

Molecular Dynamics: From Basics to Application [Article v1.0]

Luis Vollmers^{1*†}, Shu-Yu Chen^{2†§}, Maria Reif^{1¶}, Tristan Alexander Mauck¹, Martin Zacharias^{1*}

¹Technical University of Munich TUM; ²Eidgenössische Technische Hochschule Zürich

This LiveCoMS document is maintained online on GitHub at https://github.com/Foly93/MD_FromBasicsToApplication; to provide feedback, suggestions, or help improve it, please visit the GitHub repository and participate via the issue tracker.

This version dated January 6, 2025

Abstract This tutorial equips natural scientists with the essential knowledge needed to utilize and comprehend molecular dynamics simulations effectively. Beginning with the stability of integration algorithms and the conservation of energy, this article proceeds with the description of atomistic forcefields, thermodynamic ensembles, long-range electrostatics treatment, and free energy calculations within molecular dynamics simulations. It then extends to the simulation of proteins and drug discovery applications. This comprehensive overview includes numerous references to relevant publications and tackles real-world problems. The tutorial is based on a 10-week master's course typically undertaken by students in the first or second semester of their master's program.

*For correspondence:

zacharias@tum.de (MZ); luis.vollmers@tum.de (LV)

[†]These authors contributed equally to this work

Present address: [§]Informatikgest. Chemie, Inst. Mol. Phys. Wiss., Suisse; [¶]Lehrstuhl f. Theoretische Biophysik, Center for functional Protein Assemblies, Germany

1 Introduction

Molecular dynamics (MD) simulations are computational methods that model the physical movements of atoms and molecules. These simulations have become essential tools for studying organic molecules' and biological macromolecules' structure and dynamics. Many students and researchers from various scientific disciplines are increasingly interested in applying molecular mechanics and MD methods.

Molecular dynamics simulations are the computational realization of statistical thermodynamics and classical mechanics, with the first simulation attempts dating back to the 1960s.[1] The method works by assigning positions, velocities, and forces to all atoms. The forces are especially

challenging since all particles interact via non-bonded interactions and share many different interactions based on their chemical bond configurations. The collection of all forces acting on all atoms is called the forcefield and forms the primary assumption of each MD simulation, i.e., a simulation is only accurate within its forcefield. Software packages facilitating MD simulations then utilize positions, velocities, and forces to numerically integrate Newton's equations of motion. Propagating these equations in time generates a trajectory of the system of interest where time-dependent processes can be studied at high resolution and accuracy.[2] The field of computational biophysics has advanced significantly since the 1960s, and new algorithms have been developed, granting more possibilities to the user and speeding up the generation of results. Advanced analysis

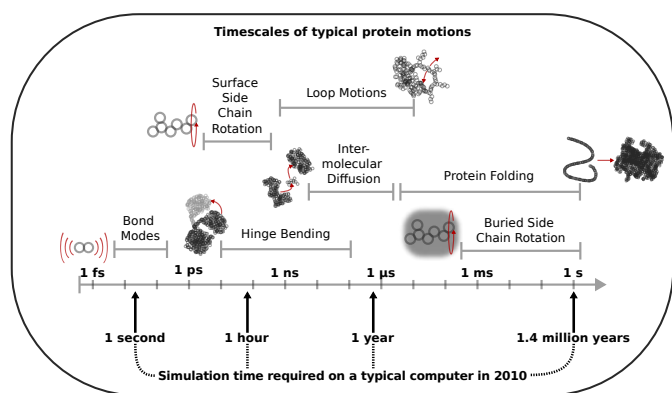


Figure 1. Protein motions occur across a wide range of timescales. Several types of protein motions are illustrated above the time axis with their corresponding time scale. The estimated computational effort required for their simulation is shown below the axis. The wall-clock time needed for molecular dynamics simulations for a typical aqueous protein system (45,000 atoms) varies drastically depending on the protein motion to be studied. The benchmarks were recorded in 2010 with a typical desktop computer. Adapted from Zwier et Chong.[4]

tools and flexible ways to set up simulations enable calculating fluctuating properties, energies, binding affinities, conformational changes, and other properties that might be interesting in a biomolecular system. Limitations of MD simulations are electron motions, including charge transfer and chemical bond formation or breakage. Also, the number of atoms and size of time periods that can be computed are finite but will grow with the evolution of algorithms and better hardware. Currently, large systems such as protein multimers can be simulated up to several microseconds.[3] Note, while movements of amino acid side chains can occur within picoseconds, intermolecular diffusion approaches times of a microsecond, and protein folding can take anything between a microsecond and a second.[4] In figure 1 timescales for typical protein motions are visualized together with the amount of time required to simulate them on a (2.6 GHz dual-core) desktop computer in the year 2010. Since then, the computational power and efficiency have grown exponentially, enabling larger scale simulations and highlighting this field's potential.

1.1 Scope

Recent advances in computational biophysics and molecular dynamics simulations have led to increasingly sophisticated simulation software. While numerous tutorials exist, they often focus on specialized applications or prioritize practicality over accessibility for beginners. The complexity of MD programs, with their extensive range of options, can overwhelm new users and may exacerbate the choice of simulation conditions and analysis methods. Focus of the

present tutorial/training hybrid article is to provide an introduction into MD fundamentals and applications by a series of hands on tutorials that guide through different applications of MD methods. Designed specifically for beginners, the series starts with an essential introduction to the Linux operating system — a prerequisite for installing and using MD simulation software. Through nine carefully structured exercises, users progress from understanding basic MD algorithms to advanced applications, including protein simulations, free energy calculations, and drug design protocols. Each exercise builds upon previous concepts, creating a logical learning progression that bridges the gap between theoretical understanding and practical implementation.

Exercise Overview

- Exercise 0: Short Introduction to Linux
- Exercise 1: Energy Conservation and Integration Algorithms
- Exercise 2: Forcefield Terms and Polymer Statistics
- Exercise 3: Temperature Coupling and Heat Capacity
- Exercise 4: Pressure and Temperature Coupling in MD Simulations
- Exercise 5: The Significance of Accurately Calculated Long-Range Interactions
- Exercise 6: Calculation of the Solvation Free Energy of Methane Using the MBAR Method
- Exercise 7: Potential of Mean Force of an Amyloid Layer Separation via Umbrella Sampling
- Exercise 8: Protein Simulation with Convergence and Principal Component Analysis
- Exercise 9: Virtual Drug Screening and MM/GBSA

The tutorial gives the user the essential practical knowledge to set up, run, and analyze MD simulations. Furthermore, this tutorial prompts the user to discuss findings for each exercise in a report and to assess the choice of specific algorithms or parameters. Combined with short discussions in the reports, they add a self-teaching element, furthering understanding and setting this tutorial apart from mere step-by-step tutorials. The tasks can also serve as a basis for developing own setups or design modifications to start own MD simulation endeavors. However, the tutorial is a practical introduction to MD simulation methods, and it does not replace the study of the appropriate literature on theoretical mechanics, statistical mechanics, and thermodynamics to become familiar with the theoretical foundations of MD simulations and analysis of MD data.

Each exercise follows a consistent format, beginning with a clear list of tasks accompanied by adequate conceptual artwork that doubles as a visual orientation. The exercises are divided into two main sections: a theoretical foundation and

detailed simulation setup instructions. The only exception to this structure is the computational drug design exercise (section 12), which takes a more application-focused approach, guiding users through various applications important to drug design methodology.

The tutorial/training hybrid article is supported by a comprehensive git repository (https://github.com/Foly93/MD_FromBasicsToApplication), structured to facilitate independent learning and classroom instruction. Each exercise has its own directory containing two subdirectories: 'teacher' and 'student.' The 'teacher' directory includes output files, analytical plots, and executable bash scripts that automatically solve the tasks. The lightweight 'student' directory contains only the necessary files for completing the exercise tasks. Additionally, each exercise directory features model reports as reference templates for students to compare their findings. However, these should be considered guidelines rather than definitive solutions since they are genuine student reports. Lecturers interested in using this document as course material are inclined to read the model reports since they are short and represent the net learning outcome.

2 Prerequisites

A computer with installed Linux OS is preferable, since many installation instructions given herein are tailored to Linux. Nevertheless, an experienced user might as well use any other operating system.

2.1 Background knowledge

This tutorial does not require any prior knowledge of the employed programs. However, basic knowledge in molecular biophysics and some knowledge on using computers is recommended. Note, the tutorial is based on a practical course on MD simulations for late stage undergraduates or graduates in biochemistry/physics/computer science or related areas.

2.2 Software/system requirements

The leading MD software used in the tutorials is the GROMACS 2018 program. Then, a Python package manager will be required for the last exercise, e.g., micromamba. Compared to the more well-known conda package manager, micromamba is faster, more lightweight, and thus recommended. The installation of micromamba can be postponed until section 12, where detailed instructions can be found. Visualizing three-dimensional structures is an essential part of molecular sciences, and a suitable program for that purpose is VMD. On their web page, <https://www.ks.uiuc.edu/Research/vmd/> installation instructions can be found, which include a quick registration via e-mail. There are other visualization programs, but VMD is recommended since some of the workflows employed in this document assume its availability. Lastly, a program to create scientific plots is required. Program suites like Python or R can be used; however, due to simplicity, the command line tool *gnuplot* (also known as *xmgnuplot*) is recommended. A summary of the requirements and their tested versions is listed below.

- Ubuntu 22.04
- GROMACS 2018
- micromamba 2.0.4 (or a recent version of conda/mamba)
- VMD 1.9.3 (recommended)
- Gnuplot 5.1 (recommended)

3 Exercise 0: Short Introduction to Linux






Figure 2. This concept art depicts the command line icon. Even though many programs have graphical user interfaces nowadays, a large proportion of scientific software utilizes the command line for speed and flexibility. Image adapted from shmector.com.

Tutorial on Linux/BASH (optional)

GNU/Linux is a UNIX-type operating system. It is free software, thus free of charge, and the source code can be downloaded and changed to fit individual needs. Several different distributions are available and many distributions offer a graphical user interface similar to Windows or MacOS. However, in the Linux world, utilizing a text-based interface (the shell) is standard, usually running in a terminal window. The shell reads and interprets user commands. After pressing enter, the commands are executed, and the output, if any, is displayed on the screen (STDOUT). In this section, a few features are introduced, offered by the Bourne-again shell (Bash), the most common shell in the Linux world. Many excellent online introductions exist for using the Unix/Linux operating system.

On webminal.org/login/, a terminal emulator can be found, i.e., a virtual terminal where Unix/Linux commands can be executed only via an e-mail registration. It also contains a tutorial section. Another terminal emulator that does not require registration but does not offer a tutorial can be found at bellard.org/jslinux/. The "UNIX Tutorial for Beginners" by Michael Stonebank at the University of Surrey can be done online (www.ee.surrey.ac.uk/Teaching/Unix). An extensive tutorial on Linux is provided by the "Cornell Virtual Workshop" (www.cac.cornell.edu/VW/Linux).

Basic Commands

The user must first open a terminal window to use the command prompt. On most Linux systems, this can be done by using the keyboard shortcut  +  + . Some of the most useful commands are listed below. Beginners should

execute some of them in the terminal. Start with the easier ones, such as `ls`, `cd`, and `echo`.

awk

While the other entries in this list are mostly programming tools, `awk` is a *Turing complete* programming language specifically designed to alter files. Some utilities might be helpful for this course.

cat <file>

The content of `<file>` is printed to STDOUT. It can also be redirected into another file (`>`) or be used as an input for a follow-up command (`|`).

cd <dir>

Change the directory to `<dir>`. `cd` changes to the home directory without an input option.

chmod <octalnum> <file>

Modifies the user permissions of `<file>` according to `<octalnum>`. `chmod` is necessary to make files executable.

cp <file> <filename>

Copies `<file>` to another file called `<filename>`. Linux assumes the users know what they are doing and overwrites any other file with the same name without question.

echo <string>

Takes `<string>` as an input text and prints it to STDOUT. It can also be redirected into another file (`>`) or be used as an input for a follow-up command (`|`).


grep <pattern> <input>

Search the `<input>`, which can be a file or a text, for the `<pattern>` and print the matching lines to STDOUT. The pattern can be a so-called *regular expression* (REGEX) or a simple text string.

history

This command prints a handy command history for keeping track or as a look-up if commands should be concatenated into an automatic script.

htop

Opens the interactive process viewer, which compares to the Windows task manager. CPU and memory usage, as well as all processes, can be monitored. Also, it makes the user feel like a hacker. Press  to exit back to the shell.

locate <pattern>

Locates files that match `<pattern>`. The pattern can either be a REGEX or any text string that is part of the file you seek.


ls <path>

List the contents of `<path>` to STDOUT. Note that `<path>` can also refer to any directory or a group of files. `ls` can also be prompted without `<path>` to list the current

directory. `ls` offers many options and is probably the most used command for Linux users.

man <executable>

The command's manual is opened, showing detailed information about the command given in <executable>.

Tap  to leave the man page.

mkdir <dirname>

Create a directory called <dirname> in the current directory.

mv <file> <filename>

Like `cp`, but the original <file> is removed in the process. Use with caution. If <filename> corresponds to a directory, the file is moved into that directory.

pwd

prints the current directory path to STDOUT.

rm <file>

This command removes <file> from the disk. Therefore, <file> is not moved to the bin like in Windows but deleted beyond recovery.

rmdir <dir>

Remove an empty directory <dir>.

sed <option> <file>

`sed` is a *turing complete* tool originally used for text filtering and manipulation and offers diverse automatic applications in this field. Mostly, this command is used to delete, insert, and substitute text in files exactly where the user wants it.

sort <input>

<input>, either a file or text string, is sorted alphabetically. Often, this command receives its input via the pipe `|`.

tail <N> <file>

The <N> last lines of <file> are printed to STDOUT. It can also be redirected into another file (`>`) or be used as an input for a follow-up command (`|`).

vim <file>

the VIM editor opens a file for editing. A steep learning curve accompanies its usage.

wc <input>

This command means *word count*; thus, it counts words in the given input. Furthermore, if given the appropriate options, it counts lines, characters, and bytes.

whoami

prints the current user to STDOUT.

Remember that these commands can change their behavior when particular options are specified. For Linux commands, these options are usually specified by `'-'` followed by the flag, e.g., `ls` list the directory content. However, `ls -l` lists the contents more verbosely and gives information about permissions, disk usage, and the latest access. Another ex-

ample would be `mv file file2`, which renames `file` to `file2` without communicating to the user. If `mv -v file file2` is specified, on the other hand, the command is more verbose and prints information to STDOUT. Most commands reveal the available options by specifying `--help`, e.g., `sort --help`.

Managing Files and Directories

Try to find out how to use the following commands, i.e., what inputs to use after each command in order to make them work:

```
# comments start with '#'; they will not be executed by bash
mkdir # create new, empty directory
touch # create new, empty file
cp # copy one or multiple files
cp -r # recursively copy one directory and its content to another
      one
rmdir # remove one empty directory
rm # remove one or multiple files
```

Use `man` or specify `--help` when in doubt. `rm` and `cp` as well as most other linux commands allow for something called *globbing*. Globbing does not require a comprehensive file name. Instead, an expression suffices resembling a variety of file names that match the so-called *glob*, e.g., use the wildcard `*` to match any pattern in the file name. For instance, `rm *.txt` will delete any file with the `.txt` extension. The question mark wildcard `?` represents exactly one character, which can be any single character. Two question marks in succession would represent any two characters in succession, and three question marks in succession would represent any string consisting of three characters. For instance, `rm ???` will delete any file whose name contains exactly three characters (including the extension). `rm ??? .txt` will delete any file with the `.txt` extension whose name contains exactly three characters. Both, `?` and `*` can be combined.

The `mv` command renames a file. The second input argument can either be a file, an absolute path, or a relative path, which holds for `cp` and `rm`.

```
mv file1 file2 # If file2 exists it will be overwritten
mv file1 dir1 # moves file1 to a directory called dir1
mv dir1/file1 ../ # move file1 from directory1 to parent folder
                 (../)
```

VIM/VI


One of the popular UNIX editors is `vi` (or VIM), which stands for visual editor. `vi` is a full-screen editor and has two main modes of operation:

- Command mode: commands to specify an action on the file.
- Insert mode: where text is inserted into the file.




In the Command mode, every character typed is a command that does something to the edited text file. Take into account

that both UNIX and vi are case-sensitive. Thus, the action performed on a file when typing y will not be the same as when typing Y. `vi <filename>` creates a new file. If the file already exists, the first lines will be shown on the screen, similar to other text editors. The following commands can be used to save and exit the file:




```
:q # quit vi without saving. This command will only work for
    unmodified files
:q! # quit vi without saving the last changes
:w # save (write) the file
:wq # save (write) the file and quit vi
:x # save and quit the file (exactly as :wq)
:w new_filename # save the current file with a different name
:wq new_filename # save the current file with a different name and
    exit. This command will only work if new_filename is not the
    name of any existing file in the current directory
:wq! new_filename # save the current file with a different name and
    exit; overwrites existing file with the name new_filename
```

To switch to the insert mode, either `i`, `I`, `a`, `A`, `o` or `O` can be used. See below for the meaning of each specific command. Once in the insert mode, every character typed is added to the text in the file. To turn off the insert mode,  can be used.

```
i # insert text before cursor
I # insert text at beginning of current line
a # append text after cursor
A # append text to end of current line
o # insert text in a new line below current line
O # insert text in a new line above current line
```

Furthermore, there is a replace mode accessible by hitting +. Switching between the insert mode and the replace mode is also possible by pressing . More specific options exist to modify file contents via VIM. Only some of them are listed below.

```
r # replace single character under cursor
x # delete single character under cursor
Nx # delete N characters, beginning with current character
dw # delete single word, beginning with current character
C # change characters in the current line
D # delete remainder of the current line
Y # copy remainder of the current line
cw # change the current word with new text
cc # change entire current line
dd # delete entire current line
yy # copy entire current line
cNw # change the next N words, beginning with the current
cNc # change the next N lines, beginning with the current
dNw # delete the next N words, beginning with the current
dNd # delete the next N lines, beginning with the current
yNy # copy the next N lines, beginning with the current
p # paste the previously copied/deleted lines/words after the
    current line
```

Undoubtedly, two of the most important commands are redo/undo, which are available in VIM by typing  for undoing and + for redoing. At this point, VIM can be used just like any other text editor. However, it is much more versatile. It can search the text for text patterns and replace them with

new text using convenient shortcuts. Some of those shortcuts are listed below. Furthermore, VIM has numerous plugins since it is open source with a lively community, e.g., you could run latex by just using VIM.

```
/word # search forward for the pattern 'word' in text.
?word # search backward for the pattern word in text
n # move to following occurrence of search string
N # move to preceeding occurrence of search string
gg # go to the first line
G # go to the last line
:%s/pattern1/pattern2/g # replace word pattern1 by pattern2
    throughout the whole file
```

For a more hands-on introduction to VIM, you may use `vimtutor`. This program gets installed simultaneously with VIM and can be opened by typing `vimtutor` in your command line. It contains seven interactive lessons, teaching the very basics of the program.

4 Exercise 1: Energy Conservation and Integration Algorithms

TASKS FOR THE FIRST EXERCISE

Finish these tasks and write a short report about your findings.

- Include plots containing E_{kin} , E_{pot} and E_{tot} for each of the 12 simulations you have to conduct: one correctly propagated simulation, one simulation that fails to conserve E_{tot} and one simulation that 'explodes'.
- Briefly explain your results and state what time step you would choose for the respective system to conserve energy but still simulate as long as possible. How can you explain the correlation between particle mass and choice for the time step?

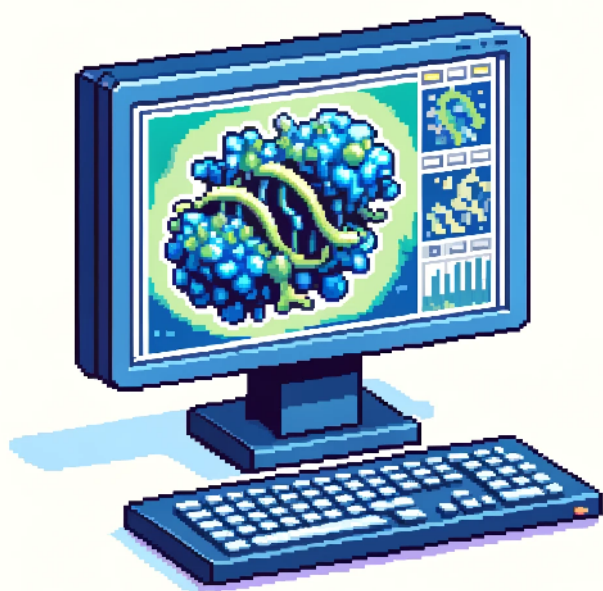
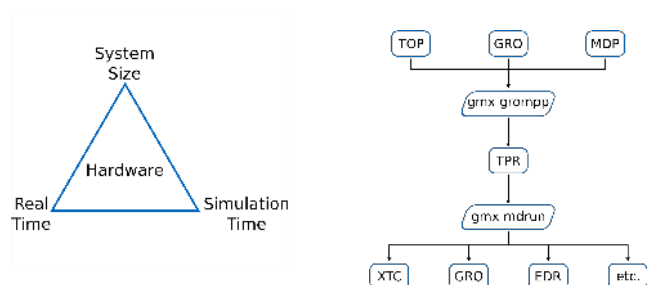


Figure 3. A computer screen showing an elusive protein structure to welcome the reader to computational biophysics. Art created with DALL-E.

Introduction to GROMACS

GROMACS[5, 6] is a highly optimized open-source software for molecular dynamics (MD) simulations of biomolecules like proteins, DNA, and lipids. A basic task of this command-line program is to integrate Newton's equations of motion to generate a time series, also known as a trajectory, of a given molecular system. The tutorial introduces first the workflow and the fundamentals of GROMACS. It will then focus



- (a) For a given hardware, the user must trade between real-time, simulation time, and the system size. The system size or the simulation time must be sacrificed for less real-time passing during the simulation.
- (b) The workflow in GROMACS uses the commands `gmx grompp` and `gmx mdrun` to generate diverse output files from three standardized input files.

on the choice of the integration time step, which is one of the most fundamental parameters of an MD simulation. Its importance originates from the limitations of computational resources.

The computational resources, hence the available hardware, determine the quality of three interdependent constraints, the most important one being the real-time needed to complete the simulation. The other two constraints are the system size (number of equations) and simulation time (number of times said equations are solved). The system size usually depends on the scientific problem at hand. Therefore, the simulation time is the most flexible of the three constraints that can also be modulated by the integration time step (see figure 4a).

Theoretical Considerations

The integration time step is related to how Newton's equations of motion are solved in the computational framework. Assuming, N particles the motions governing particle movement are dependent on the forces F on each particle and the potential energy function U_{tot} which depends on the configuration of all particles $\vec{r}^N = \{r_0, ..., r_i, ..., r_N\}$. [2] The potential energy function is also called the forcefield and presents the cardinal assumption underlying every MD simulation. It will be elaborated on in section 5.

$$F(r_i) = -\frac{\partial U_{tot}}{\partial r_i} \quad (1)$$

$$F(r_i) = m_i \frac{d^2 r_i}{dt^2} \quad (2)$$

Solving these equations analytically for more than two particles is not possible, and therefore numerical algorithms have been developed all of which rely on the user's choice of the integration time step. One of the simpler implementations is the Euler method that iteratively applies the following two equations to generate molecular configurations \vec{r}^N which are

then used to calculate the forces in equation (1).[7]

$$r_i(t + \Delta t) = r_i(t) + v_i(t)\Delta t \quad (3)$$

$$v_i(t + \Delta t) = v_i(t) + \frac{F_i(t)}{m_i}\Delta t \quad (4)$$

It can be shown, that the error resulting from this method is in the order of Δt . [8] Other algorithms like the verlet algorithm and the leap-frog algorithm (a variation of the verlet algorithm [2]) generate truncation errors that are in the order of Δt^4 which presents a substantial improvement. [9] The leapfrog integrator is the default setting that GROMACS applies and it uses the half-step velocities to generate the coordinates of the next step. [2]

$$r(t + \Delta t) = r(t) + \Delta t \cdot v(t + \frac{1}{2}\Delta t) \quad (5)$$

$$v(t + \frac{1}{2}\Delta t) = v(t - \frac{1}{2}\Delta t) + \Delta t \cdot a(t) \quad (6)$$

Another benefit of the leapfrog integrator is its time-reversibility which ensures the conservation of energy. [10]. Apart of truncation errors, other error sources include round-off errors and errors from improper sampling of the fastest movement within the molecular system. Thus, the following question emerges: How large can the time step be without destabilizing the system? Small-time steps cannot capture biological processes within reasonable real-time, and steps that are too large lead to truncation errors in the integration algorithm thereby violating energy conservation.

Setup and Simulations

This exercise aims to get accustomed to GROMACS and understand the stability of time integration algorithms and their dependency on particle mass. Herein, four systems should be simulated with different time steps. For each system, the different time steps should generate a correct trajectory, one trajectory where the energy conservation is violated, and one with a time step unreasonably high so that the simulation becomes numerically unstable resulting in chaotic motion not conserving energy and other physical quantities.






GROMACS Workflow

Unlike many everyday programs, GROMACS does not have a graphical user interface due to its manifold functionalities. Instead, it executes commands and tools via the command line. E.g., a very simple command is `gmx --version` which outputs version information about GROMACS onto the screen. GROMACS requires three different files to run a simulation: TOP-, MDP- and GRO-file. Each file contributes different information to GROMACS. The GRO-file contributes the atomic positions and velocities, the TOP-file contributes the information about the atomic and molecular attributes (the topology), and the MDP-file is short for the molecular dynamics parameters, and determines which algorithms to use and how to tune them. The

command `gmx grompp` takes these three files as input with their respective flags and outputs the TPR file that can be subsequently used to start the MD run via `gmx mdrun` (see figure 4b). In a nutshell, `gmx grompp` is the MD run preparation tool, and `gmx mdrun` calls the simulation engine. A variety of output files are generated by `gmx mdrun`, which are explained in the GROMACS command line reference (see manual.gromacs.org/documentation/2018/onlinehelp/gmx-mdrun.html for further information).

Table 1. GROMACS file formats specified by their suffices with a concise description. This table is not comprehensive. Further information are available on the GROMACS webpage.

Suffix	Short Description
.tpr	GROMACS binary run input file, contains simulation parameters and is generated via <code>grompp</code>
.gro	Coordinate file in GROMACS format. Contains positions, velocities and box dimensions.
.top	Topology file defining molecular structure such as molecules, bonds and types of interactions.
.mdp	GROMACS parameter file, contains simulation settings, like the time step, run time and thermodynamic ensemble.
.trr	Compressed binary trajectory file containing positions and velocities of all particles over the course of the simulation.
.xtc	Compressed binary trajectory file in XTC format.
.edr	Binary energy file, contains simulation energy data for analysis by GROMACS command line tools, e.g. <code>gmx energy</code> .
.log	Humand-readable log file with detailed simulation output.

The most important for today's tutorial is the EDR file outputted by `gmx mdrun`. In this file, the energetic information about the system is saved, which can be analyzed via `gmx energy` to extract specific information and plots. `gmx energy` is an interactive command that selects the energetic properties for information collection. The interface is terminal-based and typically looks like shown in figure 5. Selecting the energetic properties within the interface is conducted by typing the corresponding number followed by hitting  twice, e.g., for the kinetic energy in figure 5, type  +  + . The user interaction can be skipped by pre-defining the terms of interest with `printf` and `|`. The vertical line is called a pipe and can chain different commands. `\n` is code for a newline and is equivalent to  for the user interaction.

```
printf "4\n5\n6\n" | gmx energy -f <...>.edr -o <...>.xvg
```



```
Select the terms you want from the following list by
selecting either (part of) the name or the number or a combination.
End your selection with an empty line or a zero.
-----
1 Bond          2 LJ-(SR)       3 Coulomb-(SR)  4 Potential
5 Kinetic-En.   6 Total-Energy  7 Temperature   8 Pressure
9 Vir-XX        10 Vir-XY       11 Vir-XZ       12 Vir-YX
13 Vir-YY       14 Vir-YZ       15 Vir-ZX       16 Vir-ZY
17 Vir-ZZ       18 Pres-XX      19 Pres-XY      20 Pres-XZ
21 Pres-YX      22 Pres-YY      23 Pres-YZ      24 Pres-ZX
25 Pres-ZY      26 Pres-ZZ      27 #Surf*SurfTen 28 T-rest
```

Figure 5. User interaction after executing `gmx energy`. Multiple options can be selected at once by typing the number or the whole name of the energetic term.

Conclusively, the workflow for one system in this tutorial consists of the input file preparation (`grompp`), the actual simulations (`mdrun`), and the analysis of the energetic contributions (`energy`).

Effect of Time Step on the Stability of MD Simulations with GROMACS

Four different systems are used in this tutorial and are represented by the corresponding input files stored in the on-line repository. The coordinate file `intro.gro` represents a box of diatomic particles that should be used for all simulations. The topologies `topol_*.top` differ concerning the particle mass and the parameter file `intro.mdp` contains no value for the `dt` option, i.e., the simulation time step. This vacancy is deliberate since the main task for the user is to tinker with this option (further elaborated later on in this section). A detailed understanding of the input files is not necessary at this stage. Since the coordinates in `intro.gro` are initialized more or less randomly, their positioning might result in overlapping atoms. Thus, starting simulations from such states with very high forces results in unstable MD simulations. Therefore, an initial potential energy minimization using `em.mdp` removes sterical overlaps to converge to an energetically favorable regime. The downloadable `em.mdp` file must thus be used before applying the `intro.mdp` file in the following simulation. Commands that conduct the energy minimization and the MD simulation can look like the following lines of code. Their flags and options are detailed in table 2.

```
gmx grompp -f em.mdp -p topol_2.top -c intro.gro -o em_2.tpr
gmx mdrun -v -deffnm em_2
gmx grompp -f intro_0.001.mdp -p topol_2.top -c em_2.gro -o intro_0
.001_2.tpr -maxwarn 3
gmx mdrun -v -deffnm intro_0.001_2
gmx energy -f intro_0.001_2.edr -o intro_0.001_2.xvg
```

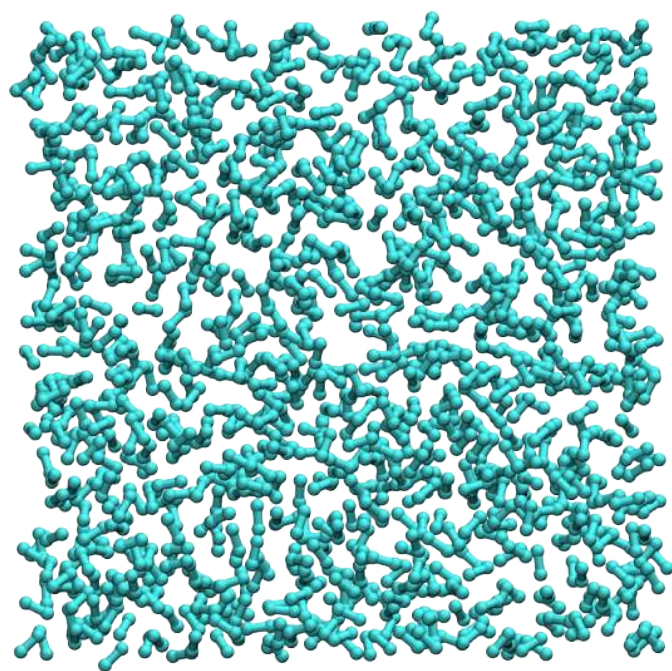


Figure 6. The particles visualised by `vmd intro.gro`. With representations set to *dynamic bonds* and *VDW* In the simulation, the bonded atoms are treated as a harmonic oscillator.

Table 2. Each flag given in the code above is explained in detail in this table using one example.

Flag (+ Option)	Meaning
<code>-f em.mdp</code>	Use the energy minimisation parameter file as <code>grompp</code> input.
<code>-p topol_2.top</code>	In this example, use the system topology with an atomic weight of 2 Dalton.
<code>-c intro.gro</code>	Use the initial coordinates from the online repository for the energy minimization.
<code>-o em_2.tpr</code>	Output the file <code>em_2.tpr</code> for usage in the <code>gmx mdrun</code> command. The name contains the parameter setup (em) and topology info (2).
<code>-maxwarn 3</code>	Choosing unreasonable options is registered and halted by GROMACS. This safety measure is based on warnings and can be circumvented for didactic purposes via this flag.
<code>-v</code>	Gives information about the time step and remaining simulation time if the simulation works. If the simulation does not work, there is no timing information.
<code>-deffnm em_2</code>	all input and output names of the corresponding <code>gmx mdrun</code> should have the prefix <code>em_2</code> which must match the <code>-o</code> option of <code>gmx grompp</code> .
<code>-f intro_0.001_2.edr</code>	Instead of an MDP file for <code>gmx grompp</code> , <code>gmx energy</code> needs an EDR file for the <code>-f</code> flag. The prefix is the same as specified in the <code>-deffnm</code> option of <code>gmx mdrun</code> .

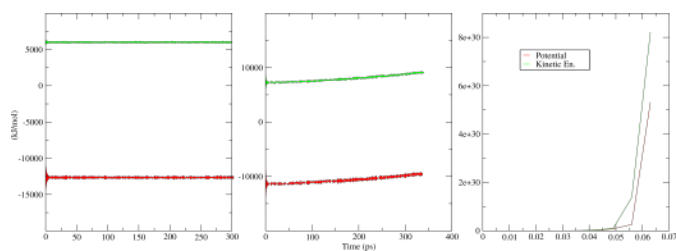


Figure 7. Example plots for one system. (left panel) total energy vs simulation time for the stable time step, (middle panel) example result for an unstable simulation, where the total energy increases over time, and lastly, a simulation that crashes after 0.06 ps due to extremely high forces (*exploding*).

Finally, the task is to choose three different time steps and compare the different outcomes. Firstly, a small time step should be selected so that the system propagates numerically stable. The second case represents a simulation where the energy conservation is violated, but the simulation still finishes. In the last case, the large time step causes the simulation to crash due to numerical instability. In this case, `gmx mdrun` will halt, and you have to abort the program with `Ctrl+C`. Eventually, 12 plots will be featured in the report. Three different time steps for each of the four particle masses (e.g. figure 7).

It is easy to generate the two extremes, however, an intermediate time step that violates energy conservation but does not crash is challenging to select. Typically, a time step of around 0.001 ps is used for classical atomistic MD simulations.[10] Start with this time step and the lightest system. Then, increase the time step in steps of 0.001 ps until the system crashes or violates energy conservation. If the system crashes, a time step in between the current and the former needs to be tried (e.g., 0.0035 ps). Otherwise, if the intermediate case is achieved, the results can be saved, and the next increment can be tried to cause a crash. In order to check the energies, `xmgrace` or any other plotting utility can be used. Python, via Jupyter Notebook, is also recommended, especially for figures to be featured in the reports. However, for having a quick look at the graphs, `xmgrace` suffices.

```
xmgrace -nxy output.xvg
```

Once all three graphs are ready for the lightest system, the second-lightest system may be tried with the intermediate time step found for the lightest system. As before, the time step has to be incremented or decremented until all three cases are achieved. The same procedure applies to the remaining systems. The graphs and a discussion of the results and apparent trends should be included in the report. Additionally, state what time step you would choose for the respective system.

5 Exercise 2: Forcefield Terms and Polymer Statistics

TASKS FOR THE SECOND EXERCISE

Finish these tasks and write a short report about your findings.

- ☐ Compare your analytical solutions for R_e to the numerical ones.
- ☐ Plot all R_e and R_{gyr} vs. the simulation time.
- ☐ Include histograms of the θ and ϕ and plot the chain averages vs. simulation time.
- ☐ Summarize all averages for θ , ϕ , R_{gyr} and R_e in one table.
- ☐ Describe and interpret your results. Include representative snapshots generated by VMD to support your conclusions.

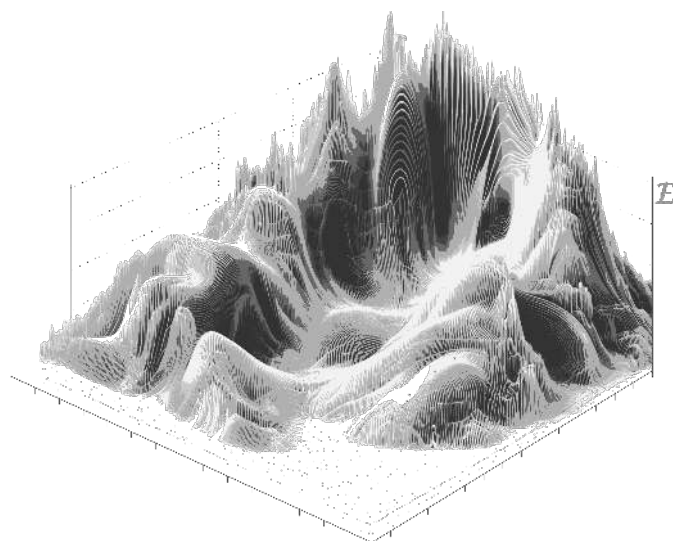


Figure 8. Conceptual depiction of the potential energy surface. The real potential energy surface is $(3N-6)$ -dimensional and cannot be illustrated in 2D.

Molecular dynamics (MD) simulations are a unique method that enables the user to visualize time-resolved processes in atomic detail. However, the method only approximates reality by parametrizing molecular properties, e.g., charges and bond lengths. The complete set of these parameters and their respective formulas is called *forcefield*, synonymous with the system's potential energy. The quality of the forcefield, as well as its completeness, determines the quality of an MD simulation's predictions. In this exercise, a simple polymer consisting of carbon atoms is simulated with different levels of forcefield completeness. The goal is to

better understand the forcefield terms and their implementation in GROMACS. Additionally, a comparison with analytical polymer models will be conducted.

Theoretical Considerations

The complete (polymer) forcefield U_{tot} can be split into an intramolecular term U_{intra} and an intermolecular term U_{inter} .

$$U_{\text{tot}} = U_{\text{inter}} + U_{\text{intra}} \quad (7)$$

On the one hand, U_{intra} consists of bond, angular and dihedral contributions. The bonded potential U_b and the angular potential U_θ are harmonic potentials and therefore defined by an equilibrium distance b_0 and angle θ_0 as well as a force constant k_b and k_θ . Usually, the dihedral term is a cosine potential defined by the force constant k_ϕ , phase ϕ_0 and periodicity n_ϕ .

$$U_{\text{intra}} = \sum_i^{\text{bonds}} U_b(b_i) + \sum_j^{\text{angles}} U_\theta(\theta_j) + \sum_k^{\text{dihedrals}} U_\phi(\phi_k) \quad (8)$$

$$U_b(b_i) = k_b (b_i - b_0)^2 \quad (9)$$

$$U_\theta(\theta_j) = k_\theta (\theta_j - \theta_0)^2 \quad (10)$$

$$U_\phi(\phi_k) = k_\phi (1 + \cos(n_\phi \phi_k - \phi_0)) \quad (11)$$

On the other hand, U_{inter} only consists of the Lennard-Jones potential $U_{\varepsilon, \sigma}$, depending on the parameters ε and σ and the interatomic distance r_{mn} .

$$U_{\text{inter}} = \sum_m^{\text{Atoms}} \sum_{n \neq m}^{\text{Atoms}} U_{\varepsilon, \sigma}(r_{mn}) \quad (12)$$

The Lennard-Jones potential is one of many standard potentials implemented in GROMACS, and the interpretation of its parameters is quite intuitive. ε corresponds to the potential well depth, and σ indicates the particle radius, which could be shown from its mathematical expression.[11]

$$U_{\varepsilon, \sigma}(r_{mn}) = 4\varepsilon \left[\left(\frac{\sigma}{r_{mn}} \right)^{12} - \left(\frac{\sigma}{r_{mn}} \right)^6 \right] \quad (13)$$

An important potential herein omitted is the Coulomb potential, which describes charge-charge interactions. The potential energy function U_{tot} has already been introduced in the calculation of forces in equation (1) in the previous exercise. That means for each numerical integration time step the forcefield must be used to calculate all forces acting on each particle. The intramolecular part is computationally less expensive than the intermolecular forcefield, due to the double sum over all Lennard-Jones interactions. The herein omitted electrostatic interactions feature a double sum as well.

For a polymer consisting of N beads at positions \vec{b}_1 to \vec{b}_N the

mean squared end-to-end distance of the polymer chain $\langle R^2 \rangle$ can be calculated using the formula below.

$$\langle R_{\text{ee}}^2 \rangle = \langle R_{\text{ee}} \cdot R_{\text{ee}} \rangle = \left\langle \sum_i r_i \sum_j r_j \right\rangle = \sum_i \sum_j \langle r_i \cdot r_j \rangle \quad (14)$$

Analytical formulas $\langle R_{\text{ee}}^2 \rangle$ can be derived for the freely jointed and freely rotating polymer models. The former is defined only by a fixed bond length L , while the latter also features a fixed angle θ between adjacent beads. Calculating the analytical solutions for both polymer models, followed by comparison to MD simulations, is an adequate way to assess the validity of the simulations.

Setup and Simulations

In this exercise, a polymer consisting of 20 atoms should be simulated in the NVT ensemble for the following cases:

1. freely jointed chain; only U_b is active.
2. freely rotating chain; U_b and U_θ are active.
3. complete intramolecular forcefield; U_{intra} is active.
4. complete forcefield; U_{tot} is active with $\varepsilon = 0.9786 \text{ kJ mol}^{-1}$ and $\sigma = 0.3401 \text{ nm}$.

The individual contributions to the forcefield are specified in the topology file, herein called `polymer.top`. It can be downloaded from the online repository and should be briefly inspected. Activating the demanded forcefield terms requires deleting the `;` in the TOP file in front of the respective lines. The `;` precedes comments that are not interpreted by GROMACS. The parameters for U_{inter} can be set in the `[atomtypes]` section.

As a reminder, GROMACS requires you to execute `gmx grompp` and `gmx mdrun` to perform a simulation. Furthermore, consistent names should be chosen for simulation files to keep directories neat. A command prompt for the freely jointed simulation could look like this:

```
gmx grompp -f polymer.mdp -c polymer.gro -p polymer_fj.top -o
polymer_fj.tpr
gmx mdrun -deffnm polymer_fj -v
```

Analysing the Trajectory

Once all simulation runs are finished, the trajectories need to be analyzed. Herein, angular properties and statistical polymer properties should be calculated. GROMACS offers the built-in tools `gmx polystat` and `gmx angle` for this purpose. Since the angular potentials U_θ and U_ϕ are not included per default, their impact on the carbon chain should be investigated. Therefore, the average angle should be plotted vs. the simulation time. Additionally, an angle histogram should be generated. Both plots can be done by GROMACS via `gmx angle`. However, this command needs information about the triplets involved in one angle provided in a custom

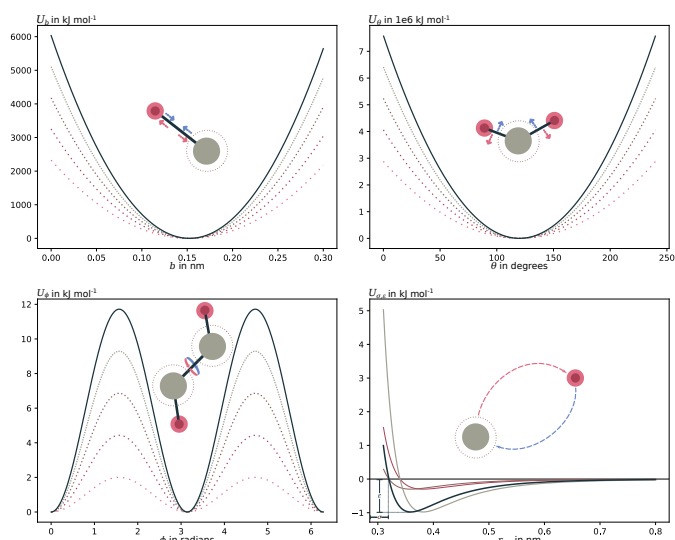


Figure 9. Exemplary plots of each forcefield term. The bonded potential is plotted on the upper left, followed by angular, torsional, and non-bonded potential. The colored, dotted lines in each panel depict the same forcefield term of decreasing strength (force constant). In the case of the non-bonded potential, this trend is visualized via the line width for aesthetic reasons. Furthermore, the epsilon and sigma values of the strongest potential are annotated close to the origin. Notice the differences in the energy scale of each contribution.

NDX file. This file lists the atom numbers making up each angle in the simulation. Therefore, it can be repurposed once it is generated. Luckily, GROMACS has a separate command, `gmx mk_angndx`, that outputs such a file automatically if angular information is present in the input TPR file. For example, for the freely jointed simulation, the necessary plots can be generated using the following commands:

```
gmx mk_angndx -s polymer_full.tpr -n angle.ndx -type angle
gmx angle -f polymer_fj.xtc -n angle.ndx -od fj_angle_hist.xvg -
type angle -ov fj_angle.xvg
```

The same steps must be taken to plot the dihedrals for the report. Thus, `gmx mk_angndx` must be executed but with the correct `-type` option, followed by the command `gmx angle`. The procedure is analogous to the one showcased above. Plotting the histograms of the individual forcefield cases into one plot is recommended to clarify differences.

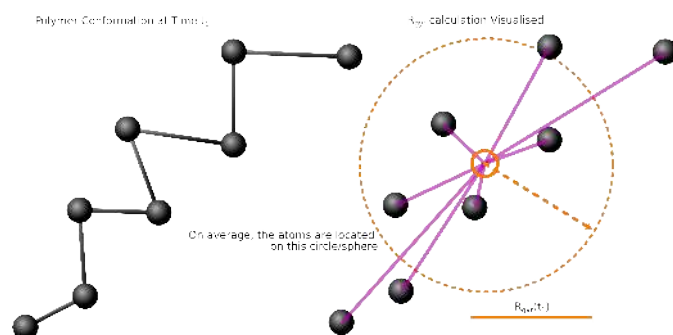


Figure 10. Visualisation of the calculation for R_{gyr} : only the atomic distances to the center of geometry of a given conformation are measured and then averaged. For extended conformations, R_{gyr} is larger than for collapsed polymer chains. However, for an equilibrated ensemble, a static ensemble average emerges $\langle R_{gyr} \rangle$.

The average end-to-end distance $\overline{R_{ee}}$ of the simulations can be validated using analytical results for comparison. From equation (14) analytical formulas shall be derived to calculate the root mean square end-to-end distance $\sqrt{\langle R_{ee}^2 \rangle}$ for the freely jointed and the freely rotating polymer model. These calculations should agree with the numerical results obtained from simulation. As aforementioned, GROMACS has its built-in tool to calculate $\overline{R_{ee}}$ and $\overline{R_{gyr}}$. Example code is given below.

```
gmx polystat -s polymer_fj.tpr -f polymer_fj.xtc -o fj_polystat.xvg
```

For the report, the time averages $\overline{R_{ee}}$ and $\overline{R_{gyr}}$, plots that feature their time development and the analytical results shall be included. All measurements should be summarised in a table for a better overview of the results. The analytical and numerical results for R_{ee} and their agreement should be discussed. Also, similarities and differences between the different simulations should be discussed concerning the underlying forcefield terms. Visual proof drawn from VMD may be incorporated.

Tutorial on Visualising a Trajectory

A standard visualization program used for molecular dynamics simulation is called VMD. For simulation output generated by GROMACS, you have to input the GRO file and the XTC file to visualize trajectories. The GRO file only specifies the atom configuration, while VMD guesses the bonds. Therefore, matching `polymer_fj.gro` and `polymer_fj.xtc` files may incorrectly depict molecular bonds. Instead `polymer.gro` can be used.

```
vmd polymer.gro polymer_fj.xtc
```

Since the VMD defaults, when loading a trajectory, can be suboptimal follow the steps below for an improved trajectory appearance. Firstly, the visibility is greatly enhanced by aligning the structure (see figure 11). Aligning entails centering the structure of each time frame on the reference

frame coordinates, followed by rotations that minimize the atomic distances between the two frames. This operation makes the trajectory appear less juddery.



Figure 11. Extensions ⇒ Analysis ⇒ RMSD Trajectory Tool ⇒ window opens ⇒ Align ⇒ close the window

Secondly, the representation of the molecule per default is set to lines, which lacks visibility. Changing this to Licorice and applying Gaussian smoothing to the trajectory enhances the user experience even further (see figure 12).

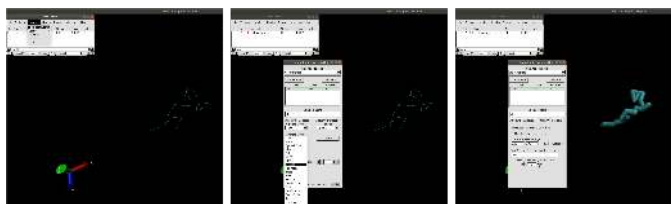


Figure 12. Graphics ⇒ Representations ⇒ window opens on Tab Draw Style
⇒ Drawing Method ⇒ Licorice (⇒ optional: adjust Bond Radius) ⇒ switch Tab to Trajectory ⇒ set Trajectory Smoothing Window Size to 3 ⇒ Close the Window

For including snapshots in the report, the Trajectory Smoothing Window Size must be reverted to zero since this setting manipulates the atom positions. The easiest and quickest way to generate images from VMD is to take screenshots.

6 Exercise 3: Temperature Coupling and Heat Capacity

TASKS FOR THE THIRD EXERCISE

Finish these tasks and write a short report about your findings.

- ☐ Include a commented thermostat-MDP file of your choice.
- ☐ For all six production runs, plot T , E_{kin} , E_{tot} and E_{pot} vs simulation time and describe them. Particularly, find differences and similarities between the thermostats.
- ☐ Calculate and compare the C_v of each production run with the literature value and interpret the results.
- ☐ Visualize the polymer trajectory with vmd and describe any unusual behavior. You may include screenshots of representative conformations.
- ☐ Compare the MDP file from the current simulation with the previous exercise's MDP file and speculate which alterations might have caused the behavior at hand.



Figure 13. Cartoon Depiction of the Flying-Ice-Cube effect. Adapted from DALL-E.

Introduction

Up to this exercise, the simulations have been conducted without emphasizing the system temperature. Most of the simulations were conducted in the microcanonical ensemble with a constant number of particles (N), constant volume (V), and constant total energy (E), also called the NVE ensemble. Implementing temperature control includes coupling the molecular system to a virtual heat bath, allowing energy exchange. Therefore, the canonical ensemble features a constant number of particles, volume, and temperature. It is also called the NVT ensemble.[12] Adding a level of temperature control to Newton's second law, thus switching to the canonical ensemble, might be desirable for various reasons. Some include matching an experimental setup, studying thermosensitive processes, and exploring the sampling space more time-efficiently with advanced methods (e.g., temperature replica exchange molecular dynamics simulations[13, 14]).

This exercise is structured in two parts. In the first part, simulations shall be setup to compare different thermostatting algorithms and their effect on different water forcefields. The second part only consists of a visual analysis of a trajectory featuring the flying-iccube-effect. This exercise aims to become familiar with different water models, visualize MD trajectories, and independently assess the drawbacks and benefits of popular thermostats by calculating the isochoric heat capacity and comparing it to its experimental values. This way, the accuracy of the thermostats and the water models can be analyzed.

Theoretical Considerations

Within MD simulations, different temperature concepts are distinguished via nomenclature. Firstly, the instantaneous temperature T is obtained from the atomic velocities of each simulation time step.

$$T(t) = \frac{1}{3Nk_B} \sum_i^N m_i v_i(t)^2 \quad (15)$$

Secondly, the ensemble average of the temperature $\langle T \rangle_{\text{NVT}}$ is a thermodynamic constant approximated by the system's time average, assuming thermodynamic equilibrium and sufficient sampling time.[15]

$$\langle T \rangle_{\text{NVT}} \approx \frac{1}{t_1 - t_0} \int_{t_0}^{t_1} \frac{1}{3Nk_B} \sum_i^N m_i v_i(t)^2 dt \quad (16)$$

Finally, the reference temperature, T_0 , is the heat bath temperature imposed on the MD system and is a user-specified parameter. Algorithmically, the heat bath's implementation is called a thermostat, a variety of which have emerged over the past decades. To showcase their function, the mathematical framework of the Berendsen thermostat[16] shall

be laid out. It is also denoted as *weak coupling* thermostat. The Berendsen thermostat is a widely used, relatively simple thermostat that employs a first-order relaxation equation. This thermostat induces a temperature change that scales proportionally with the difference between reference and instantaneous temperatures and with the coupling parameter τ_T .

$$\frac{dT}{dt} = \frac{1}{\tau_T} (T_0 - T) \quad (17)$$

$$\text{or } \Delta T = \frac{\Delta t}{\tau_T} (T_0 - T) \quad (18)$$

In the context of the numerical integration of Newton's second law, a thermostat is usually implemented by rescaling the atomic velocities since the instantaneous temperature depends on them. It can be shown that the above equations yield the following scaling law for the Berendsen thermostat.[12]

$$v_i(t) = v'_i(t) \cdot \sqrt{1 + \frac{\Delta t}{\tau_T} \left(\frac{T_0}{T(t)} - 1 \right)} \quad (19)$$

By rescaling the atomic velocities $v'_i(t)$, the Berendsen thermostat ensures that the temperature approaches T_0 following an exponential decay. Each thermostat offers certain benefits and drawbacks. A benefit of the Berendsen thermostat is its computational efficiency. Although a thermostat's quality may also be assessed with respect to time-reversibility or consistency with experimental results.[12, 17]

The isochoric heat capacity C_V is a material property that relates the amount of applied energy to the increase of temperature. Herein, C_V is used to measure the quality of the canonical ensemble generated by the temperature coupling algorithm. GROMACS calculates C_V from the total energy fluctuations $\text{RMSD}(E_{\text{tot}})$ the specified molecule number N_{mol} , the ideal gas constant R and the temperature T . [2]

$$C_V = \frac{(\text{RMSD}(E_{\text{tot}}) \cdot 1000)^2 \cdot N_{\text{mol}}}{RT^2} \quad (20)$$

Setup and Simulations

Comparing different Temperature Coupling Algorithms

The assessment of two very common thermostats is the subject of this exercise. Herein, the Berendsen and the Nose-Hoover thermostat[18–20] shall be investigated. In the online repository, all necessary input files can be found. These files comprise two different water models and three different temperature coupling algorithms, and they can thus be used to perform six different simulation sequences. In the context of an MD simulation, the water model means the collection of forcefield terms describing a its intramolecular and intermolecular interactions. Similar to the polymer

model in the last exercise the simulation sequence consists of an energy minimization, an equilibration, and a production run.[17]

This exercise task features six different MDP files: `eq_nve.mdp`, `prod_nve.mdp`, `eq_berendsen.mdp`, `prod_berendsen.mdp`, `eq_nhc.mdp` and `prod_nhc.mdp`. They can be categorized into equilibration MDP files and production MDP files. The simulations utilizing the equilibration MDP files feature fewer timesteps than the production. The purpose is to heat up the system and let it settle into its thermodynamic equilibrium such that the production run starts from an equilibrated system. Eventually, the production MDP files and their respective simulations serve the purpose of data collection. The data of interest are equilibrium properties, and their accuracy typically increases with the simulation length. Therefore, each production simulation takes quite some time to finish, and only the production output files will be used for calculating C_V and other properties.

Additionally, the six MDP files can be categorized by the thermostat used during the simulations. The three temperature-coupling algorithms used here are: None (NVE-Ensemble), Berendsen, and Nose-Hoover-Chain. They all perform very differently in generating the correct canonical ensemble, which is connected to the heat capacity. The NVE simulation is supposed to perform worst since it resembles the wrong ensemble all together.

Comparing the MDP files via `vimdiff <file1> <file2>` should indicate how GROMACS implements the different thermostats and, therefore, the different ensembles. Furthermore, for an advanced understanding of the MDP files, go through each line and explain its meaning via a comment. Consultation of the GROMACS MDP options documentations is recommended (see manual.gromacs.org/documentation/2018/user-guide/mdp-options.html). Comments can be inserted into an MDP file with `; <Comment>`. After commenting, a line could look like the following:

```
integrator = md ; Leap-Frog algorithm for time integration
```

GROMACS will ignore everything behind the semi-colon. Include one completely commented MDP file in your report.

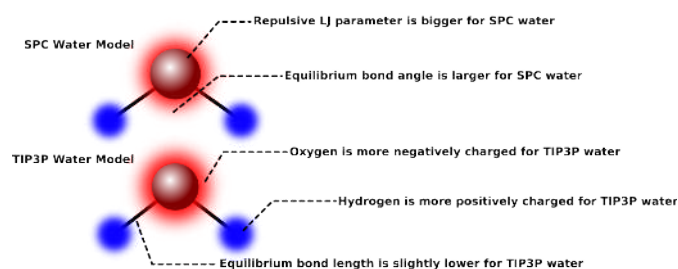


Figure 14. TIP3P and SPC water in comparison. The bond lengths, angles, charges, and Lennard-Jones parameters are true to scale and depicted by the different features of each molecule. The blue and red blurs depict the charge, and the red bubble depicts the repulsive Lennard-Jones parameter. Even though the differences are hardly visible, the impact on the accuracy is immense and highlights the importance of precise parameterization in MD. The exact parameters are listed elsewhere (see below).

The SPC and the TIP3P water model shall be used for calculation of the specific heat capacity C_V . The numerical differences between these water models can be investigated by reading their `*.itp` files or by inspecting the Wikipedia webpage: en.wikipedia.org/wiki/Water_model. Although their parameters are quite different, both belong to the three-point-water models, containing only three interaction sites and representing the three atoms of the water molecule. There are also four-point-water and five-point-water models that benefit from a better charge distribution but are computationally more expensive.

The MDP files do not contain information about the water model used in the simulation sequence. This information is stored in the GRO and TOP files, which are called `spcbox.gro`, `spc.top`, `tip3pbox.gro` and `tip3p.top`. The differences of these files can be investigated easily using the command above called `vimdiff`. As a reminder, the GRO file stores box dimensions, atom coordinates, and velocities while the topology contains forcefield information.

The following procedure describes how to conduct the simulation sequences. The first step is to minimize the water boxes. This is done analogously to previous exercises using `gmx grompp` followed by `gmx mdrun`, e.g. for SPC water:

```
gmx grompp -f em.mdp -c spcbox.gro -p spc.top -o em_spc.tpr
gmx mdrun -v -deffnm em_spc
```

The water boxes must only be minimized once, not for every temperature coupling. Once the minimization is ready, conduct the equilibration in a similar matter, e.g., for the Nose-Hoover-Chain thermostat:

```
gmx grompp -f eq_nhc.mdp -c em_spc.gro -p spc.top -o eq_nhc_spc.tpr
gmx mdrun -v -deffnm eq_nhc_spc
```

The given flags always have their command-specific meaning. E.g., `-f` needs to be followed by the (correct) MDP file for the `gmx grompp` command. `-v` prompts the `gmx mdrun` to

be more verbose about its procedures. For detailed information, consult the command line reference: manual.gromacs.org/documentation/2018/user-guide/cmdline.html.

Finally, an example of the execution of a production run can be seen below:

```
gmx grompp -f prod_nhc.mdp -c eq_nhc_spc.gro -p spc.top -o
prod_nhc_spc.tpr
gmx mdrun -v -deffnm prod_nhc6_spc
```

All consecutive runs covering all water models and temperature coupling algorithms shall be executed accordingly.

Once a production run finishes, the data collection can begin. For this purpose, the command `gmx energy` is recommended. Each simulation produces an EDR file that stores energy information throughout the simulation run. `gmx energy` uses this file to calculate all sorts of properties. The properties of interest in this exercise are the temperature T , kinetic energy E_{kin} , total energy E_{tot} and potential energy E_{pot} . Furthermore, `gmx energy` can directly calculate the heat capacity C_V if the flags `fluct_props` and `-nmol` are specified. The complete command could look like the following:

```
gmx energy -f prod_nhc_spc.edr -s prod_nhc_spc.tpr -o nhc_spc.xvg -
fluct_props -nmol <num>
```

Upon command execution, the program will define the energetic properties to be processed via user input. They can either be selected by a number or by writing out the complete name of that property. Plots of the four energetic properties (saved to `nhc_spc.xvg` for the example above) should be included in the report for each production run with a discussion of differences in the plot appearance between the temperature coupling algorithms.

Additionally, compare the six values calculated for C_V with the literature value of $74.4 \frac{\text{J}}{\text{mol K}}$ (for water at 300 K and 1 bar)[21] and report the relative deviation. If all options were set correctly, `gmx energy` prints the values to the terminal automatically.

The Flying-Icecube-Effect with a Polymer Model

For the polymer model used in the previous chapter a pre-made trajectory has been deposited online to demonstrate an interesting thermostating artifact. The three input files used to generate the MD run, `polymer.gro`, `polymer*.top`, and `polymer.mdp`, should be downloaded together with the output files. Comparison of the `polymer*.top` with the corresponding file from last exercise reveals that all forcefield terms are activated in the current run. However, visualizing the trajectory indicates a bizarre, unnatural behavior, termed *Flying-Icecube-Effect*. Some snapshots of the trajectory should be featured in the report, together with a description of the

behavior of the polymer during the simulation. Some instructions are listed below as a reminder of how to visualize trajectories nicely in VMD.

```
vmd polymer.gro polymer_non_bonded.xtc
```

Graphics ⇒ Representations ⇒ window opens on Tab
 Draw Style ⇒ Drawing Method ⇒ Licorice (⇒ optional:
 adjust Bond Radius)

Furthermore, a white background is better suited for inclusion in scientific reports.

Graphics ⇒ Colors ⇒ Display ⇒ Background ⇒ white

Given the odd behavior of the polymer, it might come in handy to display several frames at once to hint towards the system's dynamics.

Graphics ⇒ Representations ⇒ switch to Tab
 Trajectory ⇒ change the option for
 Draw Multiple Frames ⇒ E.g.: 1000:1005

After increasing the Step value from 1 to 3, the unnatural behavior is even easier to observe. Since the polymer simulation of section 5 did not exhibit the *Flying-Icecube-Effect*, there must be a difference in the input files of both runs and indeed, these differences can be found in the `polymer.mdp`. An inspection of the current and last week's `polymer.mdp` files should be conducted to find differences causing this behavior. The command `vimdiff` is recommended for a fast and error-proof comparison. The information gained from this inspection should factor into the thermostat assessment for the report.

7 Exercise 4: Pressure and Temperature Coupling in MD Simulations

TASKS FOR THE FOURTH EXERCISE

Finish these tasks and write a short report about your findings.

- For all 12 simulations, list σ_V^2 and κ_T tabularly and plot E_{tot} , V and P as a function of time. Discuss differences and similarities between barostats and choices of τ_P .
- Calculate κ_T for TIP4P and SPC/E Water and compare to the experimental value.
- Briefly, describe the differences in E_{kin} , E_{pot} , ρ , H and E_{tot} for the standard and non-standard simulation. Try to explain the differences and propose a molecular dynamics-based method to verify your explanation.

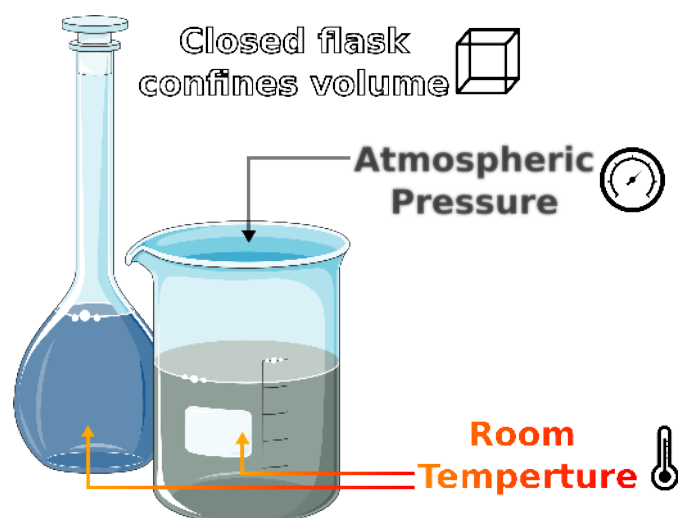


Figure 15. The application of a thermodynamic ensemble (NVT, NPT) is required to mimic the physical conditions in the wet lab. High-accuracy algorithms for barostatting and thermostatting are crucial to yield realistic results from a molecular dynamics (MD) simulation. smart.servier.com/ (CC-BY 3.0)

7.1 Introduction

In section 6, the application of thermostats was introduced, i.e., simulations in the NVT ensemble that maintains a constant temperature. This exercise focuses on pressure coupling and the isothermal-isobaric ensemble. A barostat ensures constant pressure while letting the volume fluctuate. Since most laboratory experiments are performed under constant pressure and temperature, employing the isothermal-isobaric ensemble (NPT) to achieve realistic simulations is often reasonable, i.e., a thermostat and a

barostat algorithm must be employed to amend Newton's equations of motion. Usage of the barostat only will result in a different thermodynamic ensemble.

This exercise is split into three subtasks. Firstly, a TIP3P water box is tested for two different barostats (Berendsen and Parrinello-Rahman) and a range of coupling parameters. The coupling parameter τ_P defines the barostat's stiffness towards the reference pressure, similar to the thermostat. A comparison to TIP3P's literature isothermal compressibility κ_T shall be conducted, which depends on the volume fluctuations of the simulation. Secondly, two further water models shall be simulated within the NPT ensemble, and their κ_T values shall be compared to experimental water. Thirdly, two further simulations must be conducted and analyzed for their energetic properties. The simulation setup differs only slightly. However, the results deviate significantly, and the user should find out why. This exercise aims to teach how pressure coupling works and how the water model and coupling constant influence the thermodynamic quantities. Furthermore, understanding thermodynamic ensembles and their relation to MD simulations is further clarified. After this exercise, users will have a proper overview of how pressure coupling is implemented and be able to explore further details independently.

Theoretical Considerations

Conceptually, pressure coupling uses the same approach as temperature coupling, where velocities are rescaled via coupling to a heat bath. However, a barostat algorithm rescales the positions instead of the velocities; thus, the volume, V , is no longer constant. Other than that, many principles are analogous to thermostatting. E.g., the instantaneous pressure $P(t)$, the ensemble average of the pressure $\langle P \rangle_{\text{NPT}}$ and the reference pressure P_0 have their pendants in the previous exercise. The instantaneous pressure can be calculated via the kinetic energy of the system and the so-called inner virial W_{inner} that depends on the pairwise forces F_{ij} acting on two particles i and j . [22, 23]

$$P(t) = \frac{2}{3V} (E_{\text{kin}}(t) - W_{\text{inner}}(t)) \quad (21)$$

$$P(t) = \frac{2}{3V} \left(E_{\text{kin}}(t) - \frac{1}{2} \sum_i^N \sum_{j>i}^{N-1} r_{ij}(t) \cdot F_{ij}(t) \right) \quad (22)$$

The formula is known as the virial theorem. Similarly to the barostat, the ensemble average of the system pressure can be approximated by the time average of the instantaneous pressure, given sufficient sampling. [15]

$$\langle P \rangle_{\text{NPT}} \approx \frac{1}{t_1 - t_0} \int_{t_0}^{t_1} P(t) dt \quad (23)$$

The plentifulness of barostat algorithms is up to par with thermostating, and all algorithms feature different characteristics, benefits, and drawbacks. However, to showcase the implementation of a barostat, the simple but popular Berendsen barostat will be presented in all brevity. In the case of pressure coupling, the box size and the positions are scaled with a scaling factor μ to alter the result of the virial as a pressure correction. The Berendsen algorithm introduces a pressure change via a virtual pressure bath proportional to the difference between the reference and instantaneous pressures.[16]

$$\frac{dP}{dt} = \frac{P_0 - P(t)}{\tau_P} \quad (24)$$

Analogously to temperature coupling, the user-specified constant τ_P defines how tight the pressure coupling is. It can be shown that this differential equation leads to a scaling factor μ that also depends on the isothermal compressibility κ_T .[16]

$$\mu = \sqrt[3]{1 + \frac{\Delta t \kappa_T}{\tau_P} (P(t) - P_0)} \quad (25)$$

Even when κ_T might be unknown, it can be estimated since its accuracy only affects τ_P , which can be chosen by the user over a wide range of values. Once μ is calculated from the instantaneous pressure, it is applied to the unscaled system coordinates $r'(t)$ after numeric integration and velocity rescaling in the case of thermostating.[16]

$$r(t + \Delta t) = \mu \cdot r'(t + \Delta t) \quad (26)$$

$$V(t + \Delta t) = \mu^3 \cdot V(t) \quad (27)$$

After rescaling, the updated coordinates are used to calculate the forces in the next step via equation (1). Even though pressure coupling retains the reference pressure very well on average, it can lead to substantial variations in the instantaneous pressure. The large fluctuations are due to steep pressure-volume isotherms for liquid systems, i.e., a subtle change in volume leads to a considerable change in pressure and vice versa.[24]

The isothermal compressibility is thermodynamic quantity that relates applied pressure to response in volume. The formula GROMACS uses to calculate κ_T comprises the volume fluctuations $\text{RMSD}(V)$, the temperature T , the Boltzmann constant k_B and the average volume $\langle V \rangle$.[2]

$$\kappa_T = \frac{\text{MSD}(V)}{k_B T \langle V \rangle} \quad (28)$$

Setup and Simulations

The Choice of the Coupling Constant Matters

Two barostats and six different coupling constants shall be evaluated in this sub-task. The MDP files can be found in the online directory, with missing values for the τ_P -option and

the barostat in the NPT input files. The values to be screened are listed in table 3. In the previous three exercises, the necessary input files were already provided in the online resource. In this exercise, the user needs to create the necessary GRO and TOP files. However, manually creating the two missing files is an optional sub-task. If time is limited, this task may be skipped by using `tip3pbox.gro` and `tip3p.top` from the previous exercise.

The command `gmx solvate` is a good starting point. This command offers a variety of options to solvate systems or create them from scratch. Since the command is quite versatile, reading the GROMACS documentation is recommended to figure out the crucial flags and options. Set the box dimensions to $2.504 \times 2.504 \times 2.504 \text{ nm}^3$ and even though we use TIP3P water, `spc216` needs to be specified for the `-cs` flag. In GROMACS `spc216` is the standard 3-Point water model GRO file, also representing the coordinates of the TIP3P model and all other 3-point water models. The same rationale holds for 4-Point and 5-Point water models as well. Unfortunately, `gmx solvate` regularly fails to squeeze the requested amount of solvent into the simulation box. Hence, the `-maxsol` flag should be set to 1 so that we can use the resulting one-water GRO file as input for `gmx insert-molecules`. This command fills a given box with the required number of water molecules (523), and unlike `gmx solvate`, it tries to meet the specifications as many times as given in `-try`. The generated single-water GRO file should be used for the `-ci` flag. Finally, `gmx pdb2gmx` creates the respective TOP file, herein with the TIP3P water model and the `amber99SB-ILDN` forcefield. An exemplary workflow for creating the aqueous system is shown below.

```
gmx solvate -cs spc216 -box 2.504 2.504 2.504 -maxsol 1 -o tip3p.gro
gmx insert-molecules -ci tip3p.gro -box 2.504 2.504 2.504 -nmol 523 -try 10000 -o choice_tauP.gro
gmx pdb2gmx -f choice_tauP.gro -p choice_tauP.top -water tip3p -o trash.gro # UI prompted after execution
```

As a reminder, directories and naming conventions should be kept clean and consistent within projects. Thus, names should be kept as short as possible and as informative as necessary to maintain a good overview. The simulation sequence consists of minimization, followed by an NVT simulation, and then the NPT-Ensemble. The values for τ_P can be found in the following table. For certain τ_P values, adjusting the `-maxwarn` flag of `gmx grompp` might be necessary.

Table 3. Values for the coupling constants are to be screened by the user for both barostats.

τ_P/ps^{-1}	0.05	0.1	0.5	1	5	10
-------------------------	------	-----	-----	---	---	----

For the Parrinello-Rahman barostat and the τ_P values 0.05 ps^{-1} and 0.1 ps^{-1} the additional option `nstpcouple = 1` must be inserted in the respective MDP file (position in the file is arbitrary). For each NPT simulation, plot the Volume V and Pressure P against time and calculate the variance of the volume σ_V^2 throughout the simulation. Concerning the plots, the first few frames may be excluded if they do not represent an equilibrated system using `gmx energy -b <FRAME>`. The volume fluctuations σ_V^2 are linked to the isothermal compressibility κ_T to be calculated for each τ_P as well. Tabulate the various values of σ_V^2 and κ_T and feature the plots in the report. Additionally, discuss which of the τ_P and barostats delivers results comparable to the literature value of $\kappa_T^{\text{TIP3P}} = 6.3 \times 10^{-5} \text{ bar}^{-1}$ and why others may fail.[25] The command `gmx energy` is highly recommended to conduct the result collection. The command is analogous to the one used last week to obtain the isochoric heat capacity. The *energetic* properties that need to be selected from the interactive menu can be deduced from equation (28).

```
gmx energy -f <ENERGYFILE> -o <OUTPUTFILE> -fluct_props -nmol <
NUMBER>
```

Influence of the Water Model on the Isothermal Compressibility

In this task, two water models should be simulated to investigate the influence of different water models on κ_T . Optimally, the experimental value $\kappa_T^{\text{exp}} = 4.52 \times 10^{-5} \text{ bar}^{-1}$ for 298.15 K is retained from the simulation runs.[26] The water models in question are the SPC/E and TIP4P water models. This time, the simulation procedure includes an energy minimization, an NVT- and NPT-equilibration, concluded by the pre-production and production run. It is common practice to use the Berendsen thermostat and barostat for the NVT- and NPT-equilibrations since it is robust and computationally inexpensive. The pre-production and production runs are usually simulated using the Nose-Hoover and Parrinello-Rahman algorithms for high-fidelity production data. The pre-production run facilitates the system adjusting itself to the enhanced algorithms. As shown in the previous exercise, some algorithms sample a certain ensemble more reliably, which implies that the system has to adapt to certain physical changes. The analysis should not include the transition time necessary for this adaptation since it does not represent the thermodynamic equilibrium. The data for result computation should stem only from the fully equilibrated production run.

In order to save time, `nsteps` in the MDP files can be adjusted based on the MD runs from the first task. Simulations should be long enough to collect sufficient data to compute results, but not longer. The `nsteps` option may be decreased to save time.

All necessary MDP files are given or can be repurposed, and the TOP and GRO files can be obtained in the aforementioned fashion. SPC/E is a three-site water model, and TIP4P is a four-site water model. Thus, the correct option for the `-cs` flag must be ensured.

```
gmx solvate [...] | gmx insert-molecules [...] | gmx pdb2gmx [...]
# Procedure of obtaining GRO and TOP file
```

Once the simulations have finished, `gmx energy` is recommended to evaluate which of the water models performs better concerning κ_T .

Simulation Behaviour for Large τ_T Values

In the preceding tasks, strengths and weaknesses between algorithms were uncovered, parameter choices for τ_P were examined, and the feasibility of different water models. Thus, the pressure coupling setup was the only aspect of interest. This last task serves as a reminder that a simulation consists of a delicate interplay of various algorithms and their settings.

Therefore, two simulation setups shall be compared with each other. One features a relatively tight pressure coupling and a high τ_T value, corresponding to infrequent corrections to the temperature. Meanwhile, the τ_T and τ_P values of the other setup can be considered standard.

The system itself is the same as in the second task. Therefore, the only differences lie in the MDP files deposited in the online repository. Said files, `standard.mdp` and `non_standard.mdp`, should be used to simulate the NVT-equilibrated SPC/E system from the second task. Once the simulation is done, use `gmx energy` to calculate and plot the kinetic energy E_{kin} , the potential energy E_{pot} , the density ρ , the enthalpy H and the total energy E_{tot} of the two simulations.

The plots should be included in the report, and differences in their appearances should be discussed. An explanation for the observed behavior and a simulation-related way to test this explanation should also be given.

8 Exercise 5: The Significance of Accurately Calculated Long-Range Interactions

TASKS FOR THE FIFTH EXERCISE

Finish these tasks and write a short report about your findings.

- ☐ From the SPC/E water simulation, compare the first two peak positions of the RDF for all six different potential-modifiers with the experimental values.
- ☐ Include RDF plots for all the methods with a qualitative and quantitative comparison to figure 19b.
- ☐ Include time-resolved plots and averages of the RMSDs for all the different methods with a brief discussion.
- ☐ Analyse all results regarding quality and performance and try to determine a best-practice method. Include the timings in this discussion.

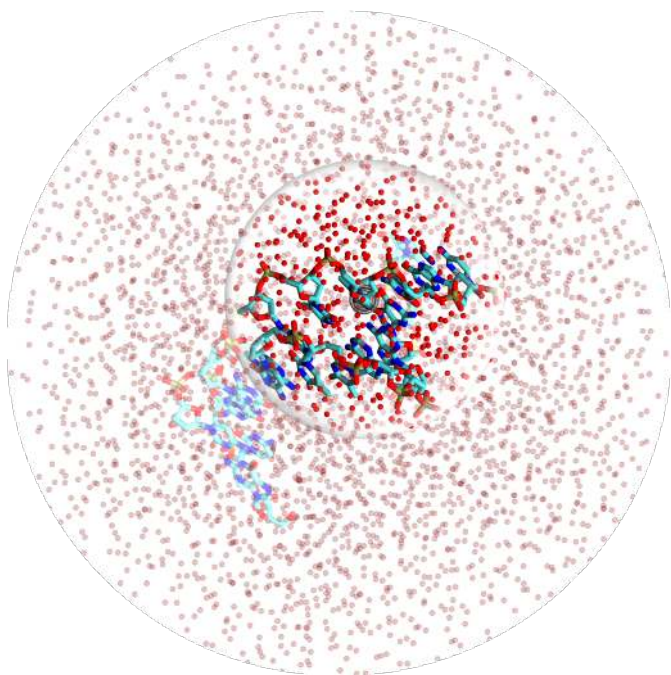


Figure 16. Simplified visualization of the cutoff radius. Every opaque particle falls within the (enlarged) central particle's cutoff, signified by the *forcefield bubble*. Non-bonded interactions with (transparent) particles outside this bubble would be omitted in a corresponding molecular dynamics (MD) simulation.

Theoretical Considerations

The most demanding task for molecular dynamics software is calculating pairwise additive interactions, i.e., the charge-

charge interaction and the London-dispersion interactions modeled most often via the Coulomb and Lennard-Jones potential. The Lennard-Jones potential was already introduced in section 5. It consists of a repulsive and an attractive term. Two constants are required to parameterize the potential, usually called σ and ε . Physically, σ is associated with the excluded volume of the respective atom, thus its radius, while ε is the maximum potential well depth.[11]

$$U_{\varepsilon,\sigma}(r_{mn}) = 4\varepsilon \left[\left(\frac{\sigma}{r_{mn}} \right)^{12} - \left(\frac{\sigma}{r_{mn}} \right)^6 \right] \quad (29)$$

The Coulomb potential is proportional to the Coulomb constant, the charge product of the two interacting particles and scales with r^{-1} . The coulomb constant consists of two other constants, π and the electric permittivity in the vacuum ϵ_0 . [27]

$$U_c(r_{mn}) = \frac{q_m q_n}{4\pi\epsilon_0 r_{mn}} \quad (30)$$

Since both of these potentials are pairwise additive, a double sum over all particles has to be calculated; therefore, the computational cost scales with N^2 where N is the number of particles in the system. Thus, the non-bonded interactions account for most of the computational time and are the main subject of algorithmic optimization. Both forces, resulting from the Coulomb potential U_c and Lennard-Jones potential $U_{\varepsilon,\sigma}$, converge to 0 for infinite interatomic distances, and the Lennard-Jones converges faster than the Coulomb potential. Therefore, a simple option is to introduce a cutoff radius r_{cutoff} beyond which the interactions are set to zero. Such a cutoff can have various forms and implementations. Either the potential energy function is abruptly set to zero after the cutoff distance, or a so-called shifting or switching function is applied, smoothly bridging the potential function to zero at the cutoff distance. The latter two options yield relatively small errors for the quickly converging Lennard-Jones potential (converges with r^{-6} towards zero). However, significant distortions must be expected for the slowly converging coulomb potential (converges with r^{-1} towards zero).[28]

More complex methods based on the Ewald summation have emerged to treat long-range electrostatics more effectively. Ewald summation was initially invented to estimate the potential energy of crystal lattices[29], and several techniques have branched out to approximate the electrostatic interactions in molecular dynamics simulations, such as P3M[30], PME[31] and SPME[32].

For the illustration of long-range electrostatic treatment, the Ewald-based methods will be omitted since their mathematical framework is beyond the scope of this document. Instead, two potential shifting functions are showcased to make the topic tangible to the user. The two functions S_1

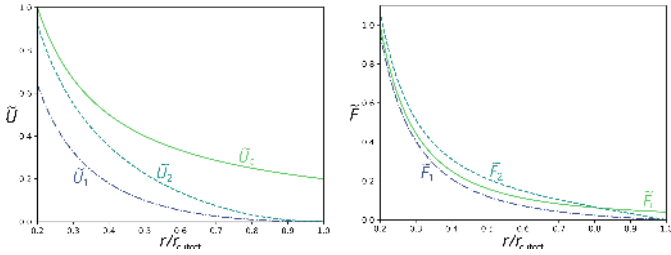


Figure 17. The reduced potential energy functions \tilde{U} and forces \tilde{F} are plotted to visualize the different behavior of the two shifting functions compared to the unfettered coulomb potential. Adapted from Prevost et al.[33]

and S_2 ensure that the electrostatic interactions cross zero when r_{cutoff} is reached.[33]

$$S_1(r_{mn}) = \begin{cases} 1 - 2\frac{r}{r_{\text{cutoff}}} + \left(\frac{r}{r_{\text{cutoff}}}\right)^2 & (r < r_{\text{cutoff}}) \\ 0 & (r \geq r_{\text{cutoff}}) \end{cases} \quad (31)$$

$$S_2(r_{mn}) = \begin{cases} 1 - 2\left(\frac{r}{r_{\text{cutoff}}}\right)^2 + \left(\frac{r}{r_{\text{cutoff}}}\right)^4 & (r < r_{\text{cutoff}}) \\ 0 & (r \geq r_{\text{cutoff}}) \end{cases} \quad (32)$$

The modified potentials U' are the product of the coulomb potential with its shifting function, and the resulting force is the negative derivative with respect to position.

$$U'(r_{mn}) = U_c(r_{mn}) \cdot S(r_{mn}) \quad (33)$$

$$F' = -\frac{\partial U'}{\partial r_{mn}} \quad (34)$$

Although the truncation largely decreases the computational cost, the modified potentials and their resulting forces deviate substantially from the original. The differences between the potentials are visualized in figure 17. Both modified potentials underestimate the energy contribution by a large margin, and for the forces, the S_1 function consistently underestimates the coulomb force, while the S_2 function overestimates short-ranged interactions. Furthermore, the radial distribution function (RDF) shall be introduced which is a measure for quantitatively testing the effect of long-range electrostatics. The RDF between particle group A and B ($g_{AB}(r)$) describes the structure of (isotropic) liquids and liquid mixtures. It works by counting the particles of type B at a certain distance from a central particle of type A. The abundance of B as a function of the distance reveals information about their intermolecular interactions and could be used to calculate several thermodynamic properties. To calculate $g_{AB}(r)$ the average particle density of particle type B for a distance r , $\langle \rho_B(r) \rangle$, is divided by the bulk density $\langle \rho_B \rangle_{\text{bulk}}$,

$$g_{AB}(r) = \frac{\langle \rho_B(r) \rangle}{\langle \rho_B \rangle_{\text{bulk}}} \quad (35)$$

The bulk density for B is usually calculated for the complete simulation box, hence dividing N_B by the box volume V_{Box} . For

a continuous distance, the average particle density B around A can be written in terms of the Kronecker-delta $\delta(\dots)$ of the particle distance r_{mn} and the surface area of the sphere,

$$\langle \rho_B(r) \rangle = \frac{1}{N_A} \sum_{i \in A} \sum_{j \in B} \frac{\delta(r_{mn}, r)}{4\pi r^2} \quad (36)$$

where $\delta(r_{mn}, r) = 1$ if $r_{mn} = r$ and $\delta(r_{mn}, r) = 0$ otherwise. The expression becomes slightly more complicated for the non-continuous case natural to computational methods. Firstly, spherical shells with a thickness of Δr are considered instead of a sphere's surface, which effectively turns $g_{AB}(r)$ into a histogram depending on n bins $g_{AB}(n)$ of bin width Δr . Let r_n and r_{n+1} be the lower and upper border of the n -th bin. Then, this histogram function can be calculated by augmenting equation (36) with the Iverson bracket notation and the formula for a spherical shell,

$$\langle \rho_B(n) \rangle = \frac{1}{N_A} \sum_{i \in A} \sum_{j \in B} \frac{[r_n < r_{mn} < r_{n+1}]}{\frac{4}{3}\pi(r_{n+1}^3 - r_n^3)} \quad (37)$$

This expression can be simplified by assuming that each shell is very thin, hence $\Delta r \ll r_n$. Given GROMACS' default bin width of 2 pm, this assumption is satisfied, resulting in the final expression of the discretized radial distribution function,

$$g_{AB}(n) = \frac{1}{N_A \langle \rho_B \rangle_{\text{bulk}}} \sum_{i \in A} \sum_{j \in B} \frac{[r_n < r_{mn} < r_{n+1}]}{4\pi r_n^2 \Delta r} \quad (38)$$

Notice that A and B can also be of the same particle type, which helps investigate the structure of pure liquids like pure water, mainly since the RDF can be obtained from scattering experiments and MD simulations. A direct comparison of the plot appearance can hint towards the forcefield quality of the water model and the methods as well.

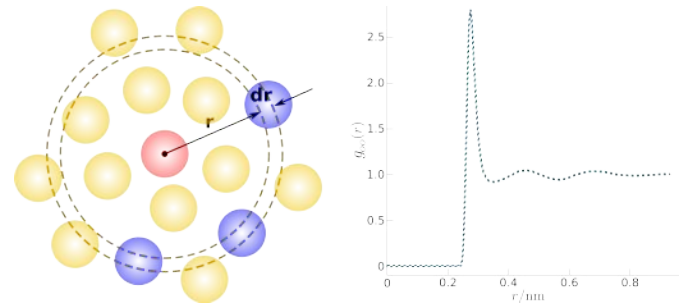


Figure 18. Left: Schematic on how the RDF is calculated. Right: The RDF of water oxygens of pure SPC water calculated from Simulation.[34]

Setup and Simulations

Water Structure Under the Influence of Different Electrostatic Interaction Schemes

The finite size of the simulation box and the periodic image limits the range of long-range interactions. Additionally,

the computational time depends largely on the number of atom pairs, for which non-bonded interactions have to be calculated. Introducing a potential-modifier using a cutoff distance r_{cutoff} to truncate non-bonded interactions remedies these issues. On the other hand, a realistic physical description must be maintained as well as possible, leading to various algorithms that mingle with this trade-off. In this exercise, the user explores some of these potential modifiers, analyses them for their strengths and weaknesses, and presents one solution as their personal recommendation.

Herein, algorithms for plain cutoff, potential shift modifiers, and the Particle-Mesh-Ewald (PME) method are investigated. The two former methods rely on cutoff truncation to simplify the calculation of electrostatic interactions. For these methods, a metric must be defined that tells GROMACS when to shift the potential and truncate the non-bonded interactions. The PME method, on the other hand, makes use of lattice Ewald summations (implemented using Fast Fourier Transforms) to simplify the calculation of long range electrostatic interactions in a periodic system. Its use requires the specification of several MDP options beyond this document's scope. The prepared MDP files can be downloaded from the online repository.

As usual, the system must be set up using GROMACS' tools. The edges of the cubic box should be 4 nm, and the SPC/E water model is supposed to be used with the amber99SB-ILDN forcefield. The MDP files for the equilibrations can be downloaded (EM, NVT, and NPT). The system can be equilibrated once, followed by the pre-production and production run for each long-range electrostatics treatment. In this exercise, six different modifiers are tested: PME, potential-shift with 8 Å and 12 Å cutoff, and the truncation method with 8 Å, 12 Å and 18 Å cutoff. The time benchmarks should be extracted from the LOG file (outputted automatically) to be included in the report. Usually, there are two different run time measurements: the wall time and the core time. The wall time measures the real-world time that elapses during the MD run, and the core time refers to the time spent by all processors to execute every command. The latter is more important in this task and should be compared for the different electrostatic interaction schemes, preferably in a table with a brief discussion.

A quantitative comparison can be conducted utilizing RDF peak positions. In figure 18, the peak locations correspond to solvation shells of water molecules. Therefore, the first peak corresponds to the location of the first, very distinct solvation layer. The second peak corresponds to the second less pronounced solvation shell. The simulation quality can be accessed by comparison with the experimental peak positions. Therefore, the report should include the timings and positions of each method's first two RDF peaks and a

quantitative comparison. Experimental values for comparing r_1 and r_2 were measured by Skinner et al.[35]. They found r_1 at 2.799 83 Å and r_2 at 4.5149 Å for a temperature of 300 K. Notice that the RDF is calculated for the water oxygen atoms, not the whole molecules. For completing this task, `gmx rdf` is highly recommended. The GROMCAS command line reference should be consulted since this command is quite versatile (see manual.gromacs.org/documentation/2018/user-guide/cmdline.html). As a hint, the `-ref` and `-sel` flags and specify the atom names given in the GRO file.

A-DNA and B-DNA Simulations with Different Electrostatic Interaction Schemes

After the investigation of pure water, the effects of different methods to calculate electrostatic interactions are probed for complex bio-molecules. Herein, A-DNA and B-DNA are simulated, both stable conformations of Watson-Crick-base-paired DNA. However, B-DNA has been proven to be more stable. Furthermore, DNA is a highly charged molecule with many anionic oxygen atoms in the phosphate backbone. Thus, it is expected that the treatment of long-range electrostatic interactions has a vast impact on DNA stability and structure.

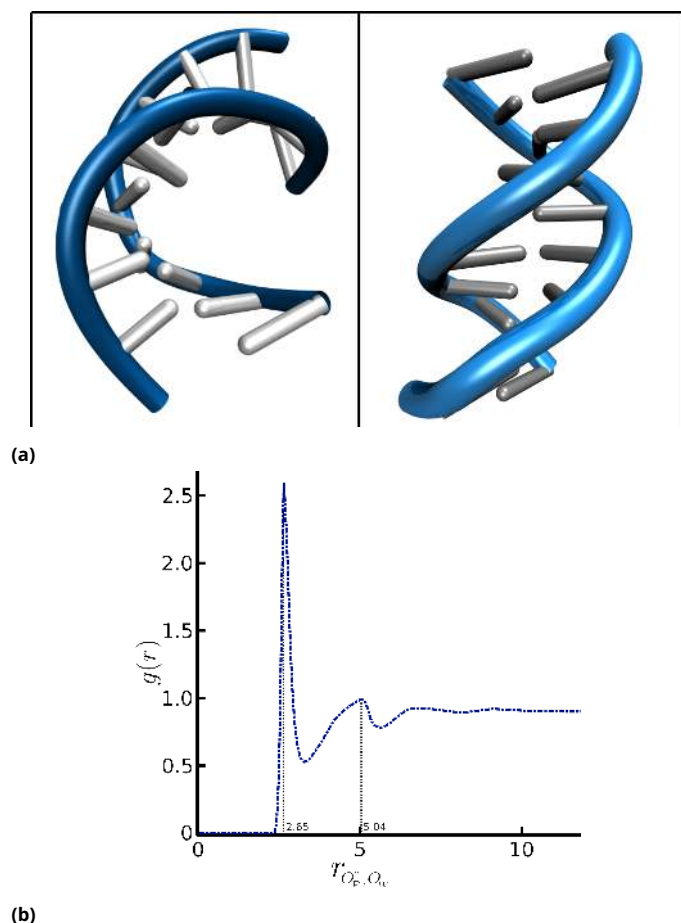


Figure 19. a) A-DNA and B-DNA conformations (visualized by VMD). B-DNA exhibits the classical helix shape. b) RDF calculated by Jaiswal et al.[36] between the uncharged phosphate oxygen atoms and the water oxygen atoms. The peak positions are indicated on the x-axis.

The files necessary for the simulations can be downloaded from the online repository. The setup consists of six electrostatic interaction schemes (same as in the prior task). Two PDB files, pictured in figure 19a, contain the coordinates of the DNA molecules. The usual GROMACS programs are used to generate a solvated simulation box from the PDB files.

```
gmx pdb2gmx [...] # Use AMBER99SB-ILDN and SPC/E water; also specify
                    flag: -ignh
gmx solvate [...] # solvates the simulation box; Use cubic box with
                    edge length 5 nm
gmx grompp [...] # just to generate tpr file for gmx genion
gmx genion [...] # replaces water with ions; just neutralise the DNA
```

After the system generation, the same MDP files as in the first section should be used to equilibrate the system. The electrostatic interaction schemes become important only for the pre-production and production run, from which the data is collected for analysis.

The information of interest consists of the RMSD, the RDF, and the timings of the production run for each method

since the calculation of non-bonded interactions is the major bottleneck of the performance. The timings are readily collected from the LOG file. In order to extract the other information from the trajectories, they need to be augmented first. Augmentation in the sense that the DNA molecules should be centered in the simulation box since they can be broken across periodic boundaries, which messes up the measurements. The GROMACS tool that can manipulate trajectories is called `gmx trjconv`, which is used in combination with an index file (NDX) to target a non-standard atom group. Herein, the non-standard group consists of one ssDNA strand, which is necessary because centering the DNA dimer can position each DNA strand on opposing periodic boundaries. GROMACS has its own tool to generate NDX files, fittingly called `gmx make_ndx`, which offers a command-line interface. Within this prompt, the user has to pick the correct atoms resembling the desired index group, e.g. 'a 1-315', followed by saving the new index group by typing `q` + `enter`. Since the website information for `gmx trjconv` is quite confusing, the correct options for the command are given below.

```
gmx trjconv -f <PROD-TRAJ>.xtc -s <PROD-STRUC>.tpr -center -n <INDEX>
            .ndx -pbc mol -o <WHOLE-TRAJ>.xtc
```

After execution, `gmx trjconv` prompts the user for which group to center (the ssDNA) and which group to output (the whole system). The RMSD of the trajectory is to be calculated in reference to its first frame, a.k.a. the last frame of the pre-production run. To make sure that the reference frame is not split across periodic boundaries, `gmx trjconv` needs to be called for `<PRE-GRO>.gro` as well.

```
gmx trjconv -f <PRE-GRO>.gro -s <PRE-STRUC>.tpr -center -n <INDEX>
            ndx -pbc mol -o <WHOLE-GRO>.gro
```

The resulting trajectory can then be used to calculate each system's RDF and RMSD curves (The RDF can also be calculated with the non-augmented trajectory). Herein, the RDF should be determined between the neutral phosphate oxygens and the water oxygens (see figure 19b for plot appearance and reference peak positions). Thus, the plot appearances should be compared qualitatively, whereas the peak locations should be compared quantitatively. The command `gmx rdf` is recommended to fulfill this task.

Regarding the RMSD, two measurements are of utmost importance: the time-resolved RMSD and the average RMSD of the production run. The reference structure that is always needed to calculate an RMSD is the prepared `<WHOLE-GRO>.gro` file and needs to be specified in the `-s` flag of `gmx rms`. The report should include RMSD plots of all the different methods and a comparison with figure 20, assessing which cutoff scheme produces realistic RMSD curves. Furthermore, a brief discussion of the average

RMSDs for each method should be included. Eventually, all results from the DNA and the water simulations should be used to select the simulation setup that appears most suitable to calculate electrostatic interactions, with a brief explanation. While timing information is crucial, the quality of the results is to be ranked as more important in this evaluation.

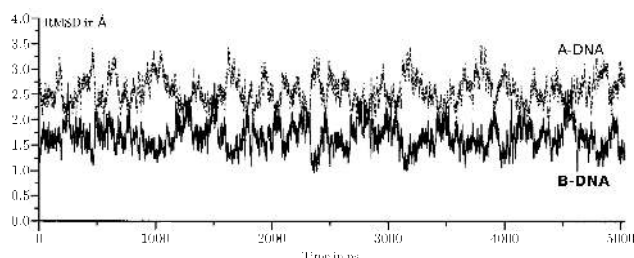


Figure 20. Exemplary time resolved RMSD. Modified from Norberg et Nilsson.[28] B-DNA is slightly more stable, which results in an overall lower RMSD. However, the fluctuations seem comparable to those seen in the visual inspection.

9 Exercise 6: Calculation of the Solvation Free Energy of Methane Using the MBAR Method

TASKS FOR THE SIXTH EXERCISE

Finish these tasks and write a short report about your findings.

- ☐ Complete the missing parts in the two bash scripts `run_lambda_simulations.sh` and `generate_mdp_files.sh` and include your code suggestions in the report (as a screenshot or similar).
- ☐ Use `gmx bar` to generate your `bar.xvg` and `barint.xvg` and include the plots in your report with a brief discussion.
- ☐ Discuss your result for ΔG_{solv} and compare it to the one in the literature.

Acknowledgement

This exercise was inspired by a tutorial offered by Justin Lemkul.[37]

Introduction

In contrast to prior exercises that focused on exploring different algorithms, this exercise centers on a specific application: calculating free energy changes. Free energy calculations are critically important in computational biophysics because they quantify various biologically relevant processes, such as stability alterations due to point mutations, small ligand binding, and solvation free energy of organic molecules.[38] Setting up and analyzing these simulations involves numerous choices experienced users must carefully select.[39] The system of interest in this exercise is an aqueous system with a single methane molecule. The goal is to calculate the solvation free energy, ΔG_{solv} , for the methane molecule. This exercise will also convey an understanding of such calculations' physical and mathematical aspects and the computational framework provided by GROMACS. While an in-depth exploration of free energy calculations is beyond the scope of this document, additional sources can be consulted for further information.[40–42]

The property to be calculated, ΔG_{solv} , was reported in 2003 by Shirts et al. They found a value of $2.42 \text{ kcal mol}^{-1}$ for the OPLS-AA forcefield.[43] By following the steps described below, it should be possible to replicate their results with reasonably good agreement.

Theoretical Considerations

Solvation free energy, ΔG_{solv} , also known as the Gibbs free energy of solvation, is a fundamental concept in physical chemistry. It quantifies the free energy change when a solute dissolves in a solvent, usually under standard conditions (pressure of 1 bar, temperature of 298.15 K, and concentration of 1 mol L⁻¹). The process of solvation entails changing the whole chemical environment of the solute. Thus, the solute might rearrange because its intra-molecular interactions adjust to the solvent contacts that suddenly present themselves. Exposed hydrophobic entities would aggregate within each other, while buried hydrophilic entities might become solvent-accessible. Especially for complex molecules and solvents (it does not have to be water), there is no way to intuitively estimate the solvation free energy given the multitude of (competing) enthalpic and entropic driving factors. Since the Gibbs free energy quantifies the spontaneity of chemical processes, finding a way to calculate this property from first principles would render such an estimate possible. Then, a negative ΔG_{solv} would indicate a readily dissolving solute. The general thermodynamic equation to calculate ΔG_{solv} for a binary solution of the solute and the solvent entails three states and their free energy. The solute alone (G_{solute}), the solvent alone (G_{solvent}) and the two components mixed (G_{solution}). The process of solvation associated with ΔG_{solv} brings them both together. Thus, it is the difference between the mixed and the unmixed systems.

$$\Delta G_{\text{solv}} = G_{\text{solution}} - G_{\text{solute}} - G_{\text{solvent}} \quad (39)$$

The control granted by computational methods enables the user to design a meticulous strategy to conduct this calculation. Technically, a simulation could be conducted, where the reverse process is probed, and the solute is magically pulled out of the solvent into the gas phase (vacuum). The force necessary to separate the two components would be related to ΔG_{solv} . However, a more elegant approach for the system of interest is the so-called *alchemical route*. Alchemical means that computational operations are conducted that are impossible in real-world physics, e.g., making a particle disappear into nothingness. Calculating the solvation free energy via the alchemical route means decoupling the van der Waals and Coulombic interactions between the solute and the solvent, effectively transforming the solute into a *ghost particle* that is formally part of the simulation but does not affect the system anymore, i.e., does not interact with its surroundings anymore. The fully interacting state is usually denoted as state A and the fully decoupled (ghost-like) state B. Despite this seemingly unphysical procedure, the alchemical route has a rigorous mathematical framework and yields fairly accurate results.^[44, 45] With the alchemical

route and equation (39) in mind, a thermodynamic cycle can be constructed to showcase the required simulation setup. Starting from the complete solution, hence G_{solution} in equation (39), the solute and the solvent are fully interacting (see figure 21). Since decoupling the solvent is computationally expensive because of the sheer number of molecules, decoupling the non-bonded parameters of the solute is the sensible choice. The next step would be to transfer the non-interacting solute out of the solvent into a vacuum. This process does not require work and thus does not contribute to ΔG_{solv} . As a last step, the intermolecular interactions are turned on again (the decoupling is reversed), and the solute adjusts its conformation according to its vacuum environment. This conformational change is non-existent for a methane molecule since directly connected atoms have bonded interactions only. Thus, the vacuum contribution $\Delta G_{\text{A} \rightarrow \text{B}}^{\text{vac}}$ results in zero as well.

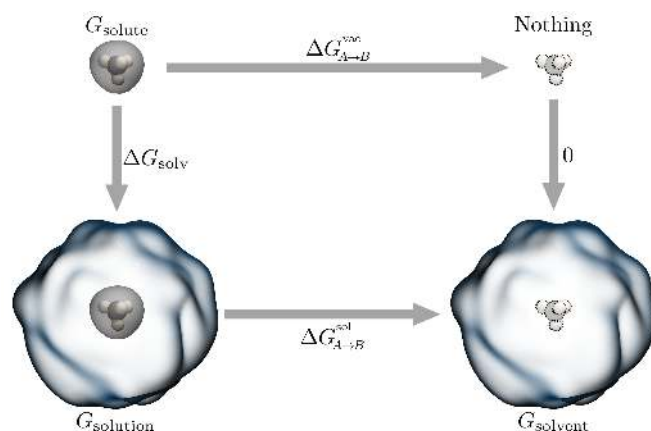


Figure 21. The solute, here a methane molecule, is transferred from a vacuum into water (left arrow). There are two intermediate systems shown on the right side. The upper one only comprises the decoupled solute, visualized by transparency and dotted outlines, and the lower system shows the decoupled solute in the solvent. Each system is labeled with its respective term from equation (39). In a thermodynamic cycle, only the end states matter; therefore, the alchemical route over two intermediate systems (three right arrows) can be chosen to indirectly obtain ΔG_{solv} . The contribution from each arrow needs to be summed up, yielding the solvation free energy.

Therefore, the only contribution that is necessary to calculate the solvation free energy is $\Delta G_{\text{A} \rightarrow \text{B}}^{\text{sol}}$, which denotes the alchemical annihilation of the solute within the aqueous solution. However, the annihilation cannot be conducted within a single simulation. A considerable energetic and conformational overlap must exist for two states to be eligible for free energy calculation.

Therefore, the solute does not vanish in one step but via a multistep approach called the coupling strategy which is usually optimized by experienced scientist to enhance results. The coupling parameter λ is introduced, which ranges be-

tween zero and one and specifies the decoupling progress from state A ($\lambda = 0$) to B ($\lambda = 1$). Theoretically, it can be applied to the complete Hamiltonian of the system containing the kinetic and potential energy or only to individual parts, e.g., the Lennard-Jones potential $U_{\varepsilon,\sigma}$ (part of the potential energy U).

$$U_{\varepsilon,\sigma}(r_{mn}, \lambda) = \frac{(1-\lambda)C_{12}^A + \lambda C_{12}^B}{r_{mn}^{12}} - \frac{(1-\lambda)C_6^A + \lambda C_6^B}{r_{mn}^6} \quad (40)$$

$$U_{\varepsilon,\sigma}(r_{mn}, \lambda) = \frac{(1-\lambda)C_{12}^A}{r_{mn}^{12}} - \frac{(1-\lambda)C_6^A}{r_{mn}^6} \quad \left| \quad C_{12}^B = C_6^B = 0 \right. \quad (41)$$

The constants C_6 and C_{12} are composite parameters containing ε and σ . The formula is identical to equation (13). The coupling strategy can be tuned to arbitrarily interpolate between both end states by selecting the appropriate λ spacing. For every λ , a molecular dynamics (MD) simulation has to be performed to sample the system in each λ -state. This procedure can be understood as yet another thermodynamic cycle, applied to the only step that remained from figure 21.

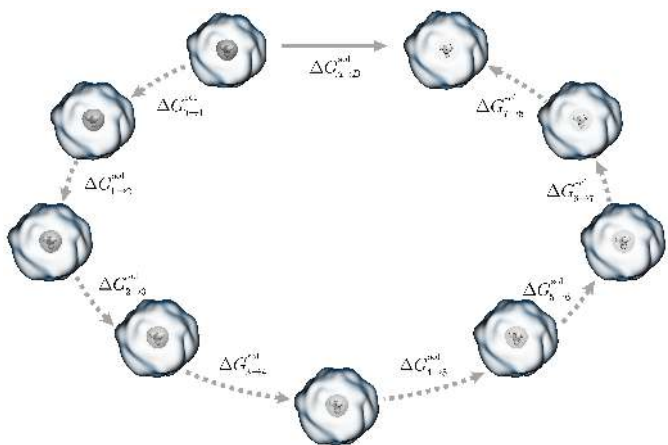


Figure 22. Visualisation of the coupling/decoupling strategy. The direct but error-prone calculation of $\Delta G_{A \rightarrow B}^{\text{sol}}$ is circumvented by a thermodynamic cycle that introduces seven unphysical intermediates utilizing the coupling parameter λ_n that assumes values between zero and one. Each λ -state is simulated, and the free energy contributions are then calculated with an analysis method, e.g., the Bennett acceptance ratio (BAR) method.

The Bennett acceptance ratio (BAR) method[42] is the method of choice for calculating the contributions showcased in figure 22, i.e. $\Delta G_{i \rightarrow j}^{\text{sol}}$ for the transition between to λ -states. The general idea is to sample configurations X_i with the Hamiltonian that corresponds to a certain λ_i , $H(\lambda_i)$. The Hamiltonian concerning a certain λ can be rewritten as U_i the potential energy since it is the only contribution affected by the decoupling strategy and for a consistent notation. The lengthy mathematical derivation can be looked up elsewhere[42]. Nonetheless, the final equation

utilizes $X_i = \{x_1, \dots, x_i, \dots, x_M\}$ and $X_j = \{x_1, \dots, x_j, \dots, x_N\}$ the configurations sampled from two adjacent Hamiltonians (i.e. simulations) with sample sizes N and M , and evaluates these configurations with their corresponding and each other's potential energy functions U_i and U_j .

$$\Delta G_{i \rightarrow j} = \ln \left[\frac{\sum_{x_i} f(U_i(x_i) - U_j(x_i) + C)}{\sum_{x_j} f(U_j(x_j) - U_i(x_j) - C)} \right] + C - \ln(N/M) \quad (42)$$

$$C = \Delta G_{i \rightarrow j} + \ln(N/M) \quad (43)$$

Here, the function $f(x)$ is the Fermi function $f(x) = \frac{1}{1+\exp(x)}$, and the constant C is to be determined by iteratively applying the above equations until the self-consistent results do not change anymore.

Setup and Simulations

Each part of this theoretical consideration has its computational counterpart. Due to the aforementioned relevancy of free energy calculations, this functionality has its own implementation via the MDP file. Also, the BAR analysis can be conducted using the `gmx bar` command. As usual, the forcefield and the system coordinates are given in the TOP and GRO files. It contains the methane molecule (called ALAB) and 750 TIP3P water molecules within the OPLS-AA force-field approach. In the downloadable TOP file, the charges of the methane molecule are set to zero, which is appropriate since the Coulombic term is negligibly small according to Shirts et al.[43]. Thus, incorporating a second decoupling strategy is here omitted.

The simulation phases comprise the usual energy minimization, NVT and NPT equilibrations, and production simulations. Every λ value requires its own set of simulations. Herein, the decoupling strategy consists of 21 equidistant λ values from 0 to 1, thus in steps of 0.05. Two additional bash scripts can be downloaded, which ease housekeeping of the variety of different output files, `generate_mdp_files.sh` and `(run_lambda_simulations.sh)`. The first one generates all necessary MDP files for running all simulation phases for each λ from the downloadable source MDP files, and the second one executes all the runs in an automated fashion. However, these scripts are incomplete and must be augmented by the user where indicated. Once both scripts are fully functional, they can be executed in the following way:

```
./generate_mdp_files.sh
```

This bash script will prompt the user to decide if a test run should be conducted first. A test run means that all MDP files will only run for 500 frames, drastically reducing the simulation time to a few minutes, even on older hardware. The dummy output files produce unreliable free energy estimates but still result in a functional BAR analysis (see

below). Running the script in test mode is highly recommended and can be followed by denying the test run to generate MDP files with the appropriate number of frames as input for `run_lambda_simulations.sh`. Files generated with `generate_mdp_files.sh` are named, e.g., `em_0.mdp`, `nvt_1.mdp` or `prod_19.mdp`. In addition to the usual options these files also specify the coupling/decoupling schemes. The corresponding free energy options provided by GROMACS are quite flexible and complex. The relevant ones are elaborated in table 4. Although working parameters have been predetermined and adjusted in the source MDP, it is important to understand the free energy-related options before proceeding. For even further details, refer to the manual: manual.gromacs.org/documentation/2023-beta/user-guide/mdp-options.html

Table 4. Tabular description of parameters related to free energy calculations in GROMACS. On the left is the MDP-parameter with its corresponding value, and on the right is a brief description.

<code>free-energy = yes</code>	Interpolate between topology A to topology B via the λ -states. Any λ -states would be ignored if this option would be set to no.
<code>init-lambda-state = 0</code>	Specifies which column of the lambda vector should be used for the current simulation run. As shown below, λ -values are compiled into lists, and the corresponding column specifies the transition degree between states A and B. Zero means starting from the first column.
<code>delta-lambda = 0</code>	Increment per time step for λ . Instead of instantiating each λ -value and thus the alchemical transformation, the whole process can happen incrementally by specifying choosing non-zero for this option. However, this can result in serious errors and is therefore omitted.
<code>calc-lambda-neighbors = 1</code>	Controls the number of lambda values for which Delta H values will be calculated. E.g., for a value of 1 and a <code>init-lambda-state</code> of 10, energy differences concerning λ -states 9 and 11 are computed.
<code>vdw-lambdas = ...</code>	A list of λ -values for the transformation of van der Waals interactions.
<code>couple-moltype = Methane</code>	Name given in the TOP file [<code>moleculetype</code>] section that is to be alchemically transformed from state A to state B.
<code>couple-lambda0 = vdw</code>	Tweaks the kind of non-bonded interactions that are present at $\lambda = 0$. Here, charges are set to zero, and van der Waals interactions are considered.
<code>couple-lambda1 = none</code>	Tweaks the kind of non-bonded interactions that are present at $\lambda = 1$. All non-bonded interactions will be turned off. Since the final λ transition switches an infinity to zero, soft-core potentials are necessary to avoid singularities. Soft-core potentials modify the Lennard-Jones potential in a way that avoids infinite values.
<code>couple-intramol = no</code>	All intra-molecular non-bonded interactions for [<code>moleculetype</code>] <code>couple-moltype</code> are replaced by exclusions and explicit pair interactions (as they would without <code>free-energy</code> set to yes).
<code>nstdhdl = 10</code>	The frequency for writing $dH/d\lambda$ and possibly Delta H to <code>dhdl.xvg</code> .

The λ spacing might be the most important part of the alchemical route and is implemented within the λ vectors in each MDP file. The λ vectors for any `*.mdp` contains 20 columns and six rows.

```
; init_lambda_state 0 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18
19 20
vdw-lambdas = 0.00 0.05 0.10 0.15 0.20 0.25 0.30 0.35 0.40 0.45
0.50 0.55 0.60 0.65 0.70 0.75 0.80 0.85 0.90 0.95 1.00
coul-lambdas = 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00
0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00
bonded-lambdas = 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00
0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00
restraint-lambdas = 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00
0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00
mass-lambdas = 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00
0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00
temperature-lambdas = 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00
0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00
```

Although `vdw-lambdas` is the only non-zero row, the others are still listed to showcase what else might be specified. The option `init-lambda-state = 0` specifies the Hamiltonian with the 0-th element in the `vdw-lambdas` list, here `0.00`. Setting `initial-lambda-state = 1` would correspond to the next element, i.e. `0.05`, while `initial-lambda-state = 20` would specify the final column, resulting in a λ -value of `1.00` for van der Waals interactions meaning they would be turned off completely according to equation (41).

The other bash script is called `run_lambda_simulations.sh`. It also needs to be completed, and one of the tasks is to fill in the gaps and append code suggestions to the report. After successful completion, `run_lambda_simulations.sh` should first create the following directory hierarchy:

```
lambda_0/
lambda_1/
lambda_2/
lambda_3/
lambda_4/
```

This way, all steps in the workflow are executed within a single directory for each value of `init-lambda-state`, which is a convenient way to organize the simulations and their output. Then, the aqueous methane system is simulated, and if no error occurs, the terminal output should look something like the below:

```
Starting minimisation for lambda = 0 ...
```

```
#####
# ENERGY MINIMIZATION STEEP #
#####

#####
# NVT EQUILIBRATION #
#####

#####
# NPT EQUILIBRATION #
#####
```

```
#####
# PRODUCTION MD #
#####
.
.
.
```

However, should there be a mistake, the bash script should report an error to the user, as seen below.

```
Starting minimisation for lambda = 0 ...
```

```
#####
# ENERGY MINIMIZATION STEEP #
#####
```

```
something went wrong, check /long/absolute/path/to/lambda_0/em/
mdrun.em
```

Notice how the bash script specifies a file containing more information on what might have caused the error. All the simulations should be executed automatically since the missing parts were filled in correctly. The whole workflow can take quite a long time (**about 8 hours on 4 CPUs**); therefore, the simulations should be started as early as possible. Once the production simulations are complete, the resulting data can be analyzed.

Utilizing bash scripts can be a double-edged sword. Firstly, writing a correctly working bash script takes time; sometimes, it could take longer than setting things up manually. Secondly, if there is an error somewhere in the bash script, simulations will not finish or end up the way they were intended, and the whole bash script must be executed again.

However, both these drawbacks can be mitigated. Firstly, looking at the workload of this exercise, there is a total of 84 simulations to run. Seamlessly executing all `gmx grompp` and `gmx mdrun` commands by hand would take a long time, so a bash script is worth it. Secondly, bash scripts have numerous benefits, like reproduction, documentation, and they can be reused, to name just a few. Last but not least, the fun and challenge of coming up with a working bash script is not to be underestimated as a motivating factor.

Analysis

The BAR module of GROMACS calculates ΔG_{solv} . Therefore, all `prod.xvg` files produced by `run_lambda_simulations.sh` must be included in the `gmx bar` command, which can be achieved by copying all the files in a shared directory or (easier) by specifying the correct GLOB. The correct command can be seen below if the directory structure is unchanged.

```
gmx bar -f lambda_*/prod.xvg -o -oi
```

In addition to producing `bar.xvg`, `barint.xvg`, and `histogram.xvg`, the program will print lots of useful information to the terminal, which should look something like this:

Detailed results in kT (see help for explanation):

```
lam_A lam_B DG +/- s_A +/- s_B +/- stdev +/-
0 1 0.08 0.00 0.03 0.00 0.04 0.00 0.24 0.00
. . . . .
19 20 -0.05 0.00 0.00 0.00 0.00 0.00 0.10 0.00
```

Final results in kJ/mol:

```
point 0 - 1, DG 0.19 +/- 0.00
. . . . .
point 19 - 20, DG -0.13 +/- 0.00

total 0 - 20, DG -9.23 +/- 0.07
```

Examining the content of the output files is essential, as they provide a comprehensive understanding of the decoupling process and the effectiveness of the sampling. From the DG values (corresponding to $\Delta G_{i \rightarrow j}^{\text{sol}}$ in figure 22), it becomes evident that some transitions lead to an increase in the free energy of solvation, while others improve the solvation of methane. The values printed for `s_A` and `s_B` represent the configurational overlap between states *A* and *B*. This information is crucial for assessing the adequacy of sampling and the reliability of the calculated free energy measures.^[46] The `bar.xvg` file contains the relative free energy differences for each λ interval (i.e., between neighboring Hamiltonians). It should look similar to figure 23 when plotted with `xmgrace`.

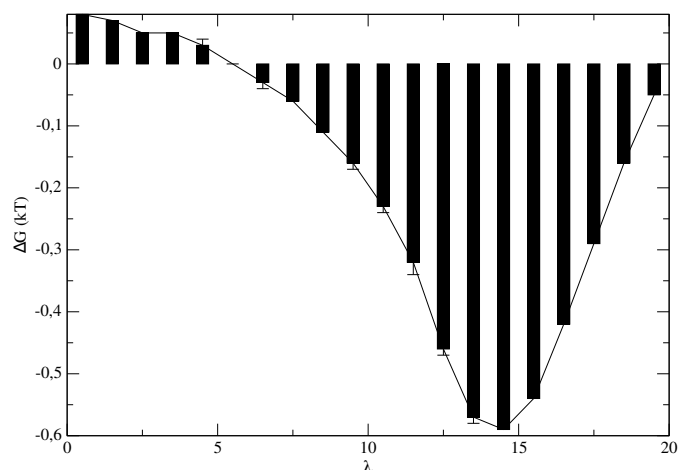


Figure 23. The x-axis contains the discrete λ_i values utilized in the decoupling process conducted here. The y-axis contains the relative free energy difference between two neighboring λ_i values shown as black bars.

The free energy differences between simulation *i* and simulation *i* + 1 are calculated via equation (42). GROMACS does not use the configurations but instead directly writes out the potential energies $U_i(x_j)$, $U_i(x_i)$ and so on, every `nstdhdl` = 10 steps. The total solvation free energy can be obtained by summing up all these intermediate free energy changes.

The `barint.xvg` file plots the cumulative ΔG as a function of λ . Thus, λ_{20} corresponds to the sum of ΔG from λ_0 to λ_{20} . In figure 24 `barint.xvg` is shown together with the data of `bar.xvg` illustrating the cumulative ΔG .

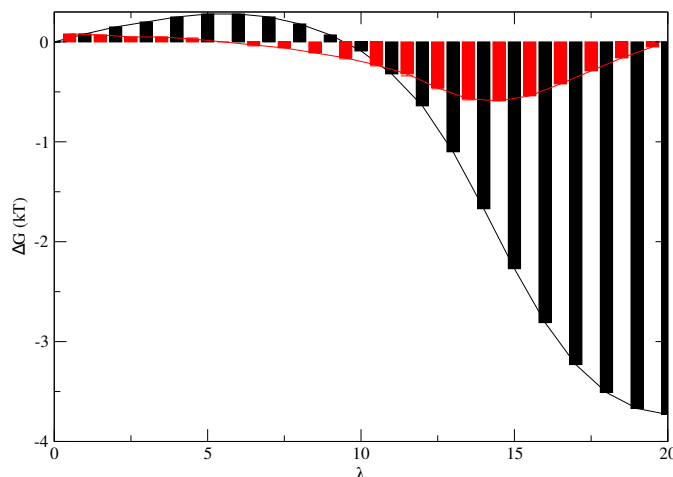


Figure 24. The x-axis contains the λ -increments, and the y-axis shows the free energy difference in units of kT . Additionally to the black bars, the cumulative free energy differences are shown as red bars.

By taking the value of ΔG at λ_{20} , we can recover the value of ΔG_{solv} by converting from kT to kcal mol^{-1} . The literature value for the solvation free energy of methane is $(2.42 \pm 0.01) \text{ kcal mol}^{-1}$ (see Shirts et al.^[43]). Report the output from `gmx bar` and how well your simulation setup reproduces the literature value. Increasing the simulation time or decreasing the lambda spacing may lead to improved agreement with the publication, but this would be voluntary. Apart from these two possibilities, the report should discuss other options that may increase the accuracy of the results.

10 Exercise 7: Potential of Mean Force of an Amyloid Layer Separation via Umbrella Sampling

TASKS FOR THE SEVENTH EXERCISE

Finish these tasks and write a short report about your findings.

- ☐ Comment `02_simprep.sh` line by line and include it in the report (as a screenshot or similar).
- ☐ Include plots of the histograms and the PMF and discuss their appearance.
- ☐ Report your result for ΔG_{bind} and compare it to the literature ($-50.5 \text{ kcal mol}^{-1}$).

Acknowledgement

This exercise was inspired by a tutorial offered by Justin Lemkul[37].

Introduction

This exercise explores the usefulness and applicability of free energy calculations, building on the previous exercise, which focused on solvation free energy via alchemical transformations. Here, we estimate the free energy of binding by sampling two binding partners at various spatial distances. The larger binding partner is usually called the receptor, and the smaller one is the ligand. Receptor-ligand complexes are crucial in biomolecular signal transduction, enzymatic catalysis, and therapeutic efficacy.[47] Binding free energy - the energy change when a ligand binds to its receptor - is fundamental in understanding these processes. As with solvation free energy, there are many approaches to obtaining the binding free energy, each with its strengths and weaknesses.

For this exercise, we analyze the A β (1-42) amyloid fibril associated with Alzheimer's disease using molecular dynamics (MD) simulations.[48] This amyloid protofilament consists of five peptides with beta sheets interconnected via their side chains. The 42 amino acid-long peptides are part of the protofilaments.[49] In a series of simulations, the terminal peptide of the protofilament is pulled away from the remaining amyloid fibril. It is extensively sampled along the distance coordinate, also known as a *collective variable*. Sampling along this collective variable is achieved using harmonic biasing potentials in a procedure known as umbrella sampling (US) simulations.[50] This method results in distance distributions that resemble an umbrella shape. The sampling data is then evaluated using an advanced

algorithm to produce a free energy surface, from which the pulling free energy is obtained.[51]

Lemkul and Revan precisely calculated the potential of mean force (ΔG_{bind}) for this system, predicting a value of $-50.5 \text{ kcal mol}^{-1}$. [52] this exercise aims to replicate this value using GROMACS built-in functionalities for conducting umbrella sampling simulations.

Theoretical Considerations

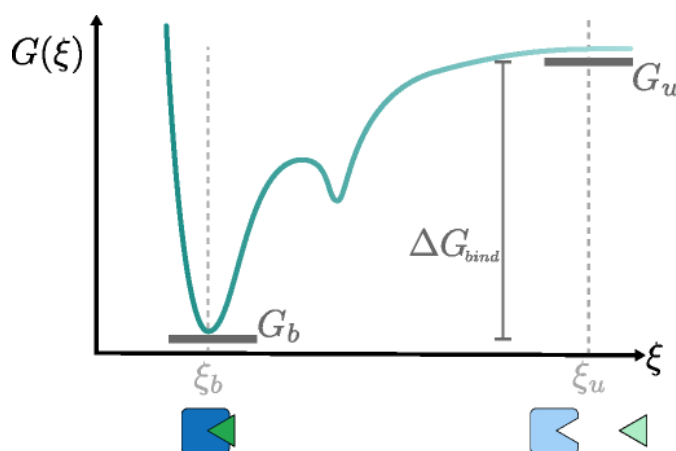


Figure 25. Visualisation of the binding process along the collective variable ξ . The free energy G approaches a minimum for the bound state, described by ξ_b . For further clarification, the bound and unbound complexes are depicted with geometrical shapes beneath the x-axis.

The binding free energy ΔG_{bind} is generally defined as the difference between the free energy of the bound state G_b and the unbound state G_u . Thus, the binding free energy represents the reversible work released during complex association. The free energies of the bound and unbound states are related to the average probability $\langle P \rangle$ of encountering these states in a given environment, such as an aqueous solution.[53]

$$\Delta G_{\text{bind}} = G_b - G_u = -k_B T \ln \left(\frac{\langle P_b \rangle}{\langle P_u \rangle} \right) \quad (44)$$

The probabilities given in equation (44) can be defined concerning an arbitrary *collective variable* ξ along which the binding can be described. For ligand binding phenomena, a classical choice for ξ is the center-of-mass (COM) distance with the receptor, denoted as r_{com} . However, any observable can be chosen for ξ , e.g., dihedral angles or RMSDs. The average probability of encountering a certain r_{com} distance can be obtained only via the partition function, Z_{NPT} .

$$\langle P(r_{\text{com}}) \rangle = \frac{\int_{\mathbf{X}} d\mathbf{X} \delta(r_{\text{com}} - R_{\text{com}}(\mathbf{X})) e^{-\beta H(\mathbf{X})}}{Z_{\text{NPT}}} \quad (45)$$

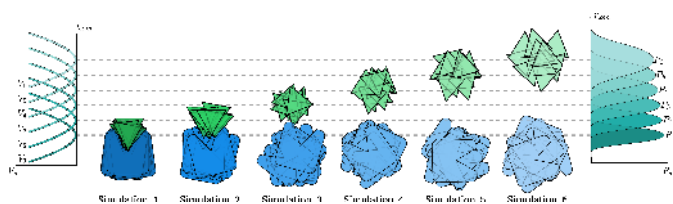


Figure 26. Conceptual visualization of umbrella sampling for an arbitrary receptor-ligand complex. The harmonic potentials on the left keep the ligand (green triangle) restrained at a certain distance from the receptor (blue rectangle). The sampling is visualized via superimposed copies of the respective molecules in the six simulation windows. The resulting COM distances are shown in the graph on the right as probability distributions.

The function $R_{\text{com}}(\mathbf{X})$ evaluates the COM of the configuration \mathbf{X} and is compared to an arbitrary r_{com} value via the Dirac delta function, $\delta(\dots)$. Integration occurs over all configurations of the system, $\int_{\mathbf{X}}$, causing the dependency on \mathbf{X} to vanish for $\langle P(r_{\text{com}}) \rangle$. The bound and unbound states can now be defined concerning the reaction coordinate: beyond a certain COM distance threshold, the intermolecular interactions cease; thus, the binding partners are considered fully separated. Unfortunately, this equation cannot obtain ΔG_{bind} from simulation since the system's partition function is practically unobtainable. The intermolecular attraction is so high for tightly bound complexes that dissociation events would never be sampled within feasible MD simulation times. Therefore, a different approach is used, employing harmonic potentials along the COM distance to enhance sampling across all regions of the binding/unbinding process. The exact *stratification strategy* may vary, but typically, for binding processes, multiple simulations (s) are conducted, each with a biasing potential U_s resembling a harmonic spring potential. This potential is defined by the spring constant k_s and the equilibrium COM distance r_{com}^s .

$$U_s(r_{\text{com}}) = k_s (r_{\text{com}}^s - r_{\text{com}})^2 \quad (46)$$

Each biased simulation results in a biased probability distribution, P_{com}^s , that preferentially samples r_{com} values close to the reference distance r_{com}^s (see figure 26). However, it can be shown mathematically that the expression for P_{com}^s still depends on the partition function Z_{NPT} and the unbiased probability distribution $\langle P(r_{\text{com}}) \rangle$.^[51]

$$\langle P_s(r_{\text{com}}) \rangle = \frac{Z_{\text{NPT}}}{Z_{\text{NPT}}^s} \langle P(r_{\text{com}}) \rangle e^{-\beta U_s(r_{\text{com}})} \quad (47)$$

This equation can be rearranged for the unbiased probability as follows:

$$\langle P(r_{\text{com}}) \rangle = \langle P_s(r_{\text{com}}) \rangle \cdot \frac{e^{-\beta U_s(r_{\text{com}})}}{e^{-\beta U_s(r_{\text{com}})}} \quad (48)$$

Seemingly, not much is achieved by performing multiple simulations with biasing potentials, as $\langle P(r_{\text{com}}) \rangle$ and the unbiased

partition function remain unknown. The partition function of the biased simulation would also need to be calculated. Nevertheless, Ferrenberg and Swendsen^[54] developed the weighted histogram analysis method (WHAM),^[55] which employs similar mathematical techniques to those implemented in the BAR method from section 9.^[42] WHAM operates using histograms of the collective variable; instead of continuous values of r_{com} , only certain histogram bins with their center values i are considered.

The lengthy and complex derivation of the WHAM equations is beyond the scope of this document. However, the resulting two equations will be discussed in detail. The objective is to accurately estimate the unbiased, discretized probability distribution $\langle P(i) \rangle$. Sampled COM distances from each simulation are collected and binned into a histogram with N_{bins} , chosen by the user. Consequently, the resulting histogram $h_s(i)$ from simulation s should contain counts mostly close to the reference distance given by the corresponding harmonic potential U_s . The sum of all simulation histograms is then divided by a self-consistent term, determined iteratively (see also figure 27).^[56]

$$\langle P(i) \rangle = \frac{\sum_s^{N_{\text{sim}}} h_s(i)}{\sum_s^{N_{\text{sim}}} n_s \cdot \exp(-\beta(U_s(i) - f_s))} \quad (49)$$

$$f_s = -k_B T \ln \left(\sum_i^{N_{\text{bins}}} \langle P(i) \rangle \cdot \exp(-\beta U_s(i)) \right) \quad (50)$$

The so-called free energy constants f_s are initialized to zero and approach convergence after a certain number of iterations.^[51] These constants correspond to the free energy of the biased simulation s but do not relate directly to the free energy of binding. To obtain ΔG_{bind} , the unbiased probability $\langle P(i) \rangle$ is used to calculate a free energy surface (FES) $G(i)$.^[56]

$$G(i) = -k_B T \ln \left(\frac{\langle P(i) \rangle}{\langle P_{\text{max}} \rangle} \right) \quad (51)$$

Dividing by the maximum of the probability distribution shifts the minimum $G(i)$ to zero. This zero-shift is common practice since the free energy surface can only be calculated up to a constant, and free energy differences are usually of interest (compare figure 25).

Setup and Simulations

The computational implementation of the described theoretical background is completely implemented in the GROMACS software. However, the specifics are still quite complicated. The steps can be categorized into obtaining equilibrated molecular coordinates and topologies, generation of initial

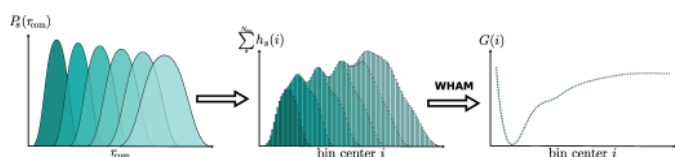


Figure 27. Depiction of data generation and the WHAM algorithm. The COM distances are collected from the N_{sims} simulations. The binned histograms are summed, and then the WHAM equations are applied iteratively until the unbiased probability distribution $\langle P(i) \rangle$ emerges, from which the free energy surface $G(i)$ can be readily obtained.

configurations along the r_{com} coordinate via a pulling simulation, and eventually, the umbrella sampling simulations, which can be subcategorized in equilibration and production simulations. The latter step can take several days. Therefore, it is recommended that these simulations start as soon as possible. Should the following analysis reveal missing bins in the histograms or generally deviating free energy results, additional sampling can be conducted in regions with insufficient sampling, so-called post-sampling. The analysis can then be rerun, incorporating the new data, hopefully leading to an improved result.

The system of interest is the A β (1-42) amyloid protofilament consisting of five peptides, and the process that shall be investigated is the dissociation of the terminal peptide. The protein complex has the PDB identifier 2BEG and can be downloaded from the PDB databank or the online repository (recommended). Apart from the original PDB file, the other necessary files entail the TOP, GRO, and MDP files. Both the TOP and GRO files can be created by GROMACS' toolkit, but the MDP files need to be downloaded from the online repository. Most US-specific differences are concentrated in the MDP file. However, this time, the TOP file will require slight alteration, and the GRO file will be elongated to better encompass the separation instead of its usual cube shape.

The GROMACS input files are generated in the usual way, firstly converting the PDB to a GRO file, specifying the box and then solvating it, followed by ion generation. The commands can be found below. The resulting file that can be forwarded for minimization is called `solv_ions.gro`.

```
gmxd pdb2gmxd -f 2BEG_model1_capped.pdb -ignh -ter -o complex.gro #
Choose GROMOS96 53A6; SPC; all N-termini: None; all C-Termini
COO-
gmxd editconf -f complex.gro -o newbox.gro -center 3.280 2.181
2.4775 -box 6.560 4.362 12 # box size complies with minimum
image convention and extend of pulling
gmxd solvate -cp newbox.gro -cs spc216.gro -o solv.gro -p topol.top
# solvate with SPC water
gmxd grompp -f ions.mdp -c solv.gro -p topol.top -o ions.tpr #
generate dummy TPR for ion generation
gmxd genion -s ions.tpr -o solv_ions.gro -p topol.top -pname NA -
nname CL -neutral -conc 0.1 # generate NaCl with c=0.1M
```

The resulting input GRO file can be using `vmd` on `solv_ions.gro`.

The box can be explicitly visualized by clicking (Extensions → Tk Console), which should open a new window. Insert the command `pbx box` and press enter. The resulting view should match the one in figure 28.

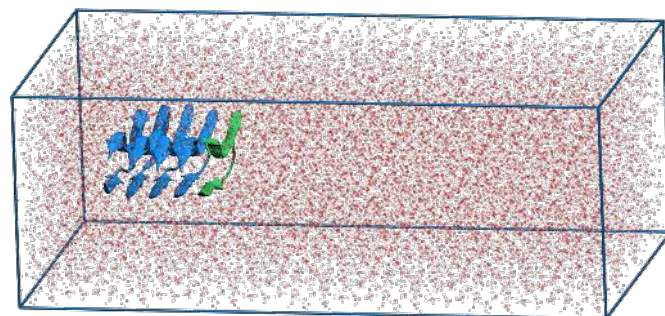


Figure 28. Example image of what the simulation box should look like for this exercise. The cartoon represents the amyloid, and the color scheme corresponds to figure 26.

Another issue needs to be addressed after the correct simulation box setup. The second chain of the amyloid fibril, Chain B, needs to be fixed in space, which can be realized by positional restraints that restrain each atom to its initial XYZ coordinates. Positional restraints differ from distance restraints because they keep atoms close to absolute reference positions, while distance restraints only restrain coordinates relative to each other. The positional restraints are activated by adding the appropriate lines to the `topol_Protein_chain_B.itp` file (`itp` stands for 'include topology'). The lines to be added are shown below. The keyword `POSRES_B` is critical since the MDP file uses this keyword to implement the positional restraints algorithmically.

```
#ifdef POSRES_B
#include "posre_Protein_chain_B.itp"
#endif
```

This last manual step concludes the input file generation. The system must be minimized and equilibrated before a pulling simulation is conducted to obtain configurations along the COM distance coordinate. The MDP files can be downloaded from the online repository and used to execute the commands as usual.

```
gmxd grompp -f em.mdp -c solv_ions.gro -p topol.top -o em.tpr
gmxd mdrun -v -deffnm em
gmxd grompp -f npt.mdp -c em.gro -p topol.top -r em.gro -o npt.tpr
gmxd mdrun -deffnm npt
```

The pulling simulation generates configurations to be used as starting frames for the individual umbrella sampling simulations. Thus, GROMACS' pulling functionality is utilized to steer the terminal peptide away from the remaining amyloid fibril in one simulation from which individual snapshots are saved within an approximately linear spacing along the COM distance coordinate (see figure 29). For GROMACS to pull

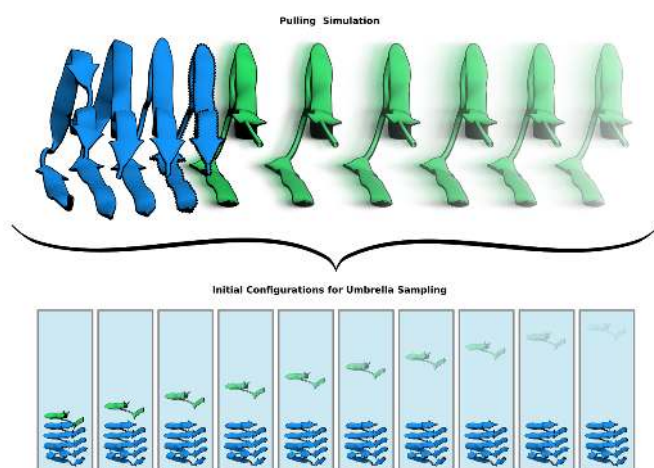


Figure 29. Concept visualization of the pulling simulation. Along the COM distance coordinate, the outermost peptide (green) is pulled away from the fibril (blue). The second outermost peptide is positionally restrained (pronounced outline). Along the reaction pathway, configurations are saved for the umbrella sampling.

a certain atom group, an appropriate index file must be created by the `gmx make_ndx` command that contains the indices of the atom groups to be separated. In this case, the atom groups of the two terminal peptides are saved to the `index.ndx` files with the names `chain_A` and `chain_B`. They correspond to the residues one to 27 and 28 to 54. The command `gmx make_ndx` can seem complex for beginners. Therefore, the correct series of commands to generate the index file can be seen below.

```
gmx make_ndx -f npt.gro
...
> r 1-27
> name 19 Chain_A
> r 28-54
> name 20 Chain_B
> q
```

The validity of `index.ndx` can be checked by opening the file via VIM and going to the end of the file (by typing `shift + G`). The non-standard index groups named `Chain_A` and `Chain_B` should be there now. GROMACS' pulling functionality is completely parameterised via the `md_pull.mdp` file (downloadable from the online repository). Below is a brief overview of the keywords that can and must be specified for successful pull simulations in table 5.

Table 5. Tabular description of parameters related to pulling simulations in GROMACS. On the left is the MDP-parameter with its corresponding value, and on the right is a brief description. For further details, refer to the GROMACS manual

<code>pull = yes</code>	All pull-related keywords are parsed by <code>gmx grompp</code> . All pull settings are ignored if set to no (default).
<code>pull-pbc-ref-prev-step-com = yes</code>	The reference distance for COM-Pulling is calculated from the periodic boundary state of the prior step. This option is required for newer GROMACS versions with <code>pull-group#-pbcatom</code> .
<code>pull_ngroups = 2</code>	Number of pull groups to be steered within the pull simulation. For distance pulling, two groups are required, while an angle would require three groups.
<code>pull_ncoords = 1</code>	Several pull coordinates can be steered simultaneously (useful for domain unfolding); however, only one pull coordinate is used here.
<code>pull_group#_name = Chain_X</code>	Numbers (#) and names of the corresponding pull groups as defined in the <code>index.ndx</code> . Two names must be provided with <code>pull_ngroups = 2</code> specified in this system. Here: Chain_A and Chain_B.
<code>pull-group#-pbcatom = 175</code>	The reference atom of both pull groups for determining the periodic boundary treatment for calculating the COM-distance between the pull groups. The atoms should be located in the center of their respective group. In this case, Atom 175 should fulfill this criterion for both amyloid chains.
<code>pull_coord1_type = umbrella</code>	Apply a harmonic potential to the pull coordinate number 1 to significantly limit the phase space accessible to the two pull groups.
<code>pull_coord1_geometry = distance</code>	For pull coordinate number 1, apply the harmonic potential to the distance of the two pull groups.
<code>pull_coord1_groups = 1 2</code>	pull coordinate number 1 should apply to pull groups 1 and 2. This option becomes increasingly important, especially if several groups and coordinates are defined.
<code>pull_coord1_dim = N N Y</code>	Apply a bias only in the z-dimension. Thus, x and y are set to "no" (N), and z is set to "yes" Y.
<code>pull_coord1_start = yes</code>	the starting value of the pull coordinate number 1 is the value it assumes in the starting configuration.
<code>pull_coord1_rate = 0.01</code>	The rate at which pull coordinate number 1 increases in nm ps^{-1} over the course of a simulation. If the pull coordinate refers to an angle/dihedral, this rate would be given in $^{\circ} \text{ps}^{-1}$.
<code>pull_coord1_k = 1000</code>	The force constant to parametrize the harmonic potential with, that was set in <code>pull_coord1_type</code> in kJ/mol/nm^2 . The corresponding variable can be found in equation (46).

Additionally, `md_pull.mdp` references the statement previously added to the `topol_Protein_chain_B.itp` file. The `define` keyword at the very beginning uses the same expression to activate the positional restraints of the Chain B (second outermost peptide). Not activating the positional restraints might lead to deformation of the amyloid fibril since the intermolecular interactions between the individual peptides are so strong that individual residues would be towed away together with Chain A. From the MDP file, it becomes apparent that throughout the 500 ps trajectory, the peptide is being pulled 5.0 nm away from the fibril. The keyword `nstxtcout`, which is set to 500 (every 1 ps), determines the frequency for writing out the compressed trajectory to the XTC file. This file will extract the starting configurations for the umbrella sampling. The bash script named `02_simpreg.sh` is downloadable from the online repository and automatically takes care of the simulation and frame extraction with a distance spacing of around 0.2 nm. Include a commented version of this bash script as part of the report. Especially the `awk` command is quite complex. Nevertheless, any attempts to correctly comment on it are appreciated. Once the run finishes, the success can be tested by plotting the contents of `pullf.xvg`, which records the force required over the simulation time. It should look similar to figure 30.

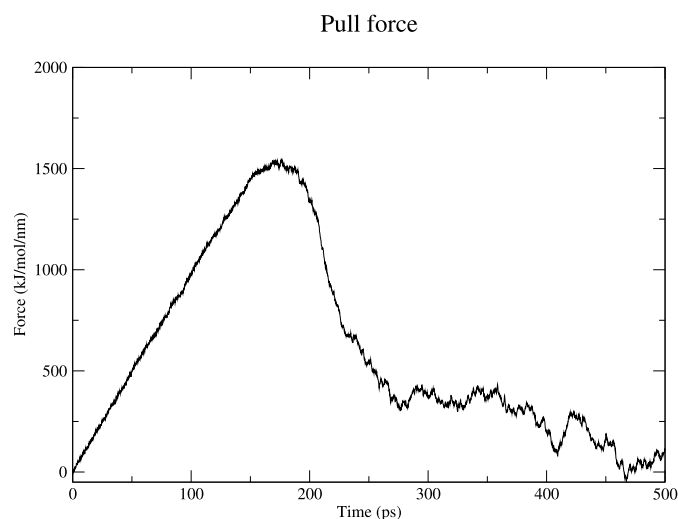


Figure 30. Force on the two pull groups throughout the simulation.

Apart from the pulling simulation outputs, the script creates two auxiliary files (`dist.xvg` and `important_frames.txt`) and a couple of GRO files, e.g., `conf191.gro`, that correspond to the frames listed in `important_frames.txt`. These GRO files serve as the starting configurations for the umbrella windows. The 0.2 nm spacing can be verified by comparing the frames listed in `important_frames.txt` with their corre-

sponding distances in `dist.xvg`.

The umbrella sampling simulations rely on the same GROMACS parameters as the pull run. Comparing `md_pull.mdp` with `md_umbrella.mdp` or `npt_umbrella.mdp` reveals that only the `pull_coord1_rate` differs by being set to zero for the umbrella sampling. Thus, the molecules are not actively pulled apart; a harmonic potential keeps the distance constant to its initial value. Each starting frame extracted by `02_simpreg` requires a quick NPT equilibration followed by a production run over 500 ps. The individual simulations are sometimes referred to as umbrella windows. A code example for executing the runs can be found below.

```
gmx grompp -f npt_umbrella.mdp -c conf<f>.gro -p topol.top -r conf<f>.gro -n index.ndx -o npt<f>.tpr
gmx mdrun -v -deffnm npt<f>
gmx grompp -f md_umbrella.mdp -c npt<f>.gro -t npt<f>.cpt -p topol.top -r npt<f>.gro -n index.ndx -o umbrella<f>.tpr
gmx mdrun -v -deffnm umbrella<f>
```

Creating a bash script to automate the simulation runs is highly recommended.

Analysis

With the umbrella sampling successfully conducted, the equilibrated data can be analyzed. Usually, ΔG_{bind} is calculated from the free energy surface (FES) obtained from MBAR or WHAM. The convenient `gmx wham` tool already implemented in GROMACS is showcased herein.[57] The value of ΔG_{bind} is simply the difference between the highest and lowest values of the FES (see figure 25), as long as the values of the FES converge to a plateau at large COM distances. It represents the potential of mean force (PMF) to separate the terminal peptide from the tip of the protofibril.

The input to be fed into `gmx wham` consists of two files to be created by the user. The first one contains line-wise entries of all TPR file names corresponding to the umbrella windows, and the other file includes the names of the `pullf.xvg` or `pullx.xvg` files, respectively. These lists could be called, e.g., `tpr-files.dat`; the contents are listed below for illustration.

```
umbrella45.tpr
umbrella123.tpr
...
umbrella487.tpr
```

Conducting the analysis is relatively simple since it only requires the user to execute the correct GROMACS command.

```
gmx wham -it tpr-files.dat -if pullf-files.dat -o -hist -unit kCal
```

The WHAM module opens each of the `umbrella*.tpr` and `umbrella*_pullf.xvg` files and runs the WHAM analysis on them. This program outputs two essential files named `histo.xvg` and `profile.xvg`, which contain the histogram

data and the resulting FES profile against the COM distance coordinate. They can be visualized using `xmgrace`.

```
xmgrace profile.xvg
xmgrace -nxy histo.xvg
```

As exemplary results, a FES is shown in figure 31. The difference from the highest to the lowest point of this FES according to equation (44) yields a ΔG_{bind} of around $-37 \text{ kcal mol}^{-1}$. The difference to the published value of $-50.5 \text{ kcal mol}^{-1}$ is quite significant; however, there are also artifacts visible in the FES around 1 nm, 1.5 nm, 3.5 nm and 4.3 nm that result from insufficient histogram overlap.

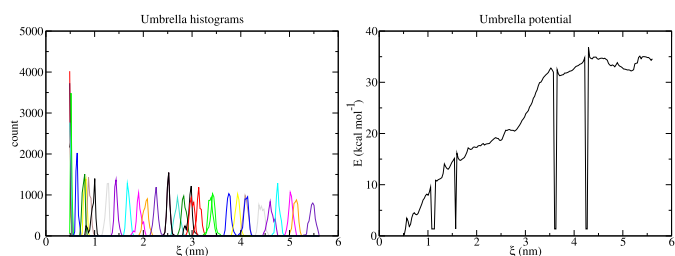


Figure 31. Distance probability distributions (histograms) for the different windows (left) and the corresponding FES along the COM distance coordinate (right).

Since each umbrella window is independent, these artifacts can be solved by post-sampling the missing distance histograms. Inserting new umbrella simulations with starting configurations centered on the defective region should mitigate the inaccuracy. The file `dist.xvg` contains frames on the first column and distance on the second column so that the correct snapshot to start a post-sampling simulation can be found. Once the desired frame is known, equilibration and production simulation can be conducted, and TPR and XVG files are added to `tpr-files.dat` and `pullx-files.dat`. Repeating the WHAM analysis should result in an FES that has fewer defects. An exemplary FES that contains no artifacts anymore after successful post-sampling is shown in figure 32. Here, ΔG_{bind} is obtained with 40 kcal mol^{-1} , which is better than before but still deviates with 20% from the literature value. There are many tools for quality control and optimization of free energy calculations[58], but for educational purposes, such a value is more than sufficient.

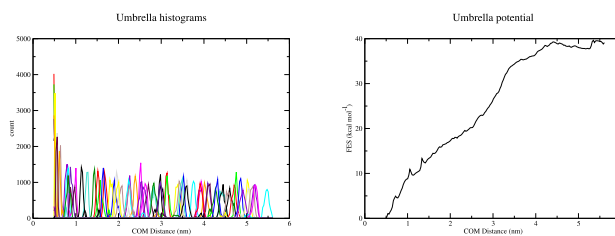


Figure 32. Free-energy profile (FES) along the reaction coordinate ξ (top panel) and corresponding distance probability distributions (histograms) for the different windows (bottom panel).

The report should include the histograms and the FES as plots, as well as the value obtained for ΔG_{bind} . A brief discussion of the plots and a comparison with the literature value should be included. Post-sampling is optional but recommended. Eventually, the FES, and thus the free energy estimate, can be further enhanced by collaborating with other users or students to collect as many TPR and PULL files as possible. After adjusting `tpr-files.dat` and `pullx-files.dat` (or `pullf-files.dat`), WHAM would analyze a vast amount of data, which could close the gap to the literature value.

11 Exercise 8: Protein Simulation with Convergence and Principal Component Analysis

TASKS FOR THE EIGHTH EXERCISE

Finish these tasks and write a short report about your findings.

- ☐ Write a Methods section describing your MD simulation setup.
- ☐ Monitor and report stability measures for the pre-production step of the simulation setup for Potential energy (EM), temperature (NVT), density, and pressure (NPT).
- ☐ Conduct a BSE analysis for R_{gyr} and RMSD(t) of your production run. Use the crystal structure as a reference and discuss the results with respect to sampling quality.
- ☐ Show the two extreme projections of the eigenvectors. Also, include a plot of the atomic contributions of eigenvectors 1-10 with a brief discussion.
- ☐ Plot PC1 vs PC2, PC1 vs PC10 and PC9 vs PC10 and discuss the plots briefly.

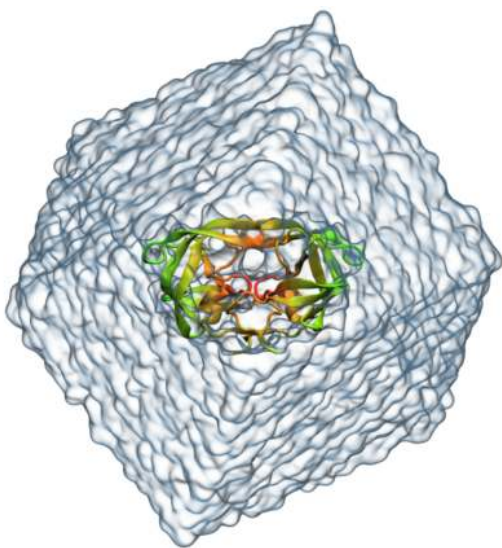


Figure 33. HIV-protease in a water box.

Introduction

In this exercise, the user conducts an equilibrium molecular dynamics (MD) simulation for the protein HIV-protease (PDB: 5YOK). As a novelty, the simulation setup in this exercise will be self-supervised. The methods applied in the prior exercises can be used as a template library and to draw inspira-

tion. Furthermore, this exercise is concerned with the convergence of the simulation with respect to the protein conformation for which a blocked standard error (BSE) analysis[59] will be conducted. Finally, the movements of the HIV-protease are of utmost importance due to their influence on reducing drug efficacy. The principal component analysis (PCA) will illuminate the global molecular movement of this drug-resistant variant.[60]

11.1 Theoretical Considerations

Assessing Sampling Quality and Convergence

Convergence in a simulation is achieved once additional sampling does not change an obtained result significantly. This definition aligns with the concept of relative convergence, which describes statistical uncertainty concerning an observable. While absolute convergence cannot be ensured or estimated in an MD simulation, relative convergence has been the subject of extensive studies, and one method to access the relative sampling quality is to conduct a blocked standard error analysis for an observable.[59]

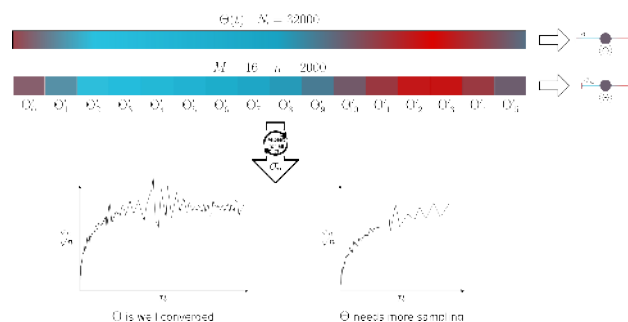


Figure 34. Concept of the analysis of block averages. When evaluated on a whole trajectory, an observable Θ yields average $\langle \Theta \rangle$ and standard deviation σ . The color indicates some arbitrary magnitude of the instantaneous quantity Θ . Let the total sample size be 32000, i.e., 32000 instantaneous values of Θ have been collected. Splitting the trajectory into M blocks, with n data points each, and then calculating each block's average yields the same overall average. However, the standard deviation of the block averages is different, $\sigma_n \neq \sigma$. Repeating this process for all possible M, n pairs produces a curve that indicates convergence for Θ . [59]

For Θ , statistical relevancy is only achieved once enough independent samples can be measured. Thus, measuring 30 consecutive frames of a slow movement will result in a sharply defined average value. However, it is not significant since the measurements can be assumed to be highly correlated. Correlated data is invalid for the application of the central limit theory, which states that for measuring infinitely many independent random variables, a normal distribution emerges. The BSE analysis transforms the initial, correlated observations of Θ by splitting the trajectory into M blocks m ,

each length n , and calculating the average of each block m , Θ'_m . From the M obtained values Θ'_m , the n -specific standard deviation is calculated.

$$\sigma_n = \sqrt{\frac{1}{M-1} \sum_{m=0}^M (\Theta'_m - \langle \Theta \rangle)^2} \quad (52)$$

At some point, by increasing n , the correlation time for Θ is exceeded, rendering Θ'_m effectively uncorrelated, which yields a normal distribution \mathcal{N} with fixed mean and variance, given a large enough sampling size.[61]

$$\lim_{M \rightarrow \infty} \Theta'_m \sim \mathcal{N}\left(\mu, \frac{\sigma_n}{\sqrt{n}}\right) \quad (53)$$

Since σ_n/\sqrt{n} resembles σ of the stable normal distribution, plotting it against the "time period" (i.e., block size n) used for block averaging, results in a horizontal plateau, but only if the samples are truly uncorrelated. For smaller values of n where Θ'_m is still correlated, the corresponding σ will steeply approach the stable standard error.

Principal Component Analysis of Atomic Movements

The Principal Component Analysis (PCA) is a two-step process that begins by calculating the covariance matrix of the atomic movements in the trajectory. An eigenvector decomposition is then performed on this matrix, resulting in $3N$ eigenvectors (principal components) and associated eigenvalues. The eigenvectors define the principal motions of the protein. These eigenvectors are typically sorted from highest to lowest eigenvalue, representing the largest and smallest protein motions, respectively. The higher principal components exhibit a higher degree of collectivity, meaning that a small number of high-level principal components account for the essential protein movement, which can provide insights into crucial functionalities.[62]

Mathematically, principal components are a set of new variables which are just linear combinations of the initial input variables. In the present description, the input variables are the atom coordinates. The coefficients of the linear combination are chosen so that the resulting principal components are uncorrelated, i.e., they carry independent information. The eigenvectors then denote the directions of largest variance or fluctuations (i.e., *information*) in the data, and the eigenvalues denote the amount of variance in each principal component.

Setup and Simulations

This time, the setup of the MD simulations is self-designed. Temperature, pressure, concentration, and other specifics should be reasonably close to physiological or laboratory conditions, as in the preceding exercises. If prior exercises

are plundered for MDP files, it is advised to reevaluate the parameters e.g. by comparing them to parameters of another MDP file or by checking against the GROMACS MDP options (<https://manual.gromacs.org/2024.4/user-guide/mdp-options.html>). Furthermore, the protein should not interact with its closest image; however, making the box dimensions too big would result in longer simulations due to excess water molecules. Therefore, the simulation design should be planned carefully. The water model and forcefield must be chosen as well. Technically, there has to be empirical evidence for the successful interplay of the forcefield and water model. However, since this is out of the scope of this exercise, mixing any options is allowed. However, the rule of thumb is that new is always better.

The report should contain a section describing the essential choices and parameters for the MD simulation, similar to how it is done for scientific publications. There are no strict conventions, but a good orientation is to describe the design in a way that is similar to the computational process of setting the system up. Thus, a reasonable order would be the system and source of the initial coordinates, force, field, solvent and box, equilibration, and production. For the latter two, each stage should at least contain information about thermo- and barostat, as well as electrostatics treatment and simulation time. Furthermore, it is good practice to mention constraints that are applied to any bonds. Of course, any information that seems relevant may be reported additionally. For guidance, the method section of any publication utilizing MD simulations can be referred to.

Furthermore, the stability of the equilibrations must be ensured and shall, therefore, be included in the report. Energy minimization optimizes potential energy, and plotting it against minimization steps should lead to a curve that converges on a minimal potential energy value. As shown in prior exercises, the canonical ensemble (NVT) keeps the temperature constant under certain fluctuations. Hence, the temperature should be measured for this equilibration step. The last part of the equilibration uses the isothermic-isobaric ensemble (NPT) that keeps the pressure constant. Since the pressure fluctuates quite strongly, the time evolution of the density would be of interest. For the report, one of these plots can be chosen and should be included as an indicator of the simulation stability. As a last remark, it should be mentioned that the PDB file contains non-standard compounds, which are not parametrized in the present forcefields. Thus, they must be deleted before building the simulation system (PyMOL or VIM is recommended).

11.2 Analysis

After successful simulation, the user shall perform a blocked standard error analysis with GROMACS which is relatively easy.

When using `gmx analyze` the flag for error estimation, `-ee` needs to be specified. As an input, the command takes an XVG file, e.g., some file created by `gmx rms`. The commands to generate the BSE analysis for the RMSD and R_{gyr} could look like the following:

```
gmx trjconv -s prod.tpr -f prod.xtc -o prod_center.xtc -pbc mol -center
gmx gyrate -f prod_center.xtc -s prod.tpr -o rgyr.xvg
gmx rms -s em.tpr -f prod_center.xtc -o rmsd.xvg
gmx analyze -f rmsd.xvg -ee bse_rmsd.xvg
gmx analyze -f rgyr.xvg -ee bse_rgyr.xvg
```

The two plots should be part of the report. In order to quantify statistical significance from the BSE plot, some intuition is necessary since the convergence onto a horizontal line is only achieved in an ideal setup. Therefore, it needs to be discussed whether the amount of sampling is enough.

Furthermore, a PCA shall be conducted with an eigenvector analysis. GROMACS provides powerful tools to analyze the principal components of a trajectory, which can aid in understanding the functioning of HIV-protease. Calculating the eigenvectors from the trajectories can be done by the following lines of code:

```
gmx covar -f prod_center.xtc -s prod.tpr -xpm
```

This command also produces the covariance matrix, eigenvectors, and eigenvalues by default. This information about the principal motions can be further processed by `gmx anaeig` to visualize different properties concerning atomic motion and convergence. The atom-wise component for each principal component is specified via the `-comp` flag, and the eigenvector projections on the trajectory with the `-proj` flag.

```
gmx_mpi anaeig -v eigenvec.trr -f prod_center.xtc -s prod.tpr -proj eigproj1_10.xvg -comp eigcomp1_10.xvg -last 10
```

The atomic components are difficult to rationalize since each atom contributes only a small amount to each principal component. Nevertheless, important information might be deduced, especially for the collective high-level principal components. The projection on the trajectory can give some clues concerning the convergence of the trajectory, albeit the information is of a qualitative nature. Given the first two principal components, which are highly collective but decorrelated motions, different sampling spaces or clusters can be identified if their trajectory projections are plotted against each other. In a well-sampled simulation, each of these clusters should show a variety of transitions. Thus, in an insufficiently sampled simulation, each cluster is sparsely populated with only a few transitions.

Since most of the atomic motion is captured in the higher principal components, it should be enough to calculate the projections and atomic components for the first 10

eigenvectors. The plots of the atomic components of all 10 eigenvectors should be included in the report with a very brief discussion. This plot is generated directly from the `gmx anaeig` output as an XVG file. The trajectory projections, however, require manual postprocessing since the corresponding XVG file lists the principal component against time information. In order to assess the sampling quality, the report should include a plot of PC1 against PC2. Additionally, plots of PC1 against PC10 and PC9 against PC10 are necessary to visualize the decreasing importance of lower eigenvectors. Discuss the plots' convergence and rationalize the usage of PC1 vs. PC2 for quality assessment over any other PC pair. Exemplary plots can be seen in figure 35.

Another useful flag of `gmx anaeig` is the `-extr`, which yields the two extreme states of the largest eigenvector. Visualizing them conveys an idea about the largest collective motion of the system, and the PDB structures that correspond to the end-states can be generated with the following command.

```
gmx anaeig -v eigenvec.trr -f prod_center.xtc -s prod.tpr -extr
```

It is recommended to visualize the end-states in VMD instead of PyMOL and to include snapshots in the report. Choosing a representation and a view that supplements a brief discussion is beneficial. Knowledge about binding sites and other structural features should be incorporated.

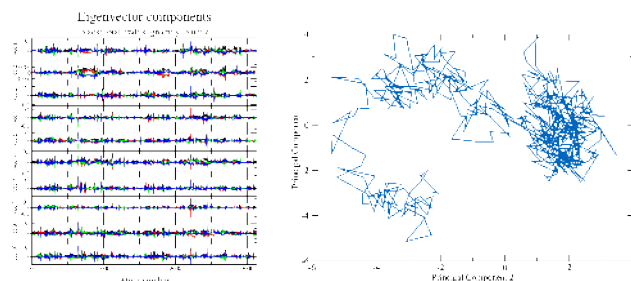


Figure 35. Exemplary results of the atomic components of the first ten eigenvectors and the trajectory projections. The atomic components indicate which atoms contribute to which global movement of the protein. The plot of the principal components 1 and 2 reveals that some clusters along the pathway are sparsely sampled.

12 Exercise 9: Virtual Drug Screening and MM/GBSA

TASKS FOR THE NINTH EXERCISE

Finish these tasks and write a short report about your findings.

- ☐ Try to complete each of the steps of this exercise
- ☐ At the end of this exercise, suggest 1-2 chemical compounds as potential protein-protein-interaction stabilizers for the receptor complex at hand.
- ☐ Discuss the suggested compounds and explain why these compounds seem promising. Use the plots and measurements in the respective steps to support your reasoning.
- ☐ In case the exercise could not be finished, describe any problems you encountered as detailed as possible. Also, document your troubleshooting attempts.

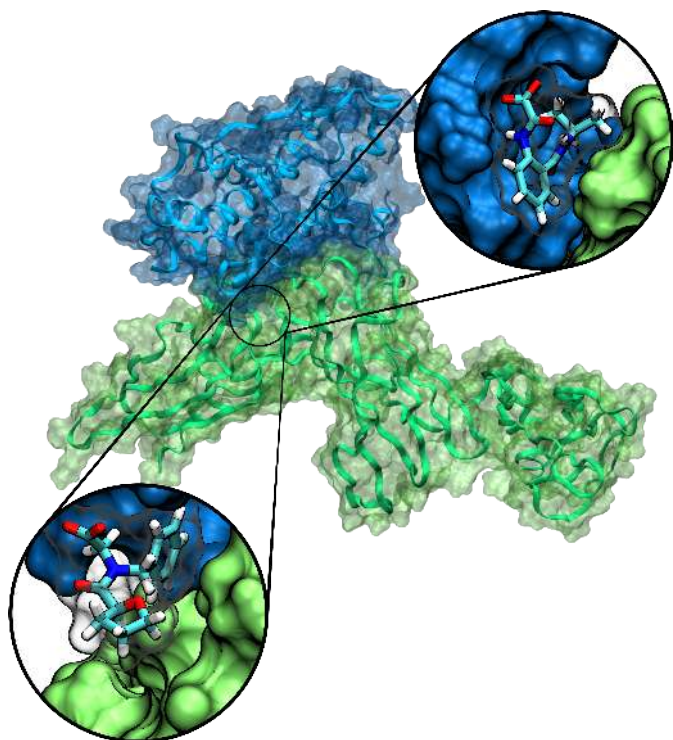


Figure 36. Concept art of protein-protein interaction stabilizers. At the interface of the depicted protein complex (green and blue macromolecules) small organic molecules can bind to fasten the connection between the complex sub-units. The small organic molecules would then act as molecular glue.

Introduction

Typically, drug design focuses on finding a drug (ligand) that binds to a given drug target (i.e. the receptor/protein/-complex). The binding occurs at a specific location on the receptor known as the binding pocket. Thus, three objects are required for a drug-discovery endeavour: the receptor, a pocket, and a ligand. Each receptor has multiple available pockets, and many compounds can fit into each pocket. The workflow itself consists of five different steps. Firstly, (1) a protein system must be chosen and pre-processed for subsequent computer-assisted tasks, i.e., its PDB file. After that, (2) a potentially druggable pocket in the protein must be found. Pocket detection is followed by (3) compiling a database of possible compounds in a process called virtual screening. Filtering this database for the best compound (4) is achieved by molecular docking, which evaluates the possible poses of a ligand in a given binding pocket and ranks them according to favourable receptor-ligand interactions, referred to as scoring. For example, a drug candidate that forms many hydrogen bonds with the receptor will have a high docking score. Finally, (5) promising drug-receptor complexes can be further investigated via molecular dynamics simulations and binding free-energy calculations. Apart from a favourable binding free energy, a viable drug candidate must also fulfil several other criteria: it must be non-toxic, selectively bind to the target without affecting other proteins, be administrable, and be feasible to synthesize.

Protein-protein interactions play pivotal roles in various biological processes, and their perturbation is associated with a wide range of diseases. Consequently, targeting these interactions presents a valuable therapeutic strategy. This exercise focuses on a novel class of drugs known as protein-protein interaction modulators, specifically protein-protein interaction stabilizers. These stabilizers are designed to strengthen the interactions between two proteins forming a complex. The goal is to identify a drug-binding pocket at the interface of the two proteins, allowing the drug to enhance its interaction effectively. Drug development is notoriously expensive, making integrating computational methods highly beneficial. Computer-assisted drug design offers both cost-effectiveness and valuable molecular insights. In this exercise, an example workflow in computer-assisted drug design is explored, illustrating how computational tools can aid in developing protein-protein interaction stabilizers.



This exercise aims to introduce and illustrate standard computational tools used in drug discovery and focuses on finding a favourably binding drug-receptor complex, instead of delving into the practical aspects of drug synthesis or administration. The goal is to appreciate the incredible versatility of computational methods in biochemistry rather



than gaining an in-depth understanding of each tool and calculation while also leveraging the knowledge gained in prior exercises.

12.1 Software Installation

The main objective of this exercise is to find drug candidates that act as a protein-protein interaction stabilizer. However, in stark contrast to the prior exercises, the workflow to generate these candidates is more complex and, therefore, mimics actual research projects characterized by multiple tasks and numerous intermediate steps. The installation of software, especially scientific software, can be quite unintuitive. For installations this exercise focuses on creating a Python environment that contains all the necessary programs and packages to accomplish the tasks. It is best practice to use a Python package manager that can create environments and manage the installation and linking of packages. Using environments helps to distinguish different projects and prevents incompatibilities within one environment caused by mutually exclusive packages and version mismatches. The widely used package manager for Python is the conda package manager.[63] However, due to its slowness and size, this exercise will use the fast and lightweight micromamba package manager, which is closely related to conda. Assuming a Linux computer with `curl` installed, installing micromamba is straightforward and can be achieved by a single command.

```
"${SHELL}" <(curl -L micro.mamba.pm/install.sh)
```

The user will then be prompted to provide some prefixes and filenames, all of which can be skipped by either typing  or , depending on the prompt in the terminal. For further information and troubleshooting, refer to mamba.readthedocs.io/en/latest/installation/micromamba-installation.html. Upon successful installation, executing `micromamba --help` should display version information and available commands. An error will occur if the installation is not successful.

The tasks at hand require two environments. Beyond their functional purpose, this situation demonstrates two different methods of creating micromamba environments. Firstly, the primary PPIS (protein-protein interaction stabilizer) environment is installed from the bash script called `ppis.sh` and can be found at the online repository in `00_install_prep/`. Upon executing the script, the user will be asked whether to create the `ppis` environment and should respond by typing  + . Secondly, another environment is necessary to install MGLTools, a graphical program incompatible with some of the packages found in the `ppis` environment. This environment will be installed via a YAML file, which is a standard configuration file format that can be downloaded from the

online repository. The commands that should result in a successful installation can be found below.

```
./ppis.sh # maybe needs to be made executable via 'chmod 777 ppis.sh'
micromamba create -f mgltools.yaml
```

One of the programs `ppis.sh` installs, is called AutoDock-GPU which requires a GPU and might cause trouble during the installation should there be none. In case no cuda-compatible GPU is available to the user, the installation of AutoDock-GPU can be skipped. Should a user prefer anaconda instead of micromamba, there is a file called `conda_env.sh` available at the online repository, however, this file is deprecated but can be used as guidance for the installation. Furthermore, to use `MODELLER` package, one needs to register for a license at salilab.org/modeller/. The license key must then replace the 'XXXX' in `~/micromamba/envs/ppis/lib/modeller-10.5/modlib/modeller/config.py` (exact path might deviate). The relevance of the programs installed by `ppis.sh`, such as `MODELLER`, is briefly explained in table 6.

Table 6. Brief description of important software packages installed by `ppis.sh`. On the left is the name of the software, and on the right is a brief description.

<code>mplcursors</code> <code>pandas</code> [65][66] <code>seaborn</code> [67] <code>numpy</code> [64]	Classic python analysis tools for data handling and visualization. The packages facilitate interactive data selection in plots, very fast arithmetic operations, handling huge data tables overseeable and efficiently and offer plot templates for aesthetic and functional data representation.
<code>jupyter</code> [68]	Enables a coding interface via <code>jupyter-notebook</code> or <code>jupyter-lab</code> . These interfaces are handy for prototyping and tutorials due to their interactive nature.
<code>modeller</code> [69]	A homology modelling package that is necessary if residues are missing from a protein structure (e.g. incomplete data from crystallography).
<code>rdkit</code>	Python software suite for chemoinformatics and machine learning. Herein, it is important for compound filtering and visualization.
<code>fpocket</code> [70]	A pocket detection program that quickly screens through the surface of your protein complex and reveals (druggable) pockets.
<code>obabel</code> [71]	Open Babel can transform structure files between PDBQT (docking format), PDB (viewing format), and MOL2 (parametrisation format).
<code>MDAnalysis</code> [72] <code>nglview</code> [74] <code>mdtraj</code> [73]	Packages for loading, analyzing and visualizing molecular structures and trajectories.
<code>ambertools</code> [75]	Package for ligand parameterisation and to generate the simulation box for molecular dynamics (MD) simulations and MM/GBSA calculation.
<code>pymol</code> [76]	A widely used tool for visualising biomolecules.
<code>autodock_gpu</code> [77]	GPU-accelerated standard software that is used for large-scale screening of ligand libraries.
<code>vina</code> [78]	Package that will be used for testing and as a substitute if <code>autodock_gpu</code> cannot be used for some reason.

In the online repository, in `00_install_prep`, some files were deposited to ensure the functionality of the micromamba environment. The first step is to activate the environment and open the Jupyter Notebook `TOY.ipynb` to verify all Python packages. None of the installed programs are available to the user without activating the environment.

```
micromamba activate ppis # '(ppis)' should appear next to the
prompt
jupyter-notebook TOY.ipynb # should open a browser window with the
Jupyter interface
```

The notebook contains *cells* that can be executed individually by hitting `Ctrl` + `Enter` after selecting them with the cursor. Understanding the code is extraneous at this point, although any efforts to accustom oneself to it are encouraged. None of the cells should result in an error, and the expected output, written as a comment in each cell, should be achieved. Some common errors and how to fix them are also elaborated on within the notebook's cells. The notebook's browser tab can be closed, and the application can be terminated by typing `Ctrl` + `C` in the respective terminal.

The second step is the validation of auxiliary programs that were installed via `ppis.sh` but are not Python packages. After activating the environment, these programs should be available from the command line, which can be tested by simply executing the commands in the shell. The output caused by executing the commands varies greatly, but none should result in a `command not found` error.

```
pymol # should open a GUI; close with ALT+F4
vina # should show a usage message
vina_split # should show a usage message
obabel # should show a usage message
fpocket # should try pocket hunting
autogrid4 # should show a usage message
parmed # should open a CLI; close with Ctrl+C
parmchk2 # should show a usage message
antechamber # should show a usage message
```

In order to test the installation of MGLtools, the current environment needs to be deactivated, and the `mglttools` environment needs to be activated subsequently. Since MGLtools is a software suite consisting of numerous programs, simply executing `mglttools` into the terminal will not work. The command to execute is called `pmv`, which means *python molecule viewer*.

```
micromamba deactivate ppis # closes ppis env
micromamba activate mglttools # starts the other env
pmv # opens a GUI; ALT+F4 to close
```

Lastly, for validating AutoDock-GPU in `00_install_prep`, a single ligand should be docked onto a receptor complex. The command for testing depends on the user's hardware architecture. However, most systems utilize a 64-bit architecture, which would lead to the following line of code for testing (after activating the environment):

```
micromamba deactivate mglttools
micromamba activate ppis
autodock_gpu_64wi -M complex.maps.fld -L ligand.pdbqt
```

In case of a successful installation, this command's STDOUT should look similar to the contents of `AutodockExpectedOutput.txt` in the same directory. This command should fail, especially for users without a functional GPU, and even if a GPU is present, `autodock_gpu` is not bound to work. For these cases, the last retreat is using the already installed `vina` program, which docks slower but fast enough for this exercise.

Always remember to activate the required environment before typing any commands. They will not work without prior activation.

Protein Construction

In the demonstration, the protein-protein complex of interleukin (IL) will be used as an example (PDBID: 3HMX[79]). The structure of the IL-12 α /IL-12 β -complex can be visualized at www.rcsb.org/3d-view/3HMX/1. As shown in figure 37, specific residues are depicted in grey instead of solid black, indicating they were not resolved in the X-ray experiment. Changing the sequence display from **Interleukin-12 subunit alpha** to **Interleukin-12 subunit beta** also reveals numerous missing residues. Therefore, the initial step involves modelling the missing residues, a helpful practice given the frequent occurrence of missing residues in the PDB.

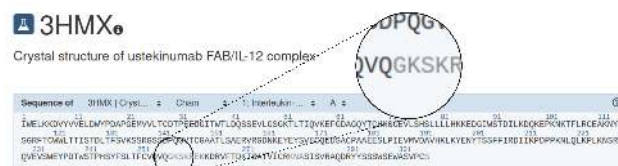


Figure 37. Webpage of RCSB with PDB code 3HMX. The missing residues of Interleukin-12 subunit alpha are light grey instead of black.

The structure and sequence are made available through the online repository in `01_receptor_prep/` under the names `3hmx.pdb` and `3hmx.fasta`, both of which may be inspected via a text editor. Only two segments of the molecular complex in `3hmx.pdb` are of interest: the Interleukin-12 subunit alpha (chain A) and Interleukin-12 subunit beta (chain B). These two chains can be written into a new file `hmx.pdb` using `grep` and regular expressions.

```
grep -E 'ATOM\s+[0-9]+\s+[A-Z]+\s+[A-Z]+\s+A' 3hmx.pdb > hmx.pdb
echo "TER" >> hmx.pdb
grep -E 'ATOM\s+[0-9]+\s+[A-Z]+\s+[A-Z]+\s+B' 3hmx.pdb >> hmx.pdb
```

Chain A and B sequences also need to be extracted into a new file called `hmx.fasta`. The sequences can be copied

from the aforementioned `3hmx.fasta`, however, the formatting needs to conform to MODELLER input rules. Therefore, the content of `hmx.fasta` should look like the following (with the complete sequences):

```
>P1;hmx
sequence:hmx:::::::::0.00:0.00
<CHAIN-A-SEQUENCE>/
<CHAIN-B-SEQUENCE>*
```

The text `hmx` is the align code used by MODELLER for the next step. The sequences of chains A and B can be found in the `3hmx.fasta` file downloaded from RCSB. The `/` symbol in the file indicates the end of a chain and the start of the next chain, and the `*` symbol indicates the end of all protein sequences.

The homology modelling software MODELLER[69] takes the sequence alignment file and the structure template to model a complete structure. Modelling the missing residues consists of two steps: sequence alignment and homology modelling.

Sequence alignment finds missing residue positions in the structural file and can be done using the Python script `salign.py` deposited in the online repository at `01_receptor_prep`. Users are encouraged to familiarise themselves with its content. This python script reads the structure from `hmx.pdb` with an align code `hmx_structure` and aligns it with the sequence from `hmx.fasta` with an align code `hmx`. With the correct environment activated, the script can be run with the following command:

```
mod10.4 salign.py # you can ignore the 'import site' failed message
```

The sequence alignment output is written to `hmx.ali` and consists of the original fasta string and the aligned string that contains gap insertion symbols `-`. The gap insertion symbol communicates missing residues to `model.py` in the next step, and therefore, any `-` must be aligned on a missing residue. However, the `-` placement can be erroneous and must be manually modified to replace the missing residues from the `hmx_structure` sequence by the dash sign. The missing residues can be looked up at the PDB website www.rcsb.org/3d-view/3HMX/1, printed in light grey instead of black, as mentioned before. Each grey letter should be replaced with a `-` in `hmx.ali`. From experience however, `hmx.ali` contains a mistake at the boundary of the two sequences `S/RNLPVATPDGMFP`. Technically, the gap insertion by `salign.py` should result in `-/-----MFP`, but usually, the first dash and the `M` is misplaced requiring a manual fix. Thus, the correctness of modeller usually requires quality control.

With a corrected sequence alignment file `hmx.ali`, the program `model.py` can be used to construct the coordinates of the missing residues. The program generates the homology

model using the `automodel` module with the DOPE scoring method. The Python script can be executed similarly for `salign.py`.

```
mod10.4 model.py
```

This generates 5 models called `hmx.B9999000[1-5].pdb`. MODELLER uses a simulated annealing method to generate the structure. This method consists of minimization, heat-up, cool-down and again minimization, which can be tracked via the following commands:

```
vi hmx.D00000001 # whole progress for the first model
tail -n 10 hmx.D* # check the final energy of all models
tail -n 10 model.log # check DOPE score of all models
```

The energy and the DOPE score guide in selecting the most promising homology model; in some cases, the first structure is not the highest-scoring one. Herein, `hmx.B99990001.pdb` serves as the target structure.

As an intermediate test, the current directory contains a Jupyter Notebook named `01_interface_visualisation.ipynb`. The Jupyter Notebook can be opened, and all cells may be executed to visualize the structure. The notebook reads the prepared receptor file and calculates the interface residues via an intermolecular, interatomic cut-off distance of 5 Å. Upon successful execution, the visualization within the notebook should appear similar to figure 38. The interface allows the user to control the visualization with the mouse keys. While the left button rotates the structure, the right button translates it. Residues or atoms can be centred in the visualization by left-clicking on a particle of choice. Zooming in can be achieved by scrolling up with the mouse wheel.

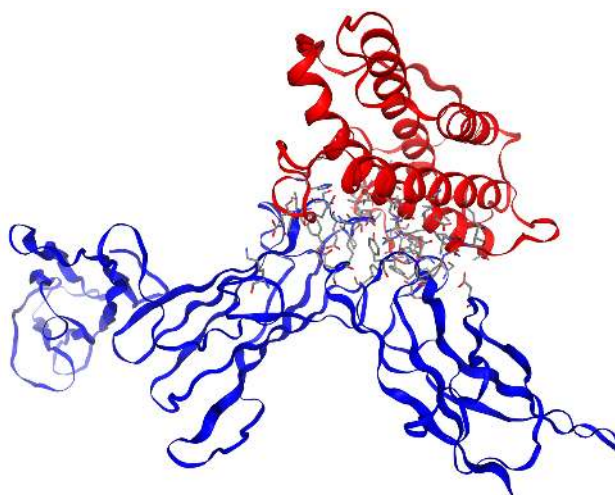


Figure 38. Visualization of the residue ID and the protein complex. The interface residues of protein A (blue cartoon) and protein B (red cartoon) are shown in the licorice representation.

Next, the PDB receptor file needs to be converted to PDBQT format in order to be compatible with `autodock_gpu` and

`vina` docking. Usage of MGLTools can be started as shown before by simply typing `pmv` in the command line.

```
pmv
```

First, the receptor molecule must be loaded by clicking `file` \Rightarrow `Read Molecule` at the upper left corner of the GUI. The path to the receptor molecule is `01_receptor_prep/hmx.B99990001.pdb`. In order to save the receptor in PDBQT format, follow the instructions shown in figure 39. There are six steps to save the PDBQT file of the receptor, which entails selecting the macromolecules and saving it with the correct name, output directory and format.

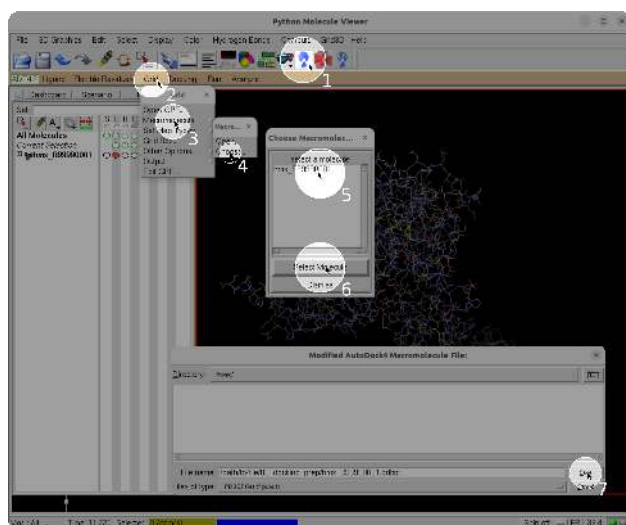


Figure 39. Generating PDBQT for the receptor using MGLTools. First, click the AutoDock-icon, then click on the newly opened `Grid` button and select `Macromolecule` and `Chose...`. After that, a new window should open with only one molecule to select, which should then be saved as `03_docking_prep/hmx.B99990001.pdbqt`.

Interface Pocket Detection

With a complete receptor available, the next step is to make out the receptor pocket for compound docking. A good PPI stabilizer needs to bind evenly to both interfaces of the complex macromolecules. A compound that binds strongly to one protein but weakly to the other acts as a competitive inhibitor, adverse to the intended behaviour. To detect a druggable pocket at the complex interface, Fpocket[70] can be used.

The algorithm behind Fpocket identifies pockets via a geometrical criterion. The program is quite sophisticated, but in all brevity, numerous so-called alpha spheres are fitted onto the protein surface and expanded until four points of the protein surface contact the respective sphere. Spheres with a large radius must be located at the flat surface and are discarded, while spheres within a crevice of the protein remain small and indicate pockets. This is followed by clus-

tering the spheres into individual pockets and ranking the spheres based on descriptor-based pocket scoring and drug-scoring methods.[70] After activating the micromamba environment, the program can be executed straightforwardly.

```
fpacket -f hmx.B99990001.pdb
```

Once successfully executed, `fpacket` generates a folder `hmx.B99990001_out/`. After changing into this directory, the pockets can be visualized by executing a custom bash script.

```
cd hmx.B99990001_out/
bash hmx.B99990001_PYMOL.sh
```

Executing said bash script should open a PyMOL window with the two protein chains and the pockets that were found by Fpocket (similar to figure 40). Further information about all detected pockets can be found in the `hmx.B99990001_out/pockets/` folder. The residues that form the pocket are written to `pocket*_atm.pdb`, and the position of the probes are written in `pocket*_vert.pqr`. Several pocket properties of the first pocket can be inspected using the following line of code:

```
head hmx.B99990001_out/pockets/pocket1_atm.pdb -n 20
```

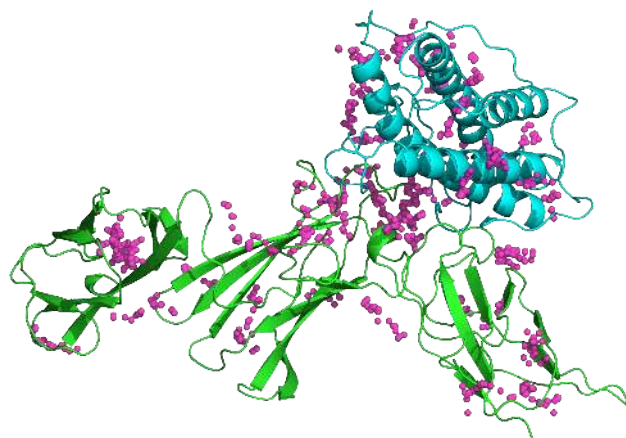


Figure 40. All pockets found by `fpacket` and visualised via `pymol`. One chain is shown in green, the other one is shown in cyan, and the pockets are shown in magenta.

Nevertheless, information about a possible protein-protein interface is not present, and therefore, a pocket that forms contacts with both proteins must be identified differently. Furthermore, the viability of each interfacial pocket might vary, and unusable pockets need to be excluded from further analysis. Therefore, another Jupyter Notebook called `02_pocket_analysis.ipynb` can be found in `01_receptor_prep` which handles the pocket assessment. When executing, the cells should calculate the solvent-accessible surface area (SASA) and buried surface area (BSA) of the two proteins and each pocket, respectively. Here, all

pocket probes resemble artificial ligands. The BSA of pocket *P* within protein A is obtained by summing up the SASA of protein A and the pocket ($SASA_A$, $SASA_P$) and then subtracting the $SASA_{AP}$ of the complex of the two.

$$BSA_{PA} = (SASA_A + SASA_P) - SASA_{AP} \quad (54)$$

The BSA of protein B and the pocket is evaluated analogously in the Jupyter Notebook. A criterion for a good stabilizer binding pocket is a large BSA with both proteins and a good pocket score and drug score. If all cells in the Jupyter Notebook are executed successfully, plots should appear similar to figure 41. In this example, the best pockets can be identified as pocket 1 and 4 since both have a high BSA_{min} and a high drug score.

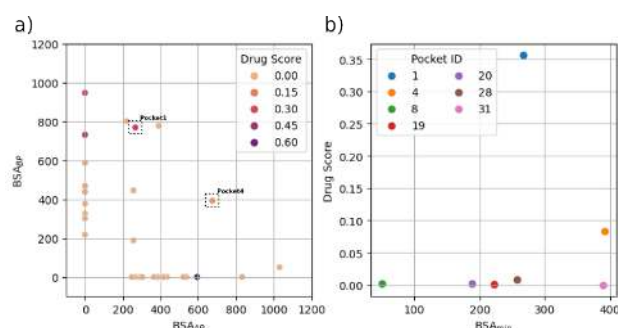


Figure 41. Data visualization of the protein-pocket BSA and drug score. **a)** Distribution of BSA_{AP} and BSA_{BP} of all detected pockets with color coded drug score. **b)** Distribution of BSA_{min} and Drug score of the interface pockets.

A closer look into figure 41 reveals that the BSA values for pocket 4 are more balanced, making it the most favourable pocket. Should two pockets' BSAs score equally well, the drug score is also reported by `fpocket` and can be used to settle ties. The primary output of `02_pocket_analysis.ipynb` is a file called `config`, which maps out the best available pocket (according to the user's assessment) in a format compatible with `autodock_gpu` and `vina`. It is recommended to generate one `config` only since the filename `config` is hard coded in `03_docking_prep/vina.sh`, a later step. Any user aiming to analyze two or more pockets must be prepared to manually alter downstream executables. Content-wise, `config` contains the information about the box centre and box dimensions in absolute coordinates; thus, changing the coordinates of the receptor PDB file would invalidate the `config` file. The Jupyter Notebook also contains cells to visually inspect eventual candidates. The cell output should look somewhat similar to figure 42.

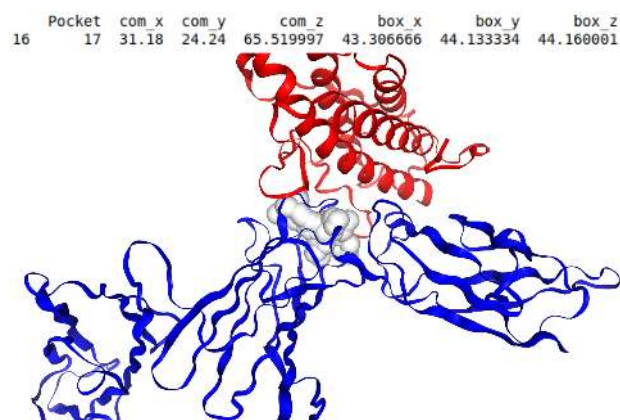


Figure 42. Structure visualization with a detected pocket. The center of mass (COM) and box size information are also printed on the top of the graphic.

With the pocket selected, it is good practice to visualize the binding pocket to check if it matches the findings in `01_receptor_prep/`. The GUI of MGLtools can be used for that purpose. As stated above, it can be opened via `pmv`. After reading-in the PDBQT receptor file, the Grid Box button can be used to visualize the volume to be used for molecular docking (see figure 39). By clicking this button, a panel of box size parameters will open. The parameters should be adjusted according to `config` in the current directory to see if the box matches the preferable binding pocket picked by the user. The contents of the `config` file should look similar to the example file printed below.

```
center_x = 27.83 # box' x-center in Angstrom
center_y = 38.78 # box' y-center in Angstrom
center_z = 56.71 # box' z-center in Angstrom
size_x = 40 # box' x-size in vina npts
size_y = 32 # box' y-size in vina npts
size_z = 40 # box' z-size in vina npts
```

For better visualization, the ribbon representation of the protein can be activated by clicking the circle under the `R` column on the left panel shown in figure 39. At the same time, deactivating the circle under the `L` column removes the atomic licorice representation. Additionally, each complex molecule can be coloured individually by clicking the triangle below the `C1` column and choosing the colouring scheme `By Chain`. The emerging structure should look similar to figure 42, particularly concerning pocket position and dimension.

Ligand Preparation

With a fully prepared receptor pocket, the next step is to prepare the ligands for docking. In theory, there are around 10^{60} drug-like compounds[80], which are impossible to screen through with current computer hardware. Instead, we want to use the ZINC20 database in this demonstration.

Although the ZINC20 database contains fewer compounds than 10^{60} , it is still not feasible to screen it whole. Therefore, some filters will be applied to further reduce the amount of compounds for docking.

Two stages of filtering are applied in this exercise. The first one is integrated into the query system of the ZINC database itself. The second filtering stage is a custom program that can be adapted to fit numerous scientific criteria and utilizes the Python package `rdkit` [81]. The ZINC database can be queried directly on the webpage (zinc20.docking.org/). The online filtering process is laid out in figure 43 and comprises $\log P$ and molecular weight filtering, which translates to hydrophobicity and size of the compounds. The process of downloading a fraction of the database's compounds is illustrated in the exact figure. This fraction still entails around 4 million compounds.

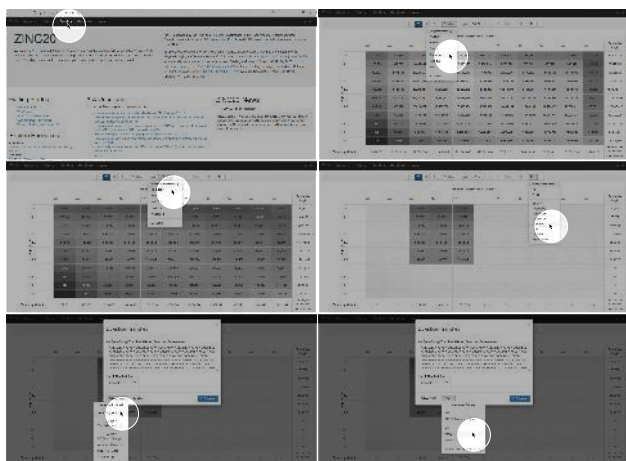


Figure 43. In these six pictures, the workflow to narrow down compounds from the ZINC20 database is shown. First, click the Tranches button to access the website. Then, the large number of compounds will be filtered with custom selections, as shown in the second, third, and fourth pictures. Finally, the compounds must be downloaded as a wget file in SMILES format. Pictures as of June 2023.

The compounds are downloaded in the SMILES string format, which is generally computer readable and can be processed with various programs, e.g. RDKit and Open Babel. The way SMILES are generated and work can be read into elsewhere (e.g. medium.com/@luis-vollmers/tutorial-to-smiles-and-canonical-smiles-explained-with-examples-fbc8a46ca29f). The file that was downloaded in the wget format is not a data file per se but an executable which will then download the respective files for the user. After moving this file from the Downloads directory to `02_ligand_prep`, which should be the current directory, it can be executed via `bash`.

```
mv ~/Downloads/ZINC-downloader-2D-smi.wget ./
bash ZINC-downloader-2D-smi.wget
```

This command should start downloading several different files containing the compounds priorly filtered from ZINC20. For consistency, a new directory should be created in `02_ligand_prep` called `SMILES/`. With the following commands, the SMI files can be transferred into `SMILES/`, and the empty directories are removed.

```
mkdir -p SMILES/
mv -v */*.smi SMILES/
rmdir */*
```

After moving all SMI files into the SMILES directory, the index row can be deleted via `sed`.



```
for i in SMILES/*.smi; do sed -i '/smiles/d' $i; done
```

The last steps in compound filtering include the execution of the Python script `filter_smiles.py`, as well as concatenating all filtered compounds into one file. The latter should not precede the former since executing the Python script on a large SMILES list can lead to memory overflow. The Python program reads in a text file containing SMILES strings in each row and applies certain filter criteria to the compounds. The criteria chosen in `filter_smiles.py` are arbitrary, and their only purpose is to show how to filter compounds in a database given prior pharmaceutical knowledge and decrease the search space. Herein, the filters encompass a limit of four hydrogen bond donors and acceptors, the presence of one carboxy group and one benzene or imidazole group. Furthermore, the last filter criterion is a Bertz complexity below 500. The Bertz complexity index measures the complexity of the molecular graph. These filters are implemented in RDKit, and users may alter them in any way. The Python program can be executed by simply calling `python` on `filter_smiles.py` repeatedly on each file. It should run quickly and yield one filtered file in `SMILES/` for each unfiltered file. The following command uses a for loop to iterate through all files in `SMILES/`, prints the filename and executes the Python script with the output name adjusted automatically. The concatenation of filtered compounds can be achieved via the `cat` command, which is short for concatenate.

```
for i in SMILES/*.smi; do echo $i; python filter_smiles.py $i ${i/.smi/_filtered.txt}; done
cat SMILES/*_filtered.txt > filtered.txt
```

The successful filtering can be assessed by printing out the line number of the output file, which should contain fewer lines than originally downloaded compounds. In this exercise, the number of downloaded compounds was around 4 million; the number of filtered compounds should be roughly 14000. The command that can be used to do so is called `wc`, which is short for word count. However, it counts lines given the `-l` flag.

```
wc -l filtered.txt
```


After filtering the compounds, the next step is generating three-dimensional molecular structures, called conformers. For this task, Open Babel is required. The following line of code should write the desired compounds to `filtered.pdbqt`. Converting 14000 compounds can take quite a lot of time, so a subset should be considered. E.g. the command can be interrupted after converting around 2000 compounds by pressing  +  on the keyboard.

```
obabel -ismi filtered.txt -opdbqt -O filtered.pdbqt --gen3d --
minimize --steps 1500 --sd --ff MMFF94
```

Sometimes `obabel` will fail to generate a 3D structure from SMILES, which results in an error that can be ignored. There will be more than enough compounds, which will result in a successful conversion. The PDBQT batch files require splitting into individual ligand files so that they can be docked individually later on. Hence, a folder needs to be created in `02_ligand_prep/` called `PDBQT` followed by execution of `vina_split` with the options shown below.

```
mkdir -p PDBQT
cd PDBQT
vina_split --input ../filtered.pdbqt --ligand ligand_
```

The resulting directory contains one PDBQT file for each ligand with a numeric ID, e.g. `ligand_0354.pdbqt`. These PDBQT files are the foundation for the virtual screening later in this exercise.

Molecular Docking

Docking Test with AutoDock Vina

With all three parts required for molecular docking prepared, i.e. the receptor, a receptor pocket and the ligands, the docking procedure can be tackled. Apart from the docking procedure, the surrounding computational tools are also explored. Firstly, a test should be performed to check whether everything is correctly set up before screening a large set of compounds with GPU-accelerated docking. For this test, a bash script called `vina.sh` is already placed in `03_docking_prep` for automatic docking of the test ligands. This bash script should be executed and will dock the ligands in the binding pocket specified in `config`. The ligands are read from `../02_ligand_prep/TEST/`, and the outputs are saved to `PDBQT/` and `PDB/`. The command for executing the bash script and the expected output are shown below.

```
./vina.sh # chmod 777 vina.sh might be necessary
```

```
#####
# If you used AutoDock Vina in your work, please cite: #
[...]
[...]
[...]
Performing search ...
0% 10 20 30 40 50 60 70 80 90 100%
```

```
|----|----|----|----|----|----|----|----|
*****
done.
Refining results ... done.
[...]
[...]
[...]
9 files output. The first is ./PDB/out_1.pdb
```

Apart from the STDOUT, this command also generates docked coordinates of the ligand molecules. PyMOL can visualise all docking results at once.

```
pymol ../01*/hmx.B99990001.pdb ./PDBQT/*
```

The representation might be improved by showing the protein in the cartoon representation and the ligands in the licorice representation and distinguishing them via colour. On the top right panel, this can be achieved by clicking the **H** icon in the `hmx_B99990001` row and then selecting `everything`. Next, clicking the **S** icon in the same row should be followed followed by clicking `cartoon`. Finally, the colouring method should be changed by clicking the **(C)** icon in the `all` row and `by chain` → `by chain (elem C)` should be chosen. The resulting graphic should be somewhat similar to figure 44.

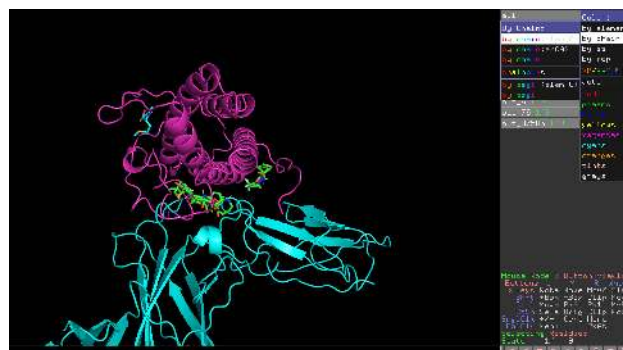


Figure 44. Visualization of the test docking output. In the lower right corner, it reads `State 1/9`. This means a maximum of 9 docking poses are available for the docked ligands, which can be navigated with the arrow keys.

The main point of the docking test here is to check if the ligands are mostly docked in the desired interface pocket. In case of substantially deviating results, the pocket size can be decreased, or another interface pocket can be selected altogether. After desirable results are ensured, the BSA between the test ligands and two proteins can be calculated for comparison, as well as their docking scores. The program `01_vinatest_analysis.ipynb` contains the analysis procedure. Executing all cells in said Jupyter Notebook should visualize the distribution of BSA and the docking score similar to the pocket analysis in figure 41. Note that the docking score here is in units of kcal/mol, and the lower the docking score, the stronger the expected binding affinity.

From figure 45, it becomes apparent that ligand 1 has the lowest docking score and the largest BSA_{min} ; thus, it is potentially the best PPI stabilizer candidate among the test ligands.

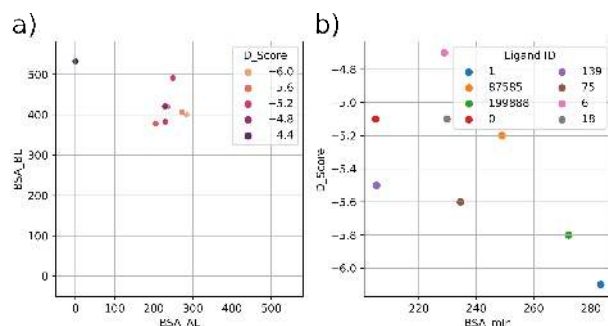


Figure 45. Visualization of the docking result from AutoDock Vina in terms of BSA and docking score.

The Jupyter Notebook also visualizes the docking poses using `nglview`. The resulting output of the visualization panel should look similar to figure 46. We can choose the ligand, and the ligand state can be changed by setting `L_path` accordingly. Because calculating the BSA for a docking pose is relatively time-consuming, only the docking pose with the lowest docking score is analysed in the code.

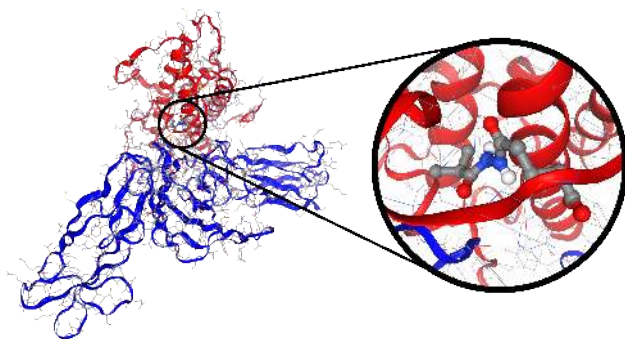


Figure 46. Visualization of the docking pose of the first docking mode of ligand 1 with a magnification of the ligand.

Large-Scale Screening with AutoDock-GPU

Although docking with a few molecules using `vina` already yields good results, screening a larger ligand database has a higher chance of discovering a better PPIs candidate. With the confirmation that the obtained docking box can successfully dock the ligands at the protein-protein interface, large-scale ligand screening can continue using `autodock_gpu`. Using a Nvidia RTX2080ti GPU enables a screening speed of up to 0.6 million compounds per week. For docking, `autodock_gpu` reads a set of grid parameter files, including `map`, `maps.fld`, `maps.xyz`. These files can be created via a ready-made GPF file called `hmx.gpf` that can be

found in `04_virtual_screening/`. It mostly contains paths to crucially important files and some options that need to be set for the docking procedure. Curious users may inspect the file; however, changing its contents is not recommended. The `autogrid4` command can automatically generate all related files, given the GPF file and the receptor file. The required commands can be seen below. The respective map files should be generated in a separate folder for tidiness reasons.

```
mkdir autogrid
cp -v ../03_docking_prep/hmx_B99990001.pdbqt hmx.gpf autogrid/
cd autogrid/
autogrid4 -p hmx.gpf
```

With the MAP files ready in `04_virtual_screening/autogrid/`, the ligands prepared in `02_ligand_pre/PDBQT/`, the next step is to generate a file for `autodock_gpu` to perform automatic docking experiments, called `docking.dat` which contains the locations of the input and output files and the location of the MAP files. Creating the file `docking.dat` and the docking itself should not be absolved manually, instead a bash script `virtual_screening.sh` downloadable from the online repository is supposed to take care of the procedure. The user should scrutinize the file and ensure that the `autodock_gpu` executable exists. The script also handles the directory infrastructure. Thus, output files can be found in the directory `04_virtual_screening/docking/`. Furthermore, the for-loop is automatically aborted after the first iteration if `virtual_screening.sh` is executed as is for testing purposes. In case of a successful execution, the `break` statement can be commented out by inserting a `#`. The following execution of `virtual_screening.sh` will then try to dock all ligands saved in the specified directory.

```
./virtual_screening.sh
```

The premier output of this docking process is DLG files and XML files for each docked ligand. The DLG file is the coordinate file, which is very similar to a PDBQT file format-wise only that each line is prefixed with a `DOCKED: .` The standard output to the terminal for a successful docking should look similar to the text below.

```
Running Job #2224:
Device: NVIDIA GeForce GTX 1080
[...]
[...]
[...]
the best molecules of the last 4 * 5 generations, 42000 generations,
or 1132076 evaluations:

Generations | Evaluations | Threshold | Average energy o...
-----+-----+-----+-----
0 | 150 | 232675.16 kcal/mol | 0.00 +/- 0...
5 | 25110 | 232675.16 kcal/mol | 0.00 +/- 0...
[...]
[...]
```

[...]

0.00 +/- 0.00 kcal/mol combined.
0 samples, best energy 232675.16 kcal/mol.

After the ligand screening, the results should be analysed like the test docking. In `04_virtual_screening`, a Jupyter Notebook can be found that contains the code for said analysis. At the beginning of the notebook `01_GPUDock_analysis.ipynb`, the docked ligand files are read, and the docking scores for the best state of each compound are saved and then plotted as a histogram. The resulting plot is saved and should look similar to figure 47. Noticeably, only compounds with a score lower than 10 are plotted since many compounds might have scores magnitudes larger, skewing the distribution.

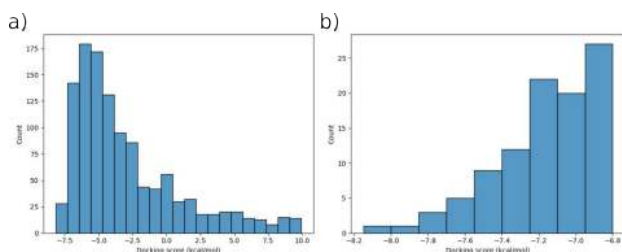


Figure 47. Sub-distribution of docking score outputs from `autdock-gpu` by reading the best-scored docking pose of each ligand.

The scoring of the compounds allows one to select the compounds. Herein, the 100 compounds with the lowest scores are picked for further processing. The notebook again features a histogram of their docking scores (see figure 47). Then it converts them from DLG to PDBQT format followed by yet another conversion into SMI, MOL2 and PDB format which is achieved by executing `convert.sh` from the notebook. For the conversion, `obabel` is used, and the script also relies on the correct directory structure and names. If the user customises any prior naming conventions, changes must be made to `convert.sh`. After both conversions, the well-known calculation of the BSA is conducted for the top compounds and is visualised in a previously described way. Again, this is done to quantify how well the ligands stay at the interface by calculating the buried surface area of protein A and protein B. An exemplary distribution of docking score, BSA_{AL} , and BSA_{BL} are shown in figure 48.

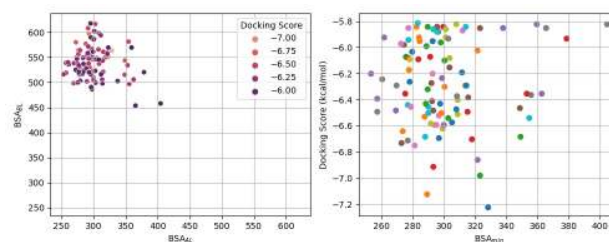


Figure 48. Data visualization of the protein-ligand BSA and docking score. On the left the distributions of BSA_{AP} and BSA_{BP} are shown of all detected pockets with color coded docking score. On the right the distribution of BSA_{min} and the docking score of the interface ligand can be seen.

Since a good PPI stabilizer must have a sufficiently high binding affinity to both proteins, a ligand should prevail with a high docking score and BSA_{min} . A mechanic offered by the python package `mplcursors` eases the selection process by enabling interactive plots. One cell in `01_GPUDock_analysis.ipynb` utilises this package to plot the same distribution shown in figure 48 with the difference, another window opens, and the ligand ID appears upon hovering over the data point with the cursor, as shown in figure 49.

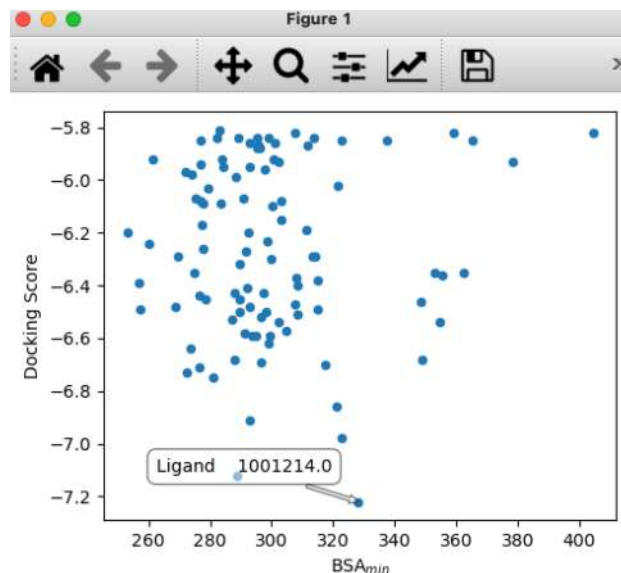


Figure 49. Data visualization with `mplcursors`.

Once a promising compound could be made out, its docking pose can be visualised via `nglview`. By setting the `L_path` to the respective PDB output path in the respective cell, the ligand docking pose can be seen at the interface as shown in figure 50.

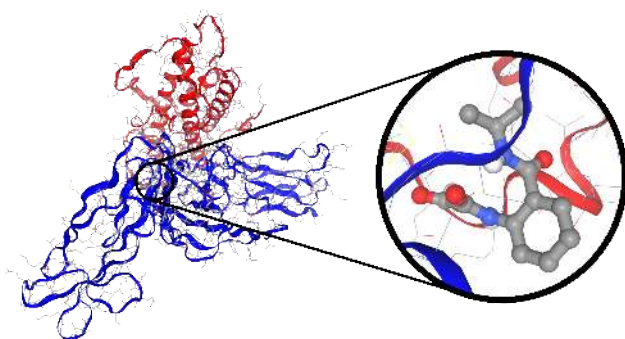


Figure 50. Visualization of Ligand 0156 docked at the interface between protein A (blue) and protein B (red).

The generated SMI files stored in `04_virtual_screening/SMI/` are used in the last step of the Jupyter Notebook. These files contain the SMILES string of each compound, which is unique to each compound (but not unique for each compound) and can be read by RDKit and used to search compounds in the ZINC20 database. Executing the respective cell in the Jupyter Notebook visualises the molecular graph of each compound in a Pandas Dataframe similar to figure 51.

ID	Score	SMILES	Molecule
0	0156	-8.15	<chem>C1(=[C][C]=[C][C]=C1NC(=O)C(=O)O)C(=O)N(C)[C]</chem>

Figure 51. Molecular graph of the docked compound 0156.

For some users, this can raise a `_get_adjustment` Attribute error that was mentioned in `TOY.ipynb` in `00_install_prep`. The bug fix described in the `TOY.ipynb` only takes effect after restarting the respective notebook (go to `kernel` → `Restart`). Apart from visualising the molecular graph, the SMILES string enables the user to easily browse the ZINC20 substance page (zinc20.docking.org/substances/home/) for the compound. In an actual scientific project, these compounds could be purchased from vendors conveniently listed on the website for experimental validation. The docking process is concluded with this step. The docked compounds can be further evaluated with MD simulations, or a more extensive screening can be conducted by increasing the number of ligands for GPU-docking before continuing.

Molecular Dynamics Simulation with GROMACS and MM/GBSA

Regarding drug screening, the whole process can be imagined as a funnel that increases sensitivity with each step. The first step was prompting the ZINC20 database more or less arbitrarily to reduce the unfathomable amount of available chemical compounds by a large margin. The next step included the application of chemical criteria that the compounds should obey. Even though they were chosen arbitrarily in this exercise, they could incorporate empirical knowledge relevant to the system. The filtering was followed by the docking experiments, which assigned each compound a binding score and estimated the buried surface area. The sensitivity of these metrics is already quite high and facilitates the dismissal of many unsuitable compounds. Only two to five best-ranking compounds should be tested to further taper the funnel.

The next step is to conduct an MD simulation with GROMACS[82] to investigate the dynamical properties of the protein-protein-ligand ternary complex as well as to estimate the binding free energy with molecular mechanics generalized Born and surface area solvation (MM/GBSA).[83] MM/GBSA is a middle ground between docking and free energy perturbation (FEP). While docking is fast, the results are unreliable; however, FEP calculations impose a ludicrous computational cost that cannot be covered on a large scale, especially with large ternary systems like the one investigated in this exercise. Therefore, MM/GBSA is the method of choice since it is especially suited for comparing the free energies of similar systems.

Ligand Parameterization and MD simulation

The working directory contains five subfolders to keep things ordered: `_exec`, `_inp`, `_output`, `_sim`, and `_trash`. The structure of these folders should not be changed since they are hard coded in downstream executables facilitating the MD simulations.

- `_exec` contains binaries to be executed or distributed into the simulation directories.
- `_inp` contains input files to be copied into the simulation directories, where they are usually modified or processed differently.
- `_output` contains output files from analysis of the simulations, like plots or data tables with calculated results.
- `_sim` contains the simulation directories. Herein, `_sim` will contain one further subdirectory for each ligand in which the actual simulation runs will be conducted.
- `_trash` contains seemingly unnecessary files. 'Seemingly' because deleted files tend to become necessary

again after being deleted. Therefore, it is recommended not to use `mv` rather than `rm`. The contents of `_trash` may be deleted in the distant future once their irrelevancy has manifested itself.

The first step in simulating the ligands is to change the directory into `_exec` and look into the `prep_setup.sh` script. At the beginning of this program, some variables might need adjustment from the user. Most noticeably, `L_ids` should be set to the ligand IDs found to be the best-ranked ones. As a reminder, ligands are ranked and selected in the prior step with `01_GPUdock_analysis.ipynb`. Anything else does not need to be changed as long as the user makes no unforeseen manual modifications. The variable declarations at the beginning of `prep_setup.sh` should be checked when in doubt.

One of the tasks that `prep_setup.sh` completes is to copy `prep_and_run.sh` in each ligand simulation directory and to modify them accordingly. `prep_and_run.sh` is the main executable and takes care of the parameterization of the ligands and the MD simulation itself. The whole process is quite complicated. Thus, the complex bash script consists of four different functions, which will not be printed in this exercise sheet but can be inspected by opening the file via `vi`. Nevertheless, the functions shall be briefly introduced. `cp_inp_files` copies all necessary input files from the input directory into the current simulation directory and adjusts them for the respective ligand and receptor.

`gen_params` calls some programs from `ambertools` and are used to generate custom parameters for the ligand molecules. This is necessary since these are likely not standard residues whose parameters are readily available in standard forcefields (like amino acids, water, and nucleic acids). There are several important output files of this function, like the `FRCMOD` file and `PDB` file of the complex.

`gen_gmxtop` takes care of file compatibility. The output from `ambertools` is not suitable for GROMACS but only for the AMBER simulation package, which is not supported within this exercise. Therefore, the output files need to be converted to GROMACS compatible files, most importantly a working `topol.top`, which is the output of this function together with the respective `GRO` file.

`run_gmx_sims` sets up the simulation box and carries out the usual simulation sequence once the `TOP` and `GRO` files are present. Three repetitions are automatically conducted to calculate more reliable average properties in the analysis.

Natively, `gen_gmxtop` and `run_gmx_sims` should be commented out with a `#` at the end of the bash script, while `cp_inp_files` and `gen_params` should be switched on. Uncommenting any other functions at this stage is unnecessary since the parametrization of the `MOL2` ligands

from `obabel` often contain artifacts. Thus, executing the `prep_and_run.sh` in the respective ligand simulation directory often leads to a fatal error in the `antechamber` program similar to the one seen below.

```
./prep_and_run.sh
[...]
```

CC1=CC=CC=C1

```
/home/user123/micromamba3/envs/ppis/bin/to_be_dispatched/antechamber
: Fatal Error!
Weird atomic valence (3) for atom (ID: 4, Name: C1).
Possible open valence.
[...]
```

The error refers to valence charges assigned to the atoms of the ligand that do not fulfill the octet rule, usually caused by incorrect bond assignment at nitrogen atoms, double bonds, or aromatic rings. This problem requires manual correction via PyMOL. The file that needs to be corrected is the `MOL2` file in the `_inp/` directory, e.g. `_inp/1234.mol2`.

```
cd path/to/_inp/
pymol 1234.mol2
```

PyMOL has a tool for building and editing molecular structures. Most valence problems can be solved by fixing the hydrogen atoms of the structure and by adding the correct aromaticity. The procedure depicted in figure 52 can be used as guidance. Once the structure has been edited, overwrite the `MOL2` file of `1234.mol2` with the edited structure and execute `prep_and_run.sh` again.

```
cd path/to/_exec/3hmx_1234/
./prep_and_run.sh
```

If the error persists, more editing might be attempted, or a different compound could be chosen for parametrization and simulation.

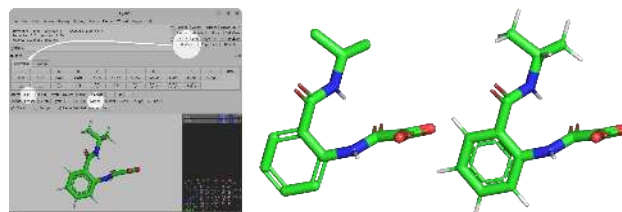


Figure 52. The process of editing a ligand with PyMOL is shown. The left picture shows how to access the builder menu. The two options `Fix H` and `Arom` were used in this specific example. The middle picture shows the ligand in its erroneous state. Hence, trying to generate parameters will fail. The right picture depicts the ligand state, resulting in a successful parametrization. Noticeably, aromaticity is added, as well as the explicit hydrogen atoms. Many other caveats could be wrong with a ligand, and some intuition is necessary for each fix.

Once the parametrization is conducted successfully, the next function `gen_gmxtop` can be switched on, and the prior two can be commented out. No user interaction is required; however, the function may show the following error, which can be ignored.


```
Traceback (most recent call last):
  File "convertTOP_amb2gmx.py", line 596, in <module>
    perm_imp = get_permutations_4_wild(dihedral[0:4])
  File "convertTOP_amb2gmx.py", line 121, in get_permutations_4_wild
    temp_a = perm[2]
IndexError: list index out of range
```

The required `topol.top` and the GRO file should still be outputted. After asserting that these files exist, the last function of `prep_and_run.sh` may be executed. This function, `run_gmx_sims`, carries out the GROMACS simulations for the user. The MDP files are automatically copied from `_inp/`, and the simulation output is automatically named. Each production run should be 2 ns long. The simulations can take some time since the system is quite large; however, three trajectories per compound should be created once they finish, named `prod_center_<REP>.xtc` containing 100 frames each. Once all simulations are finished, the trajectory analysis can start in the Jupyter Notebook `01_analyse_trajectory.ipynb` in the `_exec` folder. The analysis of the trajectory is limited. However, some essential properties are computed that reveal the stability of the receptor complex. Firstly, the RMSD over the trajectory is measured for three compounds as shown in figure 53. As a reminder, the RMSD is a measurement that reveals the average structural deviation from a reference structure for every time frame. Each run is depicted as an individual curve with the first frame of the trajectory as a reference. Considerable deviations from the reference could indicate substantial rearrangement of the receptor configuration, which should not happen given a stable receptor complex.

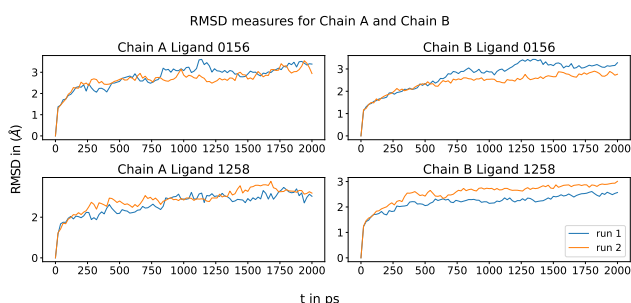


Figure 53. The RMSD is shown versus the simulation time. Each row corresponds to a different ligand, while the columns correspond to the two protein chains.

The RMSF is also calculated, giving the structural deviation as a time average per atom, which is more elucidating than the RMSD regarding structural integrity. Since stable atoms and stable conformation should exhibit a low RMSF value. In this notebook, three different ways to measure the RMSF are conducted. Firstly, the atom-wise RMSF is done to showcase the function rather than gain information. Secondly, the atomic RMSF is averaged per residue, revealing

very mobile and less mobile residues. The expectation would be that residues in loops and unhindered parts of the protein have a considerable RMSF value, and those residues buried and in the interface have a lower RMSF value. The residue-wise RMSF is also calculated for three different compounds. However, the values are averaged over the two runs. The results can be seen in figure 54 with one curve per compound. Generally, the expectation that residues within the interface have lower RMSF is validated in this plot, and significant differences are visible between each compound's trajectories. For example, The curve of Ligand 0156 is generally higher than the other two for both protein chains, and the curve of compound 1258 seems to be most stable in general. The stable regions outside the interface cannot be attributed to compound stabilization but rather to the nature of the starting structure and its individual time development. The sampling might be insufficient to properly determine the most successful PPIS but it is sufficient for demonstrating the workflow in this exercise.

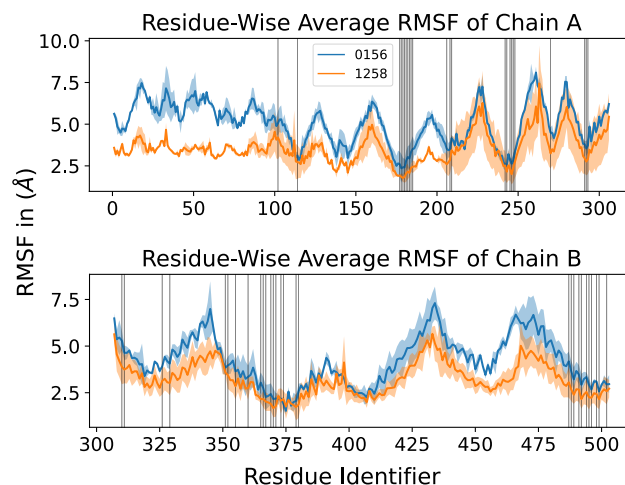


Figure 54. The residue-wise RMSF of the receptor is plotted with error bars shown as filled-in area. Although each curve follows a certain trend, the compounds largely differ concerning protein chain stability. The interface residues are shown as grey vertical lines.

The most important information is given in the last plot of the Jupyter Notebook, which consists of the atom-wise RMSF but only for the interfacial residues. Again, it can be seen that the stabilization differs between each compound. The ligands 1258 and 1306 are the best stabilizers for the interface residues (see figure 55).

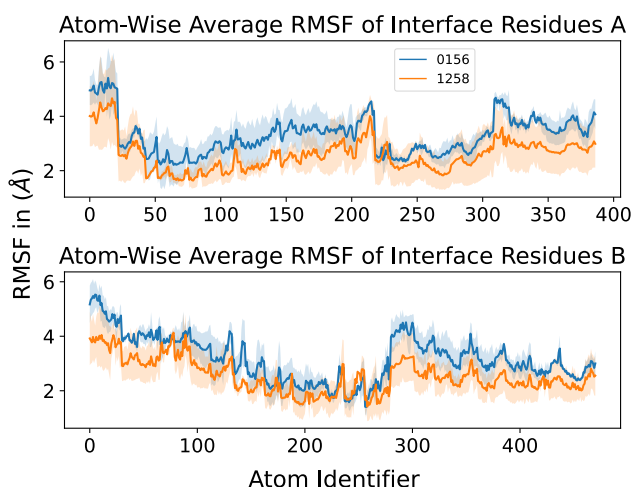


Figure 55. The atomic RMSF for the interface residues for each protein chain. In both protein chains, compound 0156 seems the least performant, while the other two compete for the most stable interface. The error bars are again shown as filled-in areas with the same color shade.

In the prior exercise, checking the convergence of a system is covered for specific measurements and how a certain sampling sufficiency can be visualized. These tests are lacking here; the system is sizeable and would need much time to converge appropriately. The ambitious user can achieve convergence by increasing the number of steps in the respective MDP files and validating it using the methods in section 11.

Binding Free Energy Estimation with MM/GBSA

The next step is to estimate the binding free energy with MM/GBSA to properly determine the most promising compound (the `ambertools` analysis function is called `MMPBSA.py` hence the name). The bash script `prep_mmpbsa.sh` is another multipurpose bash script that handles many tasks simultaneously, given that the correct environment is activated. The main functionalities include the conversion of the GROMACS trajectory to an AMBER trajectory, its decomposition into the different system components (Chain A, Chain B, and the ligand), and its setting up of the directory structure for every selected compound. The ligands and receptor variables can be found at the beginning of `prep_mmpbsa.sh` and must be adjusted accordingly before executing it.

```
vi prep_mmpbsa.sh # alter script accordingly
./prep_mmpbsa.sh
```

After successful execution, all necessary files should already be transferred into the newly created subdirectories called `<REC-NAME>_<LIG-ID>/`. One of these files is called `mmpbsa.sh`, which is used to estimate the binding free energies. Executing it will take a few hours and should result in a couple of DAT and CSV files, e.g. `AL_output_1.dat`.

The progress can be tracked by using `grep` on the MDOUT file together with a line count command. Since MM/GBSA analyses the trajectory for the complex, the receptor, and the ligand, there are 300 frames. Thus, the calculations should be done once the following code outputs 300.

```
grep 'NSTEP' _MMPBSA*.mdout.0 | wc -l
```

The DAT files contain the total binding free energy, and the CSV files contain the residue-wise decomposition of the free energy. The total amount of free energy should determine the optimal PPIS energy. The residue-wise decomposition can facilitate the lead optimization process, i.e., the knowledge-based modification of promising compounds to improve their efficacy. The Jupyter Notebook called `01_visualise_MMPBSA_results.ipynb` in `06_mmpbsa` offers guidance. The ligand's identifier must be adjusted before executing the specific cells. Also, the `ppis` environment needs to be activated.

The first plot of the notebook provides information about the total binding free energy from the MM/GBSA calculation. This notebook shows the total binding free energy for each compound run since only showing averages dilutes information. Good PPIS have small binding free energies to both protein chains while not favoring any of the two significantly (see figure 56).

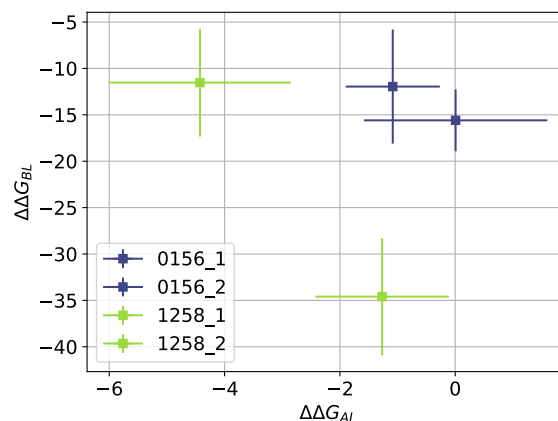


Figure 56. Exemplary distribution of the binding free energies of 2 runs of 2 ligands. The free binding energy estimate for protein A is plotted on the x-axis, and the one for protein B is plotted on the y-axis. In this case none of the compounds seem promising.

More helpful information is given in the second plot. Per default, the MM/GBSA calculation outputs a residue-wise free energy decomposition. This output is being read by the Jupyter Notebook and depicted per ligand and protein chain. Most residues do not interact significantly with the compound or contribute at all. However, the interface residues, depicted as grey, vertical lines, mainly contain non-zero

contributions for both protein A and B (see figure 57).

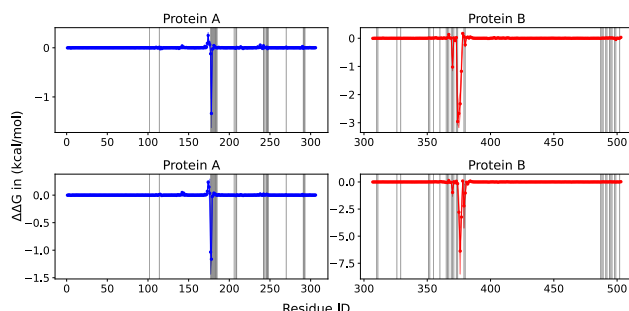


Figure 57. Energy Decomposition of binding free energy. Each row belongs to a compound, and each column is the respective protein. The black vertical lines depict interfacial protein residues.

Last but not least, the Jupyter Notebook also visualizes the structure of the complex in a particular way that illuminates why specific residues facilitate and others inhibit the compound binding. In this visualization, all interfacial residues that facilitate binding are represented in white balls and sticks, while the unfavorable residues are shown as black balls and sticks. By inspecting the residues and their behavior over the trajectory, one may optimize the drug candidate further to interact more favorably with both proteins (see figure 58).

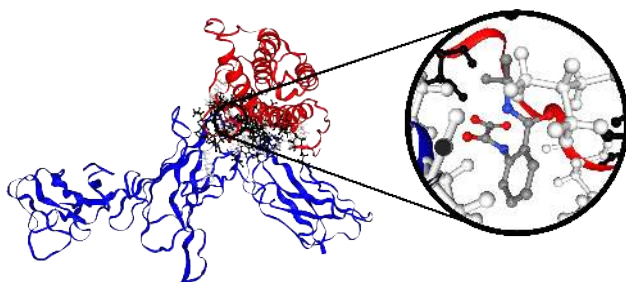


Figure 58. The complex is rendered with `nglview` once again, but this time, the residues of the interface are colored in by their contribution to the free energy of binding. In the zoomed-in section of the figure, one residue close by the ligand is interfering with the binding (black color), posing as a target for optimization.

With the information provided in the plots of the Jupyter Notebook, one compound may be presented as the most promising PPIS. The compound should be presented structurally in a brief report, and the reasons why it outperforms its competition should be elaborated to outline the decision-making process.

Conclusion

This exercise presented an example workflow in a computer-assisted drug design application to identify a potential protein-protein interaction stabilizer from a database. It

involves many steps and several computer tools must be employed, ranging from the initial preparation of target-protein files to a detailed quantitative thermodynamic characterization of the drug-target complex.

Firstly, it is imperative to understand that the approach chosen in this exercise may not be universal or optimal for each PPI. Several steps including the MM/GBSA algorithms in `06_mmpbsa/gb.in`, the number of frames in the MDP files and the scoring function in the molecular docking step may not be optimal.

A plausible outlook on such a research project could engulf a lead optimization guided by the MM/GBSA decomposition or higher precision free energy calculations, e.g., free energy perturbation. Experimentally, *in-vitro* assays could be designed and conducted to determine the actual binding behavior of the drug candidates. Nevertheless, there are many choices to think about even after the virtual drug screening is successfully conducted. This complexity makes computer-assisted drug design an exciting and challenging task.

Author Contributions

- Luis Vollmers
 - Wrote initial drafts for all exercises except section 12 and created corresponding materials in https://github.com/Foly93/MD_FromBasicsToApplication.
 - Edited section 12 and adapted the materials for that exercise in https://github.com/Foly93/MD_FromBasicsToApplication.
 - Wrote the drafts for all remaining sections.
 - Created the bibliography.
 - Wrote the pre-submission letter and handled correspondence.
 - Edited the final draft.
- Shu-Yu Chen
 - Designed the course material after which section 12 was modeled.
 - Wrote initial draft for section 12 and created the corresponding materials in https://github.com/Foly93/MD_FromBasicsToApplication.
 - Edited the final draft.
- Maria Reif
 - Designed the university course material after which this article was modeled.
 - Contributed to writing the draft, especially theoretical considerations in section 10 and section 9.
 - Supervision and guidance of the whole process.
 - Edited the final draft.
- Tristan Mauck
 - Wrote the model reports available in https://github.com/Foly93/MD_FromBasicsToApplication.
 - Edited the final draft.
- Martin Zacharias
 - Designed the university course material after which this article was modeled.
 - Supervision and guidance of the whole process.
 - Edited the final draft.

For a more detailed description of author contributions, see the GitHub issue tracking and changelog at https://github.com/Foly93/MD_FromBasicsToApplication.

Author Information

ORCID:

Luis Vollmers: [0000-0003-2768-2964](https://orcid.org/0000-0003-2768-2964)

Shu-Yu Chen: [0009-0009-1762-0269](https://orcid.org/0009-0009-1762-0269)

Maria Reif: [0000-0002-8171-3541](https://orcid.org/0000-0002-8171-3541)

Tristan Mauck: [0009-0003-0862-1725](https://orcid.org/0009-0003-0862-1725)

Martin Zacharias: [0000-0001-5163-2663](https://orcid.org/0000-0001-5163-2663)

References

- [1] **Rahman A.** Correlations in the Motion of Atoms in Liquid Argon. *Physical Review*. 1964; 136(2A):A405–A411. <https://doi.org/10.1103/physrev.136.a405>.
- [2] **Allen MP, Tildesley DJ.** Computer simulation of liquids. Reprint ed. Oxford science publications, Oxford: Clarendon Press; 1990.
- [3] **Shaw DE, Grossman JP, Bank JA, Batson B, Butts JA, Chao JC, Deneroff MM, Dror RO, Even A, Fenton CH, Forte A, Gagliardo J, Gill G, Greskamp B, Ho CR, Ierardi DJ, Iserovich L, Kuskin JS, Larson RH, Layman T, et al.** Anton 2: Raising the Bar for Performance and Programmability in a Special-Purpose Molecular Dynamics Supercomputer. In: *SC14: International Conference for High Performance Computing, Networking, Storage and Analysis* IEEE; 2014. p. 41–53. <https://doi.org/10.1109/sc.2014.9>.
- [4] **Zwier MC, Chong LT.** Reaching biological timescales with all-atom molecular dynamics simulations. *Current Opinion in Pharmacology*. 2010; 10(6):745–752. <https://doi.org/10.1016/j.coph.2010.09.008>.
- [5] **Abraham MJ, Murtola T, Schulz R, Páll S, Smith JC, Hess B, Lindahl E.** GROMACS: High performance molecular simulations through multi-level parallelism from laptops to supercomputers. *SoftwareX*. 2015; 1–2:19–25. <https://doi.org/10.1016/j.softx.2015.06.001>.
- [6] **Hess B, Kutzner C, van der Spoel D, Lindahl E.** GROMACS 4: Algorithms for Highly Efficient, Load-Balanced, and Scalable Molecular Simulation. *Journal of Chemical Theory and Computation*. 2008; 4(3):435–447. <https://doi.org/10.1021/ct700301q>.
- [7] **Atkinson K.** An introduction to numerical analysis. Brisbane, QLD, Australia: John Wiley and Sons (WIE); 1989.
- [8] **Iserles A.** Chapter 1: Euler's method and beyond. In: *A first course in the numerical analysis of differential equations*, 2 ed. Cambridge texts in applied mathematics, Cambridge University Press; 2008. p. 3–18. <https://doi.org/10.1017/cbo9780511995569.004>.
- [9] **Kim S.** Issues on the Choice of a Proper Time Step in Molecular Dynamics. *Physics Procedia*. 2014; 53:60–62. <https://doi.org/10.1016/j.phpro.2014.06.027>.
- [10] **Kim S.** Issues on the Choice of a Proper Time Step in Molecular Dynamics. *Physics Procedia*. 2014; 53:60–62. <https://doi.org/10.1016/j.phpro.2014.06.027>.
- [11] **Lennard-Jones JE.** On the determination of molecular fields I. From the variation of the viscosity of a gas with temperature. *Proceedings of the Royal Society of London Series A, Containing Papers of a Mathematical and Physical Character*. 1924; 106(738):441–462. <https://doi.org/10.1098/rspa.1924.0081>.
- [12] **Hünenberger PH.** Chapter 2: Thermostat Algorithms for Molecular Dynamics Simulations. In: *Advanced Computer Simulation* Springer Berlin Heidelberg; 2005. p. 105–149. <https://doi.org/10.1007/b99427>.
- [13] **Swendsen RH, Wang JS.** Replica Monte Carlo Simulation of Spin-Glasses. *Physical Review Letters*. 1986; 57(21):2607–2609. <https://doi.org/10.1103/physrevlett.57.2607>.
- [14] **Sugita Y, Okamoto Y.** Replica-exchange molecular dynamics method for protein folding. *Chemical Physics Letters*. 1999; 314(1–2):141–151. [https://doi.org/10.1016/s0009-2614\(99\)01123-9](https://doi.org/10.1016/s0009-2614(99)01123-9).
- [15] **Coudène Y.** Chapter 2. The Pointwise Ergodic Theorem. In: *Ergodic Theory and Dynamical Systems* Springer London; 2016. p. 15–16. <https://doi.org/10.1007/978-1-4471-7287-1>.
- [16] **Berendsen HJC, Postma JPM, van Gunsteren WF, DiNola A, Haak JR.** Molecular dynamics with coupling to an external bath. *The Journal of Chemical Physics*. 1984; 81(8):3684–3690. <https://doi.org/10.1063/1.448118>.
- [17] **Braun E, Gilmer J, Mayes HB, Mobley DL, Monroe JL, Prasad S, Zuckerman DM.** Best Practices for Foundations in Molecular Simulations [Article v1.0]. *Living Journal of Computational Molecular Science*. 2019; 1(1). <https://doi.org/10.33011/livecoms.1.1.5957>.
- [18] **Nosé S.** A molecular dynamics method for simulations in the canonical ensemble. *Molecular Physics*. 1984; 52(2):255–268. <https://doi.org/10.1080/00268978400101201>.
- [19] **Nosé S.** A unified formulation of the constant temperature molecular dynamics methods. *The Journal of Chemical Physics*. 1984; 81(1):511–519. <https://doi.org/10.1063/1.447334>.
- [20] **Hoover WG.** Canonical dynamics: Equilibrium phase-space distributions. *Physical Review A*. 1985; 31(3):1695–1697. <https://doi.org/10.1103/physreva.31.1695>.
- [21] **The-Engineering-Toolbox**, Water - Specific Heat vs. Temperature. [online]; 2004. Accessed: 2024-07-07. https://www.engineeringtoolbox.com/specific-heat-capacity-water-d_660.html.
- [22] **Clausius R.** Ueber einen auf die Wärme anwendbaren mechanischen Satz. *Annalen der Physik*. 1870; 217(9):124–130. <https://doi.org/10.1002/andp.18702170911>.
- [23] **Brenig W.** Chapter 24: Gleichverteilungssatz und Virialsatz. In: *Statistische Theorie der Wärme* Springer Berlin Heidelberg; 1996. p. 161–164. https://doi.org/10.1007/978-3-642-61038-7_24.
- [24] **Landau LD, Lifshitz EM.** Chapter XII: Fluctuations. In: *Statistical Physics* Elsevier; 1980. p. 333–400. <https://doi.org/10.1016/b978-0-08-057046-4.50019-1>.

- [25] **Bernetti M**, Bussi G. Pressure control using stochastic cell rescaling. *The Journal of Chemical Physics*. 2020; 153(11). <https://doi.org/10.1063/5.0020514>.
- [26] **Linse JB**, Hub JS. Three- and four-site models for heavy water: SPC/E-HW, TIP3P-HW, and TIP4P/2005-HW. *The Journal of Chemical Physics*. 2021; 154(19). <https://doi.org/10.1063/5.0050841>.
- [27] **Griffiths DJ**. Chapter 2: Electrostatics. In: *Introduction to Electrodynamics* Cambridge University Press; 2017. p. 59–112. <https://doi.org/10.1017/9781108333511.003>.
- [28] **Norberg J**, Nilsson L. On the Truncation of Long-Range Electrostatic Interactions in DNA. *Biophysical Journal*. 2000; 79(3):1537–1553. [https://doi.org/10.1016/s0006-3495\(00\)76405-8](https://doi.org/10.1016/s0006-3495(00)76405-8).
- [29] **Ewald PP**. Die Berechnung optischer und elektrostatischer Gitterpotentiale. *Annalen der Physik*. 1921; 369(3):253–287. <https://doi.org/10.1002/andp.19213690304>.
- [30] **Hockney RW**, Eastwood JW. Chapter8: Particle-Particle-Particle-Mesh (P3M) Algorithms. In: *Computer Simulation Using Particles* CRC Press; 2021. p. 267–305. <https://doi.org/10.1201/9780367806934>.
- [31] **Darden T**, York D, Pedersen L. Particle mesh Ewald: An Nlog(N) method for Ewald sums in large systems. *The Journal of Chemical Physics*. 1993; 98(12):10089–10092. <https://doi.org/10.1063/1.464397>.
- [32] **Essmann U**, Perera L, Berkowitz ML, Darden T, Lee H, Pedersen LG. A smooth particle mesh Ewald method. *The Journal of Chemical Physics*. 1995; 103(19):8577–8593. <https://doi.org/10.1063/1.470117>.
- [33] **Prevost M**, Van Belle D, Lippens G, Wodak S. Computer simulations of liquid water: treatment of long-range interactions. *Molecular Physics*. 1990; 71(3):587–603. <https://doi.org/10.1080/00268979000101991>.
- [34] **Berendsen HJC**, Postma JPM, van Gunsteren WF, Hermans J. Chapter 21: Interaction Models for Water in Relation to Protein Hydration. In: *Intermolecular Forces* Springer Netherlands; 1981. p. 331–342. https://doi.org/10.1007/978-94-015-7658-1_21.
- [35] **Skinner LB**, Benmore CJ, Neuefeind JC, Parise JB. The structure of water around the compressibility minimum. *The Journal of Chemical Physics*. 2014; 141(21):214507. <https://doi.org/10.1063/1.4902412>.
- [36] **Jaiswal AK**, Srivastava R, Pandey P, Bandyopadhyay P. Microscopic picture of water-ethylene glycol interaction near a model DNA by computer simulation: Concentration dependence, structure, and localized thermodynamics. *PLOS ONE*. 2018; 13(11):1–36. <https://doi.org/10.1371/journal.pone.0206359>.
- [37] **Lemkul J**. From Proteins to Perturbed Hamiltonians: A Suite of Tutorials for the GROMACS-2018 Molecular Simulation Package [Article v1.0]. *Living Journal of Computational Molecular Science*. 2019; 1(1). <https://doi.org/10.33011/livecoms.1.1.5068>.
- [38] **Mobley DL**, Chodera JD, Dill KA. On the use of orientational restraints and symmetry corrections in alchemical free energy calculations. *The Journal of Chemical Physics*. 2006; 125(8). <https://doi.org/10.1063/1.2221683>.
- [39] **Christ CD**, Mark AE, van Gunsteren WF. Basic ingredients of free energy calculations: A review. *Journal of Computational Chemistry*. 2009; 31(8):1569–1582. <https://doi.org/10.1002/jcc.21450>.
- [40] **Pohorille A**, Jarzynski C, Chipot C. Good Practices in Free-Energy Calculations. *The Journal of Physical Chemistry B*. 2010; 114(32):10235–10253. <https://doi.org/10.1021/jp102971x>.
- [41] **Villa A**, Mark AE. Calculation of the free energy of solvation for neutral analogs of amino acid side chains. *Journal of Computational Chemistry*. 2002; 23(5):548–553. <https://doi.org/10.1002/jcc.10052>.
- [42] **Bennett CH**. Efficient estimation of free energy differences from Monte Carlo data. *Journal of Computational Physics*. 1976; 22(2):245–268. [https://doi.org/10.1016/0021-9991\(76\)90078-4](https://doi.org/10.1016/0021-9991(76)90078-4).
- [43] **Shirts MR**, Pitner JW, Swope WC, Pande VS. Extremely precise free energy calculations of amino acid side chain analogs: Comparison of common molecular mechanics force fields for proteins. *The Journal of Chemical Physics*. 2003; 119(11):5740–5761. <https://doi.org/10.1063/1.1587119>.
- [44] **Kirkwood JG**. Statistical Mechanics of Fluid Mixtures. *The Journal of Chemical Physics*. 1935; 3(5):300–313. <https://doi.org/10.1063/1.1749657>.
- [45] **Zwanzig RW**. High-Temperature Equation of State by a Perturbation Method. I. Nonpolar Gases. *The Journal of Chemical Physics*. 1954; 22(8):1420–1426. <https://doi.org/10.1063/1.1740409>.
- [46] **Wu D**, Kofke DA. Phase-space overlap measures. II. Design and implementation of staging methods for free-energy calculations. *The Journal of Chemical Physics*. 2005; 123(8). <https://doi.org/10.1063/1.2011391>.
- [47] **Gumbart JC**, Roux B, Chipot C. Standard Binding Free Energies from Computer Simulations: What Is the Best Strategy? *Journal of Chemical Theory and Computation*. 2012; 9(1):794–802. <https://doi.org/10.1021/ct3008099>.
- [48] **Lühns T**, Ritter C, Adrian M, Riek-Loher D, Bohrmann B, Döbeli H, Schubert D, Riek R. 3D structure of Alzheimer's amyloid- β (1–42) fibrils. *Proceedings of the National Academy of Sciences*. 2005; 102(48):17342–17347. <https://doi.org/10.1073/pnas.0506723102>.
- [49] **Benson MD**, Buxbaum JN, Eisenberg DS, Merlini G, Saraiva MJM, Sekijima Y, Sipe JD, Westermarck P. Amyloid nomenclature 2020: update and recommendations by the International Society of Amyloidosis (ISA) nomenclature committee. *Amyloid*. 2020; 27(4):217–222. <https://doi.org/10.1080/13506129.2020.1835263>.
- [50] **Torrie GM**, Valleau JP. Nonphysical sampling distributions in Monte Carlo free-energy estimation: Umbrella sampling. *Journal of Computational Physics*. 1977; 23(2):187–199. [https://doi.org/10.1016/0021-9991\(77\)90121-8](https://doi.org/10.1016/0021-9991(77)90121-8).

- [51] **Kumar S**, Rosenberg JM, Bouzida D, Swendsen RH, Kollman PA. THE weighted histogram analysis method for free-energy calculations on biomolecules. I. The method. *Journal of Computational Chemistry*. 1992; 13(8):1011–1021. <https://doi.org/10.1002/jcc.540130812>.
- [52] **Lemkul JA**, Bevan DR. Assessing the Stability of Alzheimer's Amyloid Protofibrils Using Molecular Dynamics. *The Journal of Physical Chemistry B*. 2010; 114(4):1652–1660. <https://doi.org/10.1021/jp9110794>.
- [53] **Woo HJ**, Roux B. Calculation of absolute protein–ligand binding free energy from computer simulations. *Proceedings of the National Academy of Sciences*. 2005; 102(19):6825–6830. <https://doi.org/10.1073/pnas.0409005102>.
- [54] **Ferrenberg AM**, Swendsen RH. New Monte Carlo technique for studying phase transitions. *Physical Review Letters*. 1988; 61(23):2635–2638. <https://doi.org/10.1103/physrevlett.61.2635>.
- [55] **Roux B**. The calculation of the potential of mean force using computer simulations. *Computer Physics Communications*. 1995; 91(1–3):275–282. [https://doi.org/10.1016/0010-4655\(95\)00053-i](https://doi.org/10.1016/0010-4655(95)00053-i).
- [56] **Hub JS**, de Groot BL, van der Spoel D. g_wham—A Free Weighted Histogram Analysis Implementation Including Robust Error and Autocorrelation Estimates. *Journal of Chemical Theory and Computation*. 2010; 6(12):3713–3720. <https://doi.org/10.1021/ct100494z>.
- [57] **Grossfield A**, WHAM: the weighted histogram analysis method, version 2.0.10. Grossfield Lab; 2018. http://membrane.urmc.rochester.edu/wordpress/?page_id=126.
- [58] **Klimovich PV**, Shirts MR, Mobley DL. Guidelines for the analysis of free energy calculations. *Journal of Computer-Aided Molecular Design*. 2015; 29(5):397–411. <https://doi.org/10.1007/s10822-015-9840-9>.
- [59] **Grossfield A**, Zuckerman DM. Chapter 2: Quantifying Uncertainty and Sampling Quality in Biomolecular Simulations. In: *Annual Reports in Computational Chemistry* Elsevier; 2009. p. 23–48. [https://doi.org/10.1016/s1574-1400\(09\)00502-7](https://doi.org/10.1016/s1574-1400(09)00502-7).
- [60] **Yang L**, Song G, Carriquiry A, Jernigan RL. Close Correspondence between the Motions from Principal Component Analysis of Multiple HIV-1 Protease Structures and Elastic Network Modes. *Structure*. 2008; 16(2):321–330. <https://doi.org/10.1016/j.str.2007.12.011>.
- [61] **Flyvbjerg H**, Petersen HG. Error estimates on averages of correlated data. *The Journal of Chemical Physics*. 1989; 91(1):461–466. <https://doi.org/10.1063/1.457480>.
- [62] **David CC**, Jacobs DJ. Chapter 11: Principal Component Analysis: A Method for Determining the Essential Dynamics of Proteins. In: *Methods in Molecular Biology* Humana Press; 2013. p. 193–226. https://doi.org/10.1007/978-1-62703-658-0_11.
- [63] **Anaconda Software Distribution**. Anaconda Inc.; 2020. <https://docs.anaconda.com/>.
- [64] **Harris CR**, Millman KJ, van der Walt SJ, Gommers R, Virtanen P, Cournapeau D, Wieser E, Taylor J, Berg S, Smith NJ, Kern R, Picus M, Hoyer S, van Kerkwijk MH, Brett M, Haldane A, del Río JF, Wiebe M, Peterson P, Gérard-Marchant P, et al. Array programming with NumPy. *Nature*. 2020; 585(7825):357–362. <https://doi.org/10.1038/s41586-020-2649-2>.
- [65] **The pandas development team**, Doe J, editor, pandas-dev/pandas: Pandas. Zenodo; 2020. <https://doi.org/10.5281/zenodo.3509134>.
- [66] **Wes McKinney**. Data Structures for Statistical Computing in Python. In: Stéfan van der Walt, Jarrod Millman, editors. *Proceedings of the 9th Python in Science Conference*; 2010. p. 56 – 61. <https://doi.org/10.25080/Majora-92bf1922-00a>.
- [67] **Waskom ML**. seaborn: statistical data visualization. *Journal of Open Source Software*. 2021; 6(60):3021. <https://doi.org/10.21105/joss.03021>.
- [68] **Kluyver T**, Ragan-Kelley B, Pérez F, Granger B, Bussonnier M, Frederic J, Kelley K, Hamrick J, Grout J, Corlay S, Ivanov P, Avila D, Abdalla S, Willing C, Jupyter Development Team. Jupyter Notebooks—a publishing format for reproducible computational workflows. In: *IOS Press* IOS Press; 2016.p. 87–90. <https://doi.org/10.3233/978-1-61499-649-1-87>.
- [69] **Šali A**, Blundell TL. Comparative Protein Modelling by Satisfaction of Spatial Restraints. *Journal of Molecular Biology*. 1993; 234(3):779–815. <https://doi.org/https://doi.org/10.1006/jmbi.1993.1626>.
- [70] **Le Guilloux V**, Schmidtke P, Tuffery P. Fpocket: An open source platform for ligand pocket detection. *BMC Bioinformatics*. 2009; 10(1):168. <https://doi.org/10.1186/1471-2105-10-168>.
- [71] **O'Boyle NM**, Banck M, James CA, Morley C, Vandermeersch T, Hutchison GR. Open Babel: An open chemical toolbox. *Journal of Cheminformatics*. 2011; 3(1):33. <https://doi.org/10.1186/1758-2946-3-33>.
- [72] **Michaud-Agrawal N**, Denning EJ, Woolf TB, Beckstein O. MDAnalysis: A toolkit for the analysis of molecular dynamics simulations. *Journal of Computational Chemistry*. 2011; 32(10):2319–2327. <https://doi.org/10.1002/jcc.21787>.
- [73] **McGibbon RT**, Beauchamp KA, Harrigan MP, Klein C, Swails JM, Hernández CX, Schwantes CR, Wang LP, Lane TJ, Pande VS. MD-Traj: A Modern Open Library for the Analysis of Molecular Dynamics Trajectories. *Biophysical Journal*. 2015; 109(8):1528 – 1532. <https://doi.org/10.1016/j.bpj.2015.08.015>.
- [74] **Rose AS**, Bradley AR, Valasatava Y, Duarte JM, Prlić A, Rose PW. NGL viewer: web-based molecular graphics for large complexes. *Bioinformatics*. 2018; 34(21):3755–3758. <https://doi.org/10.1093/bioinformatics/bty419>.
- [75] **Case DA**, Aktulga HM, Belfon K, Ben-Shalom IY, Berryman JT, Brozell SR, Cerutti DS, III TEC, Cisneros GA, Cruzeiro VWD, Darden TA, Forouzesh N, Giambasu G, Giese T, Gilson MK, Gohlke H, Goetz AW, Harris J, Izadi S, Izmailov SA, et al., **Case DA**, editor, AmberTools23. D.A. Case; 2023. ambermd.org.
- [76] **Schrödinger, LLC**. The PyMOL Molecular Graphics System, Version 1.8; 2015, pymol.org, accessed 21.12.2024.

- [77] **Santos-Martins D**, Solis-Vasquez L, Tillack AF, Sanner MF, Koch A, Forli S. Accelerating AutoDock4 with GPUs and Gradient-Based Local Search. *Journal of Chemical Theory and Computation*. 2021; 17(2):1060–1073. <https://doi.org/10.1021/acs.jctc.0c01006>.
- [78] **Eberhardt J**, Santos-Martins D, Tillack AF, Forli S. AutoDock Vina 1.2.0: New Docking Methods, Expanded Force Field, and Python Bindings. *J Chem Inf Model*. 2021; 61(8):3891–3898. <https://doi.org/10.1021/acs.jcim.1c00203>.
- [79] **Luo J**, Wu SJ, Lacy ER, Orlovsky Y, Baker A, Teplyakov A, Obmolova G, Heavner GA, Richter HT, Benson J. Structural Basis for the Dual Recognition of IL-12 and IL-23 by Ustekinumab. *Journal of Molecular Biology*. 2010; 402(5):797–812. <https://doi.org/10.1016/j.jmb.2010.07.046>.
- [80] **Reymond JL**. The Chemical Space Project. *Accounts of Chemical Research*. 2015; 48(3):722–730. <https://doi.org/10.1021/ar500432k>.
- [81] RDKit: Open-source cheminformatics. RDKit; <https://www.rdkit.org>.
- [82] **Abraham M**, Alekseenko A, Bergh C, Blau C, Briand E, Doijade M, Fleischmann S, Gapsys V, Garg G, Gorelov S, Gouailardet G, Gray A, Irrgang ME, Jalalypour F, Jordan J, Junghans C, Kanduri P, Keller S, Kutzner C, Lemkul JA, et al., **Abraham M**, editor, GROMACS 2023.2 Manual. Zenodo; 2023. <https://doi.org/10.5281/zenodo.8134388>.
- [83] **Wang E**, Sun H, Wang J, Wang Z, Liu H, Zhang JZH, Hou T. End-Point Binding Free Energy Calculation with MM/PBSA and MM/GBSA: Strategies and Applications in Drug Design. *Chemical Reviews*. 2019; 119(16):9478–9508. <https://doi.org/10.1021/acs.chemrev.9b00055>.