From Proteins to Perturbed Hamiltonians: A Suite of Tutorials for the GROMACS-2018 Molecular Simulation Package [Article v1.0]

Justin A. Lemkul[1*]

[1] Department of Biochemistry, Virginia Tech, Blacksburg, VA, United States of America

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

This version dated January 2, 2019

Abstract

Molecular dynamics (MD) simulations are a popular technique for studying the atomistic behavior of any molecular system. Performing MD simulations requires a user to become familiar with the commands, options, and file formats of the chosen simulation software, none of which are consistent across different programs. Beyond these requirements, users are expected to be familiar with various aspects of physics, mathematics, computer programming, and interaction with a command-line environment, presenting critical barriers to entry in the MD simulation field. This article presents seven tutorials for instructing users in the proper methods for preparing and carrying out different types of MD simulations in the popular GROMACS simulation package. GROMACS is an open-source, free, and flexible MD package that is consistently among the fastest in the world. The tutorials presented here range from a "simple" system of a protein in aqueous solution to more advanced concepts such as force field organization and modification for a membrane-protein system, two methods of calculating free energy differences (umbrella sampling and "alchemical" methods), biphasic systems, protein-ligand complexes, and the use of virtual sites in MD simulations. In this article, users are provided the rationale and a theoretical explanation for the command-line syntax in each step in the online tutorials (available at http://www.mdtutorials.com/gmx) and the underlying settings and algorithms necessary to perform robust MD simulations in each scenario.

***For correspondence:**
jalemkul@vt.edu (JAL)

1 Introduction

Molecular dynamics (MD) simulations are widely applied to studies of structure-function relationships, disease pathways, and drug design. MD simulations rely on the relationship between a given configuration of atoms and its energy to propagate dynamics. This process is driven by empirical force fields, energy functions and associated parameter sets that are used to compute energies and forces acting in a given configuration. After calculating the forces, the configuration is updated by applying those forces over a discrete time step, δt. An assembly of many configurations, linked through time, establishes a trajectory for subsequent visualization and analysis.

Performing an MD simulation requires that many choices be made, including the starting structure of the molecule(s)

Received: 16 July 2018
Accepted: 31 August 2018

https://doi.org/10.33011/livecoms.1.1.5068
Living J. Comp. Mol. Sci. 2019, 1(1), 5068

of interest, the force field model to be applied, concentration and nature of salt in solution, the integration method, algorithms for temperature and/or pressure regulation, and length of the simulation itself. This complex series of decisions will dictate the quality, reliability, and reproducibility of the simulation, and consequently should be undertaken only after careful consideration. Beyond the conceptual issues lies the interface with the MD simulation package itself. The user must translate the knowledge of the experimental setup to actual practice by implementing these details in the framework of a complex software suite. The combination of these factors can be burdensome for new users, who may be prone to making errors or failing to consider best practices and subtle nuances of a large body of literature.

The GROMACS simulation package [1, 2] is an open-source software suite that is free for all users and available worldwide. GROMACS has for many years been among the fastest MD codes available [1], making it a popular choice for carrying out MD simulations as hardware and algorithmic development allow for ever-increasing trajectory lengths.

1.1 Scope

The seven tutorials presented here range in difficulty from basic to advanced, roughly in the order of presentation. Each of the tutorials comes with its own learning objectives and expected outcomes.

After completing Tutorial 1, "Lysozyme in Water," the user should be able to build a solvated protein system and carry out a short MD simulation. Specific learning objectives include:

1. Understand the content of a GROMACS system topology
2. Carry out basic steps of setting up a periodic system, including solvation and adding ions
3. Identify keywords and typical settings in `.mdp` input files
4. Control position restraints and topology directives via `.mdp` settings
5. Determine the appropriate algorithms for energy-minimizing and equilibrating a biomolecular system and performing a production MD simulation

After completing Tutorial 2, "$KALP_{15}$ in DPPC," the user should be able to build a simple membrane-protein system containing one lipid type. Specific learning objectives include:

1. Understand the organization and contents of GROMACS force field files and how parameters from different, but compatible, sources can be added to them
2. Apply an iterative packing routine to place phospholipids around a transmembrane protein
3. Perform lipid-specific analysis using custom index groups

After completing Tutorial 3, "Umbrella Sampling," the user should be able to perform simulations under the influence of a biasing potential. Specific learning objectives include:

1. Apply `.mdp` keywords that invoke the pull code to add a simple, one-dimensional biasing potential to selected atoms in a system
2. Define what is meant by a "reaction coordinate" and how to construct a suitable one
3. Perform restrained simulations in multiple sampling windows along a reaction coordinate
4. Compute a potential of mean force profile

After completing Tutorial 4, "Biphasic Systems," the user should be able to construct heterogeneous systems of two different liquids. Specific learning objectives include:

1. Build a simulation system containing a liquid that is not water
2. Manipulate the relative positioning of a box within a larger volume

After completing Tutorial 5, "Protein-Ligand Complex," the user should be able to construct a solvated protein system including a ligand that needs to be parametrized. Specific learning objectives include:

1. Produce a ligand topology outside of GROMACS and incorporate it into a system topology
2. Determine how to verify if a ligand topology is suitable for further simulation
3. Perform analysis of interactions between a protein and a ligand

After completing Tutorial 6, "Free Energy of Solvation," the user should be able to carry out a series of simulations for the alchemical transformation of a small molecule in water. Specific learning objectives include:

1. Understand the purpose of `.mdp` keywords related to free energy calculations
2. Use the Bennett Acceptance Ratio method to compute the free energy difference of the transformation of van der Waals terms

After completing Tutorial 7, "Virtual Sites," the user should be able to understand what virtual sites are and how they are used in specific cases. Specific learning objectives include:

1. Define a virtual site and its position relative to other atoms
2. Construct a system of a linear molecule with virtual sites

2 Prerequisites

The tutorials described in this article assume the user has installed GROMACS, version 2018 or any minor or patch release

https://doi.org/10.33011/livecoms.1.1.5068
Living J. Comp. Mol. Sci. 2019, 1(1), 5068

in the 2018.x series. Instructions for how to install GROMACS can be found in the online manual at http://manual.gromacs.org/documentation/2018-current/index.html. This tutorial article will not describe how to install GROMACS, as details vary depending on the available hardware, compilers, etc.

2.1 Background knowledge

These tutorials assume the user is familiar with basic Linux or Unix command-line interaction and navigation as well as the use of a plain-text editor such as VIM or Emacs. Background knowledge in MD simulations would be beneficial, though this article seeks to introduce new users to the fundamental concepts that are required to properly execute a simulation. These tutorials cannot be comprehensive in explaining all elements and algorithms in MD simulations, and as such, users are expected to seek outside resources (*e.g.* [3–6]) to assist in their understanding. Tutorial 1 (see Section 3.2) assumes the user has no experience with GROMACS, and therefore contains the most detail about how to carry out various tasks. In the subsequent six tutorials, it will be assumed that the user is familiar with basic operations and not all steps will be described in depth, except for instances of new or different usage.

Throughout this article, commands that are to be issued by the user are written in `monospace font` and the command prompt is indicated with `$`, which itself is not part of the command the user should enter. Some commands are long and span multiple lines; in these cases the commands should not be entered on separate lines in the terminal, rather as one continuous command. Program and file names are similarly written in `monospace font` in the text to distinguish them from the narrative.

2.2 Software/system requirements

These tutorials assume the user is working with a GROMACS version in the 2018.x series. Other versions have different command-line syntax and input file keywords and attempting to use them will result in problems.

The output of GROMACS analysis programs is, by default, written in a format compatible with XmGrace (http://plasma-gate.weizmann.ac.il/Grace/). As such, the user should be familiar with plotting data using this program.

For visualization of coordinate files and trajectories, many options are available, including Visual Molecular Dynamics (VMD, http://www.ks.uiuc.edu/Research/vmd/) [7], Chimera (https://www.cgl.ucsf.edu/chimera/) [8], and PyMOL (https://pymol.org/2/), among others. Users should be familiar with at least one of these programs for visualizing the system. The latest version of each of these programs (VMD 1.9.3, Chimera 1.12, and PyMOL 2.0 at the time this article was written) is preferred for maximum compatibility with different file formats and features.

The protein-ligand complex tutorial will make use of the Avogadro molecular editor (https://avogadro.cc/) [9] and a Python script that requires a Python 2.7.x version, the NetworkX package in the 1.11.x version series, and any compatible NumPy version. The topology conversion script is not compatible with NetworkX versions in the 2.x series.

3 Content and Links

The tutorials described in this article can be accessed at http://www.mdtutorials.com/gmx. All necessary files for completing each tutorial are provided at that location.

3.1 Overview

GROMACS comprises a set of programs for preparing, running, and analyzing MD simulations. Each program is called from within a single binary, called `gmx`, and command-line syntax is similar among all the programs. All options are preceded by a hyphen (-) and the subsequent keyword or filename must be the next argument on the command line. For instance, if a coordinate file is specified as an input file for a given command, `-f my_file.pdb` is correct syntax, but separating `-f` from `my_file.pdb` in any way with intervening text or other arguments will cause an error. Some arguments are boolean, meaning that they indicate binary, yes/no choices. For instance, the option `-princ` (from the `editconf` command, discussed below), indicates that the user wants the solute molecule to be aligned with its principal axis along the Cartesian *x*-axis. The opposite, to not require alignment, would be `-noprinc`. This syntax is the same for any GROMACS boolean argument.

At any time, the user can refer to help information for a given program by invoking `gmx help` and the name of the program or use the `-h` flag. For instance, to read the help information for the `pdb2gmx` program, the user would type `gmx help pdb2gmx` or equivalently, `gmx pdb2gmx -h`.

Every GROMACS program has a set of default options and file names that it assumes it can use if the user does not otherwise specify. That is, if any argument requires a file name (typically input and output files) and one is not provided by the user, the program will look in the working directory for a file with the default name. If that file is not found, then an error is returned. This outcome is often a point of confusion for new users. Refer to the help information for a given program to see what its default file names are, so in the event of an error, it is clear which file is missing. It is generally inadvisable to rely on GROMACS default file names, as they are very generic. The tutorials below all use descriptive file names to encourage best practices in organization and

https://doi.org/10.33011/livecoms.1.1.5068
Living J. Comp. Mol. Sci. 2019, 1(1), 5068

naming conventions.

Each tutorial is available at http://www.mdtutorials.com/gmx, and the link to each specific tutorial is given within each of the following sections. Command-line syntax and input files are available at each tutorial's URL. This article will describe the practical considerations and theoretical basis for the approaches taken here.

3.2 Tutorial 1: Lysozyme in Water

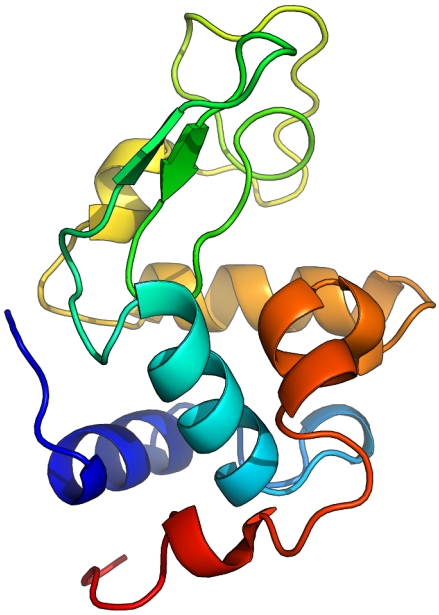

Figure 1. Crystal structure of hen egg-white lysozyme, taken from PDB 1AKI [10].

Many biomolecular simulations involve a solute that is solvated in an aqueous medium or saline solution. Thus, this tutorial guides the user through preparing a system of a protein in water, with counterions. The full tutorial is available at http://www.mdtutorials.com/gmx/lysozyme. The general protocol for any simulation is to build the coordinates and topology (a description of the properties and connectivity of all atoms in the system), perform energy minimization to relax the coordinates to a low-energy state, equilibrate, and perform a production MD simulation, during which data are collected for subsequent analysis. In this tutorial, the coordinates for the system of interest are for hen egg-white lysozyme (Figure 1), which can be downloaded from the Protein Databank, accession code 1AKI [10]. For simplicity, the user is instructed to remove crystal waters from the input PDB file:

```
$ grep -v HOH 1aki.pdb > 1AKI_clean.pdb
```

Removal of water here is to make the resulting topology less complex. Multiple blocks of water in the topology require manual correction, a task that is not well-suited to new users of the software who are likely unfamiliar with the contents and format of various files. It should be noted, however, that ad hoc removal of crystal waters is not generally a good practice, particularly in instances where a tightly bound water may have some functional significance (structural or catalytic) and must be modeled appropriately.

3.2.1 Topology Preparation

The typical approach that GROMACS takes for constructing the topology of a molecular system is to begin with generating a topology for the solute of interest, which is then amended or updated to include other components that are added to the system (water, ions, ligands, etc.). GROMACS uses two types of topology files, one with a `.top` extension and the other with an `.itp` extension. There is only ever one `.top` file for a system, hence it is called the "system topology." A topology with an `.itp` extension is called an "included topology," indicating that it can be embedded (included) within a system topology to provide the source of other parameters or other molecule definitions.

The GROMACS program most frequently used to write topologies is called `pdb2gmx`, which reads a coordinate file, determines its contents, and writes a topology for the supplied molecule(s). The name of this program is somewhat misleading, as there is no strict requirement to provide coordinates in PDB format. Refer to the help information for `pdb2gmx` for allowed formats.

The `pdb2gmx` program is best suited to linear biopolymers, with limited exceptions for branched (non-linear) bonds. Thus, it is very easy to use `pdb2gmx` for proteins and nucleic acids, but it is somewhat ill-suited for materials, branched polymers, or complex carbohydrates. However, `pdb2gmx` can handle common cases such as disulfides and heme linkages. An important point about `pdb2gmx` is that it can only process species for which parameters have been provided; it does not perform automated parametrization. Therefore, a user must carefully consider what is provided to `pdb2gmx` before attempting to process a coordinate file. Each force field provided with GROMACS has one or more `.rtp` (residue topology) files, which contain the parameters and bonded structure for each species supported by the force field. If a given species is not found in a force field `.rtp` file, `pdb2gmx` will return a fatal error. Therefore, users must always check the availability of non-standard residues in a given force field to determine its suitability for use.

Many force fields are included in GROMACS, including several variants of the AMBER force field [11–17], CHARMM22/CMAP for proteins [18, 19] and CHARMM27 for nucleic acids [20, 21], several GROMOS96 parameter sets [22–25], and OPLS-AA [26]. Also note that the CHARMM36 force

https://doi.org/10.33011/livecoms.1.1.5068
Living J. Comp. Mol. Sci. 2019, 1(1), 5068

field [27–31] is available in GROMACS format from http://mackerell.umaryland.edu/charmm_ff.shtml#gromacs.

Beyond the consideration of whether or not a force field has parameters for a given chemical species, it is important to realize that the choice of force field is perhaps the single most critical aspect to beginning a new simulation project. No existing force field perfectly treats all species, and each available force field has pros and cons for different species. It is incumbent upon the user to make an informed, justifiable choice based on literature reading and evaluation of the force field parametrization protocol to understand to which purposes a given force field is best suited. There is no catch-all answer or universally "best" force field.

Once a starting structure has been chosen and the choice of force field decided upon, run `pdb2gmx` to create the topology:

```
$ gmx pdb2gmx -f 1AKI_clean.pdb -o
    1AKI_processed.gro -water spce
```

The first prompt is for the user to select the force field that will be applied to the system. For this tutorial, choose option 15 for OPLS-AA, a widely used all-atom force field. An all-atom force field is used here for simplicity rather than introducing concepts of implicit hydrogen atoms as in united-atom force fields or coarse-grain representations. In principle, many of the force fields provided with GROMACS would be adequate for the purposes of this tutorial. The next choice the user must make is the water model that is to be used. In the tutorial, the `-water` argument is invoked from `pdb2gmx` to select the SPC/E water mode [32]; if this option is not given on the command line, the user is prompted for an interactive choice. The water model is another important consideration, though one over which the user has less freedom. Each force field was parametrized for use with a specific water model. Therefore, the user is not entirely free to choose whichever model happens to be available without due consideration. GROMACS provides a suggested water model for each force field; unless there is good evidence to choose another model, the user should follow this recommendation for greatest accuracy. However, some studies have shown that different biomolecular force fields may be accurately combined with other water models, as is the case in this tutorial. The OPLS-AA force field was originally parametrized for use with the TIP3P water model [33], but it was subsequently shown that the combination of OPLS-AA with SPC/E yielded more accurate hydration free energies for protein side chains [34], suggesting this combination is a sufficiently accurate and self-consistent model for simulating proteins. In the absence of any sufficiently strong justification or precedent, the user should always choose the recommended water model listed by `pdb2gmx`.

Often, new users are confused at having to choose a water model at this stage, particularly in the case that their provided coordinate file only contains a biomolecule such as a protein. The water model selection is planning for the subsequent steps, in which this solute is hydrated and a force field choice should be made ahead of time. By selecting the water model at this first step, `pdb2gmx` can write a complete topology that will describe all the species that will be included in the system being built.

By default, `pdb2gmx` will produce three output files: a system topology (named `topol.top` by default), a topology that specifies parameters for position restraints (named `posre.itp` by default, discussed in section 3.2.5), and a force-field compliant coordinate file (named `conf.gro` by default but in this tutorial is called `1AKI_processed.gro`). The output coordinate file is actually somewhat of a side effect of the normal function of `pdb2gmx`, which is simply to produce a topology. However, since many experimental structures determined by X-ray crystallography lack the resolution to assign hydrogen atom positions, these atoms must be built in to the model. `pdb2gmx` does this, though it cannot build in other missing atoms, requiring either a complete experimental structure or the use of modeling software such as MODELLER [35] to construct missing atoms. As a result, the processed coordinates are subsequently written out to a file to match the contents of `topol.top`. The output coordinate file can be one of several formats, the `.gro` (GROMOS87 format) is not required but it is the default in GROMACS.

After generating `topol.top`, inspect its contents. For a full description of the topology file format, refer to the online tutorial. Several keywords are worth noting here. GROMACS processes files using C preprocessor syntax and macros. Topology files will often contain statements such as `#define`, `#ifdef/#endif`, and `#include`. The `#define` macro sets the value of a variable, which can then be tested using `#ifdef` statements to call different elements of a topology conditionally. This approach is most often taken in defining rigid vs. flexible water. Many topologies have `#include` statements; these literally mean "copy and paste the contents of the named file here." CHARMM users may be familiar with this concept, which is called `stream` in that program and serves a similar function. The `#include` statements are useful for making the topology compact, rather than writing out all parameters explicitly. A parent force field, defining all necessary parameters, is the first `#include` statement, and others may reference water and ion topologies or the position restraint topology file that `pdb2gmx` created.

When `pdb2gmx` is done, it will print out the net charge of the protein to the terminal. Record this value for later use.

https://doi.org/10.33011/livecoms.1.1.5068
Living J. Comp. Mol. Sci. 2019, 1(1), 5068

3.2.2 Solvate the System

Simulating proteins *in vacuo* is not typically of biological interest, rather a condensed-phase simulation is more relevant. Thus, the next step in building the solvated protein system is to define a volume around the protein that will be filled with water. There are several important considerations in doing so, and it is also important to explain the reason that these methods are employed.

The first fundamental concept is that, in a condensed-phase system, there are no "edges" or "boundaries" to the system that is being built. If the protein was solvated in some volume of water that was simply surrounded by vacuum, ultimately the system will develop into a droplet and water molecules that are on the surface of the droplet will tend to evaporate over time. In such cases, effects such as surface tension and poor energy conservation become important. To solve this issue, most modern simulations (and certainly all that are carried out in the condensed phase) employ what are called "periodic boundary conditions" (PBC), in which identical copies of the system of interest are constructed around the central "image," which is the system that a user actually constructs. Therefore, atoms at the perceived "boundaries" of the central image will interact with periodic copies of atoms in the neighboring images. Further, any atom that diffuses "outside" of the central image will reappear on the opposite face of the central image. In truth, there is no such thing as being "outside" of a box when employing PBC, as the representation of the system is infinite.

In constructing a simulation system (the central image) for use with PBC, an important consideration is the size of that central image. Under most circumstances, it is desirable to simulate the solute of interest in dilute solution (discussion of crowded or crystalline environments is beyond the scope of this tutorial). That is, the solute should not "see" any of its periodic copies in real space. The underlying principle is called the "minimum image convention," which states that to properly calculate the force on each atom, no atom should see multiple copies of the same atom within the short-range neighbor list [3]. So how does one decide the right box size that will prevent atoms from experiencing duplicate forces? This decision requires an understanding of the requirements of the force field chosen in section 3.2.1. Each force field has a set of required nonbonded cutoffs (see discussion on values of `rvdw` and `rcoulomb` below in section 3.2.4). These cutoffs define the radius around each atom for which short-range forces are calculated. If the box is too small, such that this radius encompasses more than half of any box dimension (along the x, y, or z-axes), then forces will be double-counted, leading to artifacts. A common approach is to define a buffer around the solute of interest that is equal to the longest cutoff that will be employed in the simulation. The user should note that this type of planning is initiated even before arriving at this step in the tutorial, as it should be determined before or upon choosing a force field to represent the system.

The GROMACS program that is used for defining the box around the solute, and the subsequent positioning of the solute therein, is called `editconf`, for "edit configuration." This program allows the user to manipulate the coordinates of the solute via rotations and translations, so that molecules can be specifically positioned and oriented within a box. The most conventional usage for a system like this one is to center the solute with a box that has a defined buffer according to the minimum image convention (see below). For the lysozyme system in this tutorial, issue the following `editconf` command:

```
$ gmx editconf -f 1AKI_processed.gro -o
    1AKI_newbox.gro -c -d 1.0 -bt cubic
```

Doing so will center the protein (option `-c`, which is boolean and therefore takes no additional argument) in a cubic box (`-bt cubic`), with a minimum solute-box distance of 1.0 nm (`-d 1.0`). Note that GROMACS uses SI notation for all base units. New users often face issues in specifying distances and cutoffs if they are more familiar with software that uses Å. There are many different box shapes that can be used, specifically those that have a small volume but still satisfy the same periodic distance, but for simplicity this tutorial will only address a cubic system. Interested readers are referred to the GROMACS manual for discussion on alternate box shapes. After running `editconf`, the protein is now centered within a cubic box (Figure 2).

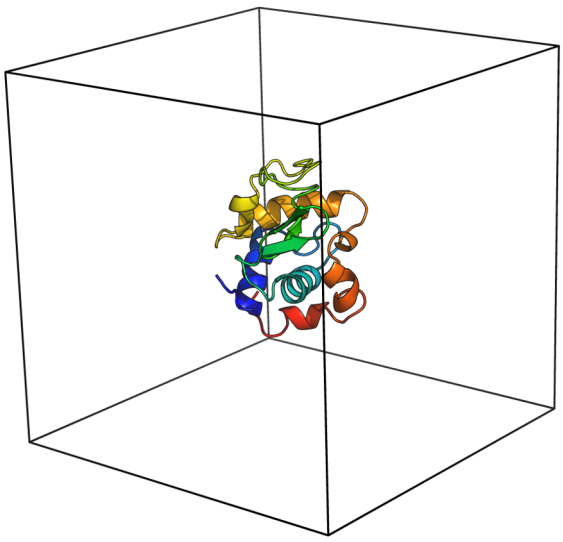

Figure 2. The structure of hen egg-white lysozyme in an empty, cubic box following the use of `editconf`.

https://doi.org/10.33011/livecoms.1.1.5068
Living J. Comp. Mol. Sci. 2019, 1(1), 5068

Now that the box size and shape have been defined, it is possible to visualize the periodic images of the system that were alluded to above. Such a representation is shown in Figure 3.

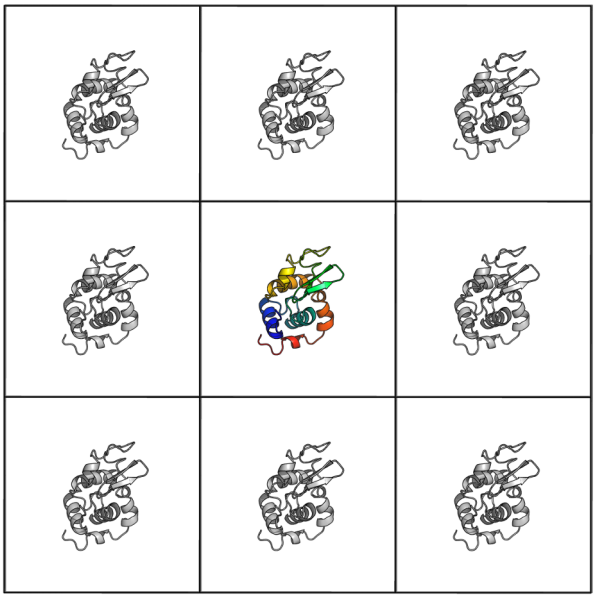

Figure 3. Periodic construction of the central image (with lysozyme colored as a rainbow gradient from blue (N-terminus) to red (C-terminus). Periodic images of the protein are shown in light gray.

Having defined a suitable volume around the protein, the system is subsequently filled with water. The program that does this is called `solvate`. This program relies on the use of a small, pre-equilibrated cubic box of water, which is then placed onto a grid that is evenly distributed through the volume of the `1AKI_newbox.gro` coordinate file. Any water molecules with atoms that overlap the protein (or any species already within the box) are deleted.

```
$ gmx solvate -cp 1AKI_newbox.gro -cs
    spc216.gro -o 1AKI_solv.gro -p topol.top
```

The `solvate` program requires both an input coordinate file for the solute (specified with the `-cp` option, conventionally the "coordinates of the protein") and the solvent (option `-cs` for "coordinates of the solvent"). In this example, the empty volume is filled using the `spc216.gro` coordinate file, which contains 216 water molecules that have been pre-equilibrated using the SPC model [36]. A common question that new users have is about the availability of coordinates for other 3-point water models such as TIP3P [33] and SPC/E [32] for the purpose of solvation. Simply, distinct files are not necessary, as any coordinate file of 3-point water can serve as a reasonable starting point for such systems. After a short equilibration period (see below, section 3.2.5), the solvent will relax according to the properties of these other models.

Where does the `spc216.gro` file come from? GROMACS has a database of pre-built coordinate files for 3-, 4-, and 5-point models of water, located in `$GMXLIB` (an environment variable that refers to the `share/gromacs/top` subdirectory of wherever GROMACS is installed on the computer). GROMACS will search in this directory for files specified on the command line before looking in the working directory.

It is important to note the use of the `-p` option to specify the name of the system topology. Doing so will cause `solvate` to automatically update the system topology with the number of water molecules it added. If the user does not supply the topology on the command line, updates must be made manually. It is strongly encouraged that users instruct GROMACS programs to do bookkeeping for them, rather than trying to make manual adjustments that can be error-prone.

3.2.3 Add Ions

With the system solvated, the last step in constructing coordinates is to add ions. Biological and *in vitro* systems often contain some amount of salt, and it is in the interest of those performing simulations to model relevant conditions as closely as possible. Beyond this point, MD simulations are typically carried out under electroneutral conditions; that is, the system does not carry net charge. Individual molecules (proteins, ions, etc.) may carry a formal charge. This convention reflects typical experimental conditions but there is another point that needs to be mentioned. Electrostatic interactions are most often evaluated using the particle mesh Ewald (PME) method [37, 38], which requires an electroneutral system. The PME method itself can supply what is known as a "uniform neutralizing plasma," in which an offsetting charge is spread uniformly across the system to neutralize it. For homogeneous systems, this approach is adequate and the addition of explicit ions is not required. However, systems of biological interest are rarely homogeneous, and relying on the neutralizing plasma can result in serious artifacts [39]. For this reason, monoatomic ions are typically added within the aqueous solution to counterbalance any net charge from the solute(s) present.

The GROMACS program for adding ions is called `genion`. It adds ions by replacing molecules in whatever group the user specifies, typically water. To do so, `genion` needs both coordinate information (to know what coordinates to assign to the added ions) as well as topological information (to know which atoms are connected, therefore defining molecules that are deleted in their entirety). Such information is present in a single file type with the extension `.tpr`. A `.tpr` file also contains simulation instructions, which are written in plain-text files with the extension `.mdp`. In this context, no instructions for performing a simulation are required, but the file format

https://doi.org/10.33011/livecoms.1.1.5068
Living J. Comp. Mol. Sci. 2019, 1(1), 5068

contains everything that is needed. The contents of an `.mdp` file will be discussed in detail in subsequent sections of this tutorial. To produce a `.tpr` file, invoke the GROMACS preprocessor, `grompp`:

```
$ gmx grompp -f ions.mdp -c 1AKI_solv.gro
    -p topol.top -o ions.tpr
```

The `grompp` program reads in coordinates (`-c`), topology (`-p`), and a simulation input file (`-f`) and produces the `.tpr` file. For the purpose of adding ions, the `.mdp` file passed to the `-f` option can contain any syntactically correct set of options. In this tutorial, a very simple file (`ions.mdp`) is supplied, to minimize the number of options that must be correctly specified. The `ions.mdp` file calls for energy minimization but this process is not actually carried out. It is also important to note that a plain cutoff method is specified for electrostatics; doing so is inadvisable for performing a simulation, but `grompp` will generate a warning if the user attempts to execute a process with PME for a non-neutral system. As counterions have not yet been added to the system, PME should not be specified. Moreover, since this `.mdp` file is not actually being used for any simulation, it need not specify rigorous simulation methods.

Recall the net charge printed by `pdb2gmx`, which should be +8. This number means that the protein has a net positive charge at neutral pH, requiring the addition of 8 anions to neutralize it. To do so, execute `genion` as follows:

```
$ gmx genion -s ions.tpr -o
    1AKI_solv_ions.gro -p topol.top -pname
    NA -nname CL -neutral
```

Choose group 13 (SOL) to replace water molecules with ions. Do not choose System or Protein, or any similar group, otherwise the output system will be completed fragmented and unphysical.

`genion` reads the `.tpr` file passed to the `-s` option and outputs (`-o`) a coordinate file called `1AKI_solv_ions.gro`. It makes changes to the topology (deleting water and adding the requested ions), therefore the use of `-p` to specify the topology is strongly encouraged. The `-pname` and `-nname` specify the residue names of positive and negative ions, respectively. This example does not require the addition of sodium ions (Na^+) but is shown for illustrative purposes. The user can specify the number of negative ions to be added with `-nn`, and if positive ions were needed, they would be specified with `-np`. Note that the charge on different systems will be different, and in some instances, it is desirable to add a higher salt concentration. To automatically neutralize a system, use the `-neutral` option, and to set a target concentration for which `genion` will determine the number of ions to be added (rather than specifying them manually), use `-conc` with a suitable value (in M).

The final, solvated system will look something like what is shown in Figure 4. Note that due to the random nature of adding ions with `genion`, the location of Cl^- ions will differ in every system.

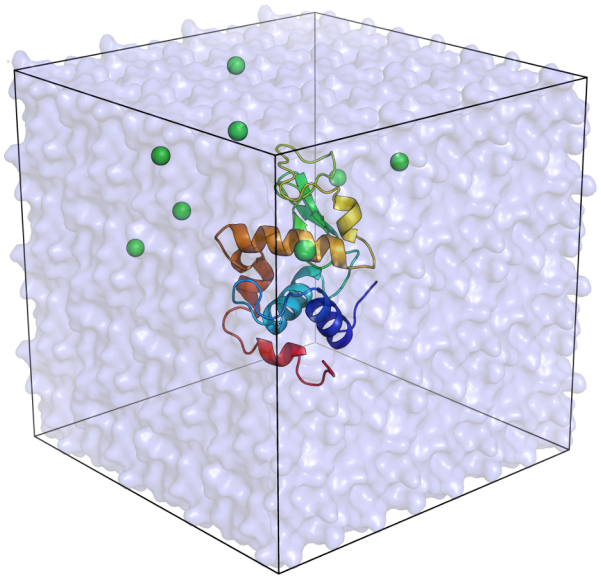

Figure 4. Rendering of the final system after solvation and adding ions. Green spheres are Cl^- ions, whose positions will vary randomly as a result of `genion`.

3.2.4 Energy Minimization

Before beginning dynamics, it is necessary to relax the solvated system to a low-energy state. Energy minimization is the process that moves the positions of atoms according to the forces acting upon them. In preparing the system, hydrogen atoms were constructed by `pdb2gmx` in assumed geometries, water molecules were added in a grid around the protein by `solvate` in a manner that may create suboptimal hydrogen bonding, and ions were added randomly by `genion` with no regard to nearby atoms with which the charges might clash. As a result, the system must be relaxed to a low-energy state, otherwise any simulation will likely be unstable. It is impossible to know whether or not the system is at its global energy minimum, but this is not necessarily the goal of energy minimization, rather it seeks to find a reasonable starting point for the simulation.

In this section, the contents of the `.mdp` file will be discussed in depth. Refer to the `minim.mdp` file, which will be used as the instructions for the energy minimization process. The first section of this input file specifies the parameters for carrying out energy minimization:

```
integrator      = steep
emtol           = 1000.0
```


https://doi.org/10.33011/livecoms.1.1.5068
Living J. Comp. Mol. Sci. 2019, 1(1), 5068

```
emstep          = 0.01
nsteps          = 50000
```

These instructions tell the GROMACS program `mdrun` (discussed below) to use the steepest descents method for energy minimization, and to terminate the process if the magnitude of the potential energy gradient is 1000.0 kJ mol^{-1} nm^{-1} or smaller. The maximum step size along the gradient is 0.01 nm, and a maximum of 50000 steps are allowed. The default value of `emstep` in GROMACS of 0.01 nm is used, but it is often necessary to use a smaller maximum step (*e.g.* 0.002 nm) for systems that have difficulty converging; the smaller step size allows for a more thorough walk over the potential energy surface, whereas a larger step may miss a path to the true minimum. It is also possible to allow the process to go on indefinitely, stopping only when convergence is reached (due to numerical precision or by actually achieving `emtol`) by setting `nsteps = -1`. Note that `emstep` has units of distance, not time. Energy minimization is not a dynamical process; time does not elapse between each step and there are no velocities. Energy minimization steps are expressed in terms of maximum displacement along the vector indicated by the force.

The remainder of the `.mdp` file contains instructions for how to compute nonbonded interactions:

```
nstlist         = 1
cutoff-scheme   = Verlet
ns_type         = grid
coulombtype     = PME
rcoulomb        = 1.0
rvdw            = 1.0
pbc             = xyz
```

The value of `nstlist` controls how many steps (minimization or MD integration) elapse between updating the neighbor list for determining which atoms contribute to the short-range forces. In MD simulations, this value is larger because atoms exchange from the neighbor list in diffusion-limited time, but for energy minimization, it needs to be set to 1 because the configuration may change considerably between each step. The `cutoff-scheme = Verlet` setting uses a buffered neighbor list, that is, atoms outside the longest cutoff are still tracked to improve energy conservation. Neighbor searching is done by checking atoms in neighboring grids (`ns_type = grid`) rather than checking every possible atom (`ns_type = simple`). All short-range nonbonded interactions (electrostatics and van der Waals) are truncated at 1.0 nm (`rcoulomb = 1.0` and `rvdw= 1.0`), and periodic boundary conditions are applied in all three dimensions (`pbc = xyz`). The PME method is used to calculate long-range electrostatic forces.

Note that in `.mdp` files, there is no difference between a hyphen (-) and underscore (_); hence in the above example, `ns-type` and `ns_type` would be equivalent, as would `cutoff-scheme` and `cutoff_scheme`.

As above, invoke `grompp` to assemble the instructions for energy minimization (`-f minim.mdp`), coordinates (`-c 1AKI_solv_ions.gro`), and topology (`-p topol.top`) to create the run input file for energy minimization (`-o em.tpr`):

```
$ gmx grompp -f minim.mdp -c
    1AKI_solv_ions.gro -p topol.top -o
    em.tpr
```

The GROMACS `mdrun` program is responsible for performing all minimization and dynamics processes. To run energy minimization, use the following syntax:

```
$ gmx mdrun -v -deffnm em
```

The `-v` option invokes "verbose" mode, in which an estimate of time remaining is printed to the screen, along with the current energy minimization step, the step size, potential energy, and maximum force. It is not a required option, and if printed to a file, may result in a large file that is saved to disk. It is, however, instructive to watch the progress of energy minimization to understand what is going on. The `-deffnm` option defines the base file name for all input and output files and eliminates the need for using individual input and output flags to specify names.

`mdrun` requires the `.tpr` file as its only input, normally passed to the `-s` flag. `mdrun` produces several file types, including an ASCII text file with a `.log` extension, a binary file with all energy values with a `.edr` extension, and trajectory files with either `.trr` (full-precision coordinates, velocities, and/or forces) or `.xtc` (reduced precision coordinates only) extensions. When using `-deffnm`, these files will be called `em`, with a corresponding file extension. It is useful to name files in this manner to avoid relying on default file names (`md.log`, `ener.edr`, `traj.trr`, `traj_comp.xtc`), which will be the same for any process carried out by `mdrun`.

When `mdrun` is done, the user will see something similar to the following:

```
Steepest Descents converged to Fmax < 1000 in 726 steps
Potential Energy  = -5.8448769e+05
Maximum force     =  9.6957593e+02 on atom 736
Norm of force     =  2.3290928e+01
```

For a system such as this, it is expected that the final potential energy will be a negative value, on the order of 10^5 - 10^6 kJ mol^{-1}. Potential energy is an extrinsic quantity, so larger systems will have larger magnitudes. For a solvated protein, the value will be negative, as favorable electrostatic interactions between water molecules dominate the potential energy terms. Smaller systems may even have positive potential energy, arising from the fact that in such cases, the intramolecular bonded terms have a larger magnitude than

https://doi.org/10.33011/livecoms.1.1.5068
Living J. Comp. Mol. Sci. 2019, 1(1), 5068

nonbonded terms. Bonded components of the potential energy function have positive values, by definition. So for small systems, these may prevail and the final, potential energy is positive. Such an outcome is not necessarily an indication that anything is wrong. It would, however, be unusual for a fully solvated protein system such as this one to have a positive potential energy.

The various energy terms saved during energy minimization are stored in the `em.edr` file, and can be extracted with the `energy` program. For example, to extract the potential energy of a system as a function of energy minimization step:

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

By default, GROMACS analysis programs write output formatted for use with the XmGrace plotting program, though this formatting can be changed with the `-xvgr` option. The output `potential.xvg` file can be plotted to produce Figure 5.

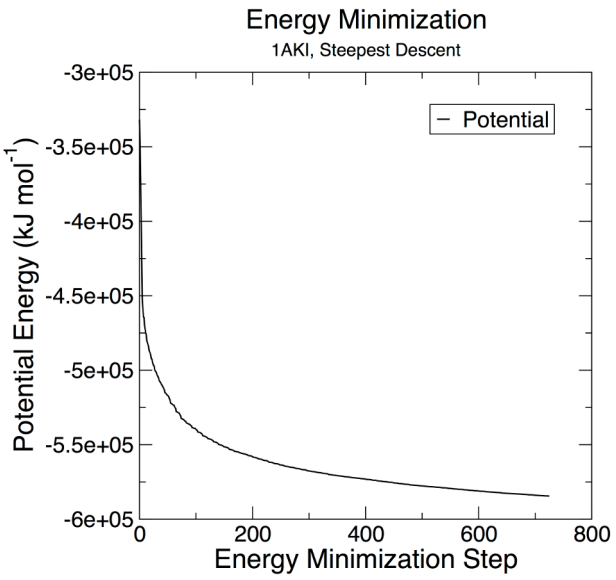

Figure 5. Potential energy of the solvated lysozyme system as a function of energy minimization step.

The solvated system is now at a reasonable energy minimum. Note that the steepest descents method is very efficient but also may have difficulty finding a true energy minimum. The settings used in this tutorial are fairly permissive but work reasonably well for most similar systems. There are applications, however, that require more exhaustive energy minimization (for instance, free energy calculations such as those in section 3.7), either through multiple rounds of minimization with different algorithms that may more carefully sample energy minima, or with stricter convergence criteria.

It is also important to note that the large changes in potential energy are often confusing to new users. During dynamics, it is expected that energy will be conserved, but the purpose of energy minimization is to change the configuration of the system in a manner that leads to a decrease in energy. As such, obtaining an average energy during energy minimization is not a useful metric, nor is any examination of energy conservation.

3.2.5 Equilibration

Following energy minimization, the system is now ready for the start of dynamics. Equilibration is carried out to obtain a stable thermodynamic ensemble for whatever conditions are desired. Most biomolecular simulations are carried out under a canonical (NVT) ensemble, in which the number of particles (N), volume (V), and temperature (T) of the system are conserved, or an isothermal-isobaric (NPT) ensemble, in which the number of particles (N), pressure (P), and temperature (T) are conserved. There are other statistical mechanical ensembles that can be employed, but this tutorial will focus on those that are most relevant to biomolecular systems. Before carrying out a simulation under a given ensemble, the system must be equilibrated under the same conditions. For this tutorial, the production MD simulation will be carried out under an NPT ensemble. It is possible to simply initiate an equilibration simulation under the same conditions, but it is often more robust to first equilibrate the temperature of the system under an NVT ensemble before applying a barostat to control the temperature. The simultaneous combination of velocity generation and coordinate scaling under the influence of the barostat can introduce instabilities in a system that may be far from equilibrium.

It is also important to note that rather than initiating the simulation at the desired temperature, it is also possible to slowly warm the system up. This technique will not be applied in this tutorial, as there is some debate as to whether or not it is necessary in most cases. Interested readers are directed to the GROMACS manual section on "simulated annealing" for information on how this process is carried out in GROMACS.

To initialize dynamics, velocities are generated by `grompp` and written to the `.tpr` file. The following settings in the `nvt.mdp` file control the generation of velocities:

```
gen_vel                 = yes
gen_temp                = 300
gen_seed                = -1
```

A new concept is the application of position restraints. A restraint is a biasing potential that is applied to selected atoms to disfavor motion. It should be noted that position restraints do not prevent motion, rather they penalize it. Position restraints are typically applied to solute non-hydrogen atoms during equilibration to prevent deformations that could be induced by the random nature of the starting velocities and subsequent collisions between the solute and

https://doi.org/10.33011/livecoms.1.1.5068
Living J. Comp. Mol. Sci. 2019, 1(1), 5068

solvent. Position restraints are specified with the following keyword in the `nvt.mdp` file:

```
define                  = -DPOSRES
```

Recall the generation of the `posre.itp` file by `pdb2gmx` in section 3.2.1, which is only invoked as part of an `#ifdef` block in `topol.top`. This construct means restraints are only applied when the user wants them, with invocation being dependent upon the use of the `define` keyword in the `.mdp` file. The presence of `define = -DPOSRES` means that the `#ifdef POSRES` condition is evaluated as true by `grompp` and the position restraint topology (`posre.itp`) is applied to the system. Note that the general syntax of `#ifdef KEYWORD` and `define = -DKEYWORD`; failing to prefix the keyword with `-D` will not lead to any error, but the `#ifdef` condition will not be matched.

Some of the settings in the `.mdp` file are the same as in energy minimization and will not be repeated here. New settings that are introduced during NVT are:

```
integrator              = md
nsteps                  = 50000
dt                      = 0.002
```

The above lines specify that this process is an MD simulation, and the `md` setting specifies the leap-frog integrator in GROMACS. A total of 50,000 integration steps will be performed, and each step corresponds to an integration time step, δt, of 0.002 ps, or 2 fs. The simulation will therefore last for 100 ps (50,000 steps $\times$ 0.002 ps per step).

Output is specified in the next section of `nvt.mdp`:

```
nstxout                 = 500
nstvout                 = 500
nstenergy               = 500
nstlog                  = 500
```

The `nstxout` and `nstvout` settings control the interval of time steps between writing coordinates (x) and velocities (v) to the trajectory (`.trr`) file. With these settings, coordinates and velocities are saved every 1 ps. Similarly, energy terms are written to the `.edr` file (`nstenergy`) and `.log` file (`nstlog`) at the same interval.

Bond constraints are applied to allow for the 2-fs integration time step specified above. The value of δt is limited by the fastest vibrational modes in the system, which are typically bonds to hydrogen atoms. As these bonds will rarely exist in excited states, it is reasonable to model them as rigid, replacing the harmonic oscillator term with a holonomic constraint. Doing so allows the value of δt to be increased from roughly 0.5 fs up to 2 fs for typical dynamics runs. In the `.mdp` file, the relevant settings are:

```
continuation            = no
constraint_algorithm    = lincs
constraints             = h-bonds
lincs_iter              = 1
lincs_order             = 4
```

The `continuation` keyword tells `mdrun` that the coordinates are not the output of a previously constrained simulation, so the constraints need to be solved at the first time step. The method for applying constraints is LINCS, the Linear Constraint Solver [40, 41]. Constraints are applied to bonds to hydrogen atoms `constraints = h-bonds`, and the remaining settings are the default LINCS parameters for accuracy of the constraint equations. It is important to note that "constraints" and "restraints" are different. As described above, a restraint is a biasing potential that disfavors motion, whereas a constraint is used to fix a distance or angle in a simulation. Older literature often uses these terms interchangeably, but in modern simulation practice, these two terms have distinct meaning.

The next relevant section is related to nonbonded interactions, which were described above in section 3.2.4. After the nonbonded settings are the keywords related to temperature control:

```
tcoupl                  = V-rescale
tc-grps                 = Protein  Non-Protein
tau_t                   = 0.1      0.1
ref_t                   = 300      300
```

The `tcoupl` keyword specifies the algorithm that is used for temperature coupling, also known as the thermostat. In this example, the velocity rescaling method of Bussi et al. is applied [42]. Another thermostat commonly applied during equilibration is the weak coupling method of Berendsen et al. [43] (specified with `tcoupl = berendsen` in the `.mdp` file). The Berendsen weak coupling method has fallen out of favor, as it has been demonstrated that it does not sample a correct canonical velocity distribution [42]. The Bussi velocity rescaling method, however, does sample a correct velocity distribution, and has the same advantages as the Berendsen method in that it quickly relaxes the system to the target temperature, making it practical and efficient for use in both equilibration and production MD simulations. The `tc-grps` keyword specifies which groups of atoms are coupled to the specified thermostat. All atoms should be coupled, but they can be separated into different groups. Strictly speaking, any separation of atoms into different groups violates the Equipartition Theorem, in that thermal energy can flow into and out of the thermal reservoir (thermostat) independently between the different groups, rather than being exchanged between the atoms in the system. The practice of coupling solute and solvent atoms separately arises from the "hot solvent/cold solute" problem in MD simulations [44], in which the solvent heats while the solute cools, due to a host of factors related

https://doi.org/10.33011/livecoms.1.1.5068
Living J. Comp. Mol. Sci. 2019, 1(1), 5068

to approximations made in MD integration. Here, the protein and all other non-protein atoms (water and ions) are coupled to separate thermostats at 300 K, with a coupling time (τ_T) of 0.1 ps, which reflects rather tight regulation, which is appropriate for equilibration.

Create the NVT equilibration `.tpr` file with `grompp`:

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

The command-line options to `grompp` are largely the same as those used above for energy minimization, but there is one addition, the `-r` flag. This option specifies the name of a coordinate file that is used as the origin for the position restraint biasing potential. That is, this coordinate file contains the reference positions (hence `-r`) at which the biasing potential on the atoms would be zero. In previous versions of GROMACS, this option was not mandatory, but in the 2018 version, it is required to prevent users from incorrectly specifying the origin of their biasing potentials. In most cases, however, the coordinates passed to `-c` and `-r` will be the same. There are advanced use cases in which different coordinate files will be used, but they are beyond the scope of this tutorial.

Perform NVT equilibration by invoking `mdrun`:

```
$ gmx mdrun -deffnm nvt
```

After the run, analyze the temperature time series to verify that the target temperature (300 K) was achieved and maintained, using the `energy` program. At the prompt, type `16 0`, then Enter. Group 16 is the temperature time series, and 0 terminates input to `energy`.

```
$ gmx energy -f nvt.edr -o temperature.xvg
```

The time series of temperature during NVT is shown in Figure 6.

As it is ultimately desirable to conduct the production MD simulation under NPT conditions (reflecting typical laboratory conditions), equilibration is continued under an NPT ensemble. Position restraints are maintained on the protein non-hydrogen atoms. New settings in the `.mdp` file related to pressure coupling are as follows:

```
pcoupl                  = Parrinello-Rahman
pcoupltype              = isotropic
tau_p                   = 2.0
ref_p                   = 1.0
compressibility         = 4.5e-5
refcoord_scaling        = com
```

The equilibration protocol utilizes a barostat method proposed by Parrinello and Rahman [45], and later modified by Nosé and Klein [46]. As with thermostats explained in the NVT equilibration step, barostats require a reference pressure (`ref_p`) and a period of time over which the pressure

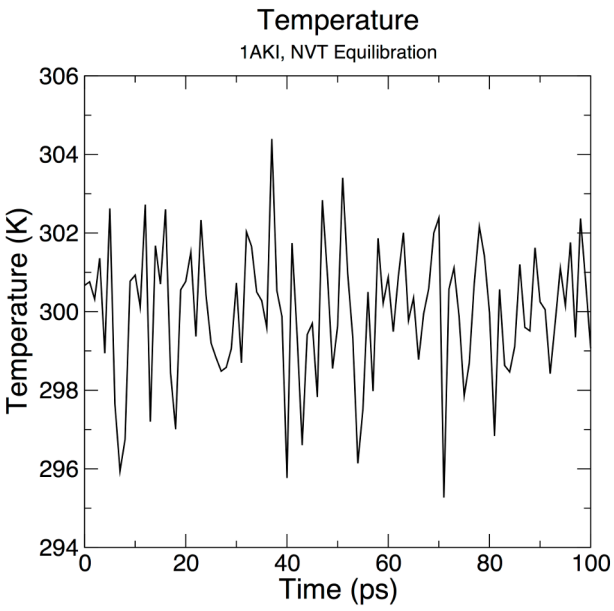

Figure 6. Temperature of the solvated lysozyme system over time during NVT equilibration.

of the system is allowed to relax towards the target (`tau_p`). The `pcoupltype` keyword controls the deformation of the box in all three dimensions. For a homogenous system such as the one constructed here, it is appropriate to scale all box vectors uniformly. Such scaling is achieved with `isotropic` pressure coupling. GROMACS has options for semi-isotropic, with *x* and *y* coupled together, *z* coupled independently, as will be used in the membrane protein tutorial (see Section 3.3). Anisotropic coupling allows all dimensions (including diagonal elements of the box) to be scaled separately (useful for crystalline materials and solid phases). The `compressibility` setting controls the speed of the response of the barostat and is, by default, set to the isothermal compressibility of water (4.5×10^{-5} bar^{-1}).

The `refcoord_scaling` setting is often confusing for new users. Pressure is scaled by actually scaling atomic coordinates as a function of increases or decreases in box size. The reference coordinates (see NVT equilibration above) set the origin of the position restrain biasing potential. If the reference coordinates are not similarly scaled, the forces applied due to the restraints will accumulate and become artificial as the updated coordinates drift slightly and are scaled, while the reference coordinates remain fixed. Reference coordinates can be scaled either based on the center-of-mass (COM) of the restrained molecule(s) or by individual atoms. In principle, either approach works sufficiently well for equilibration, but scaling relative to the COM of the restrained molecule(s) is often somewhat more numerically stable, especially for systems that are not necessarily at thermodynamic equilibrium.

https://doi.org/10.33011/livecoms.1.1.5068
Living J. Comp. Mol. Sci. 2019, 1(1), 5068

An additional difference in the `npt.mdp` file is the treatment of bonds. The `continuation` keyword is set to a value of `yes`, indicating that this simulation run is the continuation of a previous run in which constraints were applied. To preserve continuity, the constraints should not be re-solved, and the input coordinates should be assumed to satisfy the constraint algorithm being applied.

Prepare the run input file (`.tpr`) again using `grompp`. Note here that the checkpoint file from NVT equilibration (`nvt.cpt`) is passed to the `grompp -t` option. The checkpoint file contains a complete description of the thermodynamic state of the system, in high precision. To exactly continue a simulation, the checkpoint file must be passed to `grompp`. If it is omitted, information regarding velocities, thermostat variables, and high-precision coordinates will be lost. The checkpoint file is an essential component of proper continuation of MD simulations.

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

As before, execute `mdrun` to perform NPT equilibration:

```
$ gmx mdrun -deffnm npt
```

After the run has concluded, again use the `energy` program, in this case to extract the pressure time series. Enter 18 to select Pressure, and 0 to terminate input to `energy`:

```
$ gmx energy -f npt.edr -o pressure.xvg
```

The pressure time series is plotted in Figure 7. The screen output also indicates that the average pressure was 7.5 ± 160.5 bar. The target value was 1 bar. This discrepancy is often a source of concern for new users. Consider, however, the large error bar associated with the time series, and the wide range of pressure values sampled during the simulation (Figure 7). Pressure is a quantity that fluctuates wildly in a microscopic system. As a result, instantaneous values are going to vary considerably, and average values may not correspond exactly to the desired target value. However, given the huge standard deviation of pressure during NPT equilibration (160.5 bar), the average obtained during the simulation (7.5 bar) is statistically indistinguishable from the target of 1 bar. It is possible to extend the NPT equilibration phase for additional time in an attempt to obtain better agreement, but it is not necessary to do so. The existing agreement is sufficient.

An additional indicator of the convergence of NPT equilibration is the density of the system. Density is less prone to the fluctuations observed in the instantaneous pressure values. Again, use the `energy` program to extract the density time series from the `npt.edr` file, choosing 24 for density and 0 to terminate input:

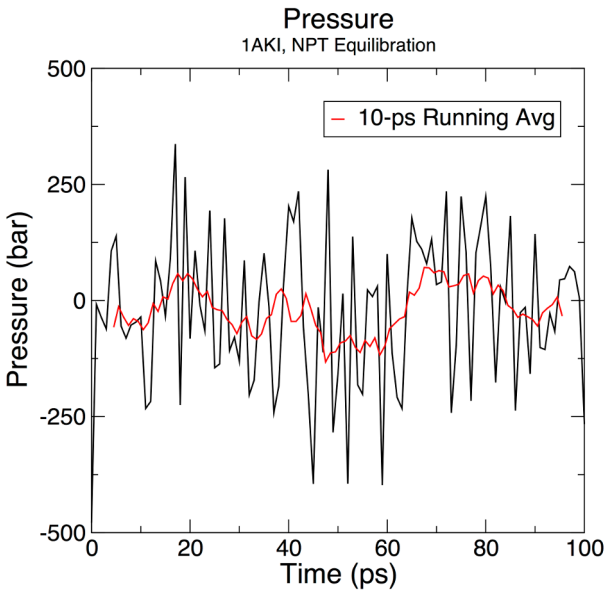

Figure 7. Pressure of the solvated lysozyme system over time during NPT equilibration.

```
$ gmx energy -f npt.edr -o density.xvg
```

The density time series is shown in Figure 8. The average value from NPT equilibration is 1019 ± 3 kg m^{-3}. Note that this value is about two percent higher than the expected value for bulk SPC/E water (998 kg m^{-3}) [32]. Such an outcome is not unusual, as the system contains a protein and ions, therefore its density will not be exactly equivalent to that of pure water.

Since the density is essentially constant over the duration of NPT equilibration, it is reasonable to conclude that the system is adequately equilibrated and unrestrained simulations can be initiated.

3.2.6 Production MD Simulation

At this point, the solvent has relaxed around the protein and the system is equilibrated under the desired statistical mechanical ensemble, NPT. Following equilibration, position restraints are removed from the protein and the simulation proceeds in an unbiased manner. As such, this phase of the MD simulation is often referred to as "unrestrained MD" or "production MD," the latter owing to the fact that trajectory data that are being collected are now considered useful for answering the scientific question(s) at hand.

Production MD proceeds like any other process carried out thus far, with the generation of a `.tpr` input file with `grompp`, providing the coordinates and checkpoint file from NPT equilibration to ensure an exact continuation:

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


https://doi.org/10.33011/livecoms.1.1.5068
Living J. Comp. Mol. Sci. 2019, 1(1), 5068

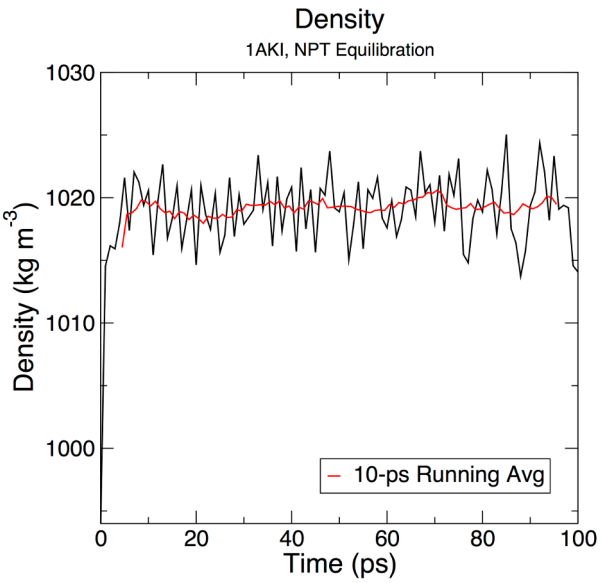

Figure 8. Density of the solvated lysozyme system over time during NPT equilibration.

Note that in this `grompp` command, there is no use of `-r`, as no restraints are being applied. The `.mdp` settings are the same as during NPT equilibration, except no restraints are specified and the run is being carried out for 500,000 steps, corresponding to 1 ns (500,000 steps × 0.002 ps per step = 1000 ps = 1 ns).

As before, it is useful to name files in accordance with the time interval and process being simulated. Here, the user will perform a 1-ns, unrestrained MD simulation, starting from t = 0 ns (equilibration time is not counted) and proceeding to t = 1 ns, hence the base file name of `md_0_1`. To run the simulation, invoke `mdrun`:

```
$ gmx mdrun -deffnm md_0_1
```

3.2.7 Data Analysis

Data analysis is the most difficult aspect of carrying out an MD simulation. It is generally advisable to plan appropriate analyses before initiating the simulation to ensure that the simulation is designed in such a way as to produce useful data that answer the scientific question(s) at hand. One should not simply execute a simulation and hope that something interesting will happen that will reveal itself in time. This tutorial cannot possibly cover all analysis methods, nor should the simple tasks here be considered required for analyzing any given MD simulation. In this tutorial, the user will conduct several basic analyses that reflect typical syntax of running various GROMACS analysis routines. Beyond being a fast MD engine, GROMACS comes pre-packaged with numerous analysis programs, facilitating the collection and processing of data.

The first step in analysis is to remove the influence of PBC that were applied during the simulation, a process often referred to as "re-imaging" or "re-wrapping." As a consequence of PBC, molecules often appear "broken" in trajectory frames, or may appear to be "outside" the unit cell. Neither is actually true. Molecules are always intact according to the topology of the system, but may be represented in a manner such that part of a molecule is at one "side" or "corner" of the box, with the remainder elsewhere. This outcome is a result of the fact that visualization convenience is irrelevant to `mdrun` in its physical calculation. Rather than wasting time re-wrapping molecules to make them appear "whole" during the run, `mdrun` writes "broken" molecules that must later be fixed, if desired. As molecules diffuse through the system, it may appear that some molecules (*e.g.* the protein) "leave" the box or appear "outside" of it. This is not the case; there is no such thing as "outside" an infinite system.

The GROMACS program that applies coordinate manipulations to account for periodicity effects is called `trjconv` ("trajectory converter") and it serves a number of functions. Beyond accounting for PBC effects, it can be used to convert file formats or save a subset of atoms from the system. In this case, only a simple invocation of `trjconv` is necessary, to simultaneously make "broken" molecules appear whole and to center the protein in the unit cell. These processes can be done in a single command:

```
$ gmx trjconv -s md_0_1.tpr -f md_0_1.xtc
    -o md_0_1_noPBC.xtc -pbc mol -center
```

The user is prompted for two selections. The first requires a choice of which group to center in the unit cell. Choose group 1 (Protein). The second prompt asks the user to choose which group should be output in the modified trajectory. For simplicity, choose group 0 (System).

Additional manipulations can be applied, such as rotational and translational fitting to produce a smoother visualization of the trajectory. Such fitting relies on a least-squares fit to a reference structure in the `.tpr` file. For the purposes of this tutorial, no such fitting will be applied, but interested readers are referred to the `-fit` option of `trjconv` and associated options. Note that one should not simultaneously use `-fit` and `-pbc`, as they are mathematically incompatible. Correct periodicity effects first, then deal with rotational and/or translational fitting.

At this point, the user should visualize the trajectory using programs like VMD [7]. Watching the progression of the trajectory is the most important part of analysis. Though GROMACS has many built-in analysis routines, the user will not know *a priori* everything that will happen during the MD simulation. Animating the trajectory can also be useful for

https://doi.org/10.33011/livecoms.1.1.5068
Living J. Comp. Mol. Sci. 2019, 1(1), 5068

determining if the `trjconv` treatment was appropriate. Many GROMACS programs are not PBC-aware, meaning if the user does not properly re-image the trajectory to account for PBC effects, analysis will be incorrect, particularly in the case of structure-related properties like those discussed here.

If one is interested in quantifying how much the protein structure changed over the course of the simulation, the root-mean-square deviation (RMSD) can be computed. RMSD is calculated following a least-squares fit to the reference structure and is plotted as a function of time. The GROMACS program that computes RMSD is called `rms`. It reads the reference coordinates, masses, and topology from the `.tpr` file passed to `-s` and computes the RMSD for each frame found in the trajectory passed to `-f`. Typically, the RMSD of a protein is calculated for its backbone (N, C_α, C) or "main chain" (N, C_α, C, O) atoms rather than the entire structure. Flexible side chains may have large RMSD that do not actually reflect meaningful structural changes. To calculate the RMSD of the protein backbone with respect to the equilibrated coordinates, invoke `rms` as follows:

```
$ gmx rms -s md_0_1.tpr -f md_0_1_noPBC.xtc
    -o rmsd.xvg -tu ns
```

When prompted, choose group 4 (Backbone) for both the group for fitting and the group for output. The RMSD thus reflects the net change in backbone configuration after accounting for global rotation and translation.

It may also be of interest to calculate the RMSD with respect to some other structure, such as a specific frame of interest or the crystal structure rather than the equilibrated (t = 0 ns) coordinates. To compute the RMSD with respect to the crystal structure, provide `rms` with the `em.tpr` file, which was the input to energy minimization. The coordinates of the protein contained therein are the crystal structure with the hydrogen atoms added by `pdb2gmx`. Invoke `rms` again:

```
$ gmx rms -s em.tpr -f md_0_1_noPBC.xtc -o
    rmsd_xtal.xvg -tu ns
```

As before, choose group 4 (Backbone) for both the group for fitting and the group for output. The two resulting RMSD time series can be plotted together, as shown in Figure 9.

The RMSD values with respect to the crystal structure and equilibrated system are not identical, with the RMSD to the crystal reference being systematically higher than the RMSD to the equilibrated structure. This outcome is not unexpected. Though restraints were applied during equilibration, it is not guaranteed that the protein will not move. Instead, it moves slightly but overall, its conformation is maintained. After about 0.5 ns, the RMSD levels off around 0.1 nm (1 Å), which is quite small and suggests the protein is very stable. It is appropriate to use this space to dispel several common misconceptions about RMSD. First, it is a degenerate metric, meaning a single value can have multiple interpretations (mathematical solutions), and as such on its own, RMSD is not very useful. Second, one cannot say that a simulation has converged simply based on RMSD, due to the first point – a structure can still be changing considerably from its starting structure even if the RMSD has largely plateaued. Therefore, one should never terminate a simulation, believing it to be "stable" or "converged" simply by looking at RMSD.

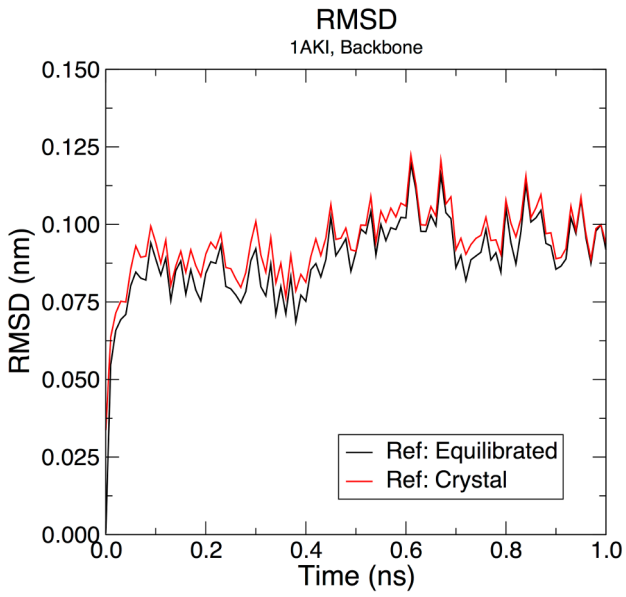

Figure 9. RMSD of the lysozyme backbone atoms during the production MD simulation following least-squares fitting to the indicated reference structures.

As a final example of analysis, calculate the radius of gyration (R_g) of lysozyme over time with the GROMACS `gyrate` program. This quantity is not particularly useful for systems of well-folded proteins, as it is a measure of compactness, a quantity expected to be largely constant for a protein with stable tertiary structure. Calculating R_g demonstrates an advanced capability of GROMACS. Invoke `gyrate` as follows:

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

The R_g time series is shown in Figure 10. This quantity is similarly stable over this short simulation, which is expected given the stable fold of lysozyme and the very short nature of the simulation. Large structural changes are not expected within 1 ns for any protein.

3.2.8 Summary and Review of Objectives

Through this tutorial, the user has been guided through the construction of a solvated protein system, including neutralizing counterions. To review, the objectives for this tutorial were as follows:

https://doi.org/10.33011/livecoms.1.1.5068
Living J. Comp. Mol. Sci. 2019, 1(1), 5068

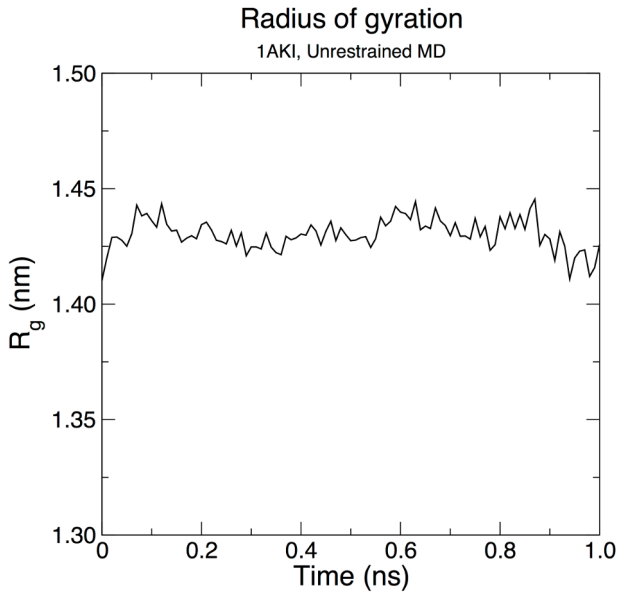

Figure 10. Radius of gyration of lysozyme during the production MD simulation.

1. Understand the content of a GROMACS system topology
2. Carry out basic steps of setting up a periodic system, including solvation and adding ions
3. Identify keywords and typical settings in `.mdp` input files
4. Control position restraints and topology directives via `.mdp` settings
5. Determine the appropriate algorithms for energy-minimizing and equilibrating a biomolecular system and performing a production MD simulation

The first step in the process was the generation of a topology for the protein, to which solvent and ions were added to describe the entire system. The topology contains a listing of the atomic properties, connectivity, and all relevant bonded interactions in the system, as well as topology directives such as `#include` to provide additional parameters or define species like water and ions, and `#ifdef...#endif` blocks that allow for selective application of parameters or restraints. Following processing of the protein by `pdb2gmx`, water was added to a cubic volume with `solvate` and ions added by `genion`. The system was energy-minimized with an efficient algorithm, steepest descents, and equilibrated under NVT and NPT ensembles. Recall the practical considerations for thermostat and barostat settings, including the use of a thermostat that properly samples an NVT distribution of kinetic energies. The tutorial concluded with rudimentary analysis that illustrates the basic syntax of GROMACS analysis tools.

3.3 Tutorial 2: $KALP_{15}$ in DPPC

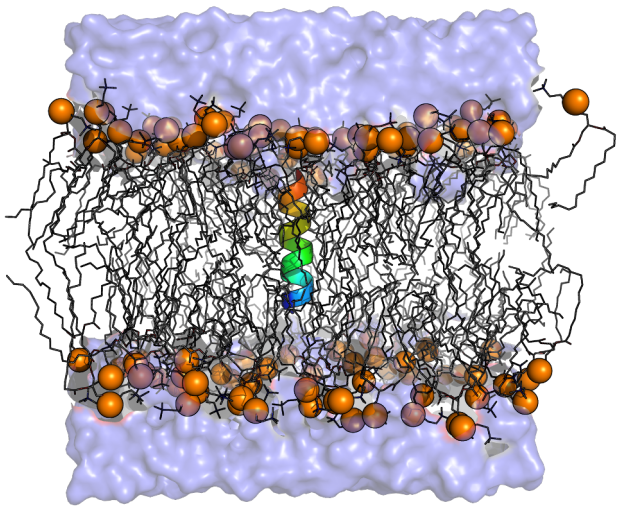

Figure 11. The $KALP_{15}$ model peptide, embedded in a hydrated DPPC bilayer.

Membrane proteins are of substantial biological interest as they modulate signaling pathways, engage in cell-cell interactions, and frequently serve as drug targets. Given their hydrophobicity, membrane proteins are difficult to crystallize or otherwise study experimentally. Thus, MD simulations serve as a useful technique in describing membrane protein conformational ensembles and protein-lipid interactions. Preparing a membrane protein system (Figure!11) is somewhat more difficult than a simple protein in water, as there are additional steps required to embed the protein in a membrane and carefully equilibrate these heterogeneous systems. This tutorial is available at http://www.mdtutorials.com/gmx/membrane_protein.

3.3.1 Prepare the Protein Topology

Model peptides are useful for studying protein-lipid interactions. The peptide that is the subject of this tutorial is called "$KALP_{15}$" and has a sequence of Ac-GKK$(LA)_4$LKKA-NH_2. The "Ac" and "NH_2" correspond to acetyl and amide groups, respectively. This peptide was the subject of an investigation on hydrophobic mismatch performed by Kandasamy and Larson [47]. That study motivated the present tutorial.

The input coordinate file for the $KALP_{15}$ peptide can be constructed with programs that can build arbitrary molecules. In this tutorial, the peptide was initially built in the xLeap program of AmberTools (http://ambermd.org/) but could also be built with CHARMM [48] if the user prefers. The initial geometry was set to that of an ideal α-helix (ϕ = -60°, ψ = -45°). This coordinate file was aligned along the *z*-axis (which, as will be shown later, corresponds to the membrane normal) by using the GROMACS `editconf` program, which has an option called

https://doi.org/10.33011/livecoms.1.1.5068
Living J. Comp. Mol. Sci. 2019, 1(1), 5068

`-princ` that aligns the longest axis of the provided coordinates along the *x*-axis. A subsequent rotation of 90°about the *y*-axis (also with `editconf`) aligns the peptide with the *z*-axis. A properly aligned $KALP_{15}$ peptide coordinate file is provided in the online tutorial.

The topology for this system will be generated with the GROMOS96 53A6 force field [24]. When executing `pdb2gmx`, the user should choose "None" for both the N- and C-termini. As described above, the N- and C-termini are capped with acetyl and amide groups, respectively. Without these groups, the peptide would have a large dipole moment as a result of the free amine ($-NH_3^+$) and carboxylate ($-COO^-$) groups that would give rise to artificial dynamics. Model peptides are often capped to avoid these spurious "end effects."

Invoke `pdb2gmx` to produce the topology:

```
$ gmx pdb2gmx -f KALP-15_princ.pdb -o
    KALP-15_processed.gro -ignh -ter -water
    spc
```

Note the use of the `-ignh` option above. The initial coordinates are an all-atom model. GROMOS96 is a united-atom force field, meaning that nonpolar hydrogen atoms are merged into their parent carbon atoms. This convention means there are fewer atoms in the system and thus the simulation will proceed more quickly. Using `-ignh` allows `pdb2gmx` to ignore all hydrogen atoms and rebuild only those required by the force field.

It is important to note here that united-atom force fields are not very commonly used. In the past, when computing power was much lower, it was desirable to limit the number of atoms in the system. All modern force fields had their origins in united-atom parameter sets [49–51]. In the present tutorial, a united-atom force field will be used as it is convenient for demonstrating the principles necessary to construct the membrane protein system and manipulate the underlying force field files. With phospholipids, the reduction in the number of atoms in the system is also quite dramatic, leading to considerable increase in simulation speed.

3.3.2 Customize the Force Field

The topology generated in Section 3.3.1 describes only the protein, and the GROMOS96 53A6 parameter set (as distributed with GROMACS) does not contain topologies or parameters for phospholipids. Recently, such parameters have been published [52]. In this tutorial, we will apply an older, but widely used, united-atom lipid force field derived by Berger, Edholm, and Jähnig [53].

The phospholipid used in this tutorial is dipalmitoylphosphatidylcholine (DPPC, Figure 12). It is a fully saturated, zwitterionic membrane lipid that is often used in simple models of membranes. The topology and force field parameters for

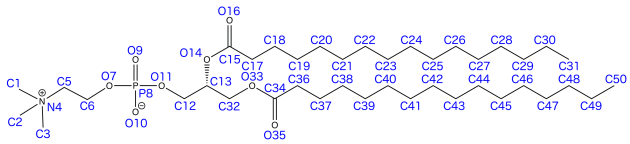

Figure 12. The chemical structure of the DPPC lipid, with atom names according to the Berger lipid force field. The *sn-1* chain begins at carbon C34 and the *sn-2* chain begins at carbon C15.

DPPC using the Berger parameters are distributed by Prof. D. Peter Tieleman at the University of Calgary. Download the following files from his website (http://wcm.ucalgary.ca/tieleman/downloads):

1. `dppc128.pdb`: Coordinates of a 128-lipid DPPC bilayer
2. `dppc.itp`: The topology of DPPC, containing its `[moleculetype]` definition
3. `lipid.itp`: The Berger lipid force field parameters

The `lipid.itp` file is a self-contained, full force field for the Berger phospholipids. It is distributed in a standalone form, such that it can be used directly for simulating pure lipid membranes. To utilize it in the context of a membrane protein system, it must be incorporated into the existing force field that describes the protein. Rather than using it directly, the user must add the parameters from `lipid.itp` into the appropriate GROMOS96 53A6 force field files. To do so, make a local copy of the force field directory into the working directory (assuming GROMACS is installed in the standard location of `/usr/local/gromacs`):

```
$ cp -r
    /usr/local/gromacs/share/gromacs/top/gromos53a6.ff/
    ./gromos53a6_lipid.ff
```

A full explanation of the force field files in this new directory is given in the online version of the tutorial. The files that will be modified as part of this tutorial are:

1. `ffbonded.itp`: The bonded force field parameters (bonds, angles, dihedrals, and improper dihedrals)
2. `ffnonbonded.itp`: The nonbonded force field parameters (atom types, Lennard-Jones parameters, pair interactions)

The modifications to the force field files must be made carefully to have a functional force field. First, copy the contents of the `[atomtypes]`, `[nonbond_params]`, and `[pairtypes]` directives in `lipid.itp` into the corresponding sections of `ffnonbonded.itp`. The `[atomtypes]` section of `lipid.itp` lack atomic numbers and must be added in. In the `[nonbond_params]` section (which defines pair-specific Lennard-Jones interactions, *e.g.* those that do not obey normal combination rules), delete the line that says `;;`

https://doi.org/10.33011/livecoms.1.1.5068
Living J. Comp. Mol. Sci. 2019, 1(1), 5068

`parameters for lipid-GROMOS interactions`. Delete this line and all the lines that follow in the `[nonbond_params]` section. These entries correspond to GROMOS87 atom types and interactions that are not necessary or appropriate when using the GROMOS96 53A6 force field. Similarly, delete or comment out (with ;) any line in `[nonbond_params]` that includes "HW," which is also not a valid atom type in GROMOS96. It indicates water hydrogen atoms, but these are simply called "H" in GROMOS96. Thus, one can also simply replace "HW" with "H" to avoid future errors. Failure to do these steps properly will lead to fatal errors when using `grompp`.

After modifying `ffnonbonded.itp`, add the contents of the `lipid.itp [dihedraltypes]` section to the equivalent section in `ffbonded.itp`.

These modifications of the force field files are fairly significant, and it is important to carefully check to make sure all steps have been completed correctly.

FORCE FIELD MODIFICATIONS

- ☐ Copy the contents of `[atomtypes]` from `lipid.itp` to `ffnonbonded.itp` and add atomic numbers to each
- ☐ Copy the contents of `[nonbond_params]` from `lipid.itp` to `ffnonbonded.itp`
- ☐ Remove the `;; parameters for lipid-GROMOS interactions` line and all subsequent lines in `[nonbond_params]` from `ffnonbonded.itp`
- ☐ Remove or comment out any line in `[nonbond_params]` in `lipid.itp` containing "HW," or rename to "H"
- ☐ Copy the contents of `[pairtypes]` from `lipid.itp` to `ffnonbonded.itp`
- ☐ Copy the contents of `[dihedraltypes]` from `lipid.itp` to `ffbonded.itp`

Next, to make use of the modified force field, an adjustment to the system topology (`topol.top`) must be made. Change the call to the force field from:

```
#include "gromos53a6.ff/forcefield.itp"
```

to:

```
#include "gromos53a6_lipid.ff/forcefield.itp"
```

Finally, add an `#include` statement to add the DPPC topology to the system topology, in the exact location shown (to avoid disrupting any other `[moleculetype]` in the topology:

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

; Include DPPC chain topology
#include "dppc.itp"

; Include water topology
#include "gromos53a6_lipid.ff/spc.itp"
```

At this point, some additional discussion on force field choice is warranted. The use of GROMOS96 53A6 to treat the protein and the Berger united-atom lipid parameters for DPPC is motivated by several factors. First, the constructed system ($KALP_{15}$ in DPPC) is a reproduction of a published and well-studied system [47]. Second, the Berger lipid parameters are compatible with the GROMOS protein parameter set. Finally, the modification of the force field files is a useful instructive tool in teaching new users how this information is organized in GROMACS, allowing for future modifications in advanced systems. However, any MD simulation must be carefully planned, which includes a critical assessment of force field parameters for all species in the system. One should not automatically choose this force field combination simply because this tutorial does. In fact, the GROMOS96 53A6 force field has been shown to under-stabilize α-helices [25], and the Berger parameters are not necessarily the best representation of lipid properties, particularly for DPPC. A useful reference is a systematic study by Piggot et al. [54], though the lipid force field literature is rich with other comparisons. Users must carefully consider available studies that critically assess force field performance for any system, but phospholipid membranes are especially challenging, and while no force field perfectly represents all relevant parameters, some force fields are demonstrably better than others.

3.3.3 Construct the System

The force field and system topology files are now fully prepared to describe the contents of the membrane-protein system being constructed. As in Tutorial 1 (Section 3.2), the next step after preparing the initial topology is solvation. In nearly all MD simulations of biomolecules, the solute of interest is embedded within some type of liquid solvent, which in this case will be comprised of both a phospholipid bilayer and water. In contrast to the simple addition of a homogeneous solvent using `solvate`, the addition of the lipids around the protein requires auxiliary methods. In this tutorial, the DPPC lipids will be packed around the $KALP_{15}$ peptide using the InflateGRO method [55]. With this approach, the lipid coordinates are scaled in the *x-y* plane, any lipids that remain overlapping with the protein are deleted, and then the lipids are progressively packed around the protein with intervening rounds of energy minimization.

The DPPC coordinate file distributed by Prof. Tieleman has "broken" lipids; all atoms are "inside" the central im-

https://doi.org/10.33011/livecoms.1.1.5068
Living J. Comp. Mol. Sci. 2019, 1(1), 5068

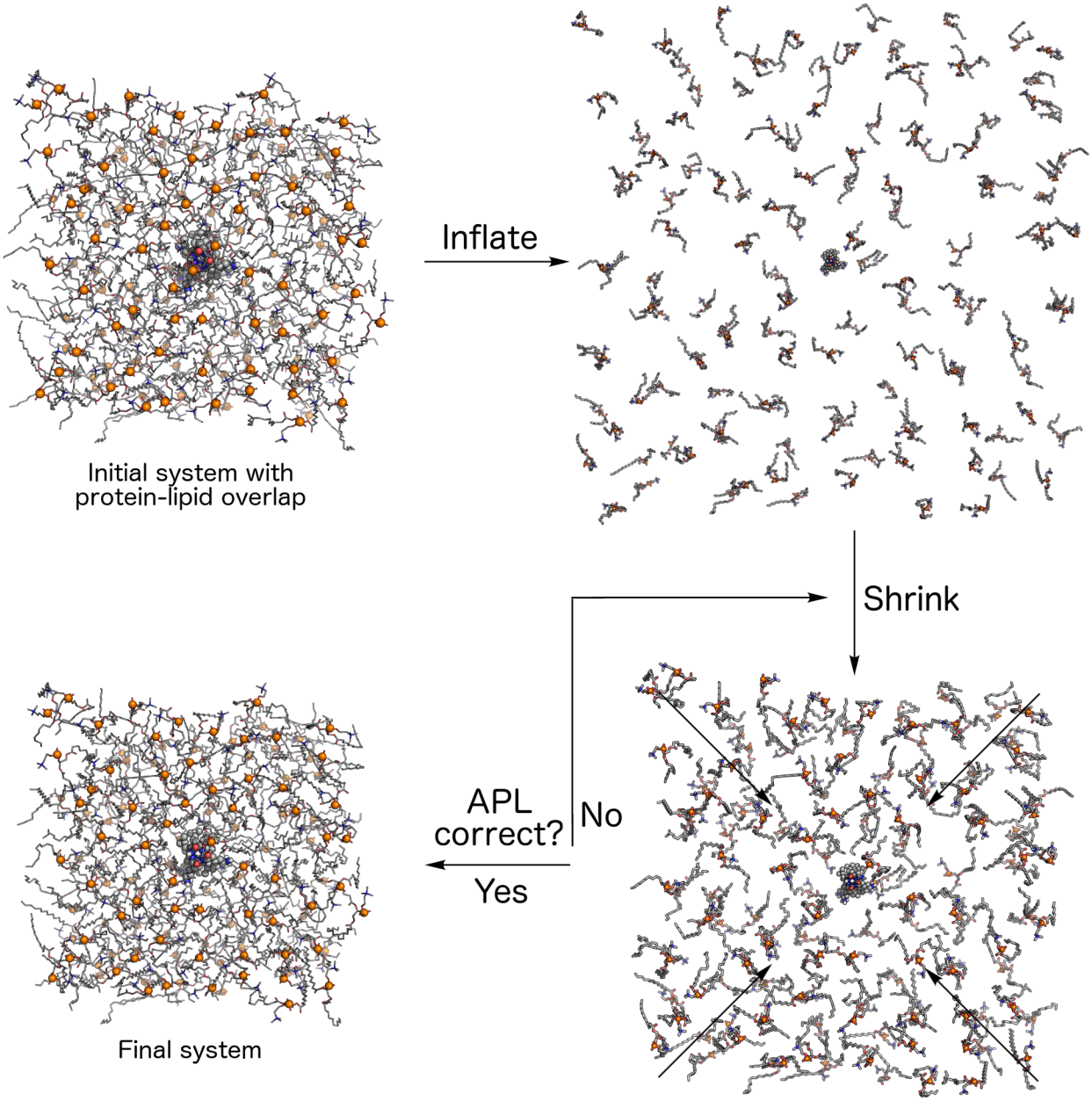

Figure 13. The process of packing DPPC lipids around the $KALP_{15}$ peptide. The initial lipid coordinates, with protein-lipid atomic overlap, are scaled outward by some inflation factor (4 in this tutorial) and any lipids that remain overlapping with the protein are deleted. The scaling vector is reduced to a value < 1 to shrink the system (inward movement denoted by arrows pointing in from corners). The shrinking process is carried out iteratively until a reasonable APL value is achieved, yielding the final, packed system that is prepared for solvation with water.

age and have the appearance that the lipid molecules are not bonded across periodic boundaries. While this representation is perfectly valid for carrying out MD simulations, InflateGRO will not work unless the molecules are "intact," that is, the distance between bonded atoms is at a minimum with respect to the periodic unit cell. To produce a correct coordinate file for InflateGRO, build a `.tpr` file that corresponds to the DPPC-water system. A topology for this system (`topol_dppc.top`) is provided as part of the online tutorial. Any syntactically valid `.mdp` file can be used for this purpose. Note that `topol_dppc.top` is only used in this step and never again. Any updates or reference to the system topology described in the tutorial are in `topol.top`.

Create the `.tpr` file:

https://doi.org/10.33011/livecoms.1.1.5068
Living J. Comp. Mol. Sci. 2019, 1(1), 5068

```
$ gmx grompp -f minim.mdp -c dppc128.pdb -p
    topol_dppc.top -o dppc.tpr
```

Invoke `trjconv`:

```
$ gmx trjconv -s dppc.tpr -f dppc128.pdb -o
    dppc128_whole.gro -pbc mol -ur compact
```

Next, the $KALP_{15}$ peptide must be centered within the same box as the DPPC bilayer, such that the membrane center and peptide center are coincident. Open `dppc128_whole.gro` in a text editor and navigate to the last line of the file (or, on the command line, use `tail -n 1 dppc128_whole.gro`). The last line of a `.gro` file contains the box vectors of the system. Center the $KALP_{15}$ peptide in a box with the same dimensions as that of the DPPC bilayer with `editconf`:

```
$ gmx editconf -f KALP-15_processed.gro -o
    KALP_newbox.gro -c -box 6.41840 6.44350
    6.59650
```

The next step in assembling the system is to prepare the input coordinates for use in InflateGRO. To do so, concatenate the protein and membrane coordinates:

```
$ cat KALP_newbox.gro dppc128_whole.gro >
    system.gro
```

The `system.gro` file is not formatted correctly. A valid `.gro` file has the following format:

```
Title
  Number of atoms
  (all lines containing atomic coordinates)
Box vectors
```

By concatenating two `.gro` files together, there are now unnecessary lines that need to be removed. Open `system.gro` in a text editor and search down for "DPPC." Above the first occurrence of lipid coordinates are three lines that need to be removed: (1) the box vectors from `KALP_newbox.gro`, (2) the title line from `dppc128_whole.gro`, and (3) the number of atoms from `dppc128_whole.gro`. Prior to removing the line containing the number of atoms, make note of it. After removing it, add this number to the second line in the file, to specify the total number of atoms in the system. Save the file and exit the text editor. Note that one can easily validate the number of atoms in the file with the Linux `wc` command:

```
$ wc -l system.gro
```

Subtract three from the number reported by `wc` (to reflect the title line, number of atoms line, and box vectors). This number should match what appears on the second line of the `.gro` file. If it does not, inspect the file to determine the source of discrepancy, and if it does not match, start over with the `cat` command above. A quick test for validity of `system.gro` is to open it in some visualization software like VMD. If the file does not open or the program reports an error, it has been constructed incorrectly.

The InflateGRO method requires strong position restraints to be placed on the protein being embedded in the lipid membrane, beyond the default value of 1000 kJ mol^{-1} nm^{-2} written in `posre.itp` from `pdb2gmx` [55]. To create a new file, with stronger restraints (100000 kJ mol^{-1} nm^{-2}), invoke the `genrestr` program, choosing "Protein" when prompted:

```
$ gmx genrestr -f KALP_newbox.gro -o
    strong_posre.itp -fc 100000 100000
    100000
```

Add an `#ifdef` statement to call the new `strong_posre.itp` file in `topol.top`:

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

; Strong position restraints for InflateGRO
#ifdef STRONG_POSRES
#include "strong_posre.itp"
#endif

; Include DPPC chain topology
#include "dppc.itp"

; Include water topology
#include "gromos53a6_lipid.ff/spc.itp"
```

The coordinates and topology are now ready to be processed with the InflateGRO Perl script:

```
$ perl inflategro.pl system.gro 4 DPPC 14
    system_inflated.gro 5 area.dat
```

The order of command-line arguments passed to InflateGRO is specific and must adhere to this exact order. The syntax is as follows:

1. `system.gro`: the input coordinate file to which scaling is applied
2. `4`: the scaling factor applied. A value > 1 indicates inflation (expansion) and a value < 1 indicates shrinking (compression)
3. `DPPC`: the residue name of the lipids to which scaling is applied
4. `14`: the cutoff radius (in Å) for searching for lipids to delete
5. `system_inflated.gro`: the output coordinate file name
6. `5`: the grid spacing for calculation of area per lipid

https://doi.org/10.33011/livecoms.1.1.5068
Living J. Comp. Mol. Sci. 2019, 1(1), 5068

7. `area.dat`: the name of the text file to which area per lipid values are printed

InflateGRO will print to the screen how many lipids are deleted. Note this value and update the `[molecules]` section of `topol.top` accordingly. It is also important to note that InflateGRO removes all water molecules. The user must not add back any water molecules until the lipids are completely packed around the protein. Therefore, `[molecules]` should refer only to the protein and the DPPC lipids. Do not invoke `solvate` until instructed to later in this tutorial.

The system now requires energy minimization. The `minim_inflategro.mdp` file (provided online) includes a line that reads:

```
define = -DSTRONG_POSRES
```

With this command, the strong position restraints will be applied to the protein heavy atoms during all energy minimizations carried out while packing the lipids around the protein with InflateGRO.

Perform energy minimization, being sure to use explicit file names. As there will be many rounds of packing and energy minimization, relying on default GROMACS file names is not recommended. Minimize the inflated system using descriptive file names:

```
$ gmx grompp -f minim_inflategro.mdp -c
    system_inflated.gro -p topol.top -r
    system_inflated.gro -o
    system_inflated_em.tpr

$ gmx mdrun -deffnm system_inflated_em
```

As before, the user must reconstruct the "broken" lipids before attempting to use the coordinate file further:

```
$ gmx trjconv -s system_inflated_em.tpr -f
    system_inflated_em.gro -o tmp.gro -pbc
    mol

$ mv tmp.gro system_inflated_em.gro
```

The use of `tmp.gro` is necessary as `trjconv` returns an error when trying to read from, and write to, the same file name when correcting for PBC effects.

After the first energy minimization, the lipids need to be packed around the protein. To do so, use InflateGRO to apply a scaling factor that is less than 1. Doing so moves the lipids towards the protein. It is wise to do this packing very slowly to prevent intermolecular clashes that will lead to an unstable system. A typical scaling factor during packing is 0.95. To perform the first shrinking/packing step, invoke InflateGRO as follows:

```
$ perl inflategro.pl system_inflated_em.gro
    0.95 DPPC 0 system_shrink1.gro 5
    area_shrink1.dat
```

Note that the cutoff radius has also been reduced to 0 Å. Such a cutoff value disables the deletion of lipids. It is only appropriate to delete lipids during inflation; if the cutoff value is retained, eventually all of the lipids (or a great many of them) will be deleted and the system will be useless. After 26 rounds of packing (shrinking the lipids inward, energy minimization, and reconstruction of "broken" lipids), an area per lipid of roughly 71 $Å^2$ should be attained. Since InflateGRO overestimates area per lipid slightly, this value is an indication that additional shrinking is not necessary. That is, it is not necessary to achieve the exact experimental value of area per lipid (62 $Å^2$). The online tutorial provides a Bash script that automates this process, but users wishing to learn via repetition can challenge themselves to carry out this process manually and become very comfortable with proper file naming. In summary, the process of packing lipids around the $KALP_{15}$ peptide is illustrated in Figure 13.

It is now time to add water to the system using `solvate`. One issue that will likely arise is that water molecules will be added within small void volumes in the hydrophobic core of the membrane. There are several strategies one can apply to deal with this situation, but the easiest is to allow `solvate` to add however many water molecules it can and subsequently delete those that fall within the membrane core. Solvate as usual with:

```
$ gmx solvate -cp system_shrink26_em.gro
    -cs spc216.gro -p topol.top -o
    system_solv.gro
```

The online tutorial provides `water_deletor.pl`, which is a simple Perl script that finds water molecules within a user-defined range of *z*-coordinate values and deletes them. The user defines a "reference" atom, which should be an atom in the ester region of the phospholipid and a "middle" atom defining the center of the membrane along its normal (typically the *z*-axis). See Figure 12 for atom naming in DPPC. From these atoms, the bilayer is divided into upper and lower leaflets, such that the reference atoms define the boundaries along the *z*-axis within which water molecules will be deleted (Figure 14). Finally, the script needs to know how many atoms constitute a water molecule. For SPC, there are 3 atoms (OW, HW1, and HW2), so this value is passed to the `-nwater` flag. Invoke the script as follows:

```
$ perl water_deletor.pl -in system_solv.gro
    -out system_solv_fix.gro -ref O33
    -middle C50 -nwater 3
```


https://doi.org/10.33011/livecoms.1.1.5068
Living J. Comp. Mol. Sci. 2019, 1(1), 5068

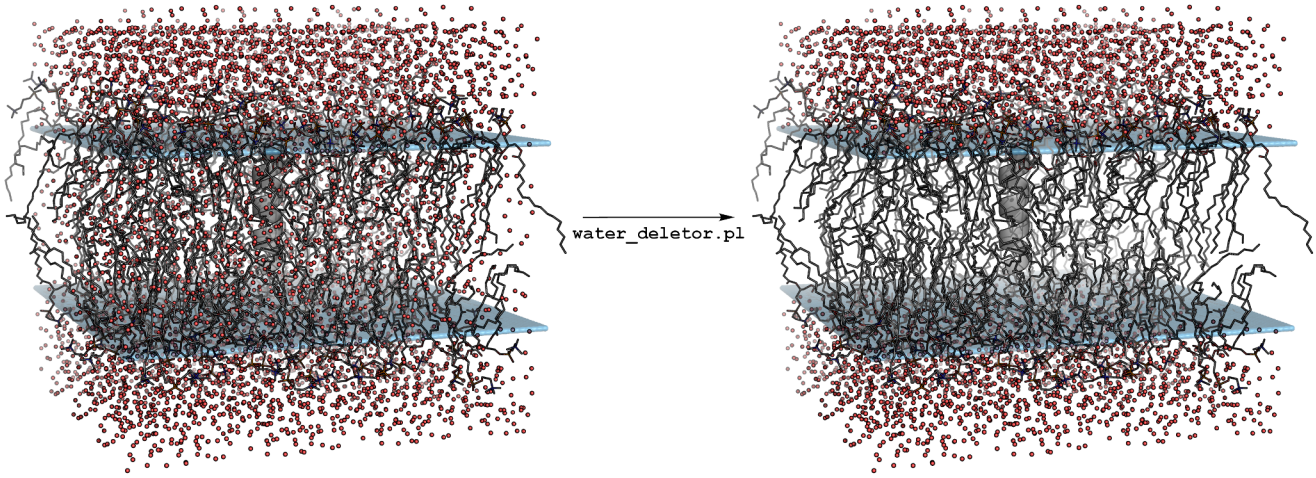

Figure 14. The action of the `water_deletor.pl` script. Water molecules with *z*-coordinates that fall between the O33 atoms (which define the translucent blue planes) are deleted, leaving a hydrated bilayer with no water molecules within the hydrophobic core.

3.3.4 Add Ions

Adding ions to the solvated $KALP_{15}$-DPPC system is the same as any other system. Invoke `grompp` and `genion`:

```
$ gmx grompp -f ions.mdp -c
    system_solv_fix.gro -p topol.top -o
    ions.tpr

$ gmx genion -s ions.tpr -o
    system_solv_ions.gro -p topol.top
    -pname NA -nname CL -neutral
```

Choose group 15 (SOL) for replacing water molecules with Cl^- ions.

3.3.5 Energy Minimization

Energy minimization is performed in the same manner as any other system:

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

$ mdrun -deffnm em
```

Energy minimization should converge rather quickly for this system, as the lipids were progressively relaxed around the protein prior to the addition of water:

```
Steepest Descents converged to Fmax < 1000 in 292 steps
Potential Energy  = -3.2078762e+05
Maximum force     =  9.4299438e+02 on atom 130
Norm of force     =  5.1295775e+01
```

3.3.6 Equilibration

Equilibration of the $KALP_{15}$-DPPC system will be carried out in two phases, as described in Section 3.2.5, an initial NVT equilibration followed by NPT equilibration. There are several important considerations specific to membrane simulations that are relevant at this stage. First, for the purposes of thermostatting and COM motion removal, the user will need to merge the atoms of the protein and DPPC groups using `make_ndx`, a program that generates custom groups of atoms:

```
$ gmx make_ndx -f em.gro -o index.ndx
```

Merge the protein (group 1) and DPPC (group 13) index groups with: `1 | 13`, type Enter, then type `q` and Enter to quit `make_ndx`.

In equilibrating the lysozyme system (Section 3.2.5), temperature coupling groups were set to:

```
tc-grps      = Protein Non-Protein
```

If the same syntax were used in the case of a membrane-protein system, the "Non-Protein" group would also contain the phospholipids. Doing so is undesirable due to the heterogeneity of the system. Lipids and water diffuse on different time scales, thus if they are treated in the same group for the purposes of thermostatting, there will be spurious contributions to the velocity scaling. As a consequence, it is common to treat the constituents of the membrane (all lipids and the embedded protein, whose diffusion depends strongly on the lipids) as a single group with respect to the thermostat. Merging the protein and DPPC lipids (above) allows the user to apply this convention. As such, the thermostat settings in `nvt.mdp` are:

```
tcoupl       = V-rescale
tc-grps      = Protein_DPPC   Water_and_ions
tau_t        = 0.1            0.1
ref_t        = 323            323
```


https://doi.org/10.33011/livecoms.1.1.5068
Living J. Comp. Mol. Sci. 2019, 1(1), 5068

The "Water_and_ions" group is a default group available in GROMACS, encompassing all SOL molecules and (in this case) Cl^- ions. Note, too, that the reference temperature is set to 323 K, above the experimental phase transition temperature of DPPC (315 K). For biological applications, it is generally desirable to perform membrane simulations above the phase transition temperature of the lipid membrane, such that it is in a liquid crystalline state. The reference temperature value should be determined not only in terms of the experimental value, but also in terms of the chosen force field. There may be some degree of error in how well a given force field models the phase transition of a given lipid, and the user must take this into account. A list of reference phase transition temperatures and associated literature references is provided in the online tutorial.

Similarly, the net COM motion removal in membrane systems (and any interfacial system, in general) must be given special consideration. Due to the differences in diffusion rates, one should set separate COM motion removal groups in the `.mdp` file. As such, in the `nvt.mdp` file, set:

```
nstcomm      = 1
comm-mode    = Linear
comm-grps    = Protein_DPPC Water_and_ions
```

Perform NVT equilibration by calling `grompp` and `mdrun`:

```
$ gmx grompp -f nvt.mdp -c em.gro -r em.gro
    -p topol.top -n index.ndx -o nvt.tpr

$ gmx mdrun -deffnm nvt
```

Following NVT, confirm that the temperature of the system stabilized at 323 K before continuing forward with NPT equilibration. It is also not uncommon to see some separation between the leaflets of the bilayer at the end of NVT. This void space will compress down and close during NPT equilibration.

In NPT equilibration, a new concept is introduced, semi-isotropic pressure coupling. In this method, the coordinate scaling factors are uniform in the *x*-*y* plane but independent along the membrane normal, coincident with the *z*-axis. This approach reflects the intrinsic anisotropy in interfacial systems such as membranes, in which the lateral forces (among lipids) differ in magnitude from forces of water acting along the *z*-axis. The relevant pressure coupling settings in `npt.mdp` are:

```
pcoupl          = Parrinello-Rahman
pcoupltype      = semiisotropic
tau_p           = 5.0
ref_p           = 1.0    1.0
compressibility = 4.5e-5     4.5e-5
```

The Parrinello-Rahman barostat was introduced previously, but is now employed in semiisotropic mode. There is only a single value of τ_p, despite the ability to assign different reference pressures (`ref_p` along *x*-*y* and *z*, respectively). The `compressibility` setting takes two values here, again along the membrane plane and normal, but the default value of the isothermal compressibility of water (4.5×10^{-5} bar^{-1}) is sufficient for this purpose. The compressibility affects the responsiveness of the barostat, not the physical properties of the medium. In principal, if one knows the lateral compressibility of the lipid, it could be substituted here, but given the dramatic range of instantaneous pressure values observed over the course of an MD simulation, it is unlikely that fine tuning this value will have any meaningful physical consequence and will not be done here.

Continue on with NPT equilibration:

```
$ gmx grompp -f npt.mdp -c nvt.gro -r
    nvt.gro -p topol.top -t nvt.cpt -n
    index.ndx -o npt.tpr

$ gmx mdrun -deffnm npt
```

After equilibration, calculate the time-average pressure value and the time series of the *x*- and *y*- box vectors (Box-X and Box-Y, respectively, in the `energy` list of terms). The box vectors are an indication of the convergence of the lateral area of the system. While only a 1-ns simulation is unlikely to yield fully converged values (which can take from 50-100 ns to fully stabilize), a simple assessment here may be useful for obtaining an initial indication of the stability of the system.

3.3.7 Production MD

Carry out a 1-ns production simulation, using the same NPT ensemble as during equilibration, but in the absence of position restraints on the protein:

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

$ gmx mdrun -deffnm md_0_1
```

It is important to note that 1 ns of time is orders of magnitude lower than what is normally required to obtain converged sampling in a membrane system. Typical simulations of these types of systems are in excess of 100 ns. The data obtained in this tutorial are for demonstration purposes only.

3.3.8 Analysis

There are many structural and dynamical properties of lipids that can be computed during an MD simulation. In the context of membrane-protein systems, many of these properties are interrelated with the dynamics of the protein, whether it is embedded (transmembrane) or peripherally associated. Lipid

https://doi.org/10.33011/livecoms.1.1.5068
Living J. Comp. Mol. Sci. 2019, 1(1), 5068

analysis requires great detail and often requires the use of custom index groups, which are user-generated selections of atoms on which the analysis is performed. In this section, several types of lipid-specific analysis will be introduced, along with proper construction of complex index groups.

Deuterium Order Parameters

The deuterium order parameter is a measure of the orientation of a C-D bond, thus reporting on the structure of hydrocarbons in layered systems. This quantity is defined as:

$$-S_{CD} = \frac{\left\langle 3cos^2\theta - 1 \right\rangle}{2} \quad (1)$$

in which θ is the angle between the C-D bond and the membrane normal (again, the *z*-axis) and the angle brackets denote a time average. The GROMACS program that computes $-S_{CD}$ is called `order`. It requires an index file containing groups with all equivalent carbon atoms of the acyl chain in separate groups. To create such groups, refer to Figure 12, in which the *sn-1* chain (referring to the "stereospecific number," *e.g.* the numbering in the glyercol backbone) includes carbon atoms C34 and C36-C50, inclusive, and the *sn-2* chain includes carbon atoms C15 and C17-C31, inclusive. To create these index groups, invoke `make_ndx`:

```
$ gmx make_ndx -f md_0_1.tpr -o sn1.ndx
> a C34
> a C36
> a C37
...
> a C50
> del 0-21
> q
```

The above commands passed to `make_ndx` create individual groups with all C34, C36, C37, ... C50 atoms. The `del 0-21` command removes all the standard GROMACS groups, leaving only the lipid carbon atoms to be analyzed. The process should be repeated to create `sn2.ndx`, selecting the atoms indicated above.

To plot the deuterium order parameters for the *sn-1* chain, invoke `order`:

```
$ gmx order -s md_0_1.tpr -f md_0_1.xtc -n
    sn1.ndx -d z -od deuter_sn1.xvg
```

It is important to note that the value of $-S_{CD}$ cannot be calculated for terminal carbon atoms, that is the carbonyl carbon (C34 in the *sn-1* chain) and the terminal methyl carbon (C50 in the *sn-1* chain). This limitation arises from the calculation of the local molecular axis. To compute this axis, the previous and next carbon atoms in the acyl chain are needed. For terminal carbon atoms, this axis is undefined. The output file (in this case, `deuter_sn1.xvg`) from `order` is numbered from 1, as the program does not necessarily know which segment of the chain is being analyzed. For plotting purposes, edit the atom numbers in `deuter_sn1.xvg` by incrementing each by 1. The resulting plot (after similarly analyzing the *sn-2* chain) will look something like what is shown in Figure 15.

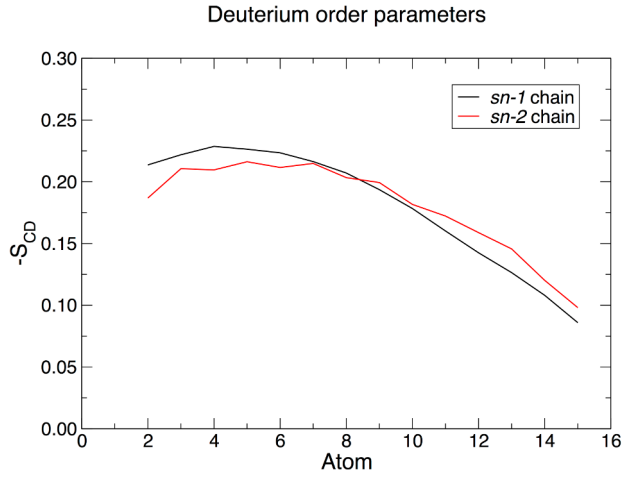

Figure 15. Deuterium order parameters, $-S_{CD}$, of the palmitoyl chains in DPPC.

The resulting values of $-S_{CD}$ are indicative of DPPC lipids in a liquid crystalline form, as the plateau region (carbons 4-8) has a value of ~0.2. This outcome is expected since (1) the starting DPPC coordinates were pre-equilibrated in a liquid crystalline state and the simulation carried out here is only for 1 ns, thus retaining this state and (2) the temperature during the simulation (323 K) was above the phase transition temperature of DPPC (315 K).

Membrane Density Profile

The structure of phospholipid membranes can be described in part by computing a density profile along the membrane normal. The resulting plots show the relative positioning of the functional groups of the lipids. In this context, it is useful to subdivide the lipids into moieties with different properties or local structure, such as the headgroups (in this case, the zwitterionic phosphatidylcholine group), the glycerol or glycerol ester atoms, and the acyl chains. To create these groups, again refer to Figure 11 for atom nomenclature and invoke `make_ndx`:

```
$ gmx make_ndx -f md_0_1.tpr -o
    density_groups.ndx
> 13 & a C1 | a C2 | a C3 | ... |  a O11
> name 22 Headgroups
> 13 & a C12 | a C13 | a O14 | a C15 |
  a O16 | a C32 | a O33 | a C34 | a O35
> name 23 Glycerol_Ester
> 13 & ! 22 & ! 23
```


https://doi.org/10.33011/livecoms.1.1.5068
Living J. Comp. Mol. Sci. 2019, 1(1), 5068

```
> name 24 Acyl_Chains
> q
```

The selections created here are somewhat more complicated than any previously encountered in these tutorials. Group 13 is a default group containing all DPPC atoms. The use of ampersand (&) denotes the intersection of selections, *i.e.* atoms belonging to both selections. The atoms are subsequently selected by their atom names ("a") and merged together using the pipe character (|) meaning "or." For the first group (Headgroups), the selection would literally read "atoms in DPPC that are also named either C1, C2, C3, ... or O11." The Glycerol_Ester group is created similarly. The remaining atoms (those not in headgroups or the glycerol ester region) belong to the lipid acyl chains. Rather than typing out all of those atom names individually, the user can again employ logical operators. In creating the Acyl_Chains group, use the logical "not" operator (!). The last selection translates to "atoms in DPPC that are not in the Headgroups group and also not in the Glycerol_Ester group."

Use the `density` program to compute the partial densities of each of these groups:

```
$ gmx density -s md_0_1.tpr -f md_0_1.xtc
    -n density_groups.ndx -o
    dens_headgroups.xvg -d Z
```

Select group number 22, then repeat the process for the glycerol ester, acyl chain, and water (SOL) groups. Plotting the files in XmGrace yields Figure 16.

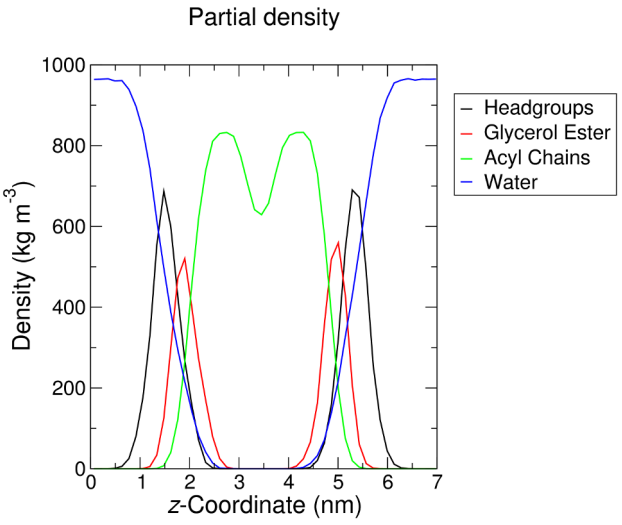

Figure 16. Partial density profile of water and lipid functional groups along the *z*-axis.

Area Per Lipid and Bilayer Thickness

The lateral area per lipid (APL) is a way of assessing lipid packing, again reflecting gel or liquid-crystalline states. APL is related to the deuterium order parameters discussed above; large values of $-S_{CD}$ reflect acyl chain elongation, close packing, and thus smaller APL. Fluid membranes have larger APL than their gel-phase counterparts. GROMACS does not have a built-in program that computes APL. For a pure membrane (no protein), APL can be calculated simply from the *x*- and *y*-box vectors, *i.e.* APL = (Box-X $\times$ Box-Y) / (number of lipids per leaflet). When a protein is present, it occupies some amount of the lateral area in one or both leaflets. Several methods are available for computing APL in such a case. GridMAT-MD [56] is a program designed to interface directly with GROMACS, processing both `.gro` and `.pdb` coordinate files (including multi-frame files). It can also compute membrane thickness as a projection over the entire *x*-*y* plane, another feature that is absent in GROMACS.

Lateral Diffusion of Lipids

Calculating the lateral diffusion of lipids (that is, within the *x*-*y* plane) is often of interest in membrane systems, and as such, GROMACS provides a program, called `msd`, that computes the self-diffusion coefficient, D_0, of particles based on their mean-square displacement (MSD) using the Einstein relation [57], *e.g.* for all particles *A* running over index *i*:

$$\lim_{t\to\infty} \langle \|\mathbf{r}_i(t) - \mathbf{r}_i(0)\|^2 \rangle_{i\in A} = 6D_A t \qquad (2)$$

The MSD can be computed by using a single atom within each molecule, or based on the COM of each molecule. Here, the MSD and resulting diffusion coefficient will be computed from a single, representative atom, the phosphorus (P8) in each DPPC lipid. First, construct an index group:

```
$ gmx make_ndx -f md_0_1.tpr -o p8.ndx
...
> a P8
> q
```

Invoke `msd`:

```
$ gmx msd -s md_0_1.tpr -f md_0_1.xtc -n
    p8.ndx -lateral z
```

The `-lateral` option instructs `msd` to calculate only the diffusion within the *x*-*y* plane, that is, the *z*-axis is normal to the plane of interest. The result from this simple system is likely to vary widely and is in no way converged or reflective of equilibrium lipid diffusion. From the limited number of data points, the statistical certainty of this value is likely to be poor, and a value like $2.7 \pm 2.7 \times 10^{-7}$ cm^2 s^{-1} might be obtained. This value is neither accurate nor precise, and as such a much longer simulation would be required to obtain a more reliable value. For the purposes of this tutorial, this calculation serves as a cautionary note about the proper time scale required for performing simulations of lipid membranes. Interested users

https://doi.org/10.33011/livecoms.1.1.5068
Living J. Comp. Mol. Sci. 2019, 1(1), 5068

are encouraged to extend the simulation out to 50 or 100 ns and compare the values obtained over these time frames.

As a final note, it is important to recognize that periodic boundary conditions and the size of the box will influence the obtained value of D_0. In fact, the result obtained here is not truly D_0, it is a size-dependent quantity termed D_{PBC} per Yeh and Hummer [58], who derived a correction factor that can be applied to obtain the true value of D_0 in periodic systems.

3.3.9 Summary and Review of Objectives

Through this tutorial, the user has been guided through the construction of a membrane-protein system, which requires modification of force field files and packing of lipids around the protein prior to solvation with water. To review, the objectives for this tutorial were as follows:

1. Understand the organization and contents of GROMACS force field files and how parameters from different, but compatible, sources can be added to them
2. Apply an iterative packing routine to place phospholipids around a transmembrane protein
3. Perform lipid-specific analysis using custom index groups

By adding parameters from an external force field file (in this case, `lipid.itp`) to an existing force field, the organization and logic of GROMACS force field files has been illustrated, with some discussion of the theory of force field compatibility. The solvation of a membrane-protein system has been approached through the use of an iterative packing scheme for the DPPC phospholipids, followed by conventional solvation with water and removal of water molecules within the hydrophobic core of the membrane with a custom script. These processes have required the user to become more familiar with GROMACS topology organization, including manual modification that was not required in the first tutorial. Finally, after performing a short simulation on the $KALP_{15}$-DPPC system, several lipid-specific analyses were performed. Though the time scale of this simulation is insufficient to obtain converged data on lipid properties, the tutorial has provided the logic behind constructing complex index groups that are required for performing lipid analysis.

3.4 Tutorial 3: Umbrella Sampling

It is often of interest to compute a free energy change that is associated with some change in geometry, such as a conformational change in a protein or the binding of two molecules as they approach each other along some path. Attempting to study such processes can be challenging with normal, unbiased MD simulations, as high-energy states (transition states or rare conformations) are not frequently sampled. Therefore, it is difficult to define accurate free energy differences, or the barriers may simply never be overcome on the timescale accessible to conventional MD. To overcome this challenge, biasing potentials can be applied to enhance sampling of these rare states and thereby more completely define free energy differences.

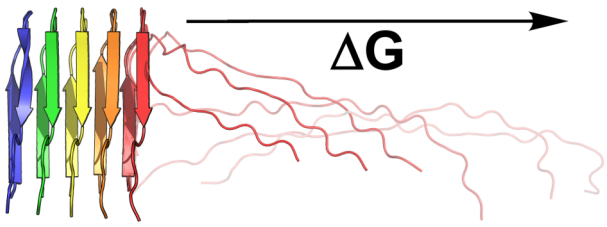

Figure 17. The terminal peptide of the Aβ_{42} protofibril dissociates along the fibril axis, the degree of freedom over which the free energy difference, ΔG, is computed.

In this tutorial (available online at http://www.mdtutorials.com/gmx/umbrella), the user will be guided through the process of calculating the free energy difference for the dissociation of a peptide from a pentameric assembly using umbrella sampling to compute the potential of mean force (PMF), or ΔG (Figure 17). The process outlined here is based on a published study that examined the impact of point mutations on the stability of Aβ_{42} protofibrils [59]. It is important to note here that these simulations are somewhat unique, as they involve extra restraints and only a single dimension along which the PMF is computed. Additional details will be provided in subsequent sections in the tutorial.

Constructing a system that will be subjected to umbrella sampling is much like any other, so the initial details of preparing the topology, defining the box, solvation, minimization, and equilibration will only highlight aspects that are unique to subsequent umbrella sampling simulations. Ultimately, to carry out these calculations, a set of configurations is generated along some path within the system (a distance, dihedral angle, etc.), called a "reaction coordinate," defining the starting points for each of a series of windows that allow sampling in discrete regions of the reaction coordinate. There are many ways to construct initial configurations in each window. In this tutorial, the reaction coordinate is defined as the *z*-axis and will employ a non-equilibrium technique called steered molecular dynamics (SMD) to generate the starting configurations for each window.

3.4.1 Prepare the Protein Topology

The system being studied here is a pentamer of the Aβ peptide, in an aggregated fibril form. The initial coordinates are taken from an NMR structure determined by Lührs et al. [60]. Each of the five protein chains is labeled (A, B, C, D, and E) in the PDB file, which is entry 2BEG. The first model

https://doi.org/10.33011/livecoms.1.1.5068
Living J. Comp. Mol. Sci. 2019, 1(1), 5068

(the lowest energy) will be used here, and the N-termini of each chain should be capped with acetyl groups to prevent spurious end effects. There are several residues in each chain that are disordered and therefore not assigned in the NMR structure. Residues 17-42 of each chain are present, so an acetyl cap should be added to Leu17 of each peptide. GROMACS has no ability to construct such terminal capping groups. As in the previous tutorial (Section 3.3.1), these groups should be constructed in AmberTools (http://ambermd.org/) or CHARMM [48]. A suitable protein PDB file is provided in the online version of the tutorial. With this file, execute `pdb2gmx`:

```
$ gmx pdb2gmx -f 2BEG_model1_capped.pdb -o
    complex.gro -ignh -ter
```

Choose the GROMOS96 53A6 parameter set as the force field, and when prompted, choose "None" for all N-termini (to indicate the fact that no modifications should be made to acetyl groups) and "COO^-" for the C-termini (as Ala42 is the true C-terminus of the Aβ_{42} peptide). The fact that there are multiple protein chains in the coordinates is not a problem for `pdb2gmx`, as it can produce a topology for each provided that the individual chains are either denoted with different chain identifiers (as is the case here) or delimited by TER cards.

Later, in Section 3.4.4, a biasing force will be applied to one of the peptides (chain A) in the pentamer. To avoid distortions to the structure, restraints will be applied to chain B to immobilize it during this process. Note that `pdb2gmx` has produced individual topologies and restraint file topologies (all with extension `.itp`) for each chain. Add the following to the end of `topol_Protein_chain_B.itp`:

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

It is important to note here that the input coordinates used in this tutorial differ from those used in our published study [59]. In that work, the pentamer was subjected to 100 ns of MD simulation to relax the initial configuration prior to being employed in PMF calculations. For simplicity in this tutorial, the input coordinates are taken directly from the NMR structure. Should the user wish to more faithfully replicate the published results, follow the protocol described in that work and use the final configuration of the system after 100 ns of MD as input into `pdb2gmx`.

3.4.2 Define the Unit Cell and Solvate

The box size for umbrella sampling simulations, like in any other system, must be set such that the minimum image convention (see Section 3.2.2) is satisfied at all times. There is an additional consideration, however, that requires the length of the biasing potential to also satisfy the minimum image convention. That is, the length of the biasing potential will be calculated from the minimum periodic distance between the restrained species that define the reaction coordinate. In this example, the reaction coordinate will be defined as the COM distance between chains A and B in the Aβ_{42} protofibril, with a final length of 5.0 nm, sufficient to lead to complete dissociation of the two chains (no interactions within the longest nonbonded cutoff). The biasing potential will only be applied along the *z*-axis, therefore the length of the box along *z* must be at least 10.0 nm (double the length of the reaction coordinate), with some additional space to satisfy the minimum image convention between the atoms in the system. The box vector along the *z*-axis will thus be set to 12.0 nm.

An orthorhombic box that defines a solute-box distance of 1.0 nm would result in a unit cell with dimensions of 6.560 nm $\times$ 4.362 nm $\times$ 4.955 nm, with the geometric center naturally following as (3.280, 2.181, 2.4775). To position the Aβ_{42} protofibril in a manner sufficient to satisfy the minimum image convention with respect to the biasing potential in the newly defined box, use `editconf`:

```
$ gmx editconf -f complex.gro -o newbox.gro
    -center 3.280 2.181 2.4775 -box 6.560
    4.362 12
```

After doing so, the protofibril will be positioned within the unit cell as shown in Figure 18.

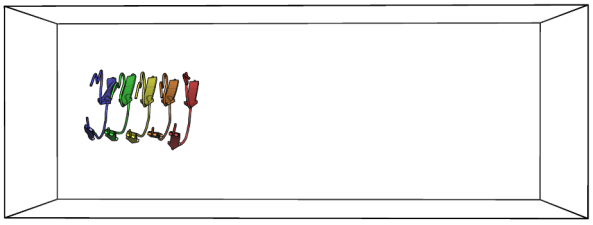

Figure 18. The position of the Aβ_{42} protofibril in the unit cell.

It is important to note here that the approach taken thus far is fairly specific to the system being considered. The biasing potential will be applied directly along the *z*-axis, which is the fibril axis for Aβ fibrils. The box size is designed to minimize the number of water molecules added, which is desirable since the box has to be large to accommodate the restraint length. It is possible to use a somewhat tight-fitting box in the $x-y$ plane since additional restraints will be applied to chain B of the protofibril. The restraints prevent the protofibril from rotating over time, therefore the box does not have to have spherical symmetry. In general, for *e.g.* protein-protein or protein-ligand binding, this approach would not

https://doi.org/10.33011/livecoms.1.1.5068
Living J. Comp. Mol. Sci. 2019, 1(1), 5068

work. The user would have to construct a cubic, rhombic dodecahedral, or truncated octahedral box and define a reaction coordinate in all three spatial dimensions. Because of the unique nature of the system being studied here, the approach is somewhat simplified in this tutorial. The biasing potential is as simple as possible and the system is relatively small so it will run quickly.

Proceed with solvation and the addition of 100 mM NaCl as in any other system:

```
$ gmx solvate -cp newbox.gro -cs spc216.gro
    -o solv.gro -p topol.top

$ gmx grompp -f ions.mdp -c solv.gro -p
    topol.top -o ions.tpr

$ gmx genion -s ions.tpr -o solv_ions.gro
    -p topol.top -pname NA -nname CL
    -neutral -conc 0.1
```

3.4.3 Energy Minimization and Equilibration

Continue preparing the system by performing energy minimization and a short *NPT* equilibration. Note that it is not strictly necessary to separate equilibration into *NVT* and *NPT* phases, as above in the lysozyme tutorial (see Section 3.2.5). The necessary `.mdp` files are provided in the online tutorial. All of the relevant keywords and settings have been discussed previously and will not be elaborated on here.

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

$ gmx mdrun -deffnm em

$ gmx grompp -f npt.mdp -c em.gro -r em.gro
    -p topol.top -o npt.tpr

$ gmx mdrun -deffnm npt
```

3.4.4 Generate Configurations

To compute the PMF along a reaction coordinate (typically denoted as ξ), a series of simulations is performed. Each of these simulations is independent from the others and is carried out under the influence of a biasing potential that restricts the sampling along some degree of freedom to a region of ξ. Commonly ξ is defined as a distance or dihedral angle. In the present example, ξ is defined as the COM distance between chains A and B of the Aβ_{42} protofibril, and as such, it is necessary to generate a series of configurations at increasing COM distance. There are many approaches to generating configurations, including unbiased simulations from which frames are extracted, manual translation of the target species along a vector that corresponds to ξ, or the application of non-equilibrium techniques like SMD to enforce motion along ξ, from which frames are extracted. It is this last approach that will be applied in this tutorial. From an SMD trajectory, frames will be extracted corresponding to regular intervals of COM distance, and then individual simulations of these configurations will be performed (Figure 19).

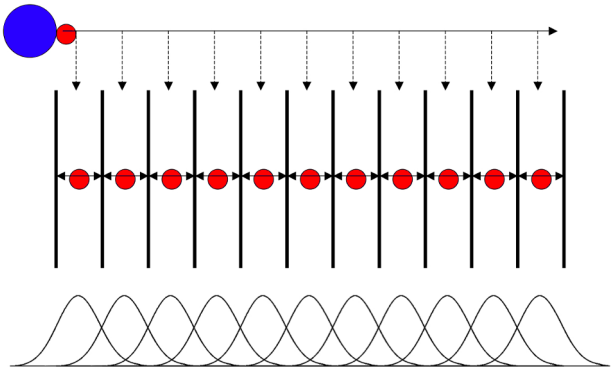

Figure 19. The process of umbrella sampling. From an initial ensemble of configurations along a reaction coordinate, several are chosen for independent simulations at intervals along the reaction coordinate. These simulations are subsequently carried out under the influence of a harmonic biasing potential, restricting their motion along the reaction coordinate to generate a restrained ensemble within a sampling window. Distributions of reaction coordinate values within each window are the "umbrellas" from which the name of the technique is derived.

To generate a series of configurations that can serve as input into the individual sampling windows, the "pull code" in GROMACS (so named for its ability to apply a biasing potential that "pulls" along a specified vector) will be employed to cause the dissociation of chain A from the remainder of the protofibril. First, create an index group for each of these chains:

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

The relevant pull code settings are listed here:

```
pull                    = yes
pull_ncoords            = 1
pull_ngroups            = 2
pull_coord1_groups      = 1 2
pull_group1_name        = Chain_A
pull_group2_name        = Chain_B
pull_coord1_type        = umbrella
pull_coord1_geometry    = distance
```


https://doi.org/10.33011/livecoms.1.1.5068
Living J. Comp. Mol. Sci. 2019, 1(1), 5068

```
pull_coord1_dim          = N N Y
pull_coord1_start        = yes
pull_coord1_rate         = 0.01
pull_coord1_k            = 1000
```

The `define = -DPOSRES_B` line in the `.mdp` file specifies that peptide chain B is to be restrained during this process. Application of restraints here models the stability of amyloid fibrils, which are orders of magnitude larger and therefore more resilient to dissociation of terminal peptides, and also enables more efficient dissociation of chain A. In the absence of position restraints, the entire pentameric protofibril responds to the imaginary spring, causing it to diffuse through the box, distort, and ultimately not produce the desired result. In general, such position restraints are not necessary, and are rarely employed in the case of, *e.g.* protein-ligand or membrane systems. The application of restraints is thus somewhat unique to the system considered here and should not be viewed as generally applicable.

The `pull` keyword tells `grompp` to read settings related to the pull code. If set to `no`, all following settings (`pull_*`) are ignored. The `pull_ncoords` keyword specifies how many reaction coordinates will be present in the simulated system. This number can be set to any value, that is, a multi-dimensional free energy surface can be constructed in any number of dimensions. For this example, there is only one reaction coordinate, which is defined by two groups of atom selections (`pull_ncoords = 2`). Next, the groups themselves are specified. Each of the groups is assigned an integer (1 through N, where N is the value of `pull_ngroups`), which are subsequently named with the `pull_groupN_name` keyword. Here, two groups are specified by the names assigned above in the index file.

The next settings define the type of biasing potential to be applied. The `pull_coord1_type` specifies a harmonic biasing potential with the keyword `umbrella`. It is important to note here that, despite the name of this setting, the current process is not umbrella sampling. The `umbrella` specification is simply a synonym for the harmonic potential that is typically employed during umbrella sampling (see below, Section 3.4.5). The reaction coordinate itself is then specified with the `pull_coord1_geometry` and `pull_coord1_dim` settings. In this case, the `distance` between chains A and B defines the geometry of the reaction coordinate, and the biasing potential only acts along the *z*-axis (`pull_coord1_dim = N N Y`), which specifies the dimensions (x, y, z) and whether or not the biasing potential acts (N = no, Y = yes). The initial value of the reaction coordinate is set very simply with `pull_coord1_start = yes`, such that `grompp` sets the initial value to whatever is computed from the input coordinate file. One can override this value with `pull_coord1_init`, which specifies a floating-point number as the initial (or reference) value of the reaction coordinate.

Finally, the harmonic biasing potential parameters are set. The `pull_coord1_rate` setting specifies the rate at which the imaginary spring connecting the two restrained groups is extended, in nm ps^{-1}. The stiffness of the spring is set with `pull_coord1_k`, according to Hooke's Law. It is important to note that the rate of extension of the spring is not necessarily equivalent to the rate of observed displacement of the pulled groups. The imaginary spring counteracts the attractive forces in the system, and builds up until these restoring forces are overcome. This concept will be explored below in the analysis of the SMD simulation.

Use the `md_pull.mdp` input file to prepare the input for the SMD simulation, and carry it out with `mdrun`:

```
$ gmx grompp -f md_pull.mdp -c npt.gro -p
    topol.top -r npt.gro -n index.ndx -t
    npt.cpt -o pull.tpr

$ gmx mdrun -deffnm pull
```

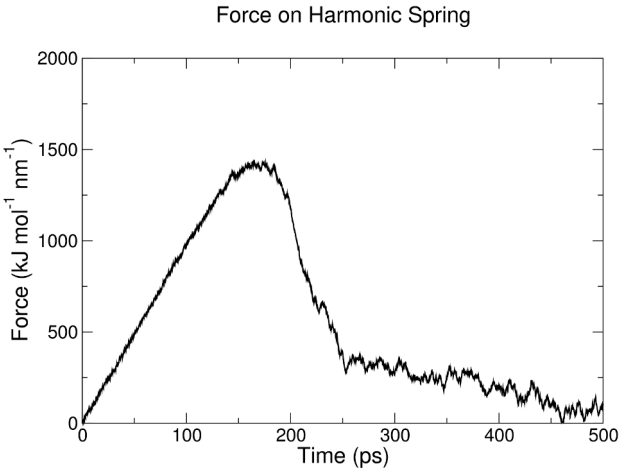

Figure 20. The force on the harmonic spring during SMD separation of chains A and B.

After the SMD simulation is finished, plot the contents of `pullf.xvg` (Figure 20). This file contains the force on the imaginary spring as a function of time. Visualize the trajectory and note that chain A does not dissociate from chain B for some time. Only after the force builds up on the spring does this dissociation begin. The Aβ_{42} protofibril is stabilized by extensive inter-peptide side-chain packing and backbone hydrogen bonding. Thus, a large force is required to disrupt all of these interactions, which are described in detail elsewhere [59].

The `pullx.xvg` file contains the length of the reaction coordinate as a function of time. It may be useful to plot the restraint forces as a function of displacement rather than as

https://doi.org/10.33011/livecoms.1.1.5068
Living J. Comp. Mol. Sci. 2019, 1(1), 5068

a function of time. To extract this information, use standard Linux commands `grep` and `awk`:

```
$ grep -v [@#] pullf.xvg | awk '{print $2}'
    > forces
$ grep -v [@#] pullx.xvg | awk '{print $2}'
    > x
$ paste x forces >
    pull_force_vs_displacement.xvg
$ rm x forces
```

These data are plotted in Figure 21.

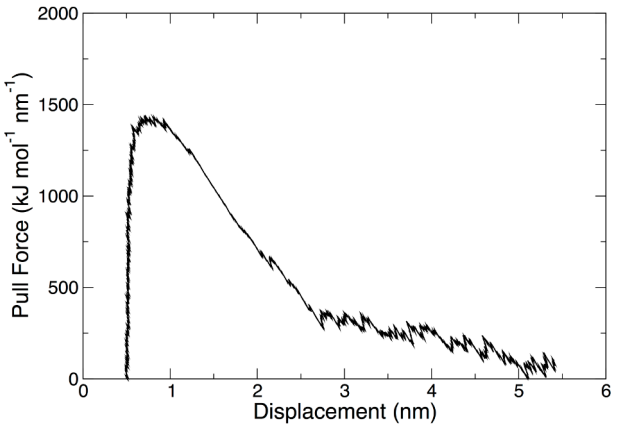

Figure 21. The force on the harmonic spring as a function of displacement during SMD.

Next, configurations must be extracted from the trajectory to define the starting points for each of the umbrella sampling windows. There are several ways to approach this problem, and here the approach will be to:

1. Write configurations of each snapshot
2. Compute the COM distance between chains A and B in each snapshot
3. Compile these COM distances as a function of snapshot index
4. Use selected frames as input into umbrella sampling

The GROMACS `trjconv` program has an option to write each frame in a trajectory to a separate file, `-sep`. Save these separate snapshots with:

```
$ gmx trjconv -s pull.tpr -f pull.xtc -o
    conf.gro -sep
```

Each snapshot will be written to a file called `confN.gro`, where N is the index of the frame. The COM distance between chains A and B in each of these frames is computed with the GROMACS `distance` program, *e.g.* for the frame at t = 0 ps:

```
$ gmx distance -s pull.tpr -f conf0.gro
    -select 'com of group "Chain_A" plus
    com of group "Chain_B"' -n index.ndx
    -oall dist0.xvg
```

Note the use of single quotes (') to enclose the entire selection string and double quotes (") to enclose group names referenced from `index.ndx`.

The online version of this tutorial provides a Bash script (`get_distances.sh`) that automates this entire process and compiles the COM distances into a single text file. This file should be inspected for snapshots at intervals of approximately 0.2 nm (perfectly even spacing is unlikely) and the frame indices should be recorded and corresponding coordinate files saved.

It is worth noting that the COM distance as a function of time can be directly calculated from the trajectory file (`pull.xtc`) for later inspection. After this analysis, the user would have to decide which frames to extract from the trajectory and invoke `trjconv` for each of these frames. As such, the approach taken in this tutorial, while somewhat redundant, may actually be viewed as more efficient. The user can move or copy the desired frames to a new location and simply delete the unnecessary snapshots, ultimately saving time.

While the reaction coordinate in this system is straightforward and justified based on a number of experiments and other mathematical models, the general application of SMD to generate configurations may yield non-equilibrium states that are sensitive to the chosen path. When using this approach in other systems, users should be forewarned that they should generate multiple pulling vectors to generate a sufficient ensemble of potential starting states, and to equilibrate thoroughly to reasonably ensure that the path sampled is the minimum free-energy path.

3.4.5 Umbrella Sampling

Once suitable snapshots have been identified, each umbrella sampling window will be prepared. Each window is equilibrated under an *NPT* ensemble for 100 ps before data collection is carried out over 10 ns. The COM pull settings are the same as what are shown above for the SMD simulation, with the exception of `pull_coord1_rate`, which is now set to zero. During these simulations, the COM distance between chains A and B is to be restrained; no net displacement is to be imposed, which is what a non-zero value of `pull_coord1_rate` specifies.

Prepare the input `.tpr` files for each *NPT* equilibration and carry out those simulations (assuming that snapshot 6 corresponds to a value of ξ = 0.5 nm):

```
$ gmx grompp -f npt_umbrella.mdp -c
    conf6.gro -r conf6.gro -p topol.top -n
    index.ndx -o npt0.tpr
```


https://doi.org/10.33011/livecoms.1.1.5068
Living J. Comp. Mol. Sci. 2019, 1(1), 5068

```
$ gmx mdrun -deffnm npt0
```

After equilibration, production umbrella sampling MD simulations are performed:

```
$ gmx grompp -f md_umbrella.mdp -c npt0.gro
    -t npt0.cpt -p topol.top -r npt0.gro -n
    index.ndx -o umbrella0.tpr

$ gmx mdrun -deffnm umbrella0
```

3.4.6 Data Analysis

To compute the PMF along the reaction coordinate, the Weighted Histogram Analysis Method (WHAM) [61] is employed, implemented in GROMACS as the `wham` program [62], which also includes algorithms to estimate errors associated with the resulting PMF. From the biased simulations (sampling windows), a histogram is constructed, $h(\xi)$. This histogram describes the probability of finding the system at individual locations along the reaction coordinate, *i.e.* a biased probability distribution, $P_b(\xi)$. The WHAM algorithm uses $h(\xi)$ to estimate the uncertainty in an unbiased $P(\xi)$ by iteratively solving a system of equations to compute the PMF that has the smallest uncertainty. The PMF produced in this analysis corresponds to the free energy along ξ.

The input to the GROMACS `wham` program comprises two text files, which list the input `.tpr` file names and either the `pullf.xvg` or `pullx.xvg` files. These lists of file names are provided in two plain-text files, created with any plain-text editor. It is important that these text files contain the list of relevant files in the order they should be assembled. Create a file called `tpr-files.dat` that contains:

```
umbrella0.tpr
umbrella1.tpr
...
umbrella22.tpr
```

Then, create a file called `pullf-files.dat` that contains:

```
umbrella0_pullf.xvg
umbrella1_pullf.xvg
...
umbrella22_pullf.xvg
```

The ellipses in both of the above blocks of text indicate that the intervening file names should similarly be specified. Next, invoke `wham` to compute the PMF:

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

The above command computes a PMF in the units of kcal mol^{-1} and writes it to a file called `profile.xvg`. The corresponding umbrella histograms, $h(\xi)$, are also produced (`histo.xvg`). These results are plotted in Figure 22.

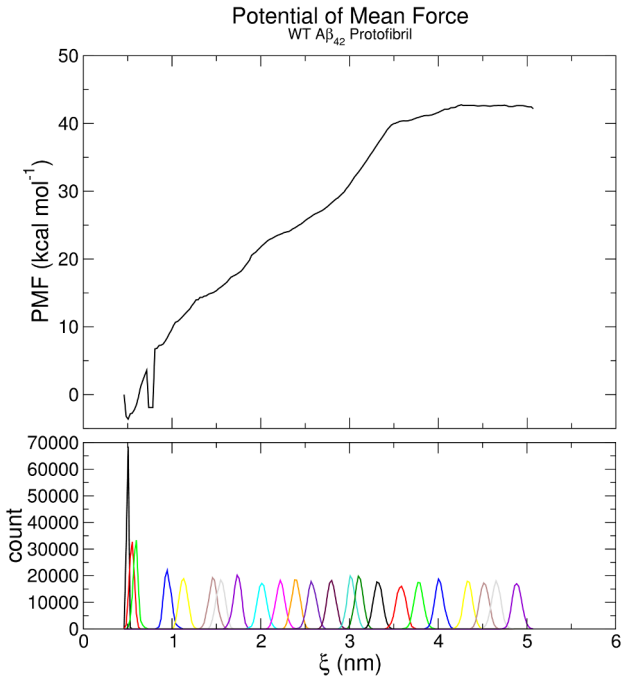

Figure 22. The PMF computed by the WHAM algorithm and the corresponding umbrella histograms.

Note that `histo.xvg` is contains multiple data series and should be plotted in XmGrace as such:

```
$ xmgrace -nxy histo.xvg
```

The PMF has an energy minimum at approximately 0.5 nm, which is the distance corresponding to inter-Cα spacing in canonical β-sheets. The position of this minimum makes sense. The resulting ΔG for dissociation is approximately 46 kcal mol^{-1}, which is close to our published value of 50.5 kcal mol^{-1} [59]. The difference can be attributed to the use of a different starting structure (as noted above) and different sampling window intervals. Note, however, the large defect in the PMF around 0.8 nm and the corresponding lack of sampling in the umbrella histograms in this region of the reaction coordinate (Figure 22). This outcome indicates that this region of the reaction coordinate has not been adequately sampled and at least one more simulation must be performed. At ξ = 0.8 nm, the pulling force in SMD was at its maximum (Figure 21), indicating that the interactions in at this ξ value are very strong and this state may reflect a very high-energy intermediate that is rarely visited. In the umbrella sampling windows employed here, only ξ = 0.7 nm and ξ = 0.9 nm were chosen, and neither sampled around 0.8 nm. To refine this region of the PMF profile, a new simulation should be carried out at ξ = 0.8 nm. The fact that umbrella sampling simulations are independent makes doing so reasonable. The existing

https://doi.org/10.33011/livecoms.1.1.5068
Living J. Comp. Mol. Sci. 2019, 1(1), 5068

simulations do not need to be re-run, as neighboring windows do not depend on one another, unlike other alchemical free energy methods (see Section 3.7) or enhanced sampling techniques like replica exchange.

It is also worth mentioning how WHAM computes the actual PMF values along ξ. By convention, the leftmost window (*i.e.* lowest value of ξ) is assigned a free energy value of zero. The energies in subsequent windows are all defined relative to this value. It may be more useful to shift some physically relevant or otherwise convenient point to zero. Such a practice does not change the result at all, it is a systematic translation of the PMF curve and the final value of ΔG is the same. The GROMACS `wham` program has the ability to shift the final PMF curve to place the zero point at a user-defined location. For example, the place the zero point at the energy minimum (ξ = 0.5 nm), invoke `wham` as follows:

```
$ gmx wham -it tpr-files.dat -if
    pullf-files.dat -o profile_shift.xvg
    -hist -unit kCal -zprof0 0.5
```

The result of this new calculation is shown in Figure 23.

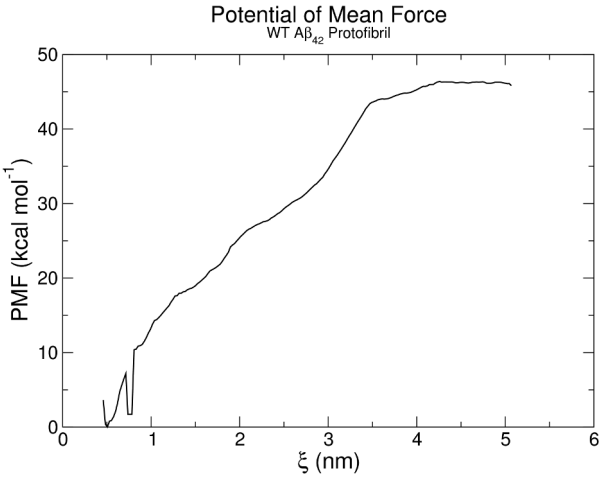

Figure 23. The PMF computed by the WHAM algorithm, with the zero point set to ξ = 0.5 nm, the energy minimum of the profile.

3.4.7 Summary and Review of Objectives

In this tutorial, the user has been guided through the process of performing an SMD simulation of one Aβ_{42} peptide from a pentameric protofibril. From the SMD trajectory, defining a one-dimensional reaction coordinate, snapshots were extracted at roughly regular intervals. These snapshots were subjected to additional equilibration and extended simulations to compute a PMF profile, which yields ΔG for dissociation of the peptide. To review, the objectives for this tutorial were:

1. Apply `.mdp` keywords that invoke the pull code to add a simple, one-dimensional biasing potential to selected atoms in a system
2. Define what is meant by a "reaction coordinate" and how to construct a suitable one
3. Perform restrained simulations in multiple sampling windows along a reaction coordinate
4. Compute a potential of mean force profile

In Section 3.4.4, the GROMACS pull code keywords were introduced and the definition of the reaction coordinate was explained in the context of these settings. In subsequent sections, umbrella sampling calculations were performed and the WHAM algorithm [61] was used to de-bias the histograms to compute the PMF.

To conclude, it is again important to emphasize that the system simulated here involves special considerations, including the use of position restraints and the fact that the reaction coordinate was one-dimensional. Restraints served two specific purposes: first, to allow efficient dissociation of two strongly interacting species and second, to mimic the stability of Aβ fibrils that are orders of magnitude larger (heavier) than the relatively small pentamer considered here. Such restraints are not typically necessary in different systems, such as protein-ligand complexes or those involving membranes. Additionally, the growth of Aβ fibrils is unidirectional, meaning that orthogonal dimensions (here, *x* and *y*) are not relevant to the fibril association/dissociation equilibrium. Hence, the reaction coordinate can conveniently be defined along only the *z*-axis, which also allows for the user of a relatively small box, extended only along the *z*-axis beyond what is necessary to satisfy the particle-particle minimum image convention. If the system being considered is a protein-ligand complex, no such assumption can be made, the biasing force should be applied along a vector in all three spatial dimensions, and the box should be sufficiently large in all dimensions. Other, special cases may similarly be unidirectional, but this case should not be considered a default assumption. The user must always consider the intrinsic geometry or symmetry of the system being studied. As such, the `.mdp` files provided with this tutorial should be adapted for use in any other system, not used directly.

3.5 Tutorial 4: Biphasic Systems

Heterogenous systems are often of interest in molecular simulations. For example, partitioning of a small molecule between aqueous and octanol phases is an experimentally useful property for understanding potential drug-like properties. Simulating such a process to calculate the associated change in free energy is important and may serve as a strict evaluation of force field quality. As such, the ability to construct

https://doi.org/10.33011/livecoms.1.1.5068
Living J. Comp. Mol. Sci. 2019, 1(1), 5068

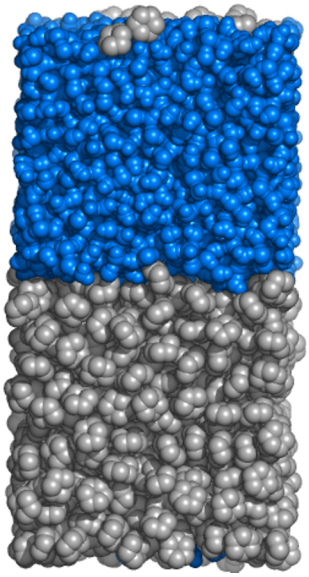

Figure 24. A layered, biphasic system of water (blue, top) and cyclohexane (gray, bottom).

such a system is of great importance, though it may not be apparent exactly how to do so. In this tutorial, a simple biphasic system will be constructed, containing an aqueous phase and a hydrophobic (cyclohexane) phase (Figure 24). This tutorial will focus solely on construction of the system rather than performing and analyzing simulations. The process for simulating biphasic systems is no different from any other simulation, with perhaps the only specific consideration being the treatment of COM motion removal groups. Refer to Section 3.3.6 for discussion on how to treat layered systems and appropriately set the `comm-grps` keyword in `.mdp` files.

The basis of this tutorial (online at http://www.mdtutorials.com/gmx/biphasic) is a recent paper [63] that explored small-molecule and amino-acid dynamics at a water-cyclohexane interface, to test the behavior of the GROMOS96 43A1 force field [22, 23] and topologies produced by the PRODRG server [64].

3.5.1 Build a Cyclohexane Box

To begin, the user needs a coordinate file containing a single cyclohexane molecule and a corresponding topology. Coordinates can be obtained from a number of different model-building programs, but the topology has to be generated, which requires advance knowledge of its parameters or some method to derive them. A simple method to obtain both coordinates and topology for use with GROMOS force fields are online servers PRODRG [64] and ATB [65]. The online version of this tutorial provides suitable coordinate and topology files. Note that if the user obtains a topology from PRODRG, the partial charges in the topology should all be corrected to zero, as required by the united-atom treatment of CH_2 groups.

There are two methods in GROMACS by which a user can build a box of pure liquid. The first uses the `insert-molecules` program to randomly insert a specified number of molecules into a box of given dimensions. Note that this approach does not guarantee that all requested molecules will be added, rather the user specifies a maximum number to be added. The output should always be inspected to ensure that the volume is adequately filled.

To build a 5-nm cubic box and attempt to fill it with up to 1200 cyclohexane molecules, invoke `insert-molecules`:

```
$ gmx insert-molecules -ci chx.gro -nmol
    1200 -box 5 5 5 -o chx_box.gro
```

The program reports that 1114 of the requested 1200 cyclohexane molecules were placed in the box. Note that default van der Waals radii used by GROMACS programs like `insert-molecules` and `solvate` were changed in version 5.0, so different versions may report different outcomes with respect to this number. As insertion is randomly seeded process, even versions in the 2018.x series may vary slightly.

The second method for constructing a box of pure liquid uses the `genconf` program to build a lattice of molecules. This approach requires that the input coordinate file have a properly defined periodic box size so that molecules can be positioned correctly without overlapping. If the box size is not adequate, the `-dist` option (which takes a full vector as its argument) can be used to space the replicated coordinates. The drawback to the `genconf` method is that the resulting system is highly artificial because it is a perfect lattice, and will require a much longer equilibration time to converge to a liquid state. To build a box containing 512 cyclohexane molecules (8^3), invoke `genconf` as follows:

```
$ gmx genconf -f chx.gro -nbox 8 8 8 -o
    chx_box.gro
```

Regardless of the method of preparing the cyclohexane box, these initial coordinates are either random or highly ordered, requiring energy minimization and subsequent equilibration. The online version of the tutorial provides an equilibrated box of 466 cyclohexane molecules in a 4.3-nm box prepared by *NVT* and *NPT* equilibration at 298 K, the latter phase lasting for 10 ns. It is this box that will be used for subsequent preparation steps in this tutorial. The online tutorial also provides a system topology, `chx.top`, corresponding to this system.

3.5.2 Add a Water Layer

At this point, the cubic cyclohexane box could simply be centered within a new, tetragonal box that is extended along one spatial dimension and the empty volume filled with water. The resulting system would have cyclohexane-water interfaces within the central unit cell, and the water layer would be

https://doi.org/10.33011/livecoms.1.1.5068
Living J. Comp. Mol. Sci. 2019, 1(1), 5068

continuous due to PBC. While perfectly viable from the perspective of physics, such a setup is somewhat inconvenient if the user is attempting to add other species (solutes, peptides, etc.) into one of the layers. As such, it is equivalent and perhaps more intuitive to position the existing cyclohexane coordinates in an elongated box such that both the water and cyclohexane layers are contiguous within the central periodic image.

As in the Umbrella Sampling tutorial (see Section 3.4.2), set the existing coordinates at their previous position within the newly extended box using `editconf`. The cyclohexane box is a 4.30795-nm cube, meaning its geometric center is located at (2.153975, 2.153975, 2.153975). Double the box length along the *z*-axis and retain this geometric center:

```
$ gmx editconf -f chx_10ns.gro -o
    chx_newbox.gro -box 4.30795 4.30795
    8.6159 -center 2.153975 2.153975
    2.153975
```

The resulting system will look like what is shown in Figure 25.

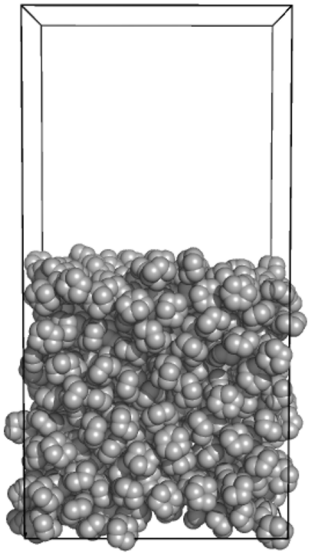

Figure 25. The equilibrated cyclohexane box is repositioned within a tetragonal unit cell that is elongated along the *z*-axis.

The next step is to solvate the empty volume with water. There are likely to be small voids within the cyclohexane layer into which water will be added by the `solvate` program, so make a copy of `vdwradii.dat` from `$GMXLIB` into the working directory. Change the default value of the C radius from 0.17 to 0.35, then run `solvate`:

```
$ gmx solvate -cp chx_newbox.gro -cs
    spc216.gro -p chx.top -o chx_solv.gro
```

There are likely to be a few stray water molecules within the cyclohexane layer, even with this large radius assigned to C atoms. These waters will be expelled quickly in a short equilibration run. Water molecules appearing at the "bottom" of the box are no concern, as they are actually continuous with the water layer in the central image via the periodic boundary along the *z*-axis. The biphasic system is now constructed and ready for energy minimization and subsequent MD simulation.

3.5.3 Other Tips and Tricks

To study partitioning behavior of peptides or small molecules, the approach taken in this tutorial can be expanded to position any other species within the aqueous layer prior to solvation. Similarly, the equilibrated cyclohexane box can be used as input to `solvate -cs` to place a solute in the hydrophobic layer. To add something to the aqueous layer, apply the same concepts as in the previous section to specify its center with `editconf`. This time, the center of the molecule along the *z*-axis is three-quarters of the box length. For example, to add the $KALP_{15}$ peptide from the Membrane Protein tutorial (see Section 3.3) in the aqueous layer, at the center of the empty volume:

```
$ gmx editconf -f peptide.gro -o
    peptide_newbox.gro -box 4.30795 4.30795
    8.6159 -center 2.153975 2.153975
    6.461925
```

Then, "solvate" the peptide with the repositioned cyclohexane layer, which has box dimensions identical to what were just assigned to the peptide, meaning the two components will fit together exactly:

```
$ gmx solvate -cp peptide_newbox.gro -cs
    chx_newbox.gro -o peptide_chx.gro
```

The resulting system appears in Figure 26, and can subsequently be solvated with water using `solvate`.

Should a larger volume be necessary to accommodate larger molecules in the aqueous phase, the cyclohexane layer can be expanded using the GROMACS `genconf` program. The existing cyclohexane coordinates can be expanded in any number of copies along any of the three spatial dimensions. To build a larger cubic box of cyclohexane:

```
$ gmx genconf -f chx_10ns.gro -nbox 2 2 2
    -o chx_bigbox.gro
```

In the event that a tetragonal layer is necessary, do not replicate along the *z*-axis, only through the *x* – *y* plane:

```
$ gmx genconf -f chx_10ns.gro -nbox 2 2 1
    -o chx_biglayer.gro
```


https://doi.org/10.33011/livecoms.1.1.5068
Living J. Comp. Mol. Sci. 2019, 1(1), 5068

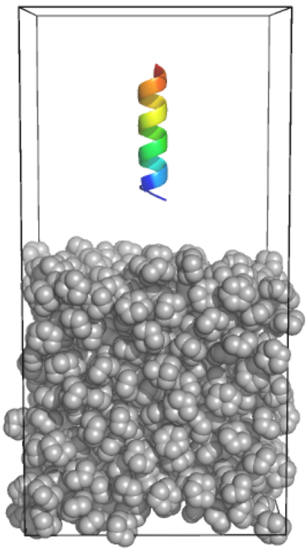

Figure 26. The $KALP_{15}$ peptide has been positioned at the middle of the empty volume that will subsequently be filled with water.

3.5.4 Summary and Review of Objectives

This tutorial has guided the user through the simple construction of a biphasic system containing two layers: water and cyclohexane. The logic applied here can be used in any similar systems. To review, the learning objectives of the tutorial are:

1. Build a simulation system containing a liquid that is not water
2. Manipulate the relative positioning of a box within a larger volume

The tutorial demonstrated how an initial system of non-aqueous liquid can be built by two different methods, and subsequently how the equilibrated box can have its relative position manipulated within a new, larger unit cell. Additional tips and tricks have been provided to demonstrate how other species can easily be added to the biphasic system, to allow for studies of interfacial dynamics and partitioning.

3.6 Tutorial 5: Protein-Ligand Complex

Simulating proteins in complex with small-molecule ligands or inhibitors (Figure 27) is an important part of understanding enzyme catalysis, drug design, and allostery. These simulations present a challenge in that the ligand topology is not likely to have been explicitly derived. Force field parametrization is an advanced topic, one that typically requires expert training. To decrease the barrier to performing these important simulations, many "general" or "drug-like" force fields exist, such as the General AMBER Force Field (GAFF) [67], CHARMM General Force Field (CGenFF) [68], and OPLS3 [69], which have well-known methodologies, parameter assignment protocols, and web servers that can generate topologies

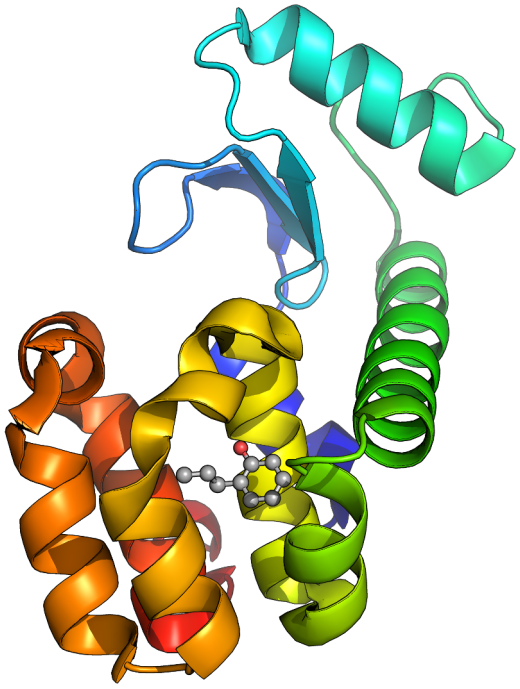

Figure 27. Crystal structure of T4 lysozyme L99A/M102Q with bound 2-propylphenol ligand, taken from PDB 3HTB [66].

for arbitrary molecules.

In this tutorial (available online at http://www.mdtutorials.com/gmx/complex), the protein topology will be generated with the CHARMM36m force field [31], a recent revision of the CHARMM36 force field [27]. The ligand topology will be generated with CGenFF via its web server (https://cgenff.paramchem.org/). Download the CHARMM36m force field in GROMACS format from the MacKerell lab website (http://mackerell.umaryland.edu/charmm_ff.shtml#gromacs), as well as the Python script called `cgenff_charmm2gmx.py`. Extract the force field archive in the working directory:

```
$ tar -zxvf charmm36-jul2017.ff.tgz
```

3.6.1 Protein Topology Preparation

Download the coordinates for T4 lysozyme L99A/M102Q from PDB entry 3HTB. Remove crystal waters and the phosphate (PO4) and β-mercaptoethanol (BME) co-solvents from the coordinate file. The online version of the tutorial provides a file called `3HTB_clean.pdb` that has already removed the unnecessary atoms.

The GROMACS program `pdb2gmx` can only generate topologies for species it recognizes, *e.g.* those encoded in residue topology (`.rtp`) files. That means non-standard species like ligands will cause the program to return a fatal error. Thus, `pdb2gmx` cannot be used to generate a topology for 2-propylphenol. It will be parametrized separately. For

https://doi.org/10.33011/livecoms.1.1.5068
Living J. Comp. Mol. Sci. 2019, 1(1), 5068

now, remove the atoms corresponding to the ligand (residue name JZ4) from the PDB file and save them in a separate file to be used later:

```
$ grep JZ4 3HTB_clean.pdb > jz4.pdb
```

After saving these coordinates, delete the JZ4 coordinate lines from `3HTB_clean.pdb` and generate the protein topology with `pdb2gmx`:

```
$ gmx pdb2gmx -f 3HTB_clean.pdb -o
    3HTB_processed.gro
```

When prompted, choose the CHARMM36 force field (the first entry in the list, "From current directory") and the default water model, TIP3P modified for use with CHARMM.

3.6.2 Ligand Topology Preparation

The principal challenge in performing a protein-ligand simulation is generating a topology for the ligand. Most biomolecular force fields include parameters for common cofactors (heme, ATP, etc.) but do not have existing parameters for every molecule that may be of interest in a simulation. The general force fields listed above seek to remedy this situation, by establishing highly transferable atom types and parameters that can be assigned to arbitrary molecules according to established rules. Doing so removes the burden of manual parametrization from the user, as these procedures often require expert training. General force fields are not necessarily perfect, and there are instances for which manual refinement or reparametrization may be necessary, but they provide a reasonable starting point for these efforts.

In this tutorial, the CGenFF parameter set will be used to generate the topology for the 2-propylphenol ligand (Figure 28). CGenFF is both a parameter set (compatible with the CHARMM force field) and a program that builds small-molecule topologies according to existing rules [70, 71]. The CGenFF program can be accessed via a web server (https://cgenff.paramchem.org/), which will be used in this tutorial. The input to the CGenFF program is a `.mol2` file, which contains atom names, coordinates, Tripos atom types, charges, and bonded connectivity. CGenFF uses these atom types and charges to perform its own parametrization.

To prepare the `.mol2` file, download and install the Avogadro molecular editor (https://avogadro.cc/) [9]. Open `jz4.pdb` within Avogadro, and from the "Build" menu, choose "Add Hydrogens." Save a `.mol2` file called `jz4.mol2`. Several corrections to this file are necessary. The second line of the file, under the `@<TRIPOS>MOLECULE` heading, reads simply "*****" in place of a residue name. Replace this text with "JZ4." The hydrogen atoms that were added are assigned a different residue name and number, which must be manually corrected. Every instance of "JZ167" or "UNL1" should be replaced with "JZ4," and every atom should have a residue number of 1. The corrected `jz4.mol2` file is provided in the online version of the tutorial.

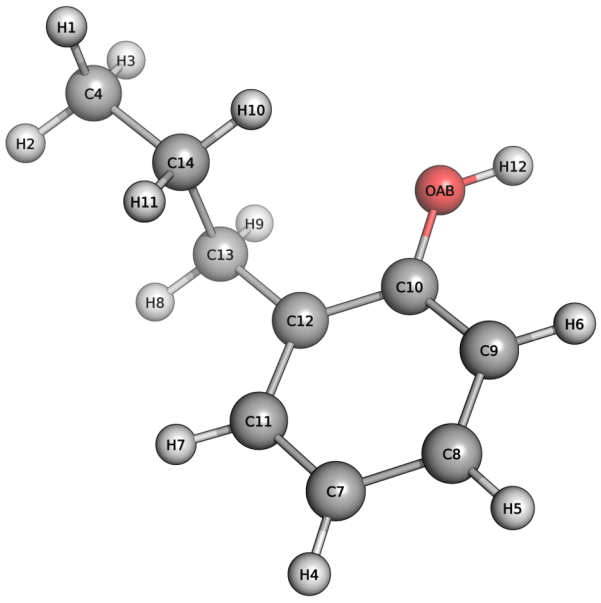

Figure 28. The structure of 2-propylphenol, with hydrogen atoms added. Each atom is labeled by name.

This file is syntactically valid, but still needs modification. The bonds in the `@<TRIPOS>BONDS` section do not appear in ascending numerical order, which will cause problems for CGenFF. To correct this bond listing, use the `sort_mol2_bonds.pl` script provided with the online tutorial:

```
$ perl sort_mol2_bonds.pl jz4.mo2
    jz4_fix.mol2
```

The final, fully corrected file `jz4_fix.mol2` is what will be used as input to the CGenFF server.

The `jz4_fix.mol2` will next be uploaded to the CGenFF web server. The use of this server is free for academic purposes, though user registration is required. Once `jz4_fix.mol2` has been uploaded, the server will return a topology and parameter file in a CHARMM-formatted stream file (`jz4.str`). This file should be downloaded from the CGenFF website so it can be converted to GROMACS format. Before continuing, the contents of `jz4.str` should be inspected. CGenFF calculates penalty scores as an indicator of the quality of the topology. Parameters are assigned by analogy, and in cases for which a strong match cannot be made, the penalty score will be large. Any penalty less than 10 indicates a reliable topology. Penalties between 10 and 50 inform the user that some refinement may be necessary, and at minimum, some basic validation should be performed (QM potential energy scans in the case of bonded parameters,

https://doi.org/10.33011/livecoms.1.1.5068
Living J. Comp. Mol. Sci. 2019, 1(1), 5068

QM water interactions for charges), and penalty scores larger than 50 indicate that significant manual refinement or reparametrization is required. The CGenFF topology for 2-propylphenol has penalties below 1 for both charges and the new dihedrals that were generated. This topology is likely suitable for use directly as it is.

To produce a GROMACS-formatted topology, use the `cgenff_charmm2gmx.py` script downloaded earlier from the MacKerell lab website. Invoke it as follows:

```
$ python cgenff_charmm2gmx.py JZ4
    jz4_fix.mol2 jz4.str charmm36-jul2017.ff
```

Recall from the introduction that this conversion script requires a Python version in the 2.7.x series, with a NetworkX version in the 1.11.x series, not the 2.x series. When finished, the conversion script will produce three important output files:

1. `jz4.itp` - the topology of 2-propylphenol (JZ4) in GROMACS format
2. `jz4.prm` - additional parameters required for the ligand topology
3. `jz4_ini.pdb` - the coordinates of the ligand

To regenerate the protein-ligand complex, the coordinates of JZ4 (with hydrogen atoms added) must be appended to those of the T4 lysozyme enzyme. First, convert the `jz4_ini.pdb` to `.gro` format:

```
$ gmx editconf -f jz4_ini.pdb -o jz4.gro
```

Copy the enzyme coordinates from `3HTB_processed.gro` to a new file, called `complex.gro`. Copy and paste the coordinates from `jz4.gro` into `complex.gro`, immediately after the protein coordinates and before the last line containing the box vectors. The JZ4 ligand contains 22 atoms, so add 22 to the number of protein atoms in the second line of `complex.gro`. There should be 2636 atoms in this coordinate file after the addition of the ligand coordinates.

Having added the coordinates of the ligand to the system, the topology must also be updated. Add the ligand topology to `topol.top` as shown here:

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

; Include ligand topology
#include "jz4.itp"

; Include water topology
#include "./charmm36-jul2017.ff/tip3p.itp"
```

Since the ligand requires new parameters to be added to the force field, the `jz4.prm` file must also be add to the topology via an `#include` statement. Such a statement must be placed specifically within `topol.top`. It must be after the `#include` statement for the CHARMM36 force field, but before the declaration of the protein `[moleculetype]` directive. Bonded parameters can only be added for known atom types, hence why `jz4.prm` must be added after the first call to the CHARMM36 force field. Similarly, all parameters in the force field must be defined before any molecule definitions can be introduced. Therefore, update `topol.top` to read as follows:

```
; Include forcefield parameters
#include "./charmm36-jul2017.ff/forcefield.itp"

; Include ligand parameters
#include "jz4.prm"

[ moleculetype ]
; Name            nrexcl
Protein_chain_A     3
```

Finally, update the contents of the `[ molecules ]` directive of `topol.top` to reflect the fact that the ligand coordinates have been added to the system:

```
[ molecules ]
; Compound        #mols
Protein_chain_A     1
JZ4                 1
```

3.6.3 Solvate the System

The remaining steps in preparing the simulation system are much like any other system of a protein in water. In this tutorial example, the protein-ligand complex will be solvated in a rhombic dodecahedral unit cell, which requires only approximately 77% of the volume of a cubic unit cell with the same periodic distance. It is therefore more efficient to use such a box shape. Place the T4 lysozyme-JZ4 complex in a rhombic dodecahedral box with `editconf` and add water with `solvate`.

```
$ gmx editconf -f complex.gro -o newbox.gro
    -bt dodecahedron -d 1.0

$ gmx solvate -cp newbox.gro -cs spc216.gro
    -p topol.top -o solv.gro
```

When visualized, it will appear that the protein-ligand complex has not been centered in the box and the added water appears to occupy a tetragonal unit cell. This point is often confusing for new users but should not be a concern. By default, GROMACS will re-wrap the coordinates into a triclinic box, which is the most efficient form for carrying out the necessary vector operations during a simulation. The coordinates of the solvated system and their relationship to a triclinic unit

https://doi.org/10.33011/livecoms.1.1.5068
Living J. Comp. Mol. Sci. 2019, 1(1), 5068

cell are shown in Figure 29. Recovering the dodecahedral box shape will be discussed later in this tutorial, but it is not necessary to make any changes to the solvated coordinates. The desired geometry will be used in the simulation.

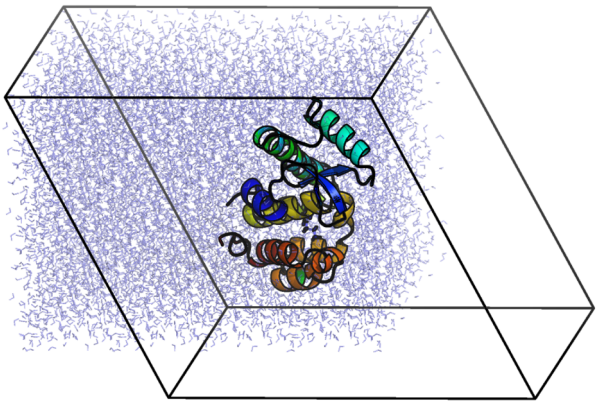

Figure 29. The coordinates of the solvated system and within a triclinic unit cell shape.

Next, add neutralizing counterions to the system. The T4 lysozyme enzyme carries a +6 charge at neutral pH, so 6 Cl^- ions will be added by `genion`.

```
$ gmx grompp -f ions.mdp -c solv.gro -p
    topol.top -o ions.tpr

$ gmx genion -s ions.tpr -o solv_ions.gro
    -p topol.top -pname NA -nname CL
    -neutral
```

3.6.4 Energy Minimization and Equilibration

The solvated, neutralized system will be energy-minimized and equilibrated in two phases, *NVT* and *NPT*, following the same protocol as the lysozyme system described above (Sections 3.2.4 and 3.2.5). There are notable differences in the `.mdp` files used here that warrant further explanation. The CHARMM36 force field requires the use of a switching function on the short-range forces, rather than a plain cutoff. The relevant settings are listed here:

```
cutoff-scheme           = Verlet
ns_type                 = grid
nstlist                 = 20
rlist                   = 1.2
vdwtype                 = cutoff
vdw-modifier            = force-switch
rvdw-switch             = 1.0
rvdw                    = 1.2
```

The short-range nonbonded cutoff is set to 1.2 nm, and a buffered neighbor list is constructed with the Verlet cutoff scheme, meaning that the actual value of `rlist` will be increased as needed to conserve energy. The van der Waals forces are truncated (`vdwtype = cutoff`) at 1.2 nm (`rvdw`), with an "inner" cutoff of 1.0 nm set with `rvdw-switch`, specifying the start of a region over which the forces are smoothly switched to zero, specified by `vdw-modifier = force-switch`. This nonbonded convention is the one for which the CHARMM force field was parametrized. It is important to utilize these settings to ensure a valid simulation. Electrostatic forces are calculated with PME and a real-space cutoff of 1.2 nm.

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

$ gmx mdrun -deffnm em
```

During equilibration, the solute non-hydrogen atoms are typically restrained. In this system, the solute includes the 2-propylphenol ligand, but no restraint topology exists for it. The GROMACS `genrestr` program can be used to create such a topology, but first an index group specifying only the non-hydrogen atoms in the ligand is required. Create the index file with `make_ndx`:

```
gmx make_ndx -f jz4.gro -o index_jz4.ndx
> 0 & ! a H*
> q
```

The expression above (`0 & ! a H*`) selects all atoms in the system (group 0) that do not (!) have names starting with H. Once this index file has created, use it to write a position restraint topology for the ligand, selecting group 3 when prompted.

```
$ gmx genrestr -f jz4.gro -o posre_jz4.itp
    -n index_jz4.ndx -fc 1000 1000 1000
```

In Section 3.2.5, the concept of thermostats was introduced. The typical strategy is to assign solute and solvent atom to separate thermostatting groups. In this case, the ligand is encompassed by the default `non-Protein` group. Given that the ligand is physically bound to the protein, its dynamics are therefore coupled to those of the protein in a way that makes it illogical to include the ligand in the generic `non-Protein` group, which includes solvent. It is more logical to couple the ligand together with the protein. To do so, again invoke `make_ndx` to merge the "Protein" and "JZ4" groups (1 and 13, respectively):

```
$ gmx make_ndx -f em.gro -o index.ndx
...
> 1 | 13
> q
```

Then, continue with *NVT* and *NPT* equilibration. In each, the thermostat has two groups (`tc-grps`). One is set to the

https://doi.org/10.33011/livecoms.1.1.5068
Living J. Comp. Mol. Sci. 2019, 1(1), 5068

merged protein-ligand group (`Protein_JZ4`) and the other is the solvent (`Water_and_ions`).

```
$ gmx grompp -f nvt.mdp -c em.gro -r em.gro
    -p topol.top -n index.ndx -o nvt.tpr

$ gmx mdrun -deffnm nvt

$ gmx grompp -f npt.mdp -c nvt.gro -t
    nvt.cpt -r nvt.gro -p topol.top -n
    index.ndx -o npt.tpr

$ gmx mdrun -deffnm npt
```

3.6.5 Production MD Simulation

Once equilibrated, the position restraints are released and a production MD simulation is performed. All settings in the `.mdp` file are familiar by now and need no repeating here. In this tutorial, the simulation will be carried out for 10 ns, which is still generally too short for most real purposes, but enough time to illustrate some principles of post-processing and analysis in the next section.

```
$ gmx grompp -f md.mdp -c npt.gro -t
    npt.cpt -p topol.top -n index.ndx -o
    md_0_10.tpr

$ gmx mdrun -deffnm md_0_10
```

3.6.6 Analysis

After the simulation is over, the effects of periodic boundary conditions must be accounted for, as in Section 3.2.7. The GROMACS `trjconv` program is again used to perform the required coordinate manipulations. Additionally, `trjconv` allows for re-wrapping of the coordinates to recover the rhombic dodecahedral box shape. Process the MD trajectory by placing all whole molecules in the box, wrapping the coordinates into their most compact form, and centering the protein:

```
$ gmx trjconv -s md_0_10.tpr -f md_0_10.xtc
    -o md_0_10_center.xtc -center -pbc mol
    -ur compact
```

Choose the "Protein" group for centering and "System" for output.

To visualize the trajectory in a program like VMD, extract the first frame and save it as a coordinate file (`.gro` or `.pdb` format):

```
$ gmx trjconv -s md_0_10.tpr -f
    md_0_10_center.xtc -o start.pdb -dump 0
```

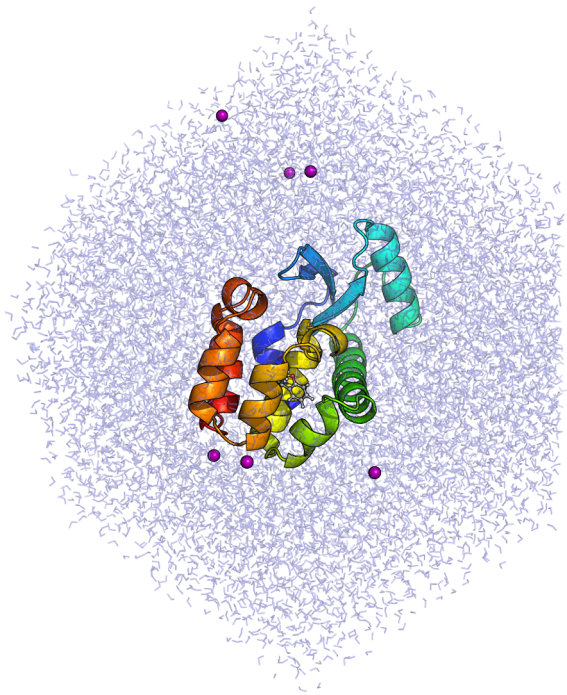

Figure 30. The coordinates of the solvated, equilibrated system after coordinate re-wrapping to yield the most compact box shape, the rhombic dodecahedron.

It is also apparent that this coordinate file has the expected box shape (Figure 30).

To make an even smoother rendering of the trajectory, perform rotational and translational fitting. Select "Backbone" for fitting and "System" for output. Note that fitting and PBC re-wrapping cannot be performed at the same time, hence there are two calls to `trjconv` needed to perform these steps. Rotational and translational fitting via `trjconv` is not required for analysis, but can be useful for visualization, particularly for long trajectories in which the protein may diffuse substantially.

```
$ gmx trjconv -s md_0_10.tpr -f
    md_0_10_center.xtc -o md_0_10_fit.xtc
    -fit rot+trans
```

The 2-propylphenol ligand principally interacts with T4 lysozyme via hydrophobic interactions, but it can form one hydrogen bond with the enzyme, between the hydroxyl group of the ligand phenol moiety and the carbonyl oxygen atom (O_ϵ) of Gln102 (Figure 31). The initial position of the phenolic hydrogen atom built by Avogadro is such that this hydrogen bond is not formed, even after energy minimization. It may be of interest to monitor the formation of this hydrogen bond during the simulation to see if it is indeed a stabilizing force, as expected. The GROMACS `hbond` program can be useful for this purpose, but in this tutorial, a different approach will be

https://doi.org/10.33011/livecoms.1.1.5068
Living J. Comp. Mol. Sci. 2019, 1(1), 5068

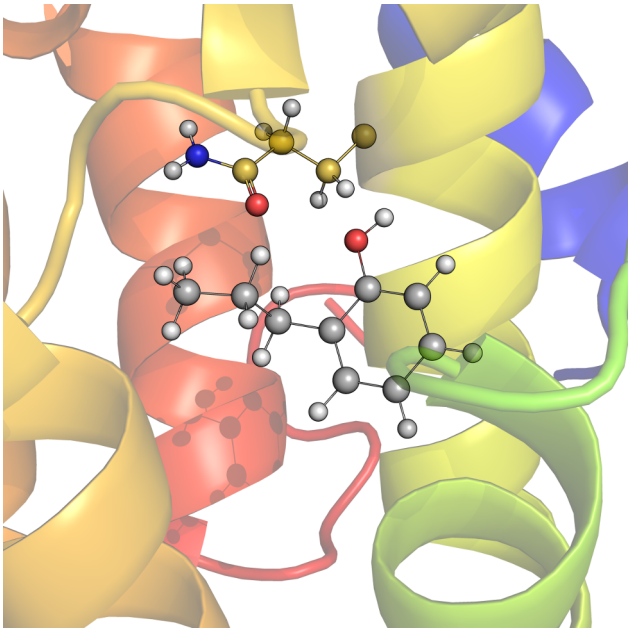

Figure 31. The 2-propylphenol ligand in the active site of T4 lysozyme after energy minimization, with nearby Gln102 shown in ball-and-stick rendering.

taken, as `hbond` will just produce a series of 0 or 1 values over time. A typical hydrogen bond is defined as having a donor-acceptor distance $\leq$ 3.5 Å (0.35 nm) and a donor - acceptor - hydrogen angle of $\leq$ 30° (alternatively defined as a donor - hydrogen - acceptor angle of $\geq$ 150°). Here, two different programs will be used to generate distance and angle time series, demonstrating how to use the GROMACS `distance` and `angle` programs, as well as additional use of `make_ndx` to create index groups needed for the angle analysis.

To compute the distance between the JZ4 phenol oxygen atom (named OAB, see Figure 28) and the Gln102 O_ϵ atom (named OE1), execute the GROMACS `distance` program:

```
$ gmx distance -s md_0_10.tpr -f
    md_0_10_center.xtc -select 'resname
    "JZ4" and name OAB plus resid 102 and
    name OE1' -oall
```

The `distance` program will produce a time series to `dist.xvg` and will print statistics to the terminal:

```
Number of samples:  1001
Average distance:   0.31031  nm
Standard deviation: 0.05078  nm
```

Next, create an index file consisting of the JZ4 OAB and H12 atoms (Figure 28) and the Gln102 OE1 atom:

```
$ gmx make_ndx -f em.gro -n index.ndx
> 13 & a OAB | a H12

Copied index group 13 'JZ4'
Found 1 atoms with name OAB
Merged two groups with AND: 22 1 -> 1
Found 1 atoms with name H12
Merged two groups with OR: 1 1 -> 2

 23 JZ4_&_OAB_H12       :     2 atoms

> 1 & r 102 & a OE1

Copied index group 1 'Protein'
Merged two groups with AND: 2614 17 -> 17
Found 14 atoms with name OE1
Merged two groups with AND: 17 14 -> 1

 24 Protein_&_r_102_&_OE1:     1 atoms

> 23 | 24

Copied index group 23 'JZ4_&_OAB_H12'
Copied index group 24 'Protein_&_r_102_&_OE1'
Merged two groups with OR: 2 1 -> 3

 25 JZ4_&_OAB_H12_Protein_&_r_102_&_OE1:     3 atoms
```

By supplying the existing `index.ndx` file to `make_ndx`, the new groups will be appended to the existing input file that was used previously for the simulations. The first selection (`13 & a OAB | a H12`) specifies the atoms in group 13 (the JZ4 ligand) that are named either OAB or H12, returning the two atoms that meet these criteria. This group is saved in the index file as group 23. The next selection returns a single atom, the OE1 atom of protein residue 102 (Gln102) from group 1 (Protein), which is saved as group 24. The final command merges the two new groups into one group consisting of three atoms, which is necessary for calculating the angle formed by these atoms. Next, invoke the `angle` program:

```
$ gmx angle -f md_0_10_center.xtc -n
    index.ndx -ov angle.xvg
```

The `angle` program also prints useful statistics to the terminal:

```
Found points in the range from 1 to 123
    (max 180)
 < angle >  = 23.3454
< angle^2 > = 632.498
Std. Dev.   = 9.35373
```

The average value is 23 $\pm$ 9°, which is somewhat unexpected; if the angle is defined as OAB - H12 - OE1 and the hydrogen bond is maintained (as it appears to from visual inspection of the trajectory), why is it that the value is so low and not somewhat closer to 180°? Inspect the contents of `index.ndx` and group 25 will contain:

```
[ JZ4_&_OAB_H12_Protein_&_r_102_&_OE1 ]
1616 2624 2636
```

GROMACS intrinsically sorts the atom indices in the groups, so the actual angle that is returned corresponds to OE1 - OAB - H12, more akin to the typical hydrogen

https://doi.org/10.33011/livecoms.1.1.5068
Living J. Comp. Mol. Sci. 2019, 1(1), 5068

bonding definition. The output from the `angle` program thus corresponds to the degree to which the hydrogen-bonded atoms deviate from linearity. The average value (23 $\pm$ 9°) matches the expectation for a hydrogen bond ($\leq$ 30°). If one were to want to obtain an actual time series of the OAB - H12 - OE1 angle, the atoms in the index group would have to be reordered as such:

```
[ JZ4_&_OAB_H12_Protein_&_r_102_&_OE1 ]
2624 2636 1616
```

Using this index group as an input into the `angle` program yields an average value of 147 $\pm$ 11°.

Another analysis that is commonly performed in simulations of protein-ligand complexes is the ligand RMSD, to determine how much the binding pose changed over time. This quantity can be useful in assessing the stability of the interactions between the ligand and the protein, and also as an indicator of the quality of the ligand topology. That is, a poor-quality topology will not preserve the native interactions and will often lead to distorted binding poses or even dissociation. Begin by creating a new index group containing only the heavy atoms of 2-propylphenol:

```
$ gmx make_ndx -f em.gro -n index.ndx
> 13 & ! a H*
> name 26 JZ4_Heavy
> q
```

This selection should look familiar, as it is the same syntax used to create an index group before generating the ligand position restraint topology. Why, then, is it necessary to recreate it here? The restraint topology requires atom numbers that match those of the `[moleculetype]` definition, which run from 1 to the number of atoms in the molecule (in this case, 22). These atom numbers differ from the global atom numbers in the coordinate file of the system. That is, in `posre_jz4.itp`, the only valid numbers are from 1-22, but the atom numbers of JZ4 in the final system are from 2615-2636, because they appear after the protein (atoms 1-2614). After creating this index group, execute the `rms` program. Choose "Backbone" for least-squares fitting and "JZ4_Heavy" for the group for RMSD calculation. Doing so computes the RMSD of the ligand in the context of overall fitting of the protein's motion, that is, how much the ligand has moved relative to the protein.

```
$ gmx rms -s em.tpr -f md_0_10_center.xtc
    -tu ns -o rmsd_jz4.xvg
```

Note the use of `em.tpr` as the reference structure. The RMSD is thus being computed using the energy-minimized crystal coordinates as reference. Over the 10-ns simulation, the ligand RMSD plateaus at a value of ~0.1 nm (1 Å). Such a low value indicates that the binding pose was very stable, at least on this short time scale.

The final analysis that will be performed is an interaction energy calculation. GROMACS provides the ability to compute the short-range nonbonded interaction energy between any groups of atoms. Note that this quantity may not have real, physical meaning if the force field was not parametrized to do so. As the CHARMM force field includes explicit targeting of QM water interactions with all species, it is intrinsically balanced in such a way that interaction energy can be a useful metric. This quantity should not, however, be confused with a "binding energy" or a free energy of any sort. It is simply a decomposition of the potential energy of the system, including only nonbonded terms between the selected atom groups.

Interaction energies can be computed using `mdrun` in conjunction with its `-rerun` option. Create a new `.mdp` file (*e.g.* a copy of `md.mdp`) that now includes the line:

```
energygrps              = Protein JZ4
```

Use `grompp` to create a new `.tpr` file, and subsequently recalculate the energies of the configurations in the trajectory with `mdrun -rerun`. Note that the `mdrun` command instructs GROMACS to compute the energies using only CPU hardware. Doing so is required because multiple energy groups are not supported on GPU.

```
$ gmx grompp -f ie.mdp -c npt.gro -t
    npt.cpt -p topol.top -n index.ndx -o
    ie.tpr

$ gmx mdrun -deffnm ie -rerun md_0_10.xtc
    -nb cpu
```

Extract the interaction energy components, Coul-SR:Protein-JZ4 (group 51) and LJ-SR:Protein-JZ4 (group 52) using the `energy` program:

```
$ gmx energy -f ie.edr -o
    interaction_energy.xvg
```

The output should indicate that the short-range Coulombic interaction energy is -20.5 $\pm$ 7.4 kJ mol^{-1} and the short-range Lennard-Jones interaction energy is -99.1 $\pm$ 7.2 kJ mol^{-1}. The decomposition of these terms, regardless of force field, has no physical meaning. Only the total interaction energy may be considered a useful metric, again only if the force field has been parametrized in such a way that makes this quantity useful. Thus, the total interaction energy between 2-propylphenol and T4 lysozyme over the 10-ns MD simulation performed here is -119.6 $\pm$ 10.3 kJ mol^{-1}.

3.6.7 Summary and Review of Objectives

This tutorial has guided the user through the process of preparing a protein-ligand system for an MD simulation. The

https://doi.org/10.33011/livecoms.1.1.5068
Living J. Comp. Mol. Sci. 2019, 1(1), 5068

preparation of a reliable ligand topology is the most challenging aspect of the process, and in this example, the CGenFF web server was used to prepare a topology for 2-propylphenol that is compatible with the CHARMM36 protein force field. After adding hydrogen atoms to the crystal coordinates of the ligand, producing a `.mol2` file defining bonded connectivity and primitive atom types, the topology was produced in CHARMM format and subsequently converted to yield a GROMACS-compatible topology. To review, the learning objectives of the tutorial are:

1. Produce a ligand topology outside of GROMACS and incorporate it into a system topology
2. Determine how to verify if a ligand topology is suitable for further simulation
3. Perform analysis of interactions between a protein and a ligand

Each of these objectives was addressed in Section 3.6.2. Other new concepts introduced in this tutorial were those related to special considerations for thermostatting groups (merging the protein and ligand groups), re-wrapping of coordinates when using a rhombic dodecahedral unit cell shape, generating index groups for ligand-specific analysis, and calculating interaction energies after the simulation has finished.

3.7 Tutorial 6: Free Energy of Solvation

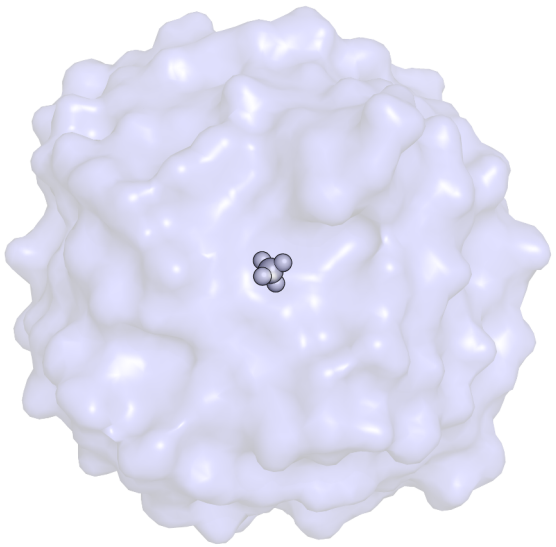

Figure 32. A single methane molecule (CH_4) in water.

In Section 3.4, the concept of computing a free energy difference along a reaction coordinate was introduced. There is another type of free energy calculation frequently employed in MD simulations, in which a set of atoms is transformed in some manner from one state to another (typically denoted A and B), yielding some difference in free energy that relates to this change in topology. Common applications include mutating one small molecule to another and computing a $\Delta\Delta G$ of binding (a relative free energy difference) or by turning off all interactions between some molecule and the remaining system, yielding ΔG of solvation in that system (an absolute free energy difference). The free energy difference between two states, A and B, was originally defined by Zwanzig [72]:

$$\Delta G(A \longrightarrow B) = G_B - G_A = -k_B T \left\langle exp\left(-\frac{E_B - E_A}{k_B T}\right)\right\rangle_A \quad (3)$$

That is, for every configuration generated in state A, the energy of the system is also evaluated in state B, from which the free energy difference, ΔG, is calculated. This approach is called free energy perturbation (FEP). The drawback to the approach is that convergence is difficult unless the difference between the two states is small. For "mutation" of one molecule into another, or to transform a molecule from a fully interacting to a non-interacting state, this approach may fail to converge. In practice, such a calculation of a free energy difference is actually a series of MD simulations, in which the molecule of interest progressively has its topology changed from state A to state B as a function of a discrete coupling factor, λ. In these simulations, the Hamiltonian $H(\mathbf{r}, \mathbf{p})$ in terms of atomic coordinates ($\mathbf{r}$) and momenta ($\mathbf{p}$) is extended to include λ, *e.g.* $H(\mathbf{r}, \mathbf{p}; \lambda)$. For a given value of λ (which takes values from 0 to 1, indicating fully in the A-state and fully in the B-state, respectively), the Hamiltonian of the system is expressed as:

$$H(\mathbf{r}, \mathbf{p}; \lambda) = (1 - \lambda)H^A(\mathbf{r}, \mathbf{p}) + \lambda H^B(\mathbf{r}, \mathbf{p}) \quad (4)$$

This approach to determining free energy differences is often called an "alchemical transformation," as it involves unphysical intermediates. For example, when determining ΔG of solvation, state A ($\lambda = 0$) would correspond to the solute in a non-interacting state. In other words, the molecule experiences intramolecular interactions but no intermolecular interactions (*e.g.* solute-solvent) such that it samples the ideal gas state. State B ($\lambda = 1$) would then correspond to the fully interacting state, in which all nonbonded interactions are at full strength. Setting $\lambda = 0.5$ indicates that intermolecular interactions are at half strength, but how does one define a molecule that is halfway interacting with its surrounding environment? These are unphysical intermediates. However, since ΔG is a state function, it does not matter what path is taken to arrive at the final answer, and the purpose of these intermediates is to define changes in energy that are small enough that they allow for convergence of the calculation.

https://doi.org/10.33011/livecoms.1.1.5068
Living J. Comp. Mol. Sci. 2019, 1(1), 5068

Another method used to evaluate free energy differences is thermodynamic integration (TI), wherein the same λ-dependent transformation is carried out, and the quantity $\frac{\partial H(\lambda)}{\partial \lambda}$ is stored. After the simulations are finished, the integral is taken over the entire λ range:

$$\Delta G = \int_0^1 \left\langle \frac{\partial H(\lambda)}{\partial \lambda} \right\rangle d\lambda \tag{5}$$

The method used for computing ΔG in this tutorial is the Bennett Acceptance Ratio (BAR) [73]. Bennett originally formulated this method for Monte Carlo (MC) simulations, however it is easily extended to MD simulations, which are essentially MC simulations in which all trial moves are accepted. The BAR method is based on an assumption that states A and B share the same microstates, though the energies of these microstates are different. Bennett subsequently demonstrated that for functions satisfying detailed balance, one can compute the free energy change between states from the Metropolis function in the simplest case:

$$M(x) \equiv min(e^{-x}, 1) \tag{6}$$

such that

$$e^{-\beta \Delta G} = \frac{\langle M(\beta(U_B - U_A))\rangle_A}{\langle M(\beta(U_A - U_B))\rangle_B} \tag{7}$$

where $\beta = (k_B T)^{-1}$ and the angle brackets denote the ensemble averages over the configurations in states A and B.

Bennett demonstrates that in the more general case, in which some function, f, is used that includes some unknown free energy offset, C:

$$e^{-\beta(\Delta G - C)} = \frac{\langle f(\beta(U_B - U_A - C))\rangle_A}{\langle f(\beta(U_A - U_B + C))\rangle_B} \tag{8}$$

that the free energy can be estimated with minimum error when the chosen function is the Fermi function:

$$f(x) = \frac{1}{1 + e^x} \tag{9}$$

which, when used in Equation 8, and solving iteratively after an initial estimate for C, yields:

$$\Delta G \approx C \tag{10}$$

This tutorial (available online at http://www.mdtutorials.com/gmx/free_energy) will guide the user through the calculation of the free energy change associated with removing van der Waals interactions of methane with water (Figure 32) using the BAR algorithm, implemented in GROMACS as the `bar` program. This quantity is part of the total transformation of methane that would be used to compute the complete free energy of solvation, ΔG_{solv}. This quantity has been rigorously calculated by Shirts et al. using the TI method [74].

3.7.1 System Contents

The system considered here contains a single methane (CH_4) molecule solvated in 596 water molecules (Figure 32). The OPLS-AA force field [26] is used to describe methane and TIP3P [33] is the water model. The coordinate and topology files for the system are available from the online version of this tutorial. A brief description of the topology contents is warranted here. The following describes the methane `[moleculetype]`:

```
[ moleculetype ]
; Name                            nrexcl
Methane                           3

[ atoms ]
; nr       type  resnr  residue  atom  cgnr    charge      mass
   1   opls_138      1     ALAB    CB     1     0.000    12.011
   2   opls_140      1     ALAB   HB1     2     0.000     1.008
   3   opls_140      1     ALAB   HB2     3     0.000     1.008
   4   opls_140      1     ALAB   HB3     4     0.000     1.008
   5   opls_140      1     ALAB   HB4     5     0.000     1.008
```

The residue is named "ALAB," from the β-carbon of alanine, for which methane serves as a model compound. All of the charges on the atoms are set to zero, though strictly speaking, it is not necessary to do so. This tutorial will only consider the van der Waals transformation (that is, turning off Lennard-Jones interactions as a function of λ). In past versions of GROMACS (e.g. 4.0 and earlier), to do such a transformation would have required a topology with the form shown here, so that Coulombic interactions are not included. As will be explained below, charges can be present in the topology but also ignored. Should the user wish to perform a full transformation to calculate ΔG_{solv}, appropriate charges for all H atoms are +0.060 and -0.240 for the C atom. The topology shown here corresponds to the A-state of the molecule, such that it is fully interacting with its environment. It is also not necessary to specify a B-state in the topology (as was the case in past GROMACS versions, as well) since the transformation of nonbonded terms is controlled entirely in the `.mdp` file. A relative free energy calculation, in which one molecule is "mutated" to another, does require explicit B-state parameters (atom types and charges) to be set. Such a calculation is beyond the scope of this tutorial.

3.7.2 Free Energy Settings

Every `.mdp` file used in this tutorial has a section with the following form:

```
free_energy              = yes
init_lambda_state        = 0
delta_lambda             = 0
calc_lambda_neighbors    = 1
couple-moltype           = Methane
couple-lambda0           = vdw
couple-lambda1           = none
couple-intramol          = no
vdw_lambdas              = 0.00 0.05 0.10 0.15 ...
```


https://doi.org/10.33011/livecoms.1.1.5068
Living J. Comp. Mol. Sci. 2019, 1(1), 5068

```
coul_lambdas              = 0.00 0.00 0.00 0.00 ...
bonded_lambdas            = 0.00 0.00 0.00 0.00 ...
restraint_lambdas         = 0.00 0.00 0.00 0.00 ...
mass_lambdas              = 0.00 0.00 0.00 0.00 ...
temperature_lambdas       = 0.00 0.00 0.00 0.00 ...
sc-alpha                  = 0.5
sc-coul                   = no
sc-power                  = 1
sc-sigma                  = 0.3
nstdhdl                   = 10
```

The `free_energy` keyword instructs `grompp` to read the relevant free energy settings. If this keyword is set to `no`, then all the options shown are ignored. The `init_lambda_state` determines the value of λ, which is actually a vector, as explained below. The `delta_lambda` keyword allows λ to change as a function of time, in what are called "slow-growth" free energy calculations. Such a feature will not be used in this tutorial. The `calc_lambda_neighbors` keyword will instruct `mdrun` on how many neighboring values of λ it should evaluate the energy of the system. Recall from Equation 8 that the energies of A- and B-states for a given configuration are needed. It is also possible to use multiple neighboring states (discussed below). For standard BAR, only a single neighbor needs its energy evaluated under the B-state topology.

The `couple-moltype` specifies the `[moleculetype]` name that will have its topology transformed as a function of λ. The `couple-lambda0` and `couple-lambda1` keywords specify which interactions are on in the A- and B-states, respectively. Valid options for these keywords are `vdw-q` (meaning both vdW and Coulombic interactions are fully on), `vdw` (only vdW), `q` (only Coulombic), and `none` (no intermolecular interactions; the coupled `[moleculetype]` is non-interacting, *i.e.* in the ideal gas state). The `couple-intramol` setting determines whether or not intramolecular nonbonded interactions are to be transformed as a function of λ. For small molecules, this option is typically set to `no` such that the fully transformed state still corresponds to the ideal gas state. Larger molecules may become trapped in a narrowly defined conformational ensemble, requiring softening of intramolecular nonbonded interactions, in which case `couple-moltype` might be set to `yes`. Methane has no intramolecular nonbonded interactions, so this setting will not affect the outcome, but for larger molecules it may be relevant.

From these options, the transformation of methane is now defined. The A-state corresponds to a molecule that interacts with water only via vdW interactions (`couple-moletype0 = vdw`). This setting is why it is irrelevant if charges are defined in the topology; even if present, they will be ignored. In the B-state, methane is non-interacting (`couple-moltype1 = none`), so the ΔG calculated in this series of simulations will correspond to the vdW contribution to $-\Delta G_{solv}$. ΔG_{solv} is formally defined as the free energy change associated with the introduction of a solute into solvent, which would be specified by inverting the definitions of the A- and B-states defined here.

The next set of options define the λ vector for nonbonded interactions (vdW and Coulombic) as well as bonded terms and restraints, masses of atoms, and the temperature of the system. Scaling temperatures is only relevant for simulated tempering [75, 76], which will not be discussed in this tutorial. These settings are used to define the λ vector, which is indexed with the `init_lambda_state` keyword. Vectors are zero-based indices, so λ[0] = (0.00, 0.00, 0.00, 0.00, 0.00, 0.00), λ[1] = (0.05, 0.00, 0.00, 0.00, 0.00, 0.00) and so on, corresponding to each column of values in the `*_lambdas` arrays. Though charges are not being transformed in this example, it is important to note that in such a case, charges should not be present on atoms that have no vdW terms to avoid numerical instability. That is, charges should only ever be "on" for atoms that have full vdW interactions. For this reason, setting `couple-moltype* = q` is very uncommon.

The options starting with `sc-` define parameters associated with "soft-core" modification of potentials [77]. As λ increasingly defines atoms with no vdW radii or very weak charges (in the absence of vdW terms), it is possible for atoms to overlap, leading to a numerical singularity (infinite force). While all λ-dependent transformations inherently sample unphysical states, a singularity makes calculation of the forces impossible and therefore must be prevented. To circumvent this problem, nonbonded potentials can be shifted, as shown in Equation 11 for the Lennard-Jones form of vdW interactions:

$$U_{ij}^{LJ}(r_{ij};\lambda) = 4\epsilon\lambda^p \left(\left[\alpha(1-\lambda)^p + \left(\frac{r_{ij}}{\sigma_{ij}} \right)^6 \right]^{-2} - \left[\alpha(1-\lambda)^p + \left(\frac{r_{ij}}{\sigma_{ij}} \right)^6 \right]^{-1} \right) \quad (11)$$

The `sc-alpha` keyword sets the value of α in Equation 11. The `sc-power` option sets the value of p; here, this parameter is set to 1 to indicate a linear dependence on λ, which has been shown to be a robust approach in conjunction with α = 0.5 [78–80]. The `sc-sigma` keyword specifies a value of σ for atoms that have force field parameter values of σ less than this specified value. This setting is important for atoms like hydrogen, which often do not have their own Lennard-Jones parameters. The final option, `sc-coul`, is not relevant here as charges are not being transformed. Typically, soft-core modifications are not applied to Coulombic interactions, if the transformation is being carried out on atoms that still have vdW interactions (the recommended approach). However, if one is transforming charges on atoms that do not have vdW interactions, the soft-core modification is recommended to avoid the same numerical singularity problems described above.

The last setting in this section of the `.mdp` file, `nstdhdl`, simply specifies the interval (in number of time steps) for writing the values of $\frac{\partial H(\lambda)}{\partial \lambda}$ to the output file.

https://doi.org/10.33011/livecoms.1.1.5068
Living J. Comp. Mol. Sci. 2019, 1(1), 5068

3.7.3 Workflow

The calculation of ΔG for removing vdW interactions between methane and water will be carried out in 21 individual simulations, one for each of the λ vectors defined in the `.mdp` files. Preceding each production simulation, the system will be energy-minimized, then equilibrated under an *NVT* and then an *NPT* ensemble for 100 ps each. Following these preparation steps, the production MD simulations (during which data are collected) will be carried out for 1 ns.

In this example, the values of λ are spaced equally from 0 to 1, thus an interval of 0.05. It is typical to cluster values of λ towards each of the end states, using a finer spacing to better resolve the energies of the fully and non-interacting states. For simplicity and in the interest of time, equal spacing is used here. As will be shown below, in this very simple case, such a λ spacing is adequate, but this may not always be the case.

The online tutorial provides a Bash script (`job.sh`) to run each of these jobs in a loop, though it is not necessary to do so. As all of the simulations are independent, it is possible to run them all simultaneously if preferred. All of the `.mdp` settings should be familiar by now, though it is important to note the use of the `sd` integrator (for "stochastic dynamics," more commonly called "Langevin dynamics"). A random frictional force is applied to all the atoms with an inverse friction coefficient specified by `tau_t`, which has been set to 1.0 ps^{-1}, to avoid over-damping of water and subsequently artificially impacting sampling. Do not use a value smaller than 1.0 here.

3.7.4 Analysis

Collect all of the output `md*.xvg` files from the production MD runs into one directory and execute the `bar` program:

```
$ gmx bar -f md*.xvg -o -oi
```

The `bar` program prints free energy differences for each λ window to the terminal (including entropy in each window, all in units of kT). At the end of the printed information will be the final results, in kJ mol^{-1}:

```
Final results in kJ/mol:

point      0 -      1,   DG  0.19 +/-  0.00
point      1 -      2,   DG  0.17 +/-  0.01
point      2 -      3,   DG  0.14 +/-  0.00
point      3 -      4,   DG  0.09 +/-  0.01
point      4 -      5,   DG  0.06 +/-  0.00
point      5 -      6,   DG  0.01 +/-  0.01
point      6 -      7,   DG -0.06 +/-  0.01
point      7 -      8,   DG -0.14 +/-  0.02
point      8 -      9,   DG -0.23 +/-  0.01
point      9 -     10,   DG -0.35 +/-  0.01
point     10 -     11,   DG -0.53 +/-  0.01
point     11 -     12,   DG -0.77 +/-  0.03
point     12 -     13,   DG -1.12 +/-  0.03
point     13 -     14,   DG -1.46 +/-  0.02
point     14 -     15,   DG -1.53 +/-  0.04
point     15 -     16,   DG -1.35 +/-  0.02
point     16 -     17,   DG -1.04 +/-  0.01
point     17 -     18,   DG -0.70 +/-  0.01
point     18 -     19,   DG -0.40 +/-  0.00
point     19 -     20,   DG -0.13 +/-  0.00

total      0 -     20,   DG -9.13 +/-  0.09
```

The transformation carried out here was the disappearance of vdW terms in methane, the opposite of the vdW contribution to the solvation free energy. Converting the obtained result to kcal mol^{-1} and inverting the sign (assuming reversibility of this process), the final result is 2.18 ± 0.02 kcal mol^{-1}, in good agreement with the value obtained by Shirts et al. of 2.21 ± 0.01 kcal mol^{-1} [74]. Small differences can be attributed to different software versions, cutoff schemes, integration algorithms, different λ spacing, and thermostats. However, the close agreement between these two results indicates that the present approach is robust and can reproduce the known quantity sufficiently.

The `-o` and `-oi` arguments produce `bar.xvg` and `barint.xvg`, the free energy differences as a function of λ state and the integral of these free energy differences (which gives the total ΔG for the transformation). It is also possible to output histograms of the free energy differences, but this output will not be discussed in this tutorial. The plot of ΔG as a function of λ is shown in Figure 33.

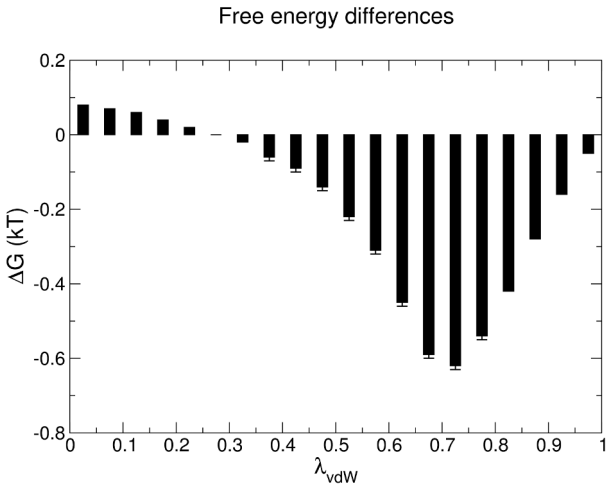

Figure 33. The ΔG values for each λ window, from the BAR algorithm.

Note that the default *x*-axis will be integers, corresponding to the indices of the λ vector. These values have been replaced in Figure 33 with the specific values of `vdw-lambdas`

https://doi.org/10.33011/livecoms.1.1.5068
Living J. Comp. Mol. Sci. 2019, 1(1), 5068

in each of these λ vectors. Recall that no other energy terms were transformed as a function of λ, so the free energy changes are all one-dimensional, and referred to simply as λ_{vdW} for clarity. Each value is plotted between two discrete values of λ. The BAR algorithm evaluates free energy differences between the neighboring λ values, so the ΔG values correspond to the free energy difference of a given configuration evaluated under the A- and B-state topologies (calculated every `nstdhdl` steps, see Section 3.7.2). The plotted BAR output corresponds to ΔG values between λ = 0 and λ = 0.05, λ = 0.05 and λ = 0.1, and so on.

The integral of these values is plotted in Figure 34, to accumulate ΔG as a function of λ. The final value is -3.69 k_BT, or -9.14 kJ mol^{-1}, since k_BT = 2.478 kJ mol^{-1} at 298 K. This value agrees with the value previously printed to the terminal by the `bar` program (-9.13 kJ mol^{-1}). The values are not exactly the same because the calculation done by the `bar` program is carried out with greater precision than using just two, rounded decimal places printed to the screen.

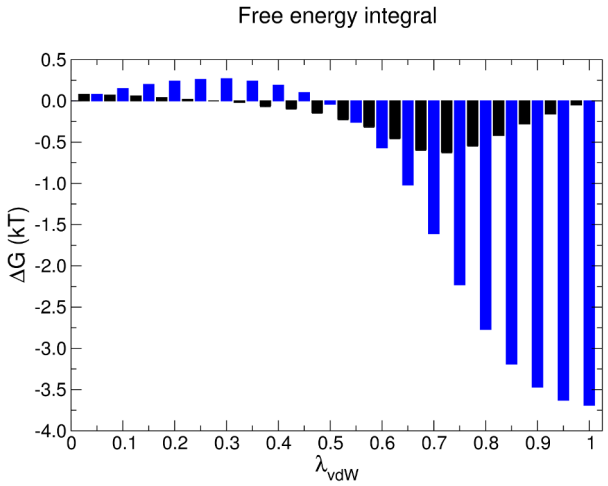

Figure 34. The cumulative ΔG value (blue), plotted along with ΔG values from each λ window (black), from the BAR algorithm.

3.7.5 More Advanced Free Energy Calculations

Common applications of free energy calculations via MD simulation include full solvation free energies and binding free energies. To obtain ΔG_{solv} for methane, the user would simply include charges in the system topology and set up a series of `coul-lambdas` in the `.mdp` files. The `bar` program can be then called in the same manner to obtain the total value of ΔG_{solv}.

Computing a binding free energy between a ligand and a protein is beyond the scope of this tutorial, but interested readers are directed to http://www.alchemistry.org/wiki/Main_Page, which has all the information necessary to perform these calculations. The most challenging aspect of these kinds of calculations is the need to restrain the ligand as it is slowly transformed. Non- or weakly interacting states of the ligand will begin to diffuse through the protein and sample totally unphysical regions of the system, inhibiting convergence of the calculations. To circumvent this problem, distance and orientational restraints can be imposed on the ligand as a function of λ. The energetic contributions of the restraints to the free energy can be corrected analytically after the simulations. An example of such a process is described by Jo et al. in their implementation of the CHARMM-GUI Ligand Binder [81]. See also references therein. The difference between the computed free energy difference from transforming the ligand in the binding site of the protein ($\Delta G_{complexation}$) and the solvation free energy of the ligand (ΔG_{solv}) yields the binding free energy, ΔG_{bind}.

3.7.6 Summary and Review of Objectives

This tutorial has guided the user through the process of calculating the free energy change associated with turning off vdW interactions between methane and water. The free energy change was computed with the BAR algorithm [73]. To review, the learning objectives of the tutorial are:

1. Understand the purpose of `.mdp` keywords related to free energy calculations
2. Use the Bennett Acceptance Ratio method to compute the free energy difference of the transformation of van der Waals terms

The user was introduced to settings related to free energy in Section 3.7.2 and analysis using BAR was performed in section 3.7.4. Several output files were plotted and their contents discussed. Interested readers may wish to refer to the Multistate BAR (MBAR) method [82], in which free energy differences are computed from all states. A Python implementation for MBAR is available from https://github.com/choderalab/pymbar and is fully compatible with GROMACS output. The user simply needs to increase the value of `calc_lambda_neighbors` in the `.mdp` file to evaluate free energy differences among all λ states.

3.8 Tutorial 7: Virtual Sites

In this tutorial (available online at http://www.mdtutorials.com/gmx/vsites), the user will be guided through the process of constructing a linear, triatomic molecule (CO_2) using two massive particles and converting the C and O atoms into virtual sites (Figure 35). A virtual site is a type of particle that has the following characteristics:

1. It has no mass
2. It may participate in Lennard-Jones and/or charge interactions

https://doi.org/10.33011/livecoms.1.1.5068
Living J. Comp. Mol. Sci. 2019, 1(1), 5068

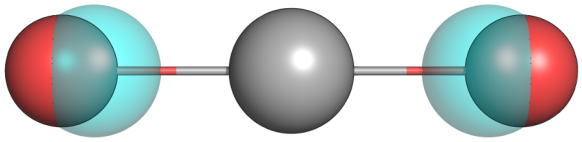

Figure 35. The CO_2 molecule, with two mass centers (cyan) that are used to construct virtual sites that replace the C and O atoms.

3. Any forces acting on virtual sites are projected onto their constructing atoms
4. Virtual site positions are not integrated, they are computed from updated positions of constructing atoms

In some literature, virtual sites are referred to as "dummy atoms," though this term is somewhat imprecise and can refer to generic, non-interacting particles. Be aware of this convention, as some articles are ambiguous in their usage.

Virtual sites have various applications in simulation literature. The first is to construct linear molecules, which present an algorithmic difficulty - the force discontinuity in evaluating the cosine of the angle *i* – *j* – *k* as it changes sign by fluctuating around 180°. It is this application that is explored in the present tutorial. Virtual sites are also used to define the positions of lone pairs on electronegative atoms, which exist as regions of electron density at predictable locations around these atoms. Examples include the modeling of the σ-hole on halogens in the OPLS3 [69] and CGenFF [83] parameter sets, lone pairs on sulfur atoms in OPLS-AA and OPLS/CM5 [84], in modeling hydrogen-bond acceptors in the Drude polarizable force field [85, 86], and in repartitioning hydrogen masses onto heavier atoms to increase the time step used in MD simulations [87, 88].

The procedure undertaken in this tutorial is quite different from all of the previous examples. There will be no use of `pdb2gmx` to write a topology, as it will be written manually by the user. There will be only one MD simulation that is performed in the absence of PBC to model the gas phase.

3.8.1 Create the Topology

To create the model for CO_2, the C and O atoms will be converted to virtual sites. They will participate in Lennard-Jones interactions and will have charges assigned to them, thereby interacting via normal Coulombic interactions with other particles. The forces acting on these virtual sites will be projected onto two mass centers, which have no charge or Lennard-Jones terms, serving only to receive the forces from the virtual sites and have their positions integrated during MD. The topology for CO_2 will be derived from the OPLS-AA force field. Charges are taken from work by Yang and Zhong [89], combined with standard OPLS-AA atom types. The model developed by doing so has not been rigorously tested in any way, but serves as a viable model for the illustrative purposes of this tutorial.

A complete, functional topology is available from the online version of the tutorial. Here, each section of the topology will be derived and explained.

The structure of CO_2 is shown in Figure 36, along with the nomenclature that will be used in this tutorial.

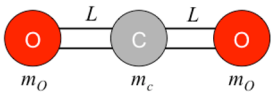

Figure 36. A schematic of the CO_2 molecule, with bond length, *L*, between each C and O atom. The masses of the C and O atoms are m_C and m_O, respectively.

Two properties have to be preserved in developing a virtual site model for CO_2: the total mass and the moment of inertia. Assigning masses to the two mass centers is straightforward. The mass of CO_2 is 44.0098 amu (1 carbon atom at 12.011 amu and 2 oxygen atoms at 15.9994 amu, according to the masses assigned in the OPLS-AA force field). Therefore, each mass center will be assigned half of this mass, or 22.0049 amu.

A new atom type, MCO, is introduced in the topology to describe the mass centers. It has no Lennard-Jones parameters associated with it. The new atom type definition and the atoms of the topology are defined as follows (note that the mass listed in `[atomtypes]` is irrelevant and is set in `[atoms]`.:

```
[ atomtypes ]
; name  bond_type mass  charge ptype    sigma   epsilon
  MCO   MCO       0.000 0.000  A        0.000   0.000

[ moleculetype ]
; name  nrexcl
CO2     2

[ atoms ]
; nr    type  resnr residue  atom cgnr    charge      mass
  1 opls_272      1     CO2     O1    1    -0.350    0.0000
  2 opls_271      1     CO2      C    1     0.700    0.0000
  3 opls_272      1     CO2     O2    1    -0.350    0.0000
  4      MCO      1     CO2     M1    1     0.000   22.0049
  5      MCO      1     CO2     M2    1     0.000   22.0049
```

A CO_2 molecule is essentially a linear, triatomic rotor, from which the moment of inertia, *I*, can be calculated as:

$$I = 2m_O L^2 \tag{12}$$

The value of L is the C=O bond length, 0.125 nm. Substituting in the value of m_O (15.9994 amu) yields a moment of inertia of 0.500 amu nm^2.

https://doi.org/10.33011/livecoms.1.1.5068
Living J. Comp. Mol. Sci. 2019, 1(1), 5068

Next, the distance between the mass centers (Figure 37) must be determined, such that the moment of inertia for the molecule is preserved.

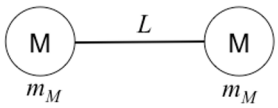

Figure 37. A schematic of the mass-carrying sites that will be used to convert CO_2 into a diatomic, rigid rotor.

The equation for *I* of a rigid rotor is:

$$I = \left(\frac{m_M m_M}{m_{Total}}\right) L^2 \tag{13}$$

Solving for *L* yields a value of 0.213173 nm, which is the length of the distance that will be assigned between the two mass centers. Rather than assigning this interaction as a normal harmonic bond (which would require the parametrization of a force constant, the physical relevance of which is not easily determined), the topology will assign a constraint such that the distance between the two mass centers is fixed. Doing so also makes it straightforward to construct the C and O virtual sites.

Thus, the constraint is defined in the topology as:

```
[ constraints ]
    4    5    1    0.213173
```

The C virtual site is assigned exactly in the middle of the two mass centers. In GROMACS, virtual sites can be constructed in a linear form by specifying the location of the virtual site as a fraction of the distance, *a*, between two constructing atoms. Therefore, the C virtual site will be exactly halfway between the two mass centers, or half of the assigned constraint length. Within a `[virtual_sites2]` directive:

```
[ virtual_sites2 ]
; site   ai   aj   funct    a
     2    4    5       1    0.5000
```

Last, the positions of the O virtual sites must be defined, again expressed as a fraction of the bond distance between the mass centers, M1 and M2. Since the C virtual site is 0.1065865 nm away from each M, and the C=O bond length is 0.125 nm, each O virtual site must be constructed 0.0184135 beyond the M1-M2 distance. This distance must be expressed as a fraction of the M1-M2 constraint length, therefore:

$$a = \frac{0.213173 + 0.0184135}{0.213173} = 1.0851116 \tag{14}$$

Now, the O virtual site constructions can be added to the `[virtual_sites2]` directive:

```
[ virtual_sites2 ]
; site   ai   aj   funct    a
     1    4    5       1    1.0851116
     2    4    5       1    0.5000
     3    5    4       1    1.0851116
```

Note that the order of the constructing atoms (M1 and M2) in this directive specifies the direction of the vector along which the O virtual sites are constructed. O site 1 is constructed beyond M2 (outside of the M1-M2 constraint) and O site 3 is constructed beyond M1 (outside the M2-M1 constraint).

3.8.2 Simple MD Simulation

To validate the approach, a small gas-phase system of CO_2 will be constructed. Place 10 molecules in a large box:

```
$ gmx insert-molecules -ci co2.pdb -nmol 10
    -box 10 10 10 -o box.pdb
```

Energy minimization and the subsequent MD simulation will be carried out without periodicity and "infinite" cutoffs, to mimic a gas-phase system. The relevant `.mdp` options in both `em.mdp` and `md.mdp` are:

```
cutoff-scheme    = group
nstlist          = 0
ns_type          = simple
rlist            = 0
coulombtype      = cutoff
rcoulomb         = 0
rvdw             = 0
pbc              = no
```

During MD, angular momentum is also conserved:

```
comm-mode        = angular
```

Setting `nstlist` to zero means the neighbor list is fixed. All interatomic interactions are considered in this simulation. This approach differs from the truncation schemes utilized in condensed-phase systems that have far more atoms, making this approach intractable if there are many atoms. For gas-phase systems, where there are few atoms and therefore few interactions, it is possible (and physically meaningful) to use the settings listed above.

Perform energy minimization:

```
$ gmx grompp -f em.mdp -c box.pdb -p
    topol.top -o em.tpr

$ gmx mdrun -nt 1 -nb cpu -deffnm em
```

Note that the use of the group cutoff scheme prevents the use of a GPU, hence the use of `-nb cpu` to tell `mdrun` to only use CPU hardware. Only one thread is started since the system

https://doi.org/10.33011/livecoms.1.1.5068
Living J. Comp. Mol. Sci. 2019, 1(1), 5068

only contains 50 particles. It will not benefit from parallelization.

Continue with MD:

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

$ gmx mdrun -nt 1 -nb cpu -deffnm md
```

When the simulation is finished, analyze the moments of inertia of CO_2 to verify that the approach worked. First, create an index file by splitting all of the CO_2 molecules into their own groups so they can be analyzed individually:

```
$ gmx make_ndx -f em.gro
> splitres 0
> q
```

Use the GROMACS `principal` program to compute the moments of inertia. Choose any of the CO_2 molecules from the index file when prompted.

```
$ gmx principal -s md.tpr -f md.trr -n
    index.ndx
```

The output file, `moi.xvg`, contains moments of inertia around the *x*-, *y*-, and *z*-axes. Rotation around *x* is zero, because this axis coincides with the M1-M2 constraint. The *y*- and *z*- moments are 0.500, in exact agreement with the value computed above. This outcome indicates that the approach taken to convert a triatomic molecule into a rigid, diatomic rotor is physically reasonable.

3.8.3 Summary and Review of Objectives

In this tutorial, the concept of virtual sites was introduced by converting a triatomic molecule, CO_2, into a linear, rigid rotor. Two mass centers were constructed using a rigid constraint, from which three virtual sites, representing C and O atoms, were constructed. To review, the learning objectives of the tutorial are:

1. Define a virtual site and its position relative to other atoms
2. Construct a system of a linear molecule with virtual sites

The concept of virtual sites was introduced in Section 3.8 and demonstrated in Section 3.8.1 through the manual construction of a topology file and coordinates. The utility and accuracy of the approach was demonstrated with a simple, gas-phase MD simulation, in which the moments of inertia of CO_2 molecules were calculated.

4 Author Contributions

JAL conceived of and wrote all the online tutorials, composed the underlying HTML for http://www.mdtutorials.com/gmx, and wrote this article.

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

5 Other Contributions

JAL thanks the many users of the GROMACS tutorials, who have provided important feedback and requests since the initial publication of the lysozyme tutorial in 2008, Dr. David R. Bevan for encouraging the development of these tutorials and hosting them on his lab web server for many years, and Dr. Anne M. Brown for providing technical assistance in maintaining tutorial accessibility while the tutorials were hosted on the Bevan lab website. JAL also thanks Dr. Michael Shirts for critical evaluation of Tutorial 6, "Free Energy of Solvation," prior to its online publication, and members of the Lemkul lab for critical review of this manuscript.

For a more detailed description of contributions from the community and others, see the GitHub issue tracking and changelog at https://github.com/jalemkul/gmx_tutorials_livecoms.

6 Potentially Conflicting Interests

JAL is a developer of the GROMACS simulation package.

7 Funding Information

JAL thanks the NSF MILES-IGERT and Virginia Tech ICTAS Doctoral Scholars programs for financial support from 2007-2012, during which time the original tutorials were created. JAL is supported by startup funding from the Virginia Tech Office of the Provost, College of Agriculture and Life Sciences, and Department of Biochemistry and USDA National Institute of Food and Agriculture.

Author Information

ORCID:

Justin A. Lemkul: 0000-0001-6661-8653

References

[1] **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.

[2] **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.

[3] **Frenkel D**, Smit B. Understanding Molecular Simulation: From Algorithms to Applications. 2nd ed. Computational Science Series, Vol. 1, Academic Press; 2001.

[4] **Leach A**. Molecular Modelling: Principles and Applications. 2nd ed. Pearson; 2001.

[5] Kukol A, editor. Molecular Modeling of Proteins, vol. 443 of Methods in Molecular Biology. Humana Press; 2008.

[6] **Braun E**, Gilmer J, Mayes HB, Mobley DL, Monroe JI, Prasad S, Zuckerman DM. Best Practices for Foundations in Molecular Simulations [Article v1.0]. Living Journal of Computational Molecular Science. 2019; 1(1):5957. https://doi.org/10.33011/livecoms.1.1.5957.

[7] **Humphrey W**, Dalke A, Schulten K. VMD: Visual molecular dynamics. Journal of Molecular Graphics. 1996; 14(1):33–38. https://doi.org/10.1016/0263-7855(96)00018-5.

[8] **Pettersen EF**, Goddard TD, Huang CC, Couch GS, Greenblatt DM, Meng EC, Ferrin TE. UCSF Chimera – A visualization system for exploratory research and analysis. Journal of Computational Chemistry. 2004; 25(13):1605–1612. https://doi.org/10.1002/jcc.20084.

[9] **Hanwell MD**, Curtis DE, Lonie DC, Vandermeersch T, Zurek E, Hutchison GR. Avogadro: an advanced semantic chemical editor, visualization, and analysis platform. Journal of Cheminformatics. 2012; 4(1):17. https://doi.org/10.1186/1758-2946-4-17.

[10] **Artymiuk PJ**, Blake CCF, Rice DW, S Wilson K. The Structures of the Monoclinic and Orthorhombic Forms of Hen Egg-White Lysozyme at 6 Angstroms Resolution. Acta Crystallographica B. 1982; 38:778–783. https://doi.org/10.1107/S0567740882004075.

[11] **Cornell WD**, Cieplak P, Bayly CI, Gould IR, Merz KM, Ferguson DM, Spellmeyer DC, Fox T, Caldwell JW, Kollman PA. A Second Generation Force Field for the Simulation of Proteins, Nucleic Acids, and Organic Molecules. Journal of the American Chemical Society. 1995; 117(19):5179–5197. https://doi.org/10.1021/ja00124a002.

[12] **Kollman PA**. Advances and Continuing Challenges in Achieving Realistic and Predictive Simulations of the Properties of Organic and Biological Molecules. Accounts of Chemical Research. 1996; 29(10):461–469. https://doi.org/10.1021/ar9500675.

[13] **Wang J**, Cieplak P, Kollman PA. How well does a restrained electrostatic potential (RESP) model perform in calculating conformational energies of organic and biological molecules? Journal of Computational Chemistry. 2000; 21(12):1049. https://doi.org/10.1002/1096-987x(200009)21:12<1049::aid-jcc3>3.3.co;2-6.

[14] **García AE**, Sanbonmatsu KY. α-Helical stabilization by side chain shielding of backbone hydrogen bonds. Proceedings of the National Academy of Sciences. 2002; 99(5):2782–2787. https://doi.org/10.1073/pnas.042496899.

[15] **Duan Y**, Wu C, Chowdhury S, Lee MC, Xiong G, Zhang W, Yang R, Cieplak P, Luo R, Lee T, Caldwell J, Wang J, Kollman P. A point-charge force field for molecular mechanics simulations of proteins based on condensed-phase quantum mechanical calculations. Journal of Computational Chemistry. 2003; 24(16):1999–2012. https://doi.org/10.1002/jcc.10349.

[16] **Hornak V**, Abel R, Okur A, Strockbine B, Roitberg A, Simmerling C. Comparison of multiple Amber force fields and development of improved protein backbone parameters. Proteins: Structure, Function, and Bioinformatics. 2006; 65(3):712–725. https://doi.org/10.1002/prot.21123.

[17] **Lindorff-Larsen K**, Piana S, Palmo K, Maragakis P, Klepeis JL, Dror RO, Shaw DE. Improved side-chain torsion potentials for the Amber ff99SB protein force field. Proteins: Structure, Function, and Bioinformatics. 2010; 78:1950–1958. https://doi.org/10.1002/prot.22711.

[18] **MacKerell AD Jr**, Bashford D, Bellott M, Dunbrack RL Jr, Evanseck JD, Field MJ, Fischer S, Gao J, Guo H, Ha S, Joseph-McCarthy D, Kuchnir L, Kuczera K, Lau FTK, Mattos C, Michnick S, Ngo T, Nguyen DT, Prodhom B, Reiher WE III, et al. All-Atom Empirical Potential for Molecular Modeling and Dynamics Studies of Proteins. Journal of Physical Chemistry B. 1998; 102(18):3586–2616. https://doi.org/10.1021/jp973084f.

[19] **Mackerell AD Jr**, Feig M, Brooks CL III. Extending the treatment of backbone energetics in protein force fields: Limitations of gas-phase quantum mechanics in reproducing protein conformational distributions in molecular dynamics simulations. Journal of Computational Chemistry. 2004; 25(11):1400–1415. https://doi.org/10.1002/jcc.20065.

[20] **Foloppe N**, MacKerell AD Jr. All-atom empirical force field for nucleic acids: I. Parameter optimization based on small molecule and condensed phase macromolecular target data. Journal of Computational Chemistry. 2000; 21(2):86–104. https://doi.org/10.1002/(sici)1096-987x(20000130)21:2<86::aid-jcc2>3.0.co;2-g.

[21] **MacKerell AD Jr**, Banavali NK. All-atom empirical force field for nucleic acids: II. Application to molecular dynamics simulations of DNA and RNA in solution. Journal of Computational Chemistry. 2000; 21(2):105–120. https://doi.org/10.1002/(sici)1096-987x(20000130)21:2<105::aid-jcc3>3.0.co;2-p.

[22] **Daura X**, Mark AE, Van Gunsteren WF. Parametrization of aliphatic CH_n united atoms of GROMOS96 force field. Journal of Computational Chemistry. 1998; 19(5):535–547. https://doi.org/10.1002/(sici)1096-987x(19980415)19:5<535::aid-jcc6>3.0.co;2-n.

[23] **Schuler LD**, Daura X, van Gunsteren WF. An improved GROMOS96 force field for aliphatic hydrocarbons in the condensed phase. Journal of Computational Chemistry. 2001; 22(11):1205–1218. https://doi.org/10.1002/jcc.1078.abs.

[24] **Oostenbrink C**, Villa A, Mark AE, Van Gunsteren WF. A biomolecular force field based on the free enthalpy of hydration and solvation: The GROMOS force-field parameter sets 53A5 and 53A6. Journal of Computational Chemistry. 2004; 25(13):1656–1676. https://doi.org/10.1002/jcc.20090.

[25] **Schmid N**, Eichenberger AP, Choutko A, Riniker S, Winger M, Mark AE, van Gunsteren WF. Definition and testing of the GROMOS force-field versions 54A7 and 54B7. European Biophysics Journal. 2011; 40(7):843–856. https://doi.org/10.1007/s00249-011-0700-9.

[26] **Kaminski GA**, Friesner RA, Tirado-Rives J, Jorgensen WL. Evaluation and Reparametrization of the OPLS-AA Force Field for Proteins via Comparison with Accurate Quantum Chemical Calculations on Peptides. The Journal of Physical Chemistry B. 2001; 105(28):6474–6487. https://doi.org/10.1021/jp003919d.

[27] **Best RB**, Zhu X, Shim J, Lopes PEM, Mittal J, Feig M, MacKerell AD Jr. Optimization of the Additive CHARMM All-Atom Protein Force Field Targeting Improved Sampling of the Backbone ϕ, ψ and Side-Chain χ_1 and χ_2 Dihedral Angles. Journal of Chemical Theory and Computation. 2012; 8(9):3257–3273. https://doi.org/10.1021/ct300400x.

[28] **Hart K**, Foloppe N, Baker CM, Denning EJ, Nilsson L, MacKerell AD Jr. Optimization of the CHARMM Additive Force Field for DNA: Improved Treatment of the BI/BII Conformational Equilibrium. Journal of Chemical Theory and Computation. 2012; 8:348–362. https://doi.org/10.1021/ct200723.

[29] **Denning EJ**, Priyakumar UD, Nilsson L, Mackerell AD. Impact of 2'-hydroxyl sampling on the conformational properties of RNA: Update of the CHARMM all-atom additive force field for RNA. Journal of Computational Chemistry. 2011; 32(9):1929–1943. https://doi.org/10.1002/jcc.21777.

[30] **Klauda JB**, Venable RM, Freites JA, O'Connor JW, Tobias DJ, Mondragon-Ramirez C, Vorobyov I, MacKerell AD Jr, Pastor RW. Update of the CHARMM All-Atom Additive Force Field for Lipids: Validation on Six Lipid Types. The Journal of Physical Chemistry B. 2010; 114(23):7830–7843. https://doi.org/10.1021/jp101759q.

[31] **Huang J**, Rauscher S, Nawrocki G, Ran T, Feig M, de Groot BL, Grubmüller H, MacKerell AD Jr. CHARMM36m: an improved force field for folded and intrinsically disordered proteins. Nature Methods. 2017; 14(1):71–73. https://doi.org/10.1038/nmeth.4067.

[32] **Berendsen HJC**, Grigera JR, Straatsma TP. The missing term in effective pair potentials. The Journal of Physical Chemistry. 1987; 91(24):6269–6271. https://doi.org/10.1021/j100308a038.

[33] **Jorgensen WL**, Chandrasekhar J, Madura JD, Impey RW, Klein ML. Comparison of simple potential functions for simulating liquid water. Journal of Chemical Physics. 1983; 79(2):926–935. https://doi.org/10.1063/1.445869.

[34] **Hess B**, van der Vegt NFA. Hydration Thermodynamic Properties of Amino Acid Analogues: A Systematic Comparison of Biomolecular Force Fields and Water Models. The Journal of Physical Chemistry B. 2006; 110(35):17616–17626. https://doi.org/10.1021/jp0641029.

[35] **Šali A**, Blundell TL. Comparative Protein Modelling by Satisfaction of Spatial Restraints. Journal of Molecular Biology. 1993; 234(3):779–815. https://doi.org/10.1006/jmbi.1993.1626.

[36] **Berendsen HJC**, Postma JPM, van Gunsteren WF, Hermans J. Intermolecular Forces. Pullman B, editor; 1981.

[37] **Darden T**, York D, Pedersen L. Particle mesh Ewald: An N·log(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.

[38] **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.

[39] **Hub JS**, de Groot BL, Grubmüller H, Groenhof G. Quantifying Artifacts in Ewald Simulations of Inhomogeneous Systems with a Net Charge. Journal of Chemical Theory and Computation. 2014; 10(1):381–390. https://doi.org/10.1021/ct400626b.

[40] **Hess B**, Bekker H, Berendsen HJC, Fraaije JGEM. LINCS: A linear constraint solver for molecular simulations. Journal of Computational Chemistry. 1997; 18(12):1463–1472. https://doi.org/10.1002/(sici)1096-987x(199709)18:12<1463::aid-jcc4>3.3.co;2-l.

[41] **Hess B**. P-LINCS: A Parallel Linear Constraint Solver for Molecular Simulation. Journal of Chemical Theory and Computation. 2008; 4(1):116–122. https://doi.org/10.1021/ct700200b.

[42] **Bussi G**, Donadio D, Parrinello M. Canonical sampling through velocity rescaling. The Journal of Chemical Physics. 2007; 126(1):014101. https://doi.org/10.1063/1.2408420.

[43] **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.

[44] **Lingenheil M**, Denschlag R, Reichold R, Tavan P. The "Hot-Solvent/Cold-Solute" Problem Revisited. Journal of Chemical Theory and Computation. 2008; 4(8):1293–1306. https://doi.org/10.1021/ct8000365.

[45] **Parrinello M**, Rahman A. Polymorphic transitions in single crystals: A new molecular dynamics method. Journal of Applied Physics. 1981; 52(12):7182–7190. https://doi.org/10.1063/1.328693.

[46] **Nosé S**, Klein ML. Constant pressure molecular dynamics for molecular systems. Molecular Physics. 1983; 50(5):1055–1076. https://doi.org/10.1080/00268978300102851.

[47] **Kandasamy SK**, Larson RG. Molecular Dynamics Simulations of Model Trans-Membrane Peptides in Lipid Bilayers: A Systematic Investigation of Hydrophobic Mismatch. Biophysical Journal. 2006; 90(7):2326–2343. https://doi.org/10.1529/biophysj.105.073395.

[48] **Brooks BR**, Brooks CL III, MacKerell AD Jr, Nilsson L, Petrella RJ, Roux B, Won Y, Archontis G, Bartels C, Boresch S, Caflisch A, Caves L, Cui Q, Dinner AR, Feig M, Fischer S, Gao J, Hodoscek M, Im W, Kuczera K, et al. CHARMM: The Biomolecular Simulation Program. Journal of Computational Chemistry. 2009; 30(10):1545–1614. https://doi.org/10.1002/jcc.21287.

[49] **Gelin BR**, Karplus M. Side-chain torsional potentials: effect of dipeptide, protein, and solvent environment. Biochemistry. 1979; 18(7):1256–1268. https://doi.org/10.1021/bi00574a022.

[50] **Weiner SJ**, Kollman PA, Case DA, Singh UC, Ghio C, Alagona G, Profeta S, Weiner P. A new force field for molecular mechanical simulation of nucleic acids and proteins. Journal of the American Chemical Society. 1984; 106(3):765–784. https://doi.org/10.1021/ja00315a051.

[51] **Jorgensen WL**, Madura JD, Swenson CJ. Optimized Intermolecular Potential Functions for Liquid Hydrocarbons. Journal of the American Chemical Society. 1984; 106(22):6638–6646. https://doi.org/10.1021/ja00334a030.

[52] **Kukol A**, Kukol A. Lipid models for united-atom molecular dynamics simulations of protein. Nature Precedings. 2011; 5(3):615–626. https://doi.org/10.1038/npre.2011.5977.1.

[53] **Berger O**, Edholm O, Jähnig F. Molecular dynamics simulations of a fluid bilayer of dipalmitoylphosphatidylcholine at full hydration, constant pressure, and constant temperature. Biophysical Journal. 1997; 72(5):2002–2013. https://doi.org/10.1016/s0006-3495(97)78845-3.

[54] **Piggot TJ**, Piñiero Á, Khalid S. Molecular Dynamics Simulations of Phosphatidylcholine Membranes: A Comparative Force Field Study. Journal of Chemical Theory and Computation. 2012; 8(11):4593–4609. https://doi.org/10.1021/ct3003157.

[55] **Kandt C**, Ash WL, Peter Tieleman D. Setting up and running molecular dynamics simulations of membrane proteins. Methods. 2007; 41(4):475–488. https://doi.org/10.1016/j.ymeth.2006.08.006.

[56] **Allen WJ**, Lemkul JA, Bevan DR. GridMAT-MD: A grid-based membrane analysis tool for use with molecular dynamics. Journal of Computational Chemistry. 2009; 30(12):1952–1958. https://doi.org/10.1002/jcc.21172.

[57] **Allen MP**, Tildesley DJ. Computer Simulations of Liquids. Oxford Science Publications; 1987.

[58] **Yeh IC**, Hummer G. System-Size Dependence of Diffusion Coefficients and Viscosities from Molecular Dynamics Simulations with Periodic Boundary Conditions. The Journal of Physical Chemistry B. 2004; 108(40):15873–15879. https://doi.org/10.1021/jp0477147.

[59] **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.

[60] **Lührs 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 of the United States of America. 2005; 102(48):17342–17347. https://doi.org/10.1073/pnas.0506723102.

[61] **Kumar S**, Bouzida D, Swendsen RH, Kollman PA, Rosenberg JM. 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.

[62] **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.

[63] **Lemkul JA**, Allen WJ, Bevan DR. Practical Considerations for Building GROMOS-Compatible Small-Molecule Topologies. Journal of Chemical Information and Modeling. 2010; 50(12):2221–2235. https://doi.org/10.1021/ci100335w.

[64] **van Aalten DMF**, Bywater R, Findlay JBC, Hendlich M, Hooft RWW, Vriend G. PRODRG, a program for generating molecular topologies and unique molecular descriptors from coordinates of small molecules. Journal of Computer-Aided Molecular Design. 1996; 10(3):255–262. https://doi.org/10.1007/bf00355047.

[65] **Malde AK**, Zuo L, Breeze M, Stroet M, Poger D, Nair PC, Oostenbrink C, Mark AE. An Automated Force Field Topology Builder (ATB) and Repository: Version 1.0. Journal of Chemical Theory and Computation. 2011; 7(12):4026–4037. https://doi.org/10.1021/ct200196m.

[66] **Boyce SE**, Mobley DL, Rocklin GJ, Graves AP, Dill KA, Shoichet BK. Predicting Ligand Binding Affinity with Alchemical Free Energy Methods in a Polar Binding Site. Journal of Molecular Biology. 2009; 294(4):747–763. https://doi.org/10.1016/j.jmb.2009.09.049.

[67] **Wang J**, Wolf RM, Caldwell JW, Kollman PA, Case DA. Development and testing of a general amber force field. Journal of Computational Chemistry. 2004; 25(9):1157–1174. https://doi.org/10.1002/jcc.20035.

[68] **Vanommeslaeghe K**, Hatcher E, Acharya C, Kundu S, Zhong S, Shim J, Darian E, Guvench O, Lopes P, Vorobyov I, MacKerell AD Jr. CHARMM General Force Field: A Force Field for Drug-Like Molecules Compatible with the CHARMM All-Atom Additive Biological Force Fields. Journal of Computational Chemistry. 2010; 31(4):671–690. https://doi.org/10.1002/jcc.21367.

[69] **Harder E**, Damm W, Maple J, Wu C, Reboul M, Xiang JY, Wang L, Lupyan D, Dahlgren MK, Knight JL, Kaus JW, Cerutti DS, Krilov G, Jorgensen WL, Abel R, Friesner RA. OPLS3: A Force Field Providing Broad Coverage of Drug-like Small Molecules and Proteins. Journal of Chemical Theory and Computation. 2016; 12(1):281–296. https://doi.org/10.1021/acs.jctc.5b00864.

[70] **Vanommeslaeghe K**, MacKerell AD Jr. Automation of the CHARMM General Force Field (CGenFF) I: Bond Perception and Atom Typing. Journal of Chemical Information and Modeling. 2012; 52(12):3144–3154. https://doi.org/10.1021/ci300363c.

[71] **Vanommeslaeghe K**, Raman EP, MacKerell AD Jr. Automation of the CHARMM General Force Field (CGenFF) II: Assignment of Bonded Parameters and Partial Atomic Charges. Journal of Chemical Information and Modeling. 2012; 52(12):3155–3168. https://doi.org/10.1021/ci3003649.

[72] **Zwanzig RW**. High-Temperature Equation of State by a Perturbation Method. I. Nonpolar Gases. Journal of Chemical Physics. 1954; 22(8):1420. https://doi.org/10.1063/1.1740409.

[73] **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.

[74] **Shirts MR**, Pitera 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.

[75] **Marinari E**, Parisi G. Simulated Tempering: A New Monte Carlo Scheme. Europhysics Letters (EPL). 1992; 19(6):451–458. https://doi.org/10.1209/0295-5075/19/6/002.

[76] **Geyer CJ**, Thompson EA. Annealing Markov Chain Monte Carlo with Applications to Ancestral Inference. Journal of the American Statistical Association. 1995; 90(431):909. https://doi.org/10.2307/2291325.

[77] **Beutler TC**, Mark AE, van Schaik RC, Gerber PR, van Gunsteren WF. Avoiding singularities and numerical instabilities in free energy calculations based on molecular simulations. Chemical Physics Letters. 1994; 222(6):529–539. https://doi.org/10.1016/0009-2614(94)00397-1.

[78] **Pitera JW**, van Gunsteren WF. A Comparison of Non-Bonded Scaling Approaches for Free Energy Calculations. Molecular Simulation. 2002; 28(1-2):45–65. https://doi.org/10.1080/08927020211973.

[79] **Steinbrecher T**, Mobley DL, Case DA. Nonlinear scaling schemes for Lennard-Jones interactions in free energy calculations. The Journal of Chemical Physics. 2007; 127(21):214108. https://doi.org/10.1063/1.2799191.

[80] **Shirts MR**, Pande VS. Solvation free energies of amino acid side chain analogs for common molecular mechanics water models. The Journal of Chemical Physics. 2005; 122(13):134508. https://doi.org/10.1063/1.1877132.

[81] **Jo S**, Jiang W, Lee HS, Roux B, Im W. CHARMM-GUI Ligand Binder for Absolute Binding Free Energy Calculations and Its Application. Journal of Chemical Information and Modeling. 2013; 53(1):267–277. https://doi.org/10.1021/ci300505n.

[82] **Shirts MR**, Chodera JD. Statistically optimal analysis of samples from multiple equilibrium states. The Journal of Chemical Physics. 2008; 129(12):124105. https://doi.org/10.1063/1.2978177.

[83] **Soteras Gutiérrez I**, Lin FY, Vanommeslaeghe K, Lemkul JA, Armacost KA, Brooks CL III, MacKerell AD Jr. Parametrization of halogen bonds in the CHARMM general force field: Improved treatment of ligand–protein interactions. Bioorganic & Medicinal Chemistry. 2016; 24(20):4812–4825. https://doi.org/10.1016/j.bmc.2016.06.034.

[84] **Yan XC**, Robertson MJ, Tirado-Rives J, Jorgensen WL. Improved Description of Sulfur Charge Anisotropy in OPLS Force Fields: Model Development and Parameterization. The Journal of Physical Chemistry B. 2017; 121(27):6626–6636. https://doi.org/10.1021/acs.jpcb.7b04233.

[85] **Harder E**, Anisimov VM, Vorobyov IV, Lopes PEM, Noskov SY, MacKerell AD Jr, Roux B. Atomic Level Anisotropy in the Electrostatic Modeling of Lone Pairs for a Polarizable Force Field Based on the Classical Drude Oscillator. Journal of Chemical Theory and Computation. 2006; 2(6):1587–1597. https://doi.org/10.1021/ct600180x.

[86] **Lemkul JA**, Huang J, Roux B, MacKerell AD Jr. An Empirical Polarizable Force Field Based on the Classical Drude Oscillator Model: Development History and Recent Applications. Chemical Reviews. 2016; 116(9):4983–5013. https://doi.org/10.1021/acs.chemrev.5b00505.

[87] **Feenstra KA**, Hess B, Berendsen HJC. Improving efficiency of large time-scale molecular dynamics simulations of hydrogen-rich systems. Journal of Computational Chemistry. 1999; 20(8):786–798. https://doi.org/10.1002/(sici)1096-987x(199906)20:8<786::aid-jcc5>3.0.co;2-b.

[88] **Bjelkmar P**, Larsson P, Cuendet MA, Hess B, Lindahl E. Implementation of the CHARMM Force Field in GROMACS: Analysis of Protein Stability Effects from Correction Maps, Virtual Interaction Sites, and Water Models. Journal of Chemical Theory and Computation. 2010; 6(2):459–466. https://doi.org/10.1021/ct900549r.

[89] **Yang Q**, Zhong C. Molecular Simulation of Carbon Dioxide/Methane/Hydrogen Mixture Adsorption in Metal-Organic Frameworks. The Journal of Physical Chemistry B. 2006; 110(36):17776–17783. https://doi.org/10.1021/jp062723w.