE_IR.sh
cp jnoe_unrestrained.sh jnoe_NOE_TAR.sh

cp jpost_noe_unrestrained.sh jpost_noe_NOE_IR.sh
cp jpost_noe_unrestrained.sh jpost_noe_NOE_TAR.sh
```

Then edit and adjust the two pairs of files, and then execute them

```
./jnoe_NOE_IR.sh
./jpost_noe_NOE_IR.sh
./jnoe_NOE_TAR.sh
./jpost_noe_NOE_TAR.sh
```

Plot the NOE distance upper-bound violations of the three MD simulations using script jxmgrace.sh.

```
./jxmgrace.sh
```

E. Print the figure of NOE distance upper bound violations and comment on the difference.

3.1.4 ^{3}J-coupling constants

Finally, we can compare simulation results with the experimental ^{3}J-coupling. For this, we will use the program `jval`. It uses a so-called Karplus relation [2] to relate the ^{3}J-coupling constant to the local molecular structure or torsional angle.

$$^3\mathrm{J}\left(\mathrm{H_N}, \mathrm{H_{C_\alpha}}\right) = a\cos^2\phi + b\cos\phi + c \quad , \tag{4}$$

where ϕ is the dihedral angle between the planes defined by the atoms (H, N, C_α) and the atoms (N, C_α, H_α). There are many different parameterizations for Karplus equations (see Table 2), inferred empirically by correlating molecular structures (e.g. from X-ray) to measured J-values (from NMR), and possibly also using some QM calculations, for different types of dihedral angles in different sets of compounds. In our calculations, we use parameters a, b, c equal to 6.4 Hz, -1.4 Hz and 1.9 Hz, respectively [11].

Before we can calculate the ^{3}J-coupling constants, we need to define the torsional angle(s) that will be used for the calculation. For this we need a ^{3}J-coupling constant restraints file. In this exercise, the ^{3}J-coupling constant restraints file `p16-31_jval.dat` is already prepared for you (see below)

```
TITLE
K278.
J-coupling constant restraints specification file for new GROMOSXX.
jozi, Jan. 2009
END
JVALRESSPEC
#
# for each J-coupling constant restraint is to be specified:
# IPJR, JPJR, KPJR, LPJR: atom sequence numbers defining the angle phi
# WJWR: weight factor of interaction
# PJR0: reference J-coupling constant value
# PSJR: phase shift delta = theta-phi
# A, B, C: Karplus parameters
# H:    0 (harmonic), 1 (attractive), -1 (repulsive)
#
#IPJR JPJR KPJR LPJR      WJVR      PJR0      PSJR         A         B         C  H
#   C    N   CA    C    ASN 16
    2    4    6   13   1.00000   7.30000  -60.0000   6.40000  -1.40000   1.90000  0
#   C    N   CA    C    TYR 17
   13   15   17   31   1.00000   6.40000  -60.0000   6.40000  -1.40000   1.90000  0
#   C    N   CA    C   HISB 18
   31   33   35   43   1.00000   7.10000  -60.0000   6.40000  -1.40000   1.90000  0
...
(continues)
...
  136  138  140  149   1.00000   6.50000  -60.0000   6.40000  -1.40000   1.90000  0
#   C    N   CA    C    LEU 29
  149  151  153  158   1.00000   6.90000  -60.0000   6.40000  -1.40000   1.90000  0
#   C    N   CA    C    VAL 30
  158  160  162  166   1.00000   6.80000  -60.0000   6.40000  -1.40000   1.90000  0
END
```

The first four columns describe the dihedral angle η from which the ^{3}J-value is derived using the Karplus relation. The weight factor in the fifth column is only used for restraining and is ignored in this case. The sixth column is used as reference ^{3}J-value in order to calculate the deviation from it. As the measured ^{3}J-coupling constant may arise from a torsional angle ϕ whose atoms, such as aliphatic hydrogens, are not explicilty represented in the GROMOS force field, the torsional angle ϕ has to be related to the torsional angle η by a phase shift $\delta = \phi - \eta$. This phase shift is read from the seventh column. In the following three columns you can specify different parameters for the Karplus relation for every individual ^{3}J-coupling constant. Finally in the last column the type of the ^{3}J-value restraining is specified. Relations between ^{3}J-coupling constants and dihedral angles occurring in polypeptides are listed in Table 2.

Table 2: Relations between ^{3}J-coupling constants and dihedral angles occurring in polypeptides.

ϕ_n	η_n	δ_n [°]		a [Hz]	b [Hz]	c [Hz]
H - N - C_α - H_α	H - N - C_α - C	-60(L)	+60(D)	6.4	-1.4	1.9
H_α - C_α - C_β - $H_{\beta 2}$	N - C_α - C_β - C_γ	-120(L)	0(D)	9.5	-1.6	1.8
H_α - C_α - C_β - $H_{\beta 3}$	N - C_α - C_β - C_γ	0(L)	+120(D)	9.5	-1.6	1.8
N - C_α - C_β - $H_{\beta 2}$	N - C_α - C_β - C_γ	+120(L)	-120(D)	4.4	-1.2	-0.1
N - C_α - C_β - $H_{\beta 3}$	N - C_α - C_β - C_γ	-120(L)	0(D)	4.4	-1.2	-0.1
H - N - C_β - C_β	C - N - C_α - C	+60(L)	-60(D)	4.7	-1.5	-0.2
H - N - C_β - C	C - N - C_α - C	-180(L,D)		5.7	-2.7	0.1

Go to the J-value analysis directory (`~/ex6/ana/jval`) and to run `jval`, you will need the input files `jval.arg` and `p16-31_jval.dat`. And the procedure will create the following output file `jval.out`. For the unrestrained simulation, the scripts `jjval_unrestrained.sh` is already provided to you.

Have a look at it

```
...
@topo ../../topo/peptide.top
@pbc r
@jval #TO_DO
@traj
...
```

The argument `@jval` determines the file which contains the torsional angle specification corresponding to the specific ^{3}J-coupling constant. What is this file? It was mentioned above. Have a look and fill it in the place of `#TO_DO`. Now execute the script `jjval_unrestrained.sh`

```
./jjval_unrestrained.sh
```

It produces the file `jval_unrestrained.out`. In the first columns it contains the information that we gave in the `p16-31_jval.dat` file, and the last five columns give the information about our calculated ^{3}J-coupling constants.

As usual, you can make two copies of this script, and modify them to calculate ^{3}J-coupling constant for the MD simulations with IR-restraining or TAR-restraining, i.e. do

```
cp jjval_unrestrained.sh jjval_NOE_IR.sh
cp jjval_unrestrained.sh jjval_NOE_TAR.sh
```

then edit and adjust the four files, and then execute them

```
./jjval_NOE_IR.sh
./jjval_NOE_TAR.sh
```

Plot the ^{3}J-coupling constant of three MD simulations using script `jxmgrace.sh`.

```
./jxmgrace.sh
```

F. Print the figure of ^{3}J-coupling constants and comment on the difference.

4 Report

4.1 Format

Please refer to the corresponding section of Exercise 1 for information on the goal and expected structure of the report.

4.2 Simulation Results

Perform the requests from text the questions A to F. Then proceed to the following questions.

G. Understanding of NOE distance restraints file: Open the `noe.dsr` file in the directory `prep_noe`, find the restraints Nr. 3 and 70. For each restraint, please answer:

   (a) Which two hydrogen atoms are restrained?

   (b) What is the upper bound of this restraint?

   (c) Which atoms are actually used to represent these two restrained hydrogen atoms in GROMOS format coordinates file?

H. Understanding of ^{3}J-coupling constants file: open `jval_unrestrained.out` in the directory `~/ex6/ana/jval`, find the ^{3}J-value Nr.6. Please answer:

   (a) What is the experimental measured ^{3}J-coupling constant?

   (b) What is the calculated ^{3}J-coupling constant?

   (c) Which atoms define the dihedral angle between the planes?

I. Now we know two NOE restraining method: instantaneous restraining (IR) and time-averaged restraining (TAR). Which method do you think is better? Why?

J. In this exercise we only implement the NOE distance restraining to perform structure refinement, however, we can also use ^{3}J-coupling constant restraining to refine structure. What will be the problem if we apply time-averaged restraints to ^{3}J-coupling constant refinement?

   Hint: Have a look at Karplus curve (Eq. 4), think about it (Figure 2). Or you can find answer in Reference [10].

4.3 Thinking Questions

Answer the following questions:

K. In the introduction, we said that the timescale of the NMR experiment is about a nanosecond. How did we get to this (rough) estimate?

L. In Eq. 2, we use observables q defined as the distance for NOEs or the J-value for the J-couplings. Alternative options would be to use r^{-n} for the NOEs or dihedral angle for the J-couplings. Comment on the feasibility/advantages/shortcomings of these two alternative choices.

M. In Eq. 2 we generally use a half-harmonic (attractive) restraint only. Why is it preferable to using a harmonic (attraction + repulsion) restraint?

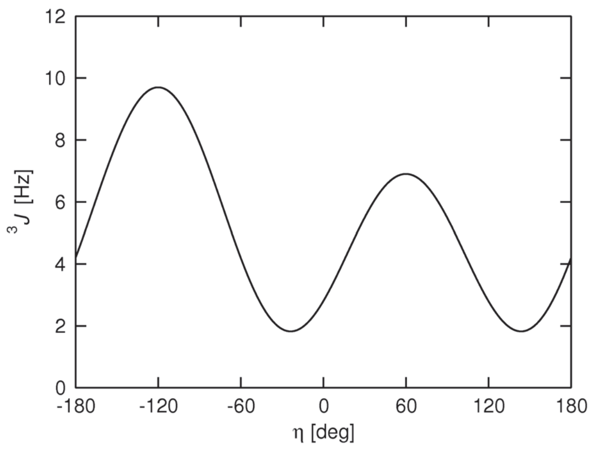


Figure 2: Example of a Karplus curve obtained using Eq. 4

N. When analysing the J-values, we saw that the source file can specify a different set of Karplus coefficients (a, b, c) for each dihedral angle specifically. We did not use this option (a, b, c set to the same values everywhere), but in which case do you think it would be useful?

O. What is the statistical mechanical ensemble sampler in an unrestrained, IR-restrained and TAR-restrained simulation? (considering a total potential energy including the distance-restraint energy and omitting the thermostat/barostat)

P. An alternative to time averaging is ensemble averaging. Can you explain briefly how you would implement such an approach?

Q. You have reached the last thinking question of the last CSBMS exercise. And this question is "how do you feel now?"

Appendix A. List of files

In the main exercise directory, you will find `ex6.pdf`, the digital version of the document you are reading and nine sub-directories:

forcefield/ Contains the GROMOS 54A7 force field file `54a7_ACE_CONH2.mtb` (two special building block of terminus ACE and CONH2 were added) and `54a7.ifp`

topo/ Contains `make_top.arg`, topology file `peptide.top`

coord/ Contains pdb file `model1_4-32GLU.pdb`, the library file to convert the protein starting coordinates to GROMOS format `pdb2g96.lib`, the GROMOS format coordinates `peptide.cnf`, and `peptide_gch.cnf` in which the coordinates of hydrogen are generated.

min/ Contains the required files to perform an energy minimization of the configuration `em.imd`, `jem.sh`, output file `em.omd`, energy minimized configuration `peptide_min.cnf`

box/ Contains the required files to get the protein solvated in a box `jsim_box.sh`, `spc.cnf`, the coordinates file after solvating `sim_box_peptide.cnf`, the requied files to perform an an energy minimization of the box `jem_solvent.sh`, `em_solvent.imd`, `sim_box_peptide.por`, `sim_box_peptide.rpr`, the output file `em_solvent.omd`, energy minimized configuration of the box `peptide_H2O.cnf`, the position restraints specification file `peptide_H2O.por`, the reference position file `peptide_H2O.rpr`

eq/ Contains the files to perform the equilibration of the peptide `equilibration.jobs`, `equilibration.imd`, `eq_mk_script.arg`, `jmk_script.sh`, `jsubmit.sh` `eq_peptide_*.run`, `eq_peptide_*.imd`, and output file `eq_peptide_*.omd`, `eq_peptide_*.trc.gz`, `eq_peptide_*.tre.gz`, `eq_peptide_*.cnf`

prep_noe/ Contains the NMR experimental NOE data `NOElist.exp`, NOE library file `noelib.54a7` which converts the `XPLOR` names into GROMOS format, NOE bound corrections `noecor.gromosXX`, the argument file to perform preparation of NOE distance upper bound `prep_noe.arg`

md/ Contains three subdirectories unrestrained, `NOE_IR`, `NOE_TAR`.

- **unrestrained**/ Contains the files to perform MD simulation `md_unrestrained.imd`, `jmk_script.sh` and `jsubmit.sh`
- **NOE_IR**/ Contains `md_NOE_IR.imd`, `jmk_script.sh` and `jsubmit.sh`
- **NOE_TAR**/ Contains `md_NOE_TAR.imd`, `jmk_script.sh`, `jsubmit.sh` and initial configuration file `md_NOE_IR_10.cnf`

ana/ Contains four subdirectories `rmsd`, `rmsf`, `noe` and `jval`.

- **rmsd**/ Contains scripts to perform RMSD calculation for MD simulation without restraints `jrmsd_unrestrained.sh`, script to plot the RMSD calculation results `jxmgrace.sh`
- **rmsf**/ Contains scripts to perform RMSF calculation for MD simulation without restraints `jrmsf_unrestrained.sh`, script to plot the RMSF calculation results `jxmgrace.sh`
- **noe**/ Contains scripts to calculate NOE distance upper bound violations for MD simulation without restraints `jnoe_unrestrained.sh`, `jprep_noe_unrestrained.sh`, script to plot the NOE distance upper bound violations `jxmgrace.sh`
- **jval**/ Contains ^{3}J-coupling constant restraints file `p16-31_jval.dat`, script to calculate ^{3}J-coupling constants for MD simulation without restraints `jjval_unrestrained.sh`, script to plot the NOE distance upper bound violations `jxmgrace.sh`

References

[1] Lipari G., Szabo A. *Biophysical Journal* **1981** 33, A307

[2] Karplus M. *Journal of the American Chemical Society* **1963** 85, 2870

[3] Torda A.E., Scheek R.M., van Gunsteren W.F. *Chemical Physics Letters* **1989** 157, 289

[4] Torda A.E., Brunne R.M., Huber T., Kessler H., van Gunsteren W.F. *Journal of Biomolecular NMR*, **1993** 3, 55

[5] Fennen J., Torda A.E., van Gunsteren W.F. *Journal of Biomolecular NMR* **1995** 6, 163

[6] Nanzer A.P., van Gunsteren W.F., Torda A.E. *Journal of Biomolecular NMR* **1995** 6, 313

[7] Nanzer A.P., Huber T., Torda A.E., van Gunsteren W.F. *Journal of Biomolecular NMR* **1996** 8, 285

[8] Keller B., Christen M., Oostenbrink C., van Gunsteren W.F. *Journal of Biomolecular NMR* **2007** 37, 1

[9] Steinmetz M.O., Jelesarov I., Matousek W.M., Honnappa S., Jahnke W., Missimer J.H., Frank S., Alexandrescu A.T., Kammerer R.A. *PNAS* **2007** 104, 7062

[10] Dolenc J., Missimer J.H., Steinmetz M.O., van Gunsteren W.F. *Journal of Biomolecular NMR* **2010** 47, 221

[11] Pardi A., Billeter M., Wuthrich K. *Journal of Molecular Biology* **1984** 180, 741