A Suite of Advanced Tutorials for the GROMOS Biomolecular Simulation Software [Article v2.0]

Bettina Lier[1], Benedict Braunsfeld[1], Radek Crha[1,2], Oriol Gracia Carmona[1], Julia Gebhardt[3], Christoph Öhlknecht[1], Peter Poliak[1,4], Peter Fraško[1], Anita de Ruiter[1], Marcelle B. M. Spera[3], Michael Gillhofer[1,2], Wilfred F. van Gunsteren[5], Chris Oostenbrink[1*,2], Niels Hansen[3*]

[1]Institute of Molecular Modeling and Simulation, University of Natural Resources and Life Sciences, Vienna, Austria; [2]Christian Doppler Laboratory for Molecular Informatics in the Biosciences, University of Natural Resources and Life Sciences, Vienna, Austria; [3]Institute of Thermodynamics and Thermal Process Engineering, University of Stuttgart, Stuttgart, Germany; [4]Institute of Physical Chemistry and Chemical Physics, Slovak University of Technology, Bratislava, Slovakia; [5]Institute of Molecular Physical Science, Swiss Federal Institute of Technology, ETH, Zürich, Switzerland

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

This version dated September 15, 2025

Abstract This tutorial describes the practical use of some recent methodological advances implemented in the GROMOS software for biomolecular simulations. It is envisioned as a living document, with additional tutorials being added in the course of time. Currently, it consists of six distinct tutorials. The first tutorial describes the use of time-averaged restraints to enforce agreement with order parameters derived from NMR experiments. The second tutorial describes the use of extended thermodynamic integration in the double-decoupling method to compute the affinity of a small molecule to a protein. The molecule involved bears a negative charge, necessitating the application of post-simulation corrections. The third tutorial is based on the same molecular system, but computes the binding free energy from a path-sampling method with distance-field distance restraints and Hamiltonian replica exchange simulations. The fourth tutorial describes the use of Gaussian accelerated MD, an enhanced sampling technique. The fifth tutorial is about the use of Accelerated Enveloping Distribution Sampling (AEDS), an enhanced version of EDS. The sixth tutorial illustrates how to run QM/MM simulations with the Buffer Region Neural Network (BuRNN) method. The tutorials are written for users with some experience in the application of molecular dynamics simulations.

***For correspondence:**
chris.oostenbrink@boku.ac.at (CO); niels.hansen@itt.uni-stuttgart.de (NH)

1 Introduction

GROMOS^{TM} is an acronym for the GROningen MOlecular Simulation computer program package for the dynamic

Received: December 18, 2024
Accepted: September 4, 2025

https://doi.org/10.33011/livecoms.6.1.3485
Living J. Comp. Mol. Sci. 2025, 6(1), 3485

modeling of (bio)molecules, which has been developed since 1978 primarily as a research vehicle for methodological development [1]. Written in the programming language C++, the latest version has a modular, object-oriented structure [2], which, together with extensive documentation [3], makes modification relatively easy. Readability of the code is prioritised over speed. The GROMOS code is freely available at www.gromos.net. The GROMOS software is to be distinguished from the GROMOS force fields for biomolecular systems. The development of the successive GROMOS force-field versions during the past 40 years has been summarised in [1, 4]. Recent work showed that time-saving approximations employed during force-field development had no effect on the parametrization in terms of agreement with experiment [5, 6]. The GROMOS software comes with a manual that consists of nine volumes. Volume 7 is a basic tutorial that introduces new users to the setup and analyses of molecular simulations with GROMOS [7]. The set of tutorials presented here is intended to build on these original tutorials released with GROMOS.

1.1 Scope

The six tutorials presented here cover some of the methodological advances that have been implemented in GROMOS over the last few years and are not treated in the basic tutorials distributed with the software. They address an advanced user who has some experience with MD simulations. Beginners in the field are advised to start with the basic tutorial of GROMOS [7]. Each of the current tutorials is based on an original publication and comes with its own learning objectives and expected outcome(s). After completing tutorial 1 "S2 order parameter restraining" the user should be able to

1. Prepare a simulation of a protein solvated in water.
2. Understand how NMR restraints are handled in GROMOS.

After completing tutorial 2 "Double decoupling method & corrections for net-charge changes" the user should be able to

1. Prepare perturbation topologies for binding free energy calculations.
2. Define distance restraints and perturbed distance restraints for simulations in GROMOS.
3. Calculate binding free energies using the double decoupling method and extended-thermodynamic integration.
4. Apply a post-simulation correction scheme to correct artifacted free-energies obtained from charge-changing perturbations.

After completing tutorial 3 "Using HREMD and distance-field" the user should be able to

1. Set up a distance-field restraining potential-energy term.
2. Perform umbrella sampling calculations in GROMOS using perturbed distance(-field) restraints.
3. Extract the binding free energy from the potential of mean force.

After completing tutorial 4 "Selective Gaussian accelerated MD (GaMD)" the user should be able to

1. Prepare GaMD acceleration input files.
2. Perform GaMD parameter searches in GROMOS.
3. Perform GaMD simulations in GROMOS.
4. Extract, reweight and analyze GaMD simulations.

After completing tutorial 5 "Accelerated Enveloping Distribution Sampling (AEDS)" the user should be able to

1. Prepare AEDS input files and perturbation topologies.
2. Perform AEDS parameter searches in GROMOS.
3. Perform AEDS simulations in GROMOS.
4. Extract, reweight and analyze AEDS simulations as a free energy method or as a chemostat.

After completing tutorial 6 "NN(QM)/MM simulations with the BuRNN approach" the user should be able to

1. Prepare QM input data and run QM calculations in MOPAC.
2. Prepare a training database and train neural networks using SchNet.
3. Run BuRNN simulations in GROMOS.

To support an efficient and accessible learning experience, the tutorials are based on rather simple molecular systems. These systems are deliberately chosen to minimize computational costs and allow users to focus on understanding core concepts and methods. While simplified, the same principles and workflows readily scale to larger, biologically relevant systems. By mastering these fundamentals, users will be well prepared to apply their knowledge to more complex biomolecular simulations in research contexts.

Due to the statistical-mechanical nature of the ensembles of molecular configurations, meaningful values of quantities are averages over configurations or trajectories. Individual trajectories are perfectly fine for instructional purposes such as in this tutorial, but are of little utility in "real" research settings, unless there is little or no variation within the configurational ensemble. For most degrees of freedom of interest in bio-molecular systems this is certainly not the case. A simple means for generating replicates is to use different seeds of the random number generator for sampling the initial velocities at equilibration.

https://doi.org/10.33011/livecoms.6.1.3485
Living J. Comp. Mol. Sci. 2025, 6(1), 3485

2 Prerequisites

The tutorials require the latest GROMOS11 version installed (1.6.1). Users can download the GROMOS source code via www.gromos.net. Users may download the source code for free, as well as the PDF files for the manual. Files required for the basic tutorials in volume 7 of the manual can also be downloaded free of charge. The Program Library Manual (volume 5) [8] contains extensive documentation of the input flags. Furthermore, after compilation of the code, one can generate local documentation using doxygen.

2.1 Background knowledge

The tutorials described in this article assume the user to be familiar with the steps described in the GROMOS basic tutorial contained in volume 7 of the manual distributed with the software [7]. Specifically, users should be familiar with the content of a GROMOS system topology, input files and analysis tools explained in detail there. Tutorial 1 (see section 3.1) repeats some of the basic system preparation steps but cannot be comprehensive in explaining all basic operations. We assume that the user is familiar with basic Linux or Unix command line interactions and tools to efficiently edit larger plain text files such as VIM or Emacs. Furthermore, a user should be able to visualize molecular structures (e.g. with PyMOL [9] or VMD [10]) and to use basic plotting tools (e.g. xmgrace, R, matplotlib).

The GROMOS software for biomolecular simulation comprises the molecular dynamics engine MD++ and the GROMOS++ suite of pre- and postprocessing programs. The program is independent of the computer architecture or force field used. The units of the various quantities are defined outside the program through a physical constants block in a force-field file. The only unit conversion performed internally by the program is between degrees and radians. The force-field files come in GROMOS units, that is SI units, but with atomic mass units for mass, nm for distance, ps for time, and electronic charge for charge [11]. No simulation protocols are prescribed. Input parameters specified by a user are not modified inside the program. A warning is displayed for inconsistencies in the input that may lead to an erroneous simulation, but could also be intentional. The interpretation of the results is simplified by extensive documentation of the implemented algorithms and their technical details in the manual available on the GROMOS web site [11, 12].

2.2 Software/system requirements

GROMOS can be compiled on almost any operating system compatible with the POSIX standard. Some of the libraries required are not available on standard operating systems and have to be installed manually as described in detail in volume 8 of the GROMOS documentation [13]. In order to use the GROMOS programs without specifying the full path, you can add them to your PATH variable, see section 3.2.2. in volume 8. For some of the analyses in this tutorial, a basic installation of Python 3 is required. Note that files edited on non-Unix-like operating systems may cause an I/O-error due to a different representation of a line break. For tutorial 6 "NN(QM)/MM simulations with the BuRNN approach", a special compilation of GROMOS with pybind11 links to SchNetPack is needed. It also relies on quantum-mechanical calculations that can be performed with any software that is preferred. In this tutorial we have used MOPAC (version MOPAC2016.22.067L), for which the output files are provided, see section 3.6.1.

3 Content and links

The tutorials described in this article can be accessed at https://github.com/biomos/gromos_tutorial_livecoms. All necessary files for completing each tutorial are provided at that location.

3.1 Tutorial 1: S^2 order parameter restraining

The backbone N-H order parameter is a measure for the spatial restriction that the N-H vector experiences in a molecular reference frame. Order parameters calculated from ensembles generated by MD simulations are not subject to a specific motional model but depend on the local flexibility inherent in the force field when solving Newton's equations of motion and on whether the assumption of internal motion being independent of overall tumbling is justified. GROMOS features a time-averaging variant of order parameter restraining that is described in detail elsewhere [14]. Such time-averaged restraining enhances the configurational sampling by forcing the molecule to surmount barriers that would, without restraining, only be surmounted rarely, that is, on longer time-scales. Moreover, a possible force-field deficiency hampering the agreement with experiment can be redressed using this restraining technique. In this way, configurational ensembles consistent with NMR data can be generated allowing a structural interpretation of experimental observations [15, 16]. We will demonstrate the use of time-averaged order parameters by means of the third IgG-binding domain of Protein G (GB3), which is a small 56-residue protein.

3.1.1 Topology

Go into the subdirectory `topo` of the directory `t_01`. The input file `make_top_GB3.arg` is already prepared. We will use

https://doi.org/10.33011/livecoms.6.1.3485
Living J. Comp. Mol. Sci. 2025, 6(1), 3485

the force field 54a7. The molecular topology file for the protein, `GB3_54a7.top`, with SPC water as a solvent can then be generated using the GROMOS++ program `make_top` by typing

```
$ make_top @f make_top_GB3.arg >
    GB3_54a7.top
```

In order to neutralize the net charge of -2e of the protein topology the next step is to build a topology file for a sodium ion using the input file `make_top_Na.arg`:

```
$ make_top @f make_top_Na.arg > Na_54a7.top
```

Next we combine the two topologies using the GROMOS++ program `com_top`

```
$ com_top @f com_top_GB3_2Na.arg >
    GB3_2Na_54a7.top
```

The file `GB3_2Na_54a7.top` contains the complete molecular topology. Using the GROMOS++ program `check_top` with the arguments `@build` and `@param` the topology can be checked against the force field. The 34 types of logical checks performed are listed in volume 5 of the documentation [8]. Be aware that `check_top` may not catch every inconsistency and that an inconsistency pointed out by `check_top` may not necessarily indicate an error in the topology. In the present case the putative inconsistency with the partial charge on atom 5 spotted by `check_top` is actually not an error because the partial charge is adapted for the N-terminus of the peptide chain. Therefore, it is important to ensure that the topology generated is the one intended.

3.1.2 Coordinates

Go into the subdirectory `coord`. The Cartesian coordinates for the protein can be downloaded from the Protein Data Bank, accession code 2OED [17]. By using the GROMOS++ program `pdb2g96` the PDB file will be converted to a GROMOS coordinate file. Before conversion, we make a copy of the downloaded pdb file `2oed.pdb` into the file `2oed_edited.pdb`. In the latter we make a change in line 1028 (replace "O " by "O1") and line 1034 (replace "OXT" by "O2 ") such that `pdb2g96` recognizes these two atoms as belonging to the carboxy terminus. When editing the PDB file, the columns must be kept aligned. The remaining differences between the nomenclature used in the PDB file and the one used in the topology are handled via the file `pdb2g96.lib`. With

```
$ pdb2g96 @f pdb2g96_GB3.arg >
    pdb2g96_GB3.cnf
```

we generate a GROMOS coordinate file. Since the NMR structure used contains more hydrogen atoms than needed by the united-atom GROMOS force field, merging aliphatic hydrogen and carbon atoms into one interaction site, a list of warnings regarding ignored hydrogen atoms is issued, which can be ignored. If the initial structure was determined using X-ray diffraction, missing hydrogen atoms can be generated with the GROMOS program `gch` as explained in the basic tutorial.

3.1.3 Energy minimization

Before putting the protein in a box of solvent, its configuration is relaxed by energy minimization *in vacuo* to release possible strain induced by small differences in bond lengths, bond angles, improper dihedral angles and short non-bonded contacts between the force-field parameters and the NMR structure. Go into the subdirectory `min` and open the shell script `em_GB3.run` to adapt the paths and the names of the files according to your system. The energy minimization of the solute in vacuo is very fast and can be run interactively by typing

```
$ ./em_GB3.run
```

Once the energy minimization is finished, the minimized coordinates are written to the file `GB3_min.cnf` and the general output file `em_GB3.omd` contains the progress of the minimization.

3.1.4 Solvating the protein in a water box

Now the protein is ready to be placed into a box and solvated for subsequent simulations under periodic boundary conditions. Go into the subdirectory `box`. The box shape will be chosen to be rectangular, the simple point charge (SPC) water model [18] will be employed (as already specified in the topology file), the minimum solute-to-wall distance will be 1.2 nm such that the closest surface atoms of two periodic copies are at least 2.4 nm apart (longer than the cutoff distance of 1.4 nm). The minimum solute-solvent distance is set to 0.23 nm. The GROMOS++ program `sim_box` is used to generate the box and to solvate the protein by executing

```
$ sim_box @f sim_box_GB3.arg >
    sim_box_GB3.cnf
```

During immersion into the solvent, water molecules may still have been placed too close or too far away relative to the protein surface. Moreover, their orientation towards the protein surface is not optimized. Therefore, we need an equilibration of the solute-solvent system using energy minimization. During this process the solute atoms will be positionally restrained around their coordinates in the initial structure using harmonic springs while the solvent molecules can move freely. The list of atoms to be positionally restrained must be specified in a file `sim_box_GB3.por`. The reference positions of these atoms must be specified in a separate file `sim_box_GB3.rpr`. To prepare these files, copy the coordinate file `sim_box_GB3.cnf` to `sim_box_GB3.por` and

https://doi.org/10.33011/livecoms.6.1.3485
Living J. Comp. Mol. Sci. 2025, 6(1), 3485

`sim_box_GB3.rpr`. Open the file `sim_box_GB3.por` in your text editor and

- In the title block, write "list of solute atoms to be positionally restrained"
- Change the keyword "POSITION" at the beginning of the atom coordinate block to "POSRESSPEC"
- Delete all the solvent atoms. This can also be conveniently achieved by using the command line instruction
  `$ sed -i "/SOLV/d" sim_box_GB3.por`

When GROMOS reads this file, it will entirely ignore the coordinates and just look at the list of atoms. Next, open `sim_box_GB3.rpr` in your text editor and

- In the title block, write "reference positions of solute atoms to be positionally restrained"
- Change the keyword "POSITION" at the beginning of the atom coordinate block to "REFPOSITION"

When GROMOS reads this file, it will only use the coordinates of the atoms listed in `sim_box_GB3.por` and ignore the rest. Now, adapt the input file `em_solvent.imd` according to the number of solvent molecules in your box by adjusting the second number in the `SYSTEM` block and the index of the last atom in the `FORCE` block. Now, adapt the paths and the names of the files in `em_solvent.run` according to your system. Then start the energy minimization of the solvent interactively by typing

```
$ ./em_solvent.run
```

This will take a few moments. Once the minimization is finished, the new coordinate file, `GB3_h2o.cnf` and the general output file `em_solvent.omd` will be written out.

3.1.5 Adding counter ions

To complete the preparation of the simulation box two sodium ions should be added. Go to the subdirectory `ion`. The two sodium ions are added to the simulation box using the GROMOS++ program `ion` such that they replace the water molecules which have the lowest electrostatic potential. You can run `ion` by typing

```
$ ion @f ion_GB3.arg > GB3_2Na_h2o.cnf
```

3.1.6 Thermalisation and equilibration

For thermalisation we will use a combination of a progressively increasing temperature and progressively decreasing position restraints on the solute atoms. The thermalisation procedure is facilitated by the use of the GROMOS++ program `mk_script`, which allows the automatic generation of successive MD jobs that (i) slightly differ in their input parameters; (ii) use the final configuration and velocities of one job as the starting configuration and velocities of the next one; (iii) automatically submit the next job upon completion of the previous one. Go into the subdirectory `eq`. Before running the script you need to adjust the number of solvent molecules and the last atom for the set of degrees of freedom in the input file `equilibration.imd` as well as the paths and names in `eq_mk_script.arg`. Moreover, new position restraint files `GB3_2Na_h2o.por` and `GB3_2Na_h2o.rpr` have to be prepared as described above based on the output file `GB3_2Na_h2o.cnf` from the `ion` program. Now the job scripts and corresponding input files are created by typing

```
$ mk_script @f eq_mk_script.arg
```

You are now ready to start the thermalisation and equilibration. Run the first job script and the others will be automatically executed as soon as the preceding script has finished.

```
./eq_GB3_1.run
```

After the equilibration is finished you can carry out some basic checks in the `eq/ana` directory. For example, you can see that the kinetic energy is increasing at every new job.

3.1.7 Unrestrained molecular dynamics simulation

The equilibration procedure produced short simulations at constant temperature and volume. Now we want to extend the simulation to 21 ns under constant temperature and pressure. Go to the directory `md` and use the `mk_script` program to create the job scripts and input files:

```
$ mk_script @f md_mk_script.arg
```

Here the simulation is split into 21 jobs that may preferably run on a computer cluster. To run the jobs interactively type

```
$ ./md_GB3_1.run
```

To facilitate the submission to a cluster, adjust the entry `lastcommand` in the file `mk_script.lib`. Depending on your cluster settings, you may also want to adjust the entry `workdir` and make sure to use a binary that runs GROMOS in parallel (MPI or openMP) or uses the GPU acceleration [13].

3.1.8 S^2-order parameter restrained molecular dynamics simulation

Starting again from the final configuration of the equilibration procedure, we now perform the S^2 order parameter restraining simulation. Go to the directory `md_S2res` and have a look at the input file `md.imd`. Compared to the unrestrained simulation it contains the additional block

```
ORDERPARAMRES
# NTOPR  NTOPRA  COPR  TAUOPR  UPDOPR  NTWOP
   -1      0     300     200     1      250
END
```


https://doi.org/10.33011/livecoms.6.1.3485
Living J. Comp. Mol. Sci. 2025, 6(1), 3485

By setting the switch NTOPR to -1 you specify that you use time-averaged restraining without individual weights for the force constant. The switch NTOPRA controls reading of the averages from the startup file. The value should be 0 for the first job and 1 for continuation jobs. The switch COPR defines the order parameter restraining force constant. With TAUOPR the relaxation time is specified. The switch UPDOPR is only relevant if the averages are not calculated using the damped memory approach but as a running average covering the last TAUOPR picoseconds of the simulation. We note that window averaging shows no advantage over the damped memory approach while requiring a sizable amount of RAM. Finally NTWOP controls how often the order parameters are written to the special trajectory. The actual settings of the switches NTOPRA, COPR and TAUOPR are defined in the joblist S2_restraining.jobs that has the following structure:

```
TITLE
S2 order parameter restraining
END
JOBSCRIPTS
job_id [...] NTOPR NTOPRA COPR TAUOPR [...]
   1    [...]    -1    0      10    200   [...]
   2    [...]    -1    1     300    200   [...]
   3    [...]    -1    1     300    200   [...]
...
END
```

In the first job we start with a small restraining force constant COPR since we want a gentle build-up of the time averages. From the second job onwards the force constant is unchanged. Similar settings of force constant and averaging time were used in previous work on this system [14]. The experimental order parameters used for the restraining are taken from Hall and Fushmann [19] and are specified in the column S0 of the file order_exp.dat. In the latter file the atoms i and j defining the bond vector need to be specified as well as the average bond length R0. In the column DS0 the flat-bottom parameter of the restraining potential-energy term is set to 0.05. Therefore, no restraining force is applied if the absolute value of the difference between simulation and experiment is smaller than or equal to this value. With WOPR individual weights can be assigned to the order parameters if the corresponding switch NTOPR in md.imd is selected. Now use the mk_script program to create the job scripts and input files:

```
$ mk_script @f md_mk_script.arg
```

The file order_exp.dat needs to be specified under the keyword order in the @files section of md_mk_script.arg. As before the simulation is split into 21 jobs that may preferably run on a computer cluster. To run the jobs interactively type

```
$ ./md_GB3_1.run
```

3.1.9 Analysis

First, we analyse the energy trajectories of the unrestrained and restrained simulations. Go into the directory ana/ene_ana/unres and run the analysis program ene_ana by typing

```
$ ene_ana @f ene_ana_unres.arg >
    ene_ana_unres.out
```

The first trajectory is excluded from the analysis to account for the fact that the system needs some additional equilibration phase when switching from a constant volume to a constant pressure simulation. The file ene_ana_unres.out contains the averages while the time series of all specified properties are contained in the .dat files. The total intramolecular energy of the protein had to be defined in the ene_ana.md++.lib file located in the subdirectory above, see line 251 in that file. Repeat the analysis for the restrained simulation and compare the results. For the latter we additionally evaluate the total restraining energy in order to check whether the contribution of the restraints is small compared to the total intramolecular energy of the protein. Note that the file ene_ana.md++.lib has to be compatible with the GROMOS version used. If you use a newer version than 1.5.0, you will find the corresponding file in the directory md++-x.y.z/data of your GROMOS installation.

Second, the atom positional root-mean-square deviation of the backbone atoms from a reference structure is calculated for the two trajectories using the GROMOS++ program rmsd. For the unrestrained simulation go to the directory ana/rmsd/unres and type

```
$ rmsd @f rmsd_unres.arg > rmsd_unres.out
```

Here, we use the last structure of the equilibration simulation as reference. The two resulting RMSD time series are shown in Figure 1.

Third, the root-mean-square fluctuation of the backbone N atoms is calculated using the program rmsf. For the unrestrained simulation go to the directory ana/rmsf/unres and type

```
$ rmsf @f rmsf_unres.arg > rmsf_unres.out
```

The two resulting plots are displayed in Figure 2 and show that the restrained simulation does not necessarily show less fluctuations compared to the unrestrained simulation.

Finally the N-H order parameters are calculated using the program nhoparam. For the unrestrained simulation go to the directory ana/nhoparam/unres/0.5 and type

```
$ nhoparam @f nhoparam_unres.arg >
    nhoparam_unres.out
```


https://doi.org/10.33011/livecoms.6.1.3485
Living J. Comp. Mol. Sci. 2025, 6(1), 3485

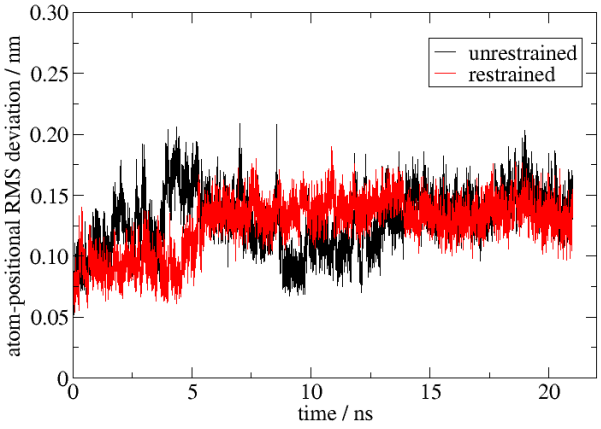

Figure 1. Backbone atom-positional root-mean-square deviation (RMSD) of GB3 with respect to the final structure of the equilibration simulation.

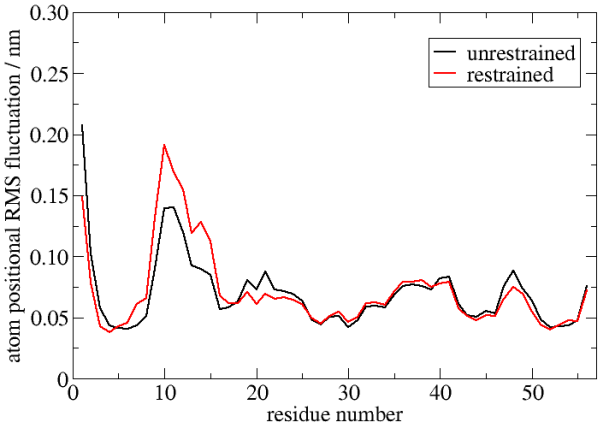

Figure 2. Backbone N atom-positional root-mean-square fluctuation (RMSF) of GB3.

We do the analysis using two averaging time windows of 0.5 and 1.0 ns, respectively. Since the order parameter is defined as long-time tail of the autocorrelation function of the bond vector, this comparison provides insight into whether the corresponding autocorrelation functions have reached their plateau values. The `nhoparam` program also calculates order parameters averaged over the entire trajectory. These values may be considerably smaller than those calculated using 1 ns averaging time. If that is the case, conformational changes occur on larger time scales and a structural interpretation based on order parameters might be difficult. Figure 3 shows a relatively small influence of the averaging time on the resulting order parameters in the present case.

3.1.10 Common errors

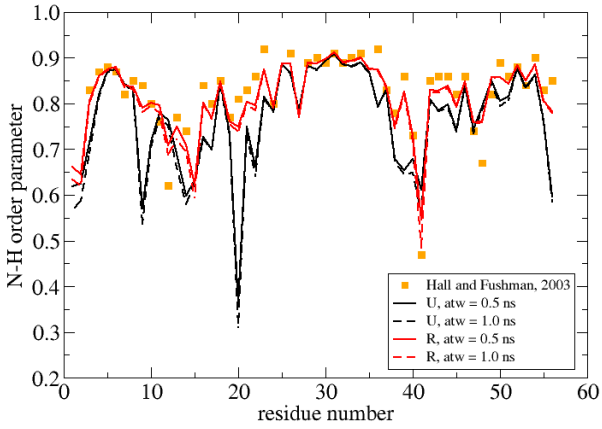

Figure 3. Comparison of backbone N-H order parameters for protein GB3, determined from unrestrained (U) and restrained (R) MD simulations using different averaging time windows (atw) in the analysis. The experimentally derived order parameters used for restraining were taken from the work of Hall and Fushman [19] (anisotropic model).

Too large initial restraining forces

Order parameters are not instantaneous observables but averages over multiple configurations of the systems. At the beginning of the simulation, these averages have not been established yet but are initialized based on the starting configuration. Depending on the value of the restraining force constant a SHAKE error might occur within the first steps of the restraining simulation. To prevent too large initial restraining forces, a gradual increase of the restraining force constant over multiple runs is usually sufficient. Note that the occurrence of this error depends on the specific system under investigation.

3.2 Tutorial 2: Double decoupling method & corrections for net-charge changes

The double decoupling method (DDM) [20] is an alchemical perturbation approach to compute binding free energies from molecular dynamics simulations by making use of a thermodynamic cycle (Figure 4). Two of the branches are determined by thermodynamic integration corresponding to the decoupling of the ligand from the system (perturbing the ligand into a non-interacting dummy molecule), both free in solution and when bound to the host. In order to avoid sampling of non-relevant phase space in the complexed system, the ligand is kept in a position that resembles that of the native bound conformation by gradually introducing a harmonic distance restraint. The free energy of the restraint removal can be evaluated analytically.

In this tutorial we will calculate the standard binding free energy of aspirin (ASA) to the protein phospholipase A_2

https://doi.org/10.33011/livecoms.6.1.3485
Living J. Comp. Mol. Sci. 2025, 6(1), 3485

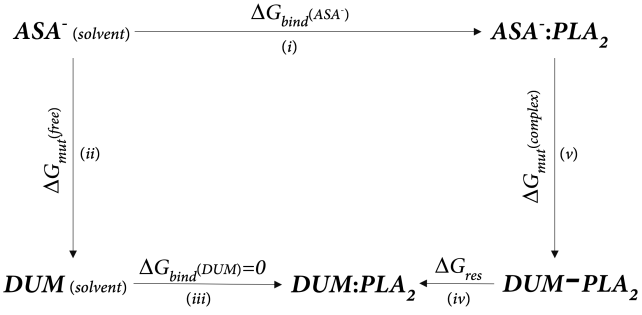

Figure 4. Thermodynamic cycle for the calculation of the standard binding free energy of aspirin (ASA^-) binding to the protein phospholipase A_2 (PLA_2). ASA^- is turned into a non-interacting dummy molecule (DUM), both in its complexed state (*v*) and free in solution (*ii*). The free energy of DUM binding to PLA_2 is zero (*iii*). An intermediate state (DUM-PLA_2) is introduced by linking both binding partners with a harmonic distance restraint. The free-energy contribution of this restraint can be calculated analytically (*iv*) via equation 6. The free-energy differences of branches (*ii*) and (*v*) are determined via (extended) thermodynamic integration (TI), enabling the calculation of the standard binding free energy (*i*).

(PLA_2) using the DDM and extended-thermodynamic integration (X-TI) [21]. For an application of X-TI and corrections of net-charge changes in current research see e. g. Ref. [22].

3.2.1 Simulation setup

Preparation of topologies and coordinate files, energy minimization, solvation in SPC water and the addition of counter ions as well as the setup of equilibrations and simulations can be performed in analogy to tutorial 1. The Cartesian coordinates for the enzyme phospholipase A_2 with bound acetyl salicylic acid (ASA) can be obtained from the Protein Data Bank with accession code 1OXR [23]. The final equilibrated structures, `eq_ASA_Na_7.cnf` and `eq_PLA2_ASA_Ca_2Na_7.cnf`, are in subdirectories `eq/eq_ASA` and `eq/eq_PLA2_ASA` of the directory `t_02`.

3.2.2 Perturbation topology

Go into the subdirectory `topo`. The topologies for the ligand (`ASA.top`) and the protein (`PLA2.top`) are already prepared. You can also find the combined topologies with sodium counter ions and a calcium ion that is important for ligand binding (`ASA_Na.top` and `PLA2_ASA_Ca_2Na.top`). For ligand decoupling, the topology for ASA in the decoupled state (`DUM.top`) was generated by changing the integer atom code (`IAC`) to 22 corresponding to the dummy type for all the atoms and setting all the charges to 0. The program `make_pt_top` can convert topologies from state A and B into a perturbation topology. The `PERTATOMPARAM` block lists all atoms with their respective force field parameters that will be alchemically perturbed during the simulation.

3.2.3 Distance restraints

Distance restraints are introduced for the calcium ion to keep it bound in the active site. For this, a distance restraint specification file `disres.dat` is set up in subdirectory `eq` containing the following block.

```
DISTANCERESSPEC
# DISH DISC
  0.1   0.153
# i    j  k  l  type i   j  k  l  type r0    w0    rah
  1208 0  0  0  0    309 0  0  0  0    0.223 1.0   0
  1208 0  0  0  0    321 0  0  0  0    0.235 1.0   0
  1208 0  0  0  0    339 0  0  0  0    0.246 1.0   0
  1208 0  0  0  0    489 0  0  0  0    0.255 1.0   0
  1208 0  0  0  0    490 0  0  0  0    0.248 1.0   0
END
```

The restraint is defined between the calcium ion and 5 atoms of residues coordinating the ion (3 amide oxygens and 2 carboxylate oxygens). `type 0` refers to explicit/real atoms. `r0` is the distance between two atoms in nm. The restraint is defined with a weight factor `w0` of 1 by which the distance restraint interaction term `CDIR` of the `DISTANCERES` block in the `imd`-file is multiplied (force constant). The parameter `rah` controls the form and dimension of the restraint, here it is set to zero, which corresponds to a full harmonic potential-energy term in *x*, *y*, *z* dimensions. Parameters `DISH` and `DISC` are the hydrogen-carbon and carbon-carbon distances, respectively. GROMOS can also apply distance restraints on virtual or pseudo atoms by setting the appropriate type and a specification of additional atoms `j`, `k` and `l`.

To keep the ligand within the active site when getting decoupled, we gradually turn on a harmonic distance restraint simultaneously to the perturbation. Go into the subdirectory `DDM/md_TI` where you will find the `disres.dat` file which contains an additional block.

```
PERTDISRESSPEC
# DISH   DISC
0.1   0.153
# i j    k    l type i     j     k     l type n
    m Ar0   Aw0   Br0   Bw0 rah
 21  211  264  529 -1 1195  1199  1203  0 -1    0
     0 0.0   0.0   0.0   1.0 0
END
```

The distance restraint is defined between the center of geometry (`type -1`) of 4 backbone atoms (`i`, `j`, `k` and `l`) of PLA_2 around the active site and the center of geometry of 3 atoms of the ligand's benzene ring with a distance of zero (`A_r0=B_r0=0`). At state A the restraint is turned off (`A_w0=0`), while at state B the restraint is retained with a weight factor of

https://doi.org/10.33011/livecoms.6.1.3485
Living J. Comp. Mol. Sci. 2025, 6(1), 3485

1 (`B_w0=1`). Parameters `n` and `m` control hidden restraints [24]. Parameters `DISH` and `DISC` are not relevant for the center of geometry and this type of pseudo atoms.

3.2.4 Extended-thermodynamic integration simulation

The thermodynamic integration approach uses the coupling parameter λ, which defines the system as a linear combination of the two end-states [25]. The coupling parameter approach formulates the Hamiltonian of the system dependent on λ by interpolating between the two states (scaling of force-field parameters).

$$V_{\mathrm{nb}}(r_{ij},\lambda) = (1-\lambda)^n V^{\mathrm{A}}(r_{ij},\lambda) + \lambda^n V^{\mathrm{B}}(r_{ij},1-\lambda) \tag{1}$$

with

$$V^{\mathrm{X}}(r_{ij},\lambda) = \frac{C_{12}^{\mathrm{X}}}{(\alpha_{\mathrm{lj}}\lambda^2 C_{126}^{\mathrm{X}} + r_{ij}^6)^2} - \frac{C_6^{\mathrm{X}}}{\alpha_{\mathrm{lj}}\lambda^2 C_{126}^{\mathrm{X}} + r_{ij}^6} + \frac{q_i^{\mathrm{X}} q_j^{\mathrm{X}}}{4\pi\epsilon_0}\left[\frac{1}{(\alpha_{\mathrm{crf}}\lambda^2 + r_{ij}^2)^{1/2}} - \frac{1/2C_{\mathrm{rf}} r_{ij}^2}{(\alpha_{\mathrm{crf}}\lambda^2 + R_{\mathrm{rf}}^2)^{3/2}} - \frac{1-1/2C_{\mathrm{rf}}}{R_{\mathrm{rf}}}\right] \tag{2}$$

where C_6^{X}, C_{12}^{X}, q_i^{X} and q_j^{X} are the Lennard-Jones parameters and partial charges for state X (A or B). r_{ij} is the distance between particles i and j, C_{rf} and R_{rf} are parameters of the electrostatic reaction field assumed outside the cutoff sphere [26]. α_{crf} and α_{lj} are soft-core parameters [27].

Go into the subdirectory `md_TI`. The input files `md_TI_ASA_Na.imd` and `md_TI_PLA2_ASA_Ca_2Na.imd` contain two additional blocks that are relevant for the free-energy calculations using X-TI. The `PERTURBATION` block controls the alchemical perturbation

```
PERTURBATION
#      NTG    NRDGL     RLAM    DLAMT
         1        0      0.0      0.0
#   ALPHLJ    ALPHC     NLAM   NSCALE
       1.0      1.0        1        0
END
```

`NTG` turns on the perturbation to calculate $\partial\mathcal{H}/\partial\lambda$ and `RLAM` is the initial value for λ. The initial value of λ could also be read from the configuration by setting `NRDGL=1`. `RLAM` will be adjusted for several different λ points using the jobs file `md_TI.jobs`. `DLAMT` controls the increase of λ with time. The parameters `ALPHLJ` and `ALPHC` are the soft-core parameters for Lennard-Jones (α_{lj}) and Coulomb (α_{crf}) interactions, respectively. `NLAM` controls the power dependence of the λ coupling (n in eq. 1) and `NSCALE` the use of interaction scaling for complete energy groups.

The `PRECALCLAM` block is relevant for the pre-calculation of intermediate non-simulated λ-points during the simulation as extension to standard TI.

```
PRECALCLAM
# NRLAM    MINLAM    MAXLAM
     81         0         1
END
```

With the settings in the above block, the energies and derivatives with respect to λ will be calculated on-the-fly at 81 points ranging from λ=0 to λ=1, resulting in λ-steps of 0.0125.

The simulation is defined in the jobs file `md_TI.jobs`. Simulations will be performed at 11 equally spaced λ-points between λ=0 and λ=1 for 5 ns each in case of the complexed system, where ASA is bound to PLA_2. This system also requires the perturbed distance restraint. The simulations of ASA free in solution will also be performed at 11 λ-points each for 0.5 ns. Note that this tutorial can also be carried out using standard TI, in which case the `PRECALCLAM` block is not required. The choice of 11 equally spaced λ-points is typically a reasonable start, but it is recommended to adjust the number of points and the spacing according to the curvature and error estimates of $\partial\mathcal{H}/\partial\lambda$. In X-TI, adjustment is often not necessary, even fewer points may be sufficient in some cases, but for standard TI usually more than 11 λ-points are needed. Therefore, we strongly recommend to use the `PRECALCLAM` block and take advantage of the pre-calculation of intermediate non-simulated λ-points and subsequent reweighting. To run the two simulations, copy the argument files required by the `mk_script` program into the two directories `md_TI_ASA` and `md_TI_PLA2_ASA`, respectively, and adapt the paths before generating the job files with the `mk_script` program and submitting the jobs to a cluster. If you prefer to continue directly, you will find the necessary energy and free-energy trajectories in the subdirectories `L_*`.

3.2.5 Free-energy analysis

The free energies can be determined via extended-thermodynamic integration (X-TI) [21] or the Bennett acceptance ratio (BAR) method [28]. The raw data for both methods can be extracted from the energy and free-energy trajectory files using the GROMOS++ program `ext_ti_ana`.

Go into the subdirectory `DDM/ana_TI/ana_TI_ASA` and run the program `ext_TI_ana` via the bash-script by typing

```
$ ./ext_ti_ana_bar.sh
```

X-TI requires the pre-calculation of free energies at non-simulated points. Free-energy derivatives at requested non-simulated λ_{P} values can be reweighted to obtain ensemble averages for λ_{P} from simulated λ_{S} points. The

https://doi.org/10.33011/livecoms.6.1.3485
Living J. Comp. Mol. Sci. 2025, 6(1), 3485

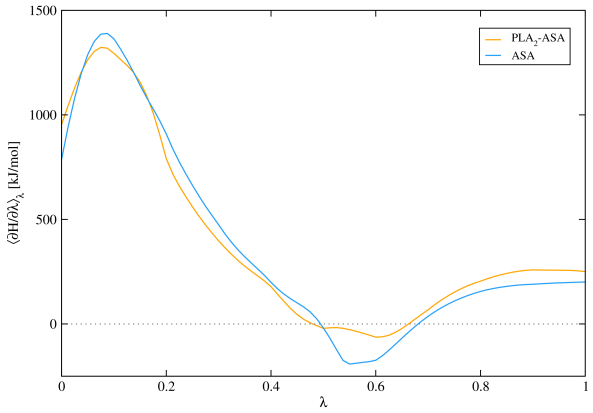

Figure 5. The reweighted property $\left\langle \frac{\partial \mathcal{H}}{\partial \lambda} \right\rangle$ for λ=0 to λ=1 for both systems ASA and PLA$_2$_ASA.

predictions from multiple simulations at λ_S points can be merged into a single TI profile by a linear reweighting scheme using the program `ext_TI_merge`. Run the program via the bash-script by typing

```
$ ./ext_ti_merge.sh
```

The program calculates the integral of the final TI-curve using the trapezoidal rule in order to obtain the free-energy estimate. The results of X-TI for both systems, ASA and PLA2_ASA, are shown in Figure 5.

BAR estimates free energies from the free-energy differences between two adjacent λ points i and j, using

$$\Delta G(\lambda_i \rightarrow \lambda_j) = k_B T \ln \frac{\langle f(E(\lambda_i) - E(\lambda_j) + C)\rangle_{\lambda_j}}{\langle f(E(\lambda_j) - E(\lambda_i) - C)\rangle_{\lambda_i}} + C \qquad (3)$$

with

$$f(x) = \frac{1}{1 + \exp(x/k_B T)} \qquad (4)$$

C is determined iteratively to ensure that the two ensemble averages from λ_i and λ_j are identical. To calculate error estimates, bootstrap sampling can be conducted. Run the program `bar` via the bash-script by typing

```
$ ./bar.sh
```

BAR is computationally more efficient and converges relatively fast compared to regular TI. The efficiency of X-TI is comparable, with the added advantage of direct visualization of the entire free-energy profile [29].

3.2.6 Thermodynamic cycle

The free energy of binding is determined according to the thermodynamic cycle shown in Figure 4 as

$$\Delta G_{bind}(ASA^-) = \Delta G_{mut}(free) + \Delta G_{bind}(DUM) - \Delta G_{res} - \Delta G_{mut}(complex) \qquad (5)$$

The free energy associated with the removal of the restraint (ΔG_{res}) for a dummy particle can be calculated analytically, including a standard state correction [30, 31]:

$$\Delta G_{res} = -k_B T \ln \frac{V^\circ}{\left(2\pi k_B T/K\right)^{3/2}} \qquad (6)$$

where K is the force constant of the harmonic distance restraint and V° is the accessible solution volume corresponding to the standard-state definition. For a molar reference concentration it is given by V° = 1.661 nm^3. Equation 6 corrects for restricted mobility and can be derived from the partition function associated with the restraining potential energy function, given that the restraint is so strong that the integration volume can be extended to the entire space [32].

3.2.7 Correction terms for net-charge changes

Due to finite box sizes, periodic boundary conditions and simplifications in the calculations of the electrostatic interactions, the calculated free energies are artifacted. In the following paragraphs, we will refer to these energies as raw free energies. We will quantify and correct these artifacted components using a free-energy correction ΔG_{cor} to yield methodology-independent values ΔG as

$$\Delta G = \Delta G_{raw} + \Delta G_{cor} \qquad (7)$$

Analogous to Reif and Oostenbrink [33], ΔG_{cor} is a combination of multiple free-energy corrections for a spurious solvent-polarization (ΔG_{pol}), the impracticality of calculating the zero of the potential under periodic boundary conditions using discrete solvent molecules (ΔG_{dsm}) and artifacted direct interactions between the ligand and the host molecule (ΔG_{dir}). The free-energy correction is calculated as

$$\Delta G_{cor} = \Delta G_{pol} + \Delta G_{dir} + \Delta G_{dsm} \qquad (8)$$

These three correction terms will be calculated in the following paragraphs. A general scheme about the calculation of the corrections based on λ-generated trajectories can be found in Figure 6 [34]. Note that within this tutorial, only the information needed for the practical part is provided, further details about the theory can be found elsewhere [33, 35, 36]. The set of the three correction terms needs to be calculated for both branches of the thermodynamic cycle. Have a look into the directory `corrections`. It contains subdirectories for Aspirin free in solution and the complex of Phospholipase A2 with Aspirin. The correction procedure will only be explained for Aspirin free in solution, the procedure for the complex is similar. Go into the directory `corrections/ASA`. First, we need to create a set of topologies for intermediate states. These topologies will have full, unperturbed van der Waals parameters but charges that scale with λ. Go

https://doi.org/10.33011/livecoms.6.1.3485
Living J. Comp. Mol. Sci. 2025, 6(1), 3485

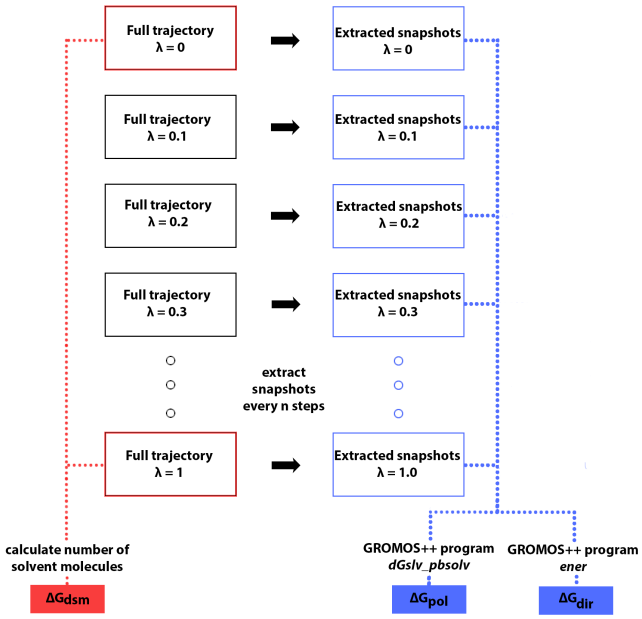

Figure 6. Schematic representation of the corrections. ΔG_{dsm} is calculated over full trajectories of the end states (however, if one of the end states has zero charges, it can be skipped). ΔG_{pol} and ΔG_{dir} are both calculated on individual snapshots only. In this tutorial, only the final snapshots of 6 chosen λ-points are used for these calculations.

further into the subdirectory `correction_topologies`. The python script `interpolate_topocharges.py` takes three input parameters: topologies at states A and B and a specific λ-value. A new topology with charges that correspond exactly to the given λ-value is written out. All other parameters remain unperturbed and are taken from topology A. We do not have to use it directly but can use the bash script `do_interpolate_topocharges.bash` instead. It creates six topologies at equidistant λ-points. These will be used for the individual correction terms explained in the following paragraphs.

3.2.8 Solvent polarisation

First, we calculate the correction estimate ΔG_{pol} for the spurious solvent polarisation. We will use a set of continuum-electrostatics calculations on configurations extracted from trajectories that were sampled at different λ points. For each of these configurations, the electrostatic potential at the atom sites of the Aspirin molecule will be calculated twice - using a cutoff scheme with a reaction-field contribution under periodic boundary conditions and using full Coulombic charges under non-periodic boundary conditions. The integrated differences between both potentials will give an estimate of ΔG_{pol}, which can be directly applied to the raw free energies that were calculated in the simulation. Note that for the sake of simplicity, in this tutorial we will use only the last configurations of six λ-generated trajectories. However, when used for "real" research questions, a more complete set of configurations should be analysed in order to achieve more reliable results.

Go into the subdirectory `corrections/ASA/dGpol`. It contains argument files for the GROMOS++ program `dGslv_pbsolv`. There are six argument files for six different λ points. Since the continuum-electrostatics calculations are computationally demanding, the output files are already provided. However, if you prefer to run the continuum-electrostatics calculations yourself, you can run them by typing

```
$ dGslv_pbsolv @f dGslv_pbsolv_L_0.0.arg >
    dGslv_pbsolv_L_0.0.out
$ dGslv_pbsolv @f dGslv_pbsolv_L_0.2.arg >
    dGslv_pbsolv_L_0.2.out
...
$ dGslv_pbsolv @f dGslv_pbsolv_L_1.0.arg >
    dGslv_pbsolv_L_1.0.out
```

Let's have a look at one of the output files. It contains four rows for the individual atoms of the charged acetyl group of the Aspirin molecule. There are several columns. Next to basic information for the individual atoms, there are six columns with potentials created under non-periodic boundary conditions in solvation (NPBC_SLV), non-periodic boundary conditions in vacuum (NPBC_VAC), periodic boundary conditions in solvation (PBC_SLV), periodic boundary conditions in vacuum (PBC_VAC), a fast Fourier transform result for the lattice-summation method under periodic boundary conditions (FFT_LS_PBC) and a fast Fourier transform result for the reaction-field method under periodic boundary conditions (FFT_RF_PBC). The potential that has to be integrated over λ reads (NPBC_SLV - NPBC_VAC) - (PBC_SLV - PBC_VAC) - (FFT_LS_PBC - FFT_RF_PBC). Note that the last term has to be used only if a cutoff scheme with reaction field contribution was applied in the simulation. You can simply use the script `integrate.py`. Type

```
$ ./integrate.py
```

We are interested in the result NPBC - PBC, which is the correction term ΔG_{pol}.

3.2.9 Direct ligand-protein interactions

In the actual MD simulation, the interaction between the ligand and the protein atoms was calculated using a cutoff scheme with a reaction field contribution. A correct scheme would involve no cutoff and purely Coulombic interactions. ΔG_{dir} accounts for the difference between the simulated and the real case. Note that in order to minimize the

https://doi.org/10.33011/livecoms.6.1.3485
Living J. Comp. Mol. Sci. 2025, 6(1), 3485

dependence of the correction terms on the conformations of the molecules the same configurations have to be used for ΔG_{dir} that were used for the calculations of ΔG_{pol}. Go into the subdirectory `corrections/ASA/dGdir`. It contains argument files for the GROMOS++ program `ener`. There are 12 argument files for six different λ-points, one for Coulombic interactions under non-periodic boundary conditions (NPBC) and one for Coulomb/Reaction-field interactions under periodic boundary conditions (PBC). You can use the provided output files or generate them yourself by typing

```
$ ener @f ener_PBC_L_0.0.arg >
    ener_PBC_L_0.0.out
$ ener @f ener_NPBC_L_0.0.arg >
    ener_NPBC_L_0.0.out
$ ener @f ener_PBC_L_0.2.arg >
    ener_PBC_L_0.2.out
...
$ ener @f ener_NPBC_L_1.0.arg >
    ener_NPBC_L_1.0.out
```

We are interested in the integrated energies NPBC-PBC. You can do it yourself or use a provided Python script. Simply type

```
$ ./integrate.py
```

The integrated result is the correction term ΔG_{dir}.

3.2.10 Potential from discrete solvent molecules

Another artifact stems from the impossibility of calculating the absolute zero potential in a periodic simulation box and the convention to average the solvent-generated potential over the exterior and the interior of the solvent molecules. As a consequence, the calculated potential differs from the "real" potential by an offset. For a rigid solvent model with a single van der Waals interaction site and any scheme relying on molecular-cutoff truncation based on this specific site, it can be shown that this offset is related to the quadrupole moment trace of the solvent model used. The free-energy correction is furthermore proportional to the water-molecule density inside the box (LS - lattice summation schemes) or within the cutoff radius (cutoff schemes with reaction-field correction - RF) and reads

$$\Delta G_{\text{dsm}}(\text{LS}) = -N_{\text{A}}(6\epsilon_0)^{-1}\gamma_{\text{S}}\Delta Q N_{\text{S}} V_{\text{B}}^{-1} \tag{9}$$

for the LS scheme and

$$\Delta G_{\text{dsm}}(\text{RF}) = -N_{\text{A}}(6\epsilon_0)^{-1}\frac{2(\epsilon_{\text{RF}}-1)}{2\epsilon_{\text{RF}}+1}\gamma_{\text{S}}\sum_{i=1}^{n}\Delta q_i \left\langle N_{\text{S}}(R_{\text{C},i})\right\rangle V_{\text{C}}^{-1} \tag{10}$$

for the RF scheme, where N_{A} is the Avogadro constant, ϵ_0 is the vacuum dielectric permittivity, ϵ_{RF} is the (relative) reaction-field dielectric permittivity, ΔQ is the net-charge change in the system, Δq_i is the net-charge change of the perturbed atom i, N_{S} is the number of solvent molecules in the box, $\left\langle N_{\text{S}}(R_{\text{C},i})\right\rangle$ is the average number of solvent molecules in the cutoff sphere of the perturbed atom i, V_{B} is the volume of the computational box and V_{C} is the volume of the cutoff sphere. γ_{S} is the quadrupole moment trace relative to the van der Waals interaction site. Values for commonly used water models can be found in table 1. Hint: the reaction-field permittivity used in the simulations can be found in the `NONBONDED` block of an `imd` file under the parameter `EPSRF`.

Table 1. Quadrupole-moment traces [e nm^2] for typical solvent models

model	γ_{S}
SPC [18]	0.008200
SPC/E [37]	0.008476
TIP3P [38]	0.007641
TIP4P [38]	0.009295
TIP5P [39]	0.002054
ST2 [40]	0.001754

Go into the subdirectory `corrections/ASA/dGdsm`. First, we need to calculate the average number of water molecules in the cutoff sphere that was used in the simulation. We will calculate this number using a radial distribution function (rdf) over the trajectory, which was generated with full charges on the perturbed atoms. The argument file for the GROMOS++ program `rdf` is already provided, as is the output file in case your simulation has not finished yet. You can run `rdf` by typing

```
$ rdf @f rdf.arg > rdf.out
```

The output file contains the densities of water particles as a function of the distance of all the perturbed atoms. To obtain the total number of water molecules, these densities need to be integrated over the distance and multiplied by the water number density $\rho = N/V_{\text{B}}$ in the simulation box. Equation 10 then gives the final correction term ΔG_{dsm} . You can calculate it by typing

```
$ ./integrate.py
```

This script reads the file `rdf.out` as well as `system.info` that contains relevant information about the box size, the cutoff, the correction field and the solvent model. Relevant information about the box size can be found by looking into one of the configuration files - the `POSITION` block contains the number of solvent molecules and the `GENBOX` block provides the box dimensions. Settings for the electrostatics can be found in the `NONBONDED` block in the `imd` files.

https://doi.org/10.33011/livecoms.6.1.3485
Living J. Comp. Mol. Sci. 2025, 6(1), 3485

3.2.11 Corrected results

Above, the three correction terms for Aspirin free in solution were calculated. According to equation 8, the sum of these three correction terms constitutes the total correction for this branch of the thermodynamic cycle. The same set of correction terms has to be calculated for the ligand bound to the host (directory `corrections/PLA2_ASA`). Both corrections can be directly added to the raw free energies, yielding methodologically independent results (see table 2). The final calculated estimate $\Delta G^{\circ}_{\text{bind}}$ = ΔG(PLA$_2$_ASA)–ΔG(ASA) = –32.3 kJ/mol agrees quite well with the experimentally determined estimate of $\Delta G_{\text{bind,exp}}$ = –29.6 kJ/mol [23].

Table 2. Results from Double Decoupling with corrections. All values are reported in kJ/mol.

System	ΔG_{raw}	ΔG_{res}	ΔG_{pol}	ΔG_{dir}	ΔG_{dsm}	ΔG
ASA	-371.2	-	12.3	-9.1	77.5	-290.5
PLA$_2$_ASA	-383.2	18.2	-13.3	23.4	32.1	-322.8

3.2.12 Common errors

Insufficient λ-point resolution

Large variation in free-energy estimates across replicates may indicate an insufficient λ-point resolution, especially in regions with high curvature in $\partial\mathcal{H}/\partial\lambda$.

To diagnose this, plot the predicted free-energy derivatives from neighboring simulated λ-points. If the predictions of two neighboring pre-calculated values differ significantly, intermediate λ-points should be added to improve accuracy.

Coarse grid in ΔG_{pol} correction term

Solvent polarization corrections using Poisson–Boltzmann (PB) calculations are computationally expensive. To reduce computational costs, one might increase the grid spacing. However, this can introduce significant errors if the grid does not fully enclose the solute or fails to resolve its charge distribution accurately.

We recommend validating the chosen grid by checking whether further refinement leads to noticeable changes in the correction.

3.3 Tutorial 3: Using HREMD and distance-field

Another way to calculate the binding free energy of a ligand to a protein is to pull the ligand out of the active site. There are several methods available to perform such calculations, however, most of them require the *a priori* knowledge of the dissociation path. Even if the dissociation path is determined during the simulation, it is often assumed that it is linear, or that only a single dissociation path exists. For an accurate estimate of the binding free energy the simulations should be performed such that the binding and unbinding of the ligand can be sampled reversibly. In this tutorial we will use the distance-field (DF) distance as the reaction coordinate and combine this with Hamiltonian-replica-exchange molecular dynamics (HREMD) simulations [41]. The advantage of the DF distance is that we will be able to pull the ligand back into the active site, even if it moved to the other side of the protein. The HREMD simulations allow for multiple paths to be sampled reversibly [42].

3.3.1 Preparations

As in the tutorial above, we will use the PLA$_2$-ASA complex as a test system. The preparation of the topology, coordinates, energy minimization and equilibration of the system are very similar to the tutorial above. The only difference is that we will use a larger cubic box to allow the ligand to move freely in the solvent. The final equilibrated structure can be found in the directory `eq` with the name `eq_PLA2_ASA_Ca_2Na_7.cnf`.

3.3.2 Slow-growth

In order to get some initial coordinates for each of the replicas of the HREMD simulations, we will perform a short slow-growth simulation where the ligand is pulled out of the active site in a non-equilibrium manner. The exact pulling speed and force constant are not relevant in this case as we are not trying to calculate the binding free energy from the slow-growth simulations. It is, however, important that the structure of the protein does not get disrupted during this initial simulation. The slow-growth simulation is started from the final configuration of the equilibration. Go to the directory `slowgrowth` and have a look at the `PERTURBATION` block in the input file `slowgrowth.imd`.

```
PERTURBATION
#      NTG    NRDGL     RLAM    DLAMT
         1        0      0.0    0.001
#   ALPHLJ    ALPHC     NLAM   NSCALE
       0.0      0.0        1        0
END
```

With `NTG` set to `1`, you specify that a free energy calculation will be performed. The starting λ point `RLAM` is set to `0`. With each timestep during the simulation, λ will be increased. The rate of increase in ps^{-1} is set to `DLAMT = 0.001`. Together with `NSTLIM = 500000` (from the `STEP` block) and an integration time step of 0.002 ps, this results in a simulation of 1 ns where λ is continuously changed from 0 to 1. `ALPHLJ` and `ALPHC` are the softness parameters of the Lennard-Jones and Coulombic interactions, respectively. These are set to 0 here, because we are not changing any nonbonded interactions, only distance restraints. In addition to that, we have to specify which kind of restraints will be used. In this case we will use distance-field distance restraints:

https://doi.org/10.33011/livecoms.6.1.3485
Living J. Comp. Mol. Sci. 2025, 6(1), 3485

```
DISTANCEFIELD
# NTDFR
      1
#  GRID PROTEINOFFSET PROTEINCUTOFF PROTECT
    0.2         15           0.2          0
# UPDATE  SMOOTH   RL   NTWDF     PRINTGRID
    100         1  1.0    1000            0
END
```

With NTDFR set to 1, we turn on distance-field (DF) restraining during the simulation. Typically the grid size (GRID) for the DF calculation is set to 0.2 nm. PROTEINOFFSET is the penalty which is added to the DF distance if the path crosses the host. This value has to be large enough, such that paths through the protein always result in larger DF distances than around the protein. If the length of paths around the protein is larger than the value of PROTEINOFFSET, the paths that go through the protein may become competitive and forces will point into the protein, rather than along the surface. Setting very large values of PROTEINOFFSET however, may lead to large forces at the surface of the protein, especially if the SMOOTH option is not used (see below). Here we have chosen PROTEINOFFSET = 15 nm. PROTEINCUTOFF = 0.2 determines that any grid point which is within 0.2 nm of a protein atom will be flagged as protein. In cases where the binding site is quite small, it can happen that the zero distance point is often flagged as protein, in these cases it might be necessary to use PROTECT > 0. This is the radius around the zero-distance point in which a grid point will not be flagged as protein. For our simulation, we will leave this value at 0.

In order to speed up the simulation, the grid is calculated only every UPDATE = 100 steps. The SMOOTH parameter is used to prevent very large forces acting at the surface of the protein due to the large penalties. With each smoothening step the non-protein grid points are identified which have a direct neighbouring grid point flagged as protein. These are on the edge of the protein and we will recalculate their DF distance but now without the protein penalty. This removes the large forces pointing away from the protein, but because the smoothening steps are performed after the updating step, they do not influence the optimal route. We will set the number of smoothening rounds to 1. Forces change linearly for distances larger than 1 nm as set by RL. We will write DF distances and forces to an external file (special trajectory file, *.trs) every NTWDF = 1000 steps. We will not print the grid to the final configuration, so we use PRINTGRID=0. The definition of the distance restraint that will be modified during the slow-growth simulation, is specified in the distance restraint specification file disres.dat. There are two blocks in this file. The first one, DISTANCERESSPEC specifies the distance restraints between the Ca^{2+} ion and its surrounding residues, which prevents the Ca^{2+} from drifting away. The second block PERTDFRESSPEC specifies the perturbed distance-field restraints.

```
PERTDFRESSPEC
# DISH   DISC
  0.1    0.153
# PROTEINATOMS   A_R0   K_A   B_R0   K_B   M   N
           1190    0.0   500    5.0   500   0   0
#   TYPE_I   NUM_I  ATOM_I[0] .. ATOM_I[NUM_I]
         -1       7     16 190 249 312 486 632
            1208
#   TYPE_J   NUM_J  ATOM_J[0] .. ATOM_J[NUM_J]
         -1       2   1194 1203
END
```

DISH = 0.1 nm specifies the carbon-hydrogen distance and DISC = 0.153 nm specifies the carbon-carbon distance. These are used to compute the position of some types of pseudo or virtual atoms, respectively, from the positions of explicitly represented atoms. PROTEINATOMS specifies the last atom number of the protein which will be used to flag protein atoms. A_R0 and B_R0 are the restraining distances in nm at λ = 0 and λ = 1, respectively. We will use a force constant of 500 kJ mol^{-1} nm^{-2} which remains the same upon changing λ (K_A = K_B = 500). We will not use hidden distance restraints, so M = N = 0.

The distance between pseudo atom i and pseudo atom j will be restrained, where pseudo atom i is defined as the center of geometry of 7 atoms (NUM_I = 7) with atom numbers 16, 190, 249, 312, 486, 632 and 1208. These atom numbers correspond to the C_α atoms of residues L2, W18, A22, C28, D48, Y63 (residue numbers according to the topology) and the calcium ion. Pseudo atom j is the center of geometry of atoms C2 and C7 (topology names) of the aspirin ligand. All input files are now prepared and we can generate the run file with:

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

Make sure the generated slowgrowth_PLA2_ASA_1.run file is ready to be submitted to a cluster. After running the slow-growth simulation, we can analyze the system by using trs_ana. We will use this program to read out the DF distances and forces from the special trajectory, *.trs, file. An example of the argument file is prepared in trs_ana.arg. You can run the program with

```
$ trs_ana @f trs_ana.arg
```

The DF distance as a function of time is written out to the file df_dist.dat. Have a look at the file with e.g., Xmgrace and make sure no sudden jumps are present. We also need to make sure that the protein was not disrupted during the

https://doi.org/10.33011/livecoms.6.1.3485
Living J. Comp. Mol. Sci. 2025, 6(1), 3485

pulling process. For this, calculate the root-mean-square deviation (RMSD) as a function of time with

```
$ rmsd @f rmsd_bb.arg > rmsd_bb.dat
$ rmsd @f rmsd_all.arg > rmsd_all.dat
```

Have a look at the RMSD of the backbone atoms (`rmsd_bb.dat`) as well as for all protein atoms (`rmsd_all.dat`). When both the `df_dist.dat` and the RMSD plots look normal, we can continue to prepare the starting structures for the replica-exchange simulations. If not, the slow-growth simulation should be performed again with some variables modified. For example, you can decrease `DLAMT` (and increase `NSTLIM` accordingly) in order to pull slower. Changing the force constant of the DF restraints (`K_A` and `K_B` in the `PERTDFRESSPEC` block) or choosing different atoms to apply the DF restraints to can also help to avoid any disruption of the protein.

3.3.3 Hamiltonian-replica-exchange simulations

One of the first choices that we have to make when starting a HREMD simulation is how many replicas will be used. For performance issues it is best to keep the number of replicas equal to the number of CPUs available (preferably on a single computational node). In the prepared example, we used 24 replicas. Have a look at the prepared input file for the replica-exchange simulation (`HREMD.imd` in the directory `HREMD`). You will find that the `PERTURBATION` block is still present, but with `DLAMT` now set to `0`. This means we are still calculating free energies, but we are no longer changing the λ parameter during a single simulation. The `DISTANCEFIELD` block is largely unchanged, apart from `NTWDF` because we will write out the DF data more often. You will also find an additional block:

```
REPLICA
# NRET
  1
# RET(1..NRET)
  298.0
# LRESCALE
  0
# NRELAM
  24
# RELAM(1..NRELAM)
 0.0000  0.0435  0.0870  0.1304  0.1739  0.2174
     0.2609  0.3043  0.3478  0.3913  0.4348
     0.4783  0.5217  0.5652  0.6087  0.6522
     0.6957  0.7391  0.7826  0.8261  0.8696
     0.9130  0.9565  1.0000
# RETS(1..NRELAM)
 0.002   0.002   0.002   0.002   0.002   0.002
     0.002   0.002   0.002   0.002   0.002
     0.002   0.002   0.002   0.002   0.002
     0.002   0.002   0.002   0.002   0.002
     0.002   0.002   0.002
# NERTRIAL
  100
# NREQUIL
  0
# CONT
  1
END
```

Because we will perform Hamiltonian-replica-exchange and not temperature replica exchange, the number of replica exchange temperatures is set to `NRET=1`. We thus also only have one `RET` value which is the temperature of each replica, in this case equal to 298 K. We do not need to scale temperatures after exchange trials, so `LRESCALE=0`. `NRELAM` is the number of replica-exchange lambda values, which is set to 24 here. For each of the replicas you have to specify at which λ value it should be simulated. These values are given at `RELAM`. We do not know the optimal spread of the λ values of the replicas before actually running the simulations, so as an initial guess we just spread them evenly between λ = 0 and 1. We will keep the standard timestep of 0.002 ps for each λ-replica, as given by `RETS`. In order to keep the simulation time per run short, we set the number of exchange trials per run to `NERTRIAL=100`. Prolonging the simulations can then be done by performing multiple runs sequentially. `NREQUIL` sets the number of exchange periods for equilibration, during which no switches between the replicas are allowed. This would be especially beneficial if you start the HREMD simulations from a single configuration and you need time to equilibrate. Since we start from multiple configurations which are close to their respective λ values, we will set this to 0. `CONT=1`, as we start from multiple configuration files, instead of a single configuration. The next step will be to select the appropriate configurations from the slow-growth simulation as initial configurations for each of the replicas.

The script `write_start_files.py` in the directory `starting_coordinates` will help with this. The program will find the λ values of the replicas, look for the DF restraint in the `PERTDFRESSPEC` block in the distance restraints file and determine the restraining distances for each of the replicas from this information. It will then search for the frame in the slow-growth simulation which has a DF distance that is closest to the restraining distance of this replica. This frame will be written to a separate file for each of the replicas, named `start_{X}.cnf`, where `X` will be the replica number. An example argument file is provided which lists the topology of the system, the DF distances over time of the slow-growth simulation (`df_dist.dat`), the coordinate trajectory from the slow-growth simulation, the HREMD

https://doi.org/10.33011/livecoms.6.1.3485
Living J. Comp. Mol. Sci. 2025, 6(1), 3485

input file and the distance restraint specification file as will be used for the HREMD simulation. To run the script:

```
$ ./write_start_files.py
    @write_start_files.arg
```

The starting coordinates for the HREMD simulation are now available and we can prepare the run files with `mk_script`:

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

The HREMD simulations are split into multiple runs, in order to prevent an extremely long single simulation. Go to the first directory, `run1` and submit `HREMD_1.run` to a computer cluster, preferably with the same number of CPUs as we have replicas, which would be 24 in the prepared example.

3.3.4 Analysis of HREMD

All analyses for HREMD simulations will be performed in the subfolder `analysis`. In order to make sure the choice of replicas is appropriate, we will analyze whether there were sufficient switches between the replicas. This can be done already after only a few runs have finished. The switching information of the replicas is written out to the `replica.dat` file for each of the runs separately. You can combine them into a single file by using the provided script:

```
$ ./gather_replica_data.sh [nr_runs]
    [nr_trials]
```

Here, you need to replace [nr_runs] with the number of runs that have already been performed and [nr_trials] with the number of exchange trials per run, which was set to 100 in the example files. This script results in the file `replica_all.dat`. The switches of the replicas can now be visualized by running

```
$ ./rep_ana_mpi.py replica_all.dat
```

The resulting `rep_change.out` file can be opened with Xmgrace. An example of such a file is shown in Figure 7.

In case there is a pair of λ points for which not enough switches are occurring, you have two options to resolve this. You can either insert another λ point or make the difference between the λ points smaller. The former option may be quicker to set up, but requires longer simulation time because of the additional replica. The latter option does not require more replicas, but it is not guaranteed that your small change improves the switching probabilities and that you do not introduce another region of low switching probability due to the change. Of course you can also use more elaborate methods to optimize the λ-spacing [43].

If you are happy with the switching probabilities, you can start preparing for the calculation of the Free Energy Curve (FEC). First, we have to write out the measured DF distances

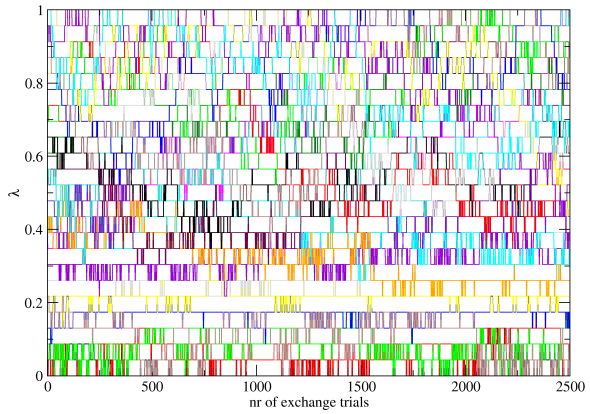

Figure 7. Replica exchanges during time. Different colors represent different replicas.

for each replica. Then we calculate the distributions of these and on this data we can perform the weighted histogram analysis method (WHAM) which will result in the FEC. We will again use the program `trs_ana` to extract the DF distance from the special trajectories (`*trs.gz`). A small script is provided which runs this program for each of the replicas, thereby collecting data from each of the runs.

```
$ ./do_all_trs_ana.sh [nr_runs]
    [nr_replicas]
```

This will generate a subdirectory called `df_dist` which then contains `df_dist_X.dat` files for each replica X. From these distance files, we will first generate the distributions, which can then be used to determine the FEC by applying WHAM. The program `tcf` can generate distributions for each of the `df_dist` files. We will set the boundaries to 0 and 5 nm (the same range as the restraining distances) and use 200 bins. Especially the boundaries should be adjusted when working on different systems. Again, a small script is prepared which will perform the program `tcf` on each of the replicas:

```
$ ./do_tcf.sh [nr_replicas]
```

One should always check whether the distributions at adjacent λ-values are sufficiently overlapping and whether the individual distributions are sufficiently sampled. We can then determine the FEC $F(r)$ by using the WHAM program. Note that the FEC contains the Jacobian contribution, whereas a PMF does not [44]. As input parameters, the WHAM program needs the temperature of the simulation, the restraining distance for each replica and the force constant of the restraints. All this information is read from the `HREMD.imd` and `disres.dat` files as specified in the `do_wham.sh` file:

```
$ ./do_wham.sh
```


https://doi.org/10.33011/livecoms.6.1.3485
Living J. Comp. Mol. Sci. 2025, 6(1), 3485

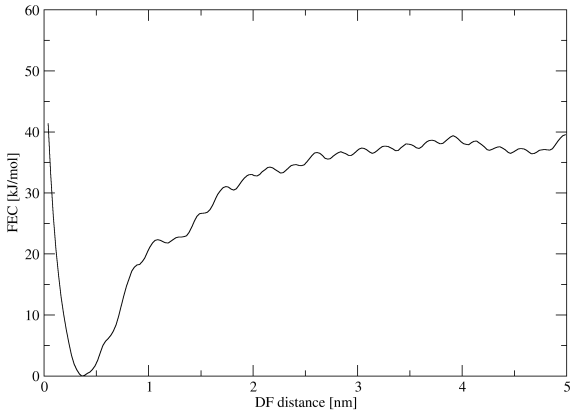

Figure 8. Free Energy Curve (FEC) along the DF distance as obtained from HREMD simulations, for the PLA2-ASA system.

This script also moves the final FEC on the y-axis such that its minimum is placed at 0 kJ/mol. The file `wham_FEC_200bins_5000iter_min0.dat` now contains the final FEC, which is shown in Fig. 8. As you can see, the curve is not completely flat at larger distances, but is rather noisy. Ideally, you would have to change the spacing of the replicas, add more replicas in the unbound range, or lower the maximum distance which you restrain such that the replicas are placed more densely in the unbound range. However, we can still see where the plateau of the unbound range is, so we will go ahead with the calculation of the binding free energy.

3.3.5 Calculation of binding free energy

To derive the binding free energy from the FEC in Fig. 8 we cannot simply take the value of ΔG at the plateau around the unbound state. We will need to integrate the bound and unbound ranges of the FEC and we will need to include the standard state correction of

$$\Delta G_{\text{std}} = -k_{\text{B}}T \ln\left(\frac{V_{\text{u}}}{V^{\circ}}\right) \tag{11}$$

Here, V° is the standard state volume of 1.661 nm^3 and V_{u} is the unbound volume which is sampled by the ligand in the unbound range. This range is defined by the plateau observed in the FEC curve.

In order to determine V_{u} we need to select the configurations of the trajectory which contributed to the plateau range of the FEC. In this example, the plateau can be observed between 3 and 5 nm. The script `select_frames_unbound_region.sh` (which you can find in the subdirectory `unboundVolume`) will select the appropriate frames by reading in the `df_dist_X.dat` files which were generated using `do_all_trs_ana.sh`. Since the DF distances are written out every 50 steps (`NTWDF=50`) and the coordinates only every 1000 steps (`NTWE=1000`), the script filters the `df` distance file such that it matches the timesteps of the coordinate trajectory. In order to run the program, specify the number of replicas, the number of runs and the boundaries of the unbound range:

```
$ ./select_frames_unbound_region.sh 24
    [nr_runs] 3.0 5.0
```

A list of the selected configurations (`list_frames_unbound_region.txt`) as well as the trajectory with only these configurations (`unbound_region_frames.trj`) are written to separate directories for each of the replicas. We will now combine all the configurations from the unbound range and determine how much volume was sampled by the ligand by using the program `iondens`. `iondens` calculates the average density of ions (ligand in our case) from a trajectory file. For the example, here you can start it with

```
$ ./do_iondens.sh
```

where we use the final configuration from the equilibration run as a reference configuration. Some parameters in `do_iondens.sh` have been set as appropriate for the current system. For example, the particle that we will be monitoring now will not be an ion, but the centre of geometry (cog) of the atoms C2 and C3 of the aspirin ligand. The grid spacing is set to 0.1 nm, such that a single grid point corresponds to 1 Å^3. The thresholds are set very low, such that we pick up single occupancies of the grid points. The results are written out to multiple files, but we are interested only in the file `grid.pdb`. This file contains one line for each of the grid points that have been sampled by the particle at least once. Since we have chosen the grid spacing such that each point corresponds to 1 Å^3, the number of different grid points that have been visited (number of lines in the file) corresponds to the unbound volume (in Å^3) which was sampled by the ligand during the simulations. For the current example (5 ns HREMD simulation with the unbound range chosen between 3 and 5 nm), the number of visited grid points is 11 258 which equals to a sampled unbound volume of 11.3 nm^3.

We can now determine the raw binding free energy from the WHAM results and determine the standard state correction with the sampled unbound volume which we have just obtained. To perform this calculation, we will use the program `calc_dG_corrected.py` which you can find in the `analysis` directory. Before running the program, be sure to modify the argument file `calc_dG_corrected.arg` to your data. It should contain the file name of the WHAM results, the start of the bound range (in nm), the end of the bound range (in nm), the start of the unbound range (in nm), the end of the unbound range (in nm) and the sampled unbound volume (in nm^3), each on a separate line. You can now run the program with

https://doi.org/10.33011/livecoms.6.1.3485
Living J. Comp. Mol. Sci. 2025, 6(1), 3485

```
$ ./calc_dG_corrected.py
    @calc_dG_corrected.arg
```

This program will determine the raw binding free energy from the FEC $F(r)$ obtained with WHAM, the standard state correction and the final binding free energy:

$$\begin{aligned}\Delta G^{\circ}_{\text{bind}} &= \Delta G^{\text{raw}}_{\text{bind}} + \Delta G_{\text{std}} \\ &= -k_{\text{B}}T \ln\left(\frac{\int_{\text{b}} \text{d}r\, e^{-F(r)/k_{\text{B}}T}}{\int_{\text{u}} \text{d}r\, e^{-F(r)/k_{\text{B}}T}}\right) - k_{\text{B}}T \ln\left(\frac{V_{\text{u}}}{V^{\circ}}\right)\end{aligned} \tag{12}$$

It also prints the standard state correction and the final binding free energy. Note that in [41], $F(r)$ is referred to as $\Delta G_{\text{WHAM}}(r)$. The expression (Eq. 12) used in that paper to calculate the binding free energy is obtained when shifting $F(r)$ to become $\hat{F}(r) = F(r) + C$, such that $\int_{\text{b}} \text{d}r\, e^{-\hat{F}(r)/k_{\text{B}}T}$ equals 1. This is achieved for

$$C = k_{\text{B}}T \ln \int_{\text{b}} \text{d}r\, e^{-F(r)/k_{\text{B}}T} \tag{13}$$

Note that the minus sign in Eq. 12 of ref [41] should actually be a plus sign.

We have performed the prepared HREMD simulations for 5 ns and obtained $\Delta G^{\text{raw}}_{\text{bind}}$ = –31.9 kJ/mol, ΔG_{std} = –4.7 kJ/mol and $\Delta G^{\circ}_{\text{bind}}$ = –36.7 kJ/mol. The final result is similar to what we found in the previous tutorial (-32.3 kJ/mol), but deviates a bit more from the experimental estimate of -29.6 kJ/mol [23]. As mentioned before, the spacing of the replicas is not optimal in the current example. This can influence both the convergence of the FEC and final binding free energies. An improvement of the accuracy of the final binding free energy can thus likely be obtained by optimizing the spacing of the replicas, adding more replicas and/or prolonging the simulations.

3.3.6 Common errors

Closing of the binding site

In some cases the binding site closes up, preventing a DF path from the active site to the solvent. As a result, all grid points outside the protein will get the additional protein penalty as defined with the parameter `PROTEINOFFSET`. If the site remains buried for some time, this becomes problematic especially if the ligand is unbound. Since all grid points outside the protein have the large `PROTEINOFFSET`, the DF distance basically becomes a radial distance again, and ligands could be forced to move through the protein instead of around it.

This issue is easily detected by a large jump in the DF distance as soon as the binding site closes.

A proper definition of the zero-distance point helps prevent the closure of the binding site. This point is typically the center of geometry of several atoms in the binding site. If your binding is closing during the simulation, try the following tips for the definition of the zero-distance point:

- Choose more stable atoms (e.g. backbone atoms) to ensure that the zero-distance point remains fixed and unaffected by side chain movements.
- Use more atoms to define the zero-distance point to distribute the restraint force and reduce distortion of the active site.

3.4 Tutorial 4: Selective Gaussian accelerated MD (GaMD)

Molecular dynamics simulations often struggle to sufficiently sample the events of interest in the accessible simulation timescales [45]. This is due to high energy barriers separating the desired minima of the energy landscape [46]. Enhanced sampling techniques make it possible to cross these barriers by adding a bias. Several enhanced sampling techniques have been developed and improved over the years, each of them with their respective advantages and limitations [47]. Gaussian accelerated MD is a recently developed technique that allows increasing the sampling without the need for *a priori* knowledge of the cause of the energy barriers. GaMD works by adding a boosting potential-energy term that flattens the energy landscape.

$$\Delta V(r) = \begin{cases} \frac{1}{2}K\,(E - V(r))^2 & \text{if } V(r) < E \\ 0 & \text{if } V(r) \geq E \end{cases} \tag{14}$$

Where $\Delta V(r)$ is the boosting potential energy added to the system, E is the energy threshold and K is the force constant [48]. The potential-energy term used leads to a Gaussian distribution of ΔV, making the reweighting process easier through the use of cumulant expansion to the second order [49]. Both acceleration parameters (E and K) can be easily obtained through a search simulation. The energy threshold E, can be defined as $E = V_{max}$ (lower bound) or as $E = V_{min} + \frac{1}{K}$, where V_{max} and V_{min} are the maximum and minimum energy observed in the search simulation. K is defined as:

$$K \equiv K_0 \frac{1}{V_{max} - V_{min}} \tag{15}$$

Where K_0 can be calculated as:

$$K_0 = min\left(1.0, \frac{\sigma_0}{\sigma_V}\frac{V_{max} - V_{min}}{V_{max} - V_{avg}}\right) \tag{16}$$

When E is set to the lower bound or as:

$$K_0 = K_0'' \equiv \left(1 - \frac{\sigma_0}{\sigma_V}\right)\frac{V_{max} - V_{min}}{V_{max} - V_{avg}} \tag{17}$$

When E is set to the upper bound. If K_0'' is not found between 0 and 1 then K_0 is calculated with Eq. (16). σ_0 corresponds

https://doi.org/10.33011/livecoms.6.1.3485
Living J. Comp. Mol. Sci. 2025, 6(1), 3485

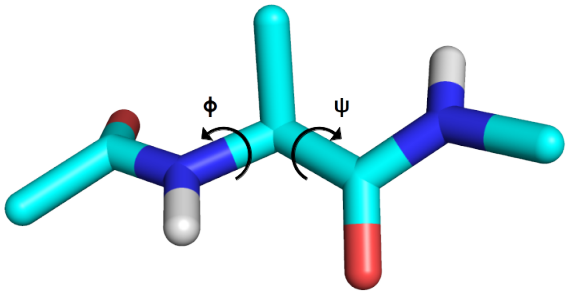

Figure 9. Stick representation of alanine dipeptide with the ϕ and ψ dihedrals highlighted.

to a user-specified upper limit for the $\sigma_{\Delta V}$ to ensure a narrow distribution of the boosting potential energy. In addition, GaMD offers different types of acceleration. One can accelerate the whole potential-energy term, only the dihedral term, or both separately (dual boosting).

Recent improvements of the methodology have been developed, among them, "selective" GaMD in which instead of adding the boosting energy to the potential energy of the whole system, the boosting is selectively applied to a subset of the atoms of the system. These selective approaches allow for a stronger enhancement of the events of interest with no additional computational cost [50–53]. The cited methodologies allow selective acceleration of small ligands, bound peptides and the interactions between two proteins. The selective GaMD methodology that will be used in this tutorial focuses on offering full flexibility in defining the regions that one wants to accelerate.

In this tutorial, we will use both the standard GaMD approach as well as a showcase of the selective GaMD functionality on the alanine dipeptide. All the boosting potential-energy terms used follow the dual boosting approach and the energy threshold is set to the lower bound, since it is the most commonly used setup. GROMOS is compatible with all the other approaches, and the tutorial can be run with any of them by simply adapting the input files.

3.4.1 Preparations

In this tutorial, we will use alanine dipeptide as a test system. The alanine dipeptide is an extensively studied system that has been used to validate different GaMD implementations [48, 54, 55]. In this tutorial, we will compute the potential of mean force (PMF) over the dihedrals ϕ and ψ of the alanine dipeptide (Figure 9).

The preparation of the topology, coordinates, the energy minimization and the equilibration of the system follow the same procedure as described in Tutorial 1. The final equilibrated structure can be found in the directory `eq` with the name `aladip_equilibrated.cnf`.

3.4.2 GaMD regions definition

The first step in performing GaMD simulations is to create a GaMD input file containing all the information about which groups of interactions to accelerate and how they need to be accelerated. In the directory `gamd_setup`, you will find two files, one with the definitions to run standard GaMD, `ala.gamd`, and one showcasing the selective GaMD functionality, `selective_ala.gamd`. Take a look at the block `GAMDATOMS` of both files.

```
# Standard GaMD:
------------------------------
GAMDATOMS
1
#  INATOM    FINATOM    AGROUP
         1       3840         1
END
------------------------------
# Selective GaMD:
------------------------------
GAMDATOMS
2
#  INATOM    FINATOM    AGROUP
         1         12         1
        13       3840         2
END
------------------------------
```

With `GAMDATOMS` you specify the number of groups of atoms to account for separately. The next line indicates the index of the first atom of the group, `INATOM`, the final atom, `FINATOM`, and an integer that will be assigned to that group, `AGROUP` (groups of atoms with the same `AGROUP` index will be considered a single group). In the case of standard GaMD, only one set of atoms is defined. For selective GaMD, the user can define as many atom groups as desired. In this tutorial, two atom groups are defined, one containing the solute (`AGROUP` = 1), and one containing the solvent (`AGROUP` = 2). Now take a look at the second block of the `.gamd` files.

```
# Standard GaMD:
------------------------------
GAMDGROUPS
1
# AGROUP_1    AGROUP_2    ACCELGROUP
  1           1           1
END
------------------------------
# Selective GaMD:
------------------------------
GAMDGROUPS
1
# AGROUP_1    AGROUP_2     ACCELGROUP
```


https://doi.org/10.33011/livecoms.6.1.3485
Living J. Comp. Mol. Sci. 2025, 6(1), 3485

```
  1            1            1
  1            2            1
END
------------------------------
```

This block defines the number of acceleration potential-energy terms to use, and to which interactions they should be applied. In a similar fashion as with the previous block, GAMDGROUPS specifies the number of distinct acceleration potential-energy terms to use. The next lines assign each group of interactions to their corresponding acceleration group where AGROUP_1 and AGROUP_2 are the indexes of the atom groups defined in the previous block and ACCELGROUP is the index for the acceleration potential-energy term. In the case of standard GaMD, we have only one acceleration group containing the interactions of all the atoms of the system. For the selective GaMD, the interactions of the alanine dipeptide with itself (AGROUP_1 = 1, AGROUP_2 = 1) and the interactions of the alanine dipeptide with the solvent (AGROUP_1 = 1, AGROUP_2 = 2) are assigned to one acceleration group. The interactions of the solvent with itself (AGROUP_1 = 2, AGROUP_2 = 2) are not assigned to any group and thus not accelerated. The same behaviour can be obtained by using ACCELGROUP = 0, which is never accelerated.

3.4.3 Acceleration parameter search

In order to get the acceleration parameters (*E* and *K*) a search simulation needs to be performed. The search starts with a conventional MD (cMD) simulation in which the necessary statistics are recorded. After the cMD search, a starting set of *E* and *K* parameters is computed and applied to the system. After an equilibration phase, the statistics are collected again and the acceleration parameters are constantly updated. After the adaptive GaMD search, the final acceleration parameters are collected to be used for the production simulations. In this tutorial we will use 0.1 ns of cMD search followed by 0.3 ns of GaMD search. For a real-case scenario, the search phase must be run for longer times, at least 10 times longer than the searches described here. Go to the directory `search`, where you will find two directories, one for the search of the standard GaMD methodology `gamd`, and one for the selective approach `selective_gamd`. Take a look at the GAMD block of the input file.

```
GAMD
  1
# SEARCH   FORM   THRESH   NTIGAMDS
       1      1        1          1
# AGROUPS   IGROUPS
       1     1
# DIHSTD   TOTSTD
  24.79    24.79
#ED
  0
#ET
  0
#KD
  0
#KT
  0
#EQSTEPS
  0
#WINDOW
  0
END
```

The fourth line specifies the general parameters for the GaMD simulation, such as the search algorithm to use (SEARCH), whether the dihedral term has to be accelerated separately (FORM), whether the energy threshold used is an upper or lower bound (THRESH), and whether the search needs to be initialized (NTIGAMDS). For this tutorial, we will accelerate dihedral and total potential-energy terms separately (dual boosting approach) and the energy threshold will be set at the lower bound ($E = V_{max}$). The next line contains the number of atom groups and interaction groups or acceleration groups defined in the `gamd` file. DIHSTD and TOTSTD correspond to the values of σ_0 for the dihedral term boosting and the total potential-energy term boosting respectively. The variables *ED*, *ET*, *KD* and *KT* are the acceleration parameters that define the boosting energy that will be applied to the system. *ED* and *ET* correspond to the list of energy thresholds for the dihedral terms and the potential energy of the system respectively, while *KD* and *KT* correspond to the list of force constants of the boosting energy applied to the dihedral term and to the potential-energy term. Since this is a search run, all the parameters can be set to 0, for a production run, these parameters will need to be updated for the ones found during the search. Since each acceleration region requires its own parameters, if more than one acceleration region is defined, one must provide extra *ED*, *ET*, *KD* and *KT* parameters. Finally, EQSTEPS corresponds to the number of equilibration steps to use before starting to collect statistics of the search simulation and WINDOW is the size of the window used to collect the parameters, when WINDOW equals 0 the whole simulation is used to compute the needed statistics.

All input files are now prepared and we can generate the run file with:

```
$ mk_script @f mkscript_gamd_search.arg
```

The job file `gamd_search.jobs` provided to `mk_script` shows the parameters that change between search runs.

https://doi.org/10.33011/livecoms.6.1.3485
Living J. Comp. Mol. Sci. 2025, 6(1), 3485

For this tutorial, we provide two `mk_script` libraries, one to prepare the run files for a cluster using SLURM as a queueing system (`libsmkscript.lib`), and one for running the simulations locally (`libsmkscript_local.lib`). If the CUDA acceleration code is used, one simply needs to uncomment the block `INNERLOOP` from the `imd` files before running `mk_script`.

After running the GaMD simulation, we can extract the acceleration parameters by using `ene_ana`. The analysis files are provided in the folder `ana`. We will use this program to read out the energy thresholds and force constants from the last energy trajectory.

You can run the program with

```
$ ene_ana @f ene_ana.arg
```

The file `ene_ana.arg` contains the information about which parameters to extract from the energy trajectory. The definitions of those parameters can be found in the library file `ene_ana.md++.lib`. This program will generate the trajectories of the selected parameters. The name of the parameters use the following syntax, "gamd_" (to indicate that it is a gamd parameter), followed by parameter to extract (in lower case), followed by the number of the acceleration region. For example, the energy threshold for the dihedrals (*ED*) for the first acceleration group will be `gamd_ed1`.

3.4.4 GaMD production run

Now that the search is complete, we can run the production run. Go to the directory `prod`. Similar to the `search` directory, you will find two directories, one for the standard GaMD methodology `gamd`, and one for the selective approach `selective_gamd`. The same `imd` file used in the search runs can be used for the production ones. However, the search algorithm needs to be turned off (`SEARCH` = 0) and the acceleration parameters obtained from the search run need to be added. For this tutorial, since the search run was not long enough to obtain optimal acceleration parameters, the `imd` files already contain all the necessary acceleration parameters, estimated from a longer GaMD search. We will run a production simulation for a total of 1 ns, although a true production run must be much longer.

The run files can be created in the same way as in the search runs by using `mk_script` with its corresponding `.arg` file.

3.4.5 GaMD analysis

All the analyses for the GaMD simulations will be performed in the subfolder `ana`. In this section of the tutorial, we will calculate the reweighted PMF for the ϕ and ψ dihedrals of the alanine dipeptide. The same procedure can be used with any other collective variable of interest. To extract the dihedral angle time series, first we need to run the program `tser` with the correct argument files. You will find two argument files in the folder, one for each dihedral of interest, e.g.: `phi_dihedral_tser.arg`.

```
$ tser @f phi_dihedral_tser.arg > phi.out
```

The argument files already contain the information needed to compute the dihedrals over time accounting for periodicity, providing values from -180 to 180 degrees. The next step is to extract the trajectory of the biasing potential-energy term used to be able to reweight the obtained dihedral time series. This can be achieved by using the program `ene_ana` in a similar fashion to the extraction of the acceleration parameters from the search. The values of interest are `totgamd_dV` which contains the total boosting potential energy in kJ/mol, `totunbiased`, which contains the total potential energy excluding the boosting energy and `totpot` which contains the total potential energy. The boosting energy for each acceleration region can also be extracted independently by adding `gamd_dVi` to the `ene_ana` property list `@prop` with the `i` sub-index indicating the acceleration group of interest. After obtaining the collective variables (CV) and biasing potential-energy trajectories, the next step is to reweight and plot them. We will do this using both exponential reweighting and cumulant expansion to the second order. In the exponential reweighting, the time series of observed values of X (in this case the dihedral angles) sampled during the biased simulation R can be reweighted to the unbiased state Y using the following equation:

$$\langle X \rangle_Y = \frac{\langle X \exp\left[-\beta\left(V_Y - V_R\right)\right]\rangle_R}{\langle \exp\left[-\beta\left(V_Y - V_R\right)\right]\rangle_R} = \langle X \exp\left[-\beta\left(V_Y - V_R - \Delta F_{YR}\right)\right]\rangle_R \tag{18}$$

where $\Delta F_{YR} = F_Y - F_R$.

The GROMOS toolkit offers a program called `reweight` that can be used to reweight time series of observed properties and produce a histogram of the selected number of bins. During the reweighting process special care is taken in order to avoid overflow [56]. A sample input file is provided under the name `reweight.arg`. Note that the reweight program only accepts a single time series at a time. The script `ExponentialReweighing.py` provided in the `scripts` folder can be used to reweigh the probability distribution of interest $p_R(\phi, \psi)$. `ExponentialReweighing.py` first discretizes the two-dimensional dihedral angle matrix into a one-dimensional time series. The probability distribution of this time series is then computed and reweighted using the `reweight` program. Finally, the one dimensional reweighted probability distribution is mapped back to a two dimensional probability distribution, $p_Y(\phi, \psi)$, and converted to energies to be able to plot them.

https://doi.org/10.33011/livecoms.6.1.3485
Living J. Comp. Mol. Sci. 2025, 6(1), 3485

```
$ python ExponentialReweighing.py --cv1
    phi.out --cv2 psi.out --vr totpot.dat
    --vy totunbiased.dat --xdim -180 180 10
    --ydim -180 180 10
```

An alternative way of reweighting the trajectories would be to use cumulant expansion to the second order to approximate the reweighting factor. The reweighted free energy $F_Y(\phi, \psi)$ will then be calculated as:

$$F_Y(\phi, \psi) = F_R(\phi, \psi) - \sum_{k=1}^{2} \frac{\beta^k}{k!} C_k + F_c \tag{19}$$

where F_c is a constant, $F_R(\phi, \psi)$ is the free energy obtained from the unreweighted trajectory, $F_R(\phi, \psi) = -k_B T \ln p_R(\phi, \psi)$ and the first two cumulants are:

$$\begin{aligned} C_1 &= \langle \Delta V \rangle, \\ C_2 &= \langle \Delta V^2 \rangle - \langle \Delta V \rangle^2 = \sigma_v^2 \end{aligned} \tag{20}$$

This can be achieved by using the `pyreweight` script that is provided in the `scripts` directory or that can be downloaded from its original repository [49].

The version provided in the `scripts` folder contains a small modification to provide the results in kJ/mol to ease the comparison with the results provided by GROMOS. The script takes as input a file with the CVs in columns and another file with the biasing potential energy in three columns, biasing in kJ/mol, time, and biasing in kcal/mol. An example of how to run the script can be found in the `scripts` folder in the file `pyreweight_command.txt`.

Because the trajectories performed are too short to obtain converged PMFs, the dihedral and energy trajectories of a 100 ns run are provided in the subfolder `data`. The same analysis can be performed on the files provided there, producing the 2D-PMFs in Figure 10.

3.4.6 Common errors

Selection of parameters

A common source of error in GaMD simulations is an undesired or erroneous parameter selection for acceleration and interaction groups. If the selected inputs are incompatible, the program will terminate with an error. More concerning, however, is the possibility that the simulation completes successfully but produces accelerations that deviate from those intended by the user. It is therefore recommended to ensure the following:

- All atoms in the system are assigned to at least one interaction group.
- No atom is assigned to more than one acceleration group.

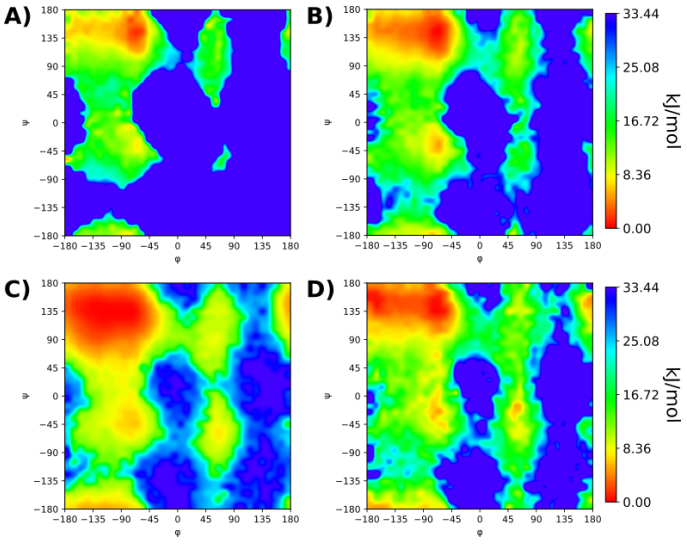

Figure 10. Potential of Mean Force (PMF) profiles over the ϕ and ψ angles of the alanine dipeptide. A) Simulation using the standard GaMD approach reweighted with cumulant expansion to the second order. B) Simulation using the standard GaMD approach reweighted with exponential reweighing. C) Simulation using the selective GaMD approach reweighted with cumulant expansion to the second order. D) Simulation using the selective GaMD approach reweighted with exponential reweighting.

- All interactions between the defined interaction groups are assigned to one acceleration group. If no acceleration is desired for a given interaction group, use acceleration group 0.
- In total, one should have:

$$\binom{n+1}{2} = \frac{n(n+1)}{2} \tag{21}$$

sets of interaction pairs where n is the number of interaction groups.
- A set of force constants (K) and energy thresholds (E) should be provided for each defined acceleration group.

SHAKE errors

Since GaMD pumps energy into the system, although rarely, this can lead to the system blowing up, resulting in SHAKE errors.

A solution for this problem is to decrease the amount of acceleration used on the system or to decrease the integration time step (e.g. from 2 fs to 1 fs) for a few steps to allow the system to stabilize.

3.5 Tutorial 5: Accelerated Enveloping Distribution Sampling (AEDS)

Alchemical free-energy methods have been established as an irreplaceable tool in computational drug design. A vast vari-

https://doi.org/10.33011/livecoms.6.1.3485
Living J. Comp. Mol. Sci. 2025, 6(1), 3485

ety of methods to perform alchemical transformations exists. We have already seen the extended-TI approach outlined in section 3.2.4. One family of free-energy methods is the free-energy perturbation (FEP) based on Zwanzig's equation (eq. 22) [57]. Defining the free-energy difference $\Delta G_{i,j}$ between two states i and j, represented by Hamiltonians H_i and H_j over the positions $\vec{r}$ as an ensemble average over state i.

$$\begin{aligned}\Delta G_{i,j} &= G_j - G_i \\ &= -k_B T \ln \left\langle e^{-(H_j(\vec{r})-H_i(\vec{r}))/k_B T} \right\rangle_i \end{aligned} \tag{22}$$

As mentioned in section 3.2 sampling relevant phase-space is of utmost importance to converge to plausible results. While methods like TI [25] or BAR [28] aim to connect physical states through non-physical states, one-step perturbation methods take a different approach. Here, a reference Hamiltonian, constructed from the end-states, is sampled. This avoids spending simulation time on non-relevant states. This method fits perfectly to investigate closely related systems like derivates. In this tutorial, we will take a look at accelerated enveloping distribution sampling (AEDS) [58, 59]. In enveloping distribution sampling a reference Hamiltonian $H_R(\vec{r})$ is constructed by combining several different end-state Hamiltonians $H_i(\vec{r})$. To achieve equal sampling of each of the end-states, $H_i(\vec{r})$ is biased by an offset ΔF_i^R. The partition function of $H_R(\vec{r})$ results now as the sum of the biased end-state partition functions (eq. 23).

$$H_R(\vec{r}) = -k_B T \ln \left(\sum_{i=1}^{N} e^{-(H_i(\vec{r})-\Delta F_i^R)/k_B T} \right) \tag{23}$$

Since this approach can lead to high energy barriers between the minima and thus to sampling problems, the original EDS version introduced a smoothing parameter s (eq. 24) to smoothen the energy landscape.

$$H_R(\vec{r}) = -\frac{k_B T}{s} \ln \left(\sum_{i=1}^{N} e^{-s(H_i(\vec{r})-\Delta F_i^R)/k_B T} \right) \tag{24}$$

This can lead to a deformation of the energy landscape, without preserving the position of the energy minima. Therefore, a refined approach using a boosting potential-energy term was proposed [58, 59]. Similar to Gaussian-accelerated MD, a harmonic boosting potential-energy term is used to accelerate defined regions, namely the non-bonded interactions of the perturbed atoms. The accelerated reference Hamiltonian $H_R^*(\vec{r})$ is defined as shown in equation 25, where the acceleration range is defined by E_{max} and E_{min}.

$$H_R^*(\vec{r}) = \begin{cases} H_R(\vec{r}) - \frac{E_{max}-E_{min}}{2} & \text{if } H_R(\vec{r}) \geq E_{max} \\ H_R(\vec{r}) - \frac{1}{2(E_{max}-E_{min})}(H_R(\vec{r}) - E_{min})^2 & \text{if } E_{min} < H_R(\vec{r}) < E_{max} \\ H_R(\vec{r}) & \text{if } H_R(\vec{r}) \leq E_{min} \end{cases} \tag{25}$$

To obtain parameters for E_{max}, E_{min}, and the offsets, a parameter search simulation is necessary. In the search simulation, the currently sampled state is defined as the state with the lowest energy for $H_i(\vec{r}) - \Delta F_i^R$. E_{max} is defined as the maximum observed transition energy after a full state round trip, meaning that each state was sampled at least once. E_{min} is based on the fluctuations of $H_i(\vec{r})$ of the state with the lowest energy. The offsets have different definitions depending on the search algorithm in use. Generally, the difference between two offsets $\Delta\Delta F_{ij}$ is the free-energy difference between the accelerated end-states i and j. In older iterations of the search algorithm, the offsets were estimated as the free-energy difference between an accelerated end-state and the accelerated reference state and afterward anchored to one of the end-states. The search algorithm used in this tutorial uses an approach similar to local elevation by continuously adding to the offsets based on their sampling probability. In this approach, all offsets are averaged around zero. This has proven to be a more robust search algorithm and removed the dependency on the anchor state. The parameters obtained from the search simulation are then used in a production run.

3.5.1 Preparations

In this tutorial, we will use two SPC water molecules as a test system. We will use AEDS to calculate free-energy differences and use it as a tool for water probing in a protein-ligand system [60].

The preparation of the topology, coordinates, energy minimization, and equilibration of the system follows the same procedure as described in Tutorial 1. The final equilibrated structures can be found in the directory `eq`. The perturbation topologies can be created as shown in Tutorial 2 and are provided in the directory `topo`.

3.5.2 Acceleration parameter search

To get the acceleration parameters(E_{max}, E_{min}, offsets), a search simulation needs to be performed. The search run starts from an equilibrated structure. If no initial parameters are given, the search starts as a conventional MD simulation. After each simulation step E_{max}, E_{min}, and the offsets are updated. After approximately 5-25 ns, the parameter search simulation converges. The needed simulation time depends mostly on the complexity of the system and the perturbation since we are dependent on visiting all end-states to adjust

https://doi.org/10.33011/livecoms.6.1.3485
Living J. Comp. Mol. Sci. 2025, 6(1), 3485

the acceleration parameters. In this tutorial, we will run a very short search simulation of 0.5 ns and use parameters from a longer simulation for the production run.

Go to the directory `search`. In this folder, you will find three files, namely `aeds.arg`, `aeds.job`, and `aeds.imd`. Let's start with the `aeds.imd` file. This file is a simulation input file we already know from previous tutorials. At the bottom of the file, we have an additional `AEDS` block.

```
AEDS
# AEDS
     1
# FORM   NUMSTATES
     5          4
# EMAX   EMIN
     0      0
# EIR [1..NUMSTATES]
    0    0    0    0
# NTIAEDSS   RESTREMIN    BMAXTYPE     BMAX
          1           1          2        2
# ASTEPS     BSTEPS
    1000      10000
END
```

In the second line, we set `FORM` to 5, specifying an all-parameter search using the adaptive search. Later we will change this to `FORM` 1 to run a production run. Other options are described in the GROMOS manual. `NUMSTATES` specifies the number of end-states in this simulation. Since we perturb two SPC water molecules, A and B, we get four possible end-states as all combinations of A and B being present or dummies as shown in table 3.

In the third line, we set both `EMAX` and `EMIN` to zero as well as for the offsets in the line underneath. If we had prior information about these values we could give them as a starting point for the search instead of starting from scratch. Line five is specific to search runs. With `NTIAEDSS` 1 we initialize the search run. You will see that we turn it off in the jobscript after the first initialization. `RESTREMIN` restricts E_{min} to the minimum average end-state energy before all states have been visited at least once. `BMAXTYPE` and `BMAX` are used to control the estimated maximum barrier height. By default, we set both of them to 2. `ASTEPS` is used to control the change rate in offsets. A higher number means a faster change in offsets. `BSTEPS` is used for initial free-energy guesses in other search algorithms. In the argument file, we need to specify the folder location and the location of the md binaries as absolute paths. The `imd` file uses the `INNERLOOP` block for CUDA support. If the used GROMOS binary does not support CUDA, comment out the block using hashtags. The jobfile is used to create ten `imd` and `run` files of the same simulation length as specified in `aeds.imd`. The only changes in the settings are

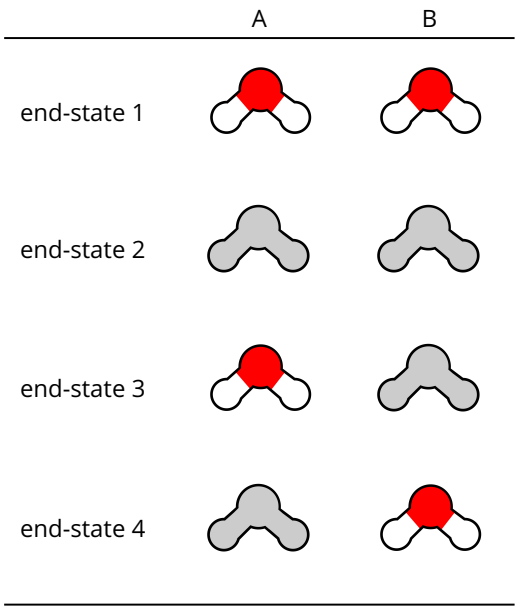

Table 3. Showing the resulting end-states 1-4 for the perturbation of two SPC water molecules A and B.

that `NTIAEDSS` is set to zero after the first jobscript.

With all input files prepared, we can now generate the run files using `mk_script` from `gromos++` package with:

```
$ mk_script @f aeds.arg
```

For this tutorial, we provide two `mk_script` libraries, one to prepare the run files for a cluster using SLURM as a queueing system (`libs/mk_script.lib`) and one for running the simulations locally (`libs/mk_script_local.lib`).

After running the parameter search simulation, we can extract the acceleration parameters using `ene_ana`. The analysis files are provided in the subfolder `ana` in the search directory. We will use this program to read out the energy and offset of each end-state as well as E_{max} and E_{min}.

You can run the program with

```
$ ene_ana @f ene_ana.arg
```

The file `ene_ana.arg` contains the information about which parameters to extract from the defined energy trajectories. The definitions of those parameters can be found in the library file `libs/ene_ana.md++.lib`. `ene_ana` will generate trajectories of the selected parameters. We could plot `eds_emin.dat`, `eds_emax.dat`, and `e*r.dat` to check for convergence, where the `*` in `e*r.dat` is to be replaced by the end-state number. These files contain the offsets for each end-state. Another way is to use `search.py` in the `scripts` folder. This script splits the trajectories into ten blocks and gives us the average E_{min} and E_{max} of the last block in the first line. Below, the offsets with an error estimate for each end-state averaged for each block are shown. It also calculates

https://doi.org/10.33011/livecoms.6.1.3485
Living J. Comp. Mol. Sci. 2025, 6(1), 3485

the theoretical offsets, as the free-energy difference to the accelerated reference state, shown in the third column. In an ideal case, the offset values converge to a single value in the later blocks and have a small error estimate. In addition, they should be similar to the theoretical offset meaning the free-energy difference between the accelerated end-state and the accelerated reference state. Keep in mind that the offsets in column one are averaged around zero. To call `search.py` in the `search/ana` folder, do the following:

```
$ python ../../scripts/search.py
```

The output of the longer search run can be found in `search/ana/long`. We will use E_{min} and E_{max} as well as the offset parameters from the last block of each end-state, found in `search.out`.

3.5.3 AEDS production run

With the parameters from the search run, we can now start a production run. From the production run, we can calculate our relative free energies. Go into the directory `prod`. Here, we see two files, `aeds.arg` and `aeds.imd`. We do not use the `@joblist` argument in the argument file, but the `@script` argument to specify the number of jobs. If we take a short look into `aeds.imd` we will spot a few differences:

```
AEDS
# AEDS
     1
# FORM   NUMSTATES
     1          4
# EMAX   EMIN
    43.08   -187.95
# EIR [1..NUMSTATES]
 16.32    -43.66    12.99     14.35
# NTIAEDSS   RESTREMIN    BMAXTYPE     BMAX
          0           1           2        2
# ASTEPS     BSTEPS
    1000      10000
END
```

We now use `FORM` 1 for a production run. We have also inserted values for E_{max}, E_{min} and the offsets. We also set `NTIAEDSS` to zero, which is optional since this line is not read in a production run. Similar to the search run, we now use `mk_script` to create our input files. Again, the `INNERLOOP` block in the `imd` file has to be commented out if the used `gromos` binary does not support `CUDA`. Since we use well converged parameters we can get away with a short production run of 1 ns. In practice, production runs are mostly between 20-100 ns.

3.5.4 Production run analysis

The analysis will be done in the subfolder `prod/ana`. We will do two kinds of analysis. Firstly, we will take a look at the free-energy difference between the end-states using `ene_ana` and `dfmult`. Secondly, we will take a look into the fractional occupancy of the end-states. We will use this information in the water probing section later on. All values shown are from a longer production run. The files can be found in the subfolder `prod/ana/long`. To start with analysis, we will run the following commands:

```
$ ene_ana @f ene_ana.arg
```

and after that we use

```
$ dfmult @f df.arg > df.out
```

`ene_ana` generates the trajectories of the selected parameters as before. We use `dfmult` to calculate free energies and save them in `df.out`. If we take a look at the `df.out` file we will see the following:

```
             #DF (kJ/mol)                 err
DF_1_R      2.1428295e+01      6.9032914e-02
DF_2_1      5.3410399e+01      7.9494696e-02
DF_3_1      2.5792504e+01      3.0323318e-01
DF_4_1      2.6096480e+01      1.9175297e-01
DF_2_R      7.4838694e+01      3.8460413e-02
DF_3_2     -2.7617895e+01      2.9776361e-01
DF_4_2     -2.7313919e+01      1.8298067e-01
DF_3_R      4.7220799e+01      2.9514515e-01
DF_4_3      3.0397569e-01      3.4513025e-01
DF_4_R      4.7524775e+01      1.7868847e-01
```

In each row, we see a specifier like `DF_1_R` meaning end-state 1 to the reference state, the relative free energy in kJ/mol, and an error estimate in kJ/mol. From two waters present to no water present we get a free-energy difference of 53.3 ± 0.1 kJ/mol. From two waters to one water present and one not present we get 25.9 ± 0.3 kJ/mol and 26.2 ± 0.3 kJ/mol. Turning two different single waters into dummies produces free energies in good agreement with each other as well as doing two at the same time which is roughly double the solvation free energy of a single water molecule. The values also agree with values from the literature [60]. To estimate the quality of our free energies as well as to estimate the time of occupancy for each end-state we use the `prevalence.py` script like:

```
$ python ../../scripts/prevalence.py
```

This script gives us an output as follows:

```
ENDSTATE     WEIGHT       PERCENTAGE   #FRAMES
Endstate 1   53399.95     26.7         37491
Endstate 2   52681.49     26.34        58881
```


https://doi.org/10.33011/livecoms.6.1.3485
Living J. Comp. Mol. Sci. 2025, 6(1), 3485

Endstate 3	42163.19	21.08	5621
Endstate 4	50356.36	25.88	7325

Showing the accumulated probability of each end-state over all frames (WEIGHT), the same value in percentage (PERCENTAGE) and the total number of frames that contributed to the final free energy within a cutoff of one k_BT (#FRAMES). The more frames contribute to the free energy the more certain we are to sample the true minimum of that state. Free energy estimates based on just a few frames are less trustworthy. If we use too much acceleration, this leads to a very flat energy surface and we struggle to sample the real minima. In such a case, one would still find a lowest energy for that end-state which does not necessarily represent the true minimum, since the information was lost due to the flattening of the energy surface. As you can see the parameters used resulted in a roughly equal sampling of all of the end-states. This means that we converged to values that result in an equilibrium between our end-states during the production run. Endstate 3 is slightly underrepresented in comparison and one could try to refine the parameter set to get more equal sampling. We will use this set of parameters in the water probing to catch the influence of a change in the environment on the sampling times.

3.5.5 Water probing

To capture the influence the environment has on our sampled water, we set up a production run with the parameter set optimized for the water environment in a protein-ligand environment. The Woodhead-2 system (pdb:2XJG[61]) is used in this tutorial. Go into the folder `water_probing`. Similar to the production run in water, we have two files, `aeds.arg` and `aeds.imd` in the directory. Looking into `aeds.arg`, we will see a few changes. Besides using the topology and coordinates for the Woodhead-2 system we also use a different perturbation topology since the perturbation file depends on the atom number in the topology which is changed because we now have a protein in our system. Additionally, we are also using distance restraints on the water molecules. The reason is that the end-states with the water turned into dummies have no interactions to keep them in place and the dummies would move out of the binding site. If we now take a look into the `aeds.imd` file, we can see that besides the system-specific changes like the number of atoms, a distance restraint block is added. For more information about distance restraints read Tutorial 2 or check in the GROMOS manual. If we take a look at the AEDS block we will see that the same parameters are used as in the production run in water. Since this system is quite large we provide the output files from `ene_ana` in the `water_probing/ana/long` subdirectory.

After running `dfmult` and the prevalence script we will get the following:

ENDSTATE	WEIGHT	PERCENTAGE	#FRAMES
Endstate 1	3646.0	7.29	3634
Endstate 2	0.06	0.00	23413
Endstate 3	46353.94	92.71	25127
Endstate 4	0.00	0.00	3368

We see that in the protein environment opposing the production run in water environment, where end-states 1,2,4 were equally likely and end-state 3 a little less likely, that end-state 3 is solely favorable in the protein environment, and almost all of the simulation time is spent to sample this state. As we used a parameter set that gives us roughly equal sampling over all end-states in a water environment this shift in sampling is due to the change in environment. This indicates that just a single water at the position of water A is favored to be present in the binding site.

3.5.6 Common errors

General errors in topologies

A prevalent issue arises from incorrect or incomplete topologies and perturbation topologies. Since AEDS equilibrates the system with only one end-state active at a time, structural or topological inconsistencies in the other end-states may remain unnoticed until the AEDS search simulation. A telltale symptom is a consistent SHAKE failure during the first few steps of the simulation. This typically indicates faulty geometries, missing parameters or unrealistic interactions in one or more inactive end-states.

Mismatched atom indices in perturbation topologies

A subtype of the first error involves incorrectly matched atom indices to the atoms to be perturbed. A frequent mistake occurs when users copy a ligand-in-water perturbation topology for use in the protein complex, without updating the atom indices to match the protein context. Since the perturbation relies on these indices to define the alchemical transformation, a mismatch results in nonsensical transformations, again leading to SHAKE errors at the start of the simulation.

Methodological instabilities due to acceleration

Even with correct topologies, AEDS simulations can encounter SHAKE failures due to methodological factors. The application of boosting potential-energy terms can amplify interatomic forces, leading to bond length violations that exceed the SHAKE tolerance. In addition, the potential energy surface of the AEDS reference state may exhibit a cusp, leading to a sudden increase in the gradients and accordingly high forces.

https://doi.org/10.33011/livecoms.6.1.3485
Living J. Comp. Mol. Sci. 2025, 6(1), 3485

In both cases, restarting the simulation with a reduced time step (e.g., 1 fs) and appropriately adjusted trajectory write-out frequency is often sufficient to stabilize the system. If reducing the time step is insufficient to resolve the error, reinitializing the velocities of the affected atoms may help stabilize the simulation. However, persistent SHAKE errors even at reduced time steps may still indicate underlying topological issues, such as missing exclusions or incorrect charge assignments.

3.6 Tutorial 6: NN(QM)/MM simulations with the BuRNN approach

The Buffer Region Neural Network (BuRNN) approach [62] is a hybrid quantum mechanics/molecular mechanics (QM/MM) [63, 64] simulation method. The system is partitioned into regions having different levels of theory, a QM (inner) and a classical MM (outer) region.

In between the two regions, an additional buffer region is introduced to be treated at both levels of theory (Figure 11, a). The inner region and the interactions between the inner region and the buffer region are described by an atomistic neural network (NN) model, approximating a QM energy.

The total potential energy of the system is calculated as follows:

$$V_{tot} = V^{QM}_{\mathbb{I}+\mathbb{B}} - V^{QM}_{\mathbb{B}} + V^{MM}_{\mathbb{B}} + V^{MM}_{\mathbb{O}} \quad (26)$$

The energy difference of the first two terms is then directly described by a neural network potential-energy term:

$$V_{tot} \cong V^{NN}_{\mathbb{I}+\Delta\mathbb{B}} + V^{MM}_{\mathbb{B}+\mathbb{O}} \quad (27)$$

As the NN model we use SchNet [65, 66], a continuous filter convolutional NN. The deep learning architecture SchNetPack [67] is interfaced to the MD engine of GROMOS.

In this tutorial, we will use methanol in water as a model system. In the context of BuRNN, methanol will be the inner region, while water molecules within a radius of 0.5 nm around the methanol will form the buffer region. The remaining water molecules will serve as the outer region.

We will learn how to install GROMOS with the interface to SchNetPack, generate a training dataset, train the NN model and run the BuRNN simulation.

3.6.1 Installation

Prerequisites

Before compiling the NN-enabled version of GROMOS to run Schnetpack (SPK) NN models, we recommend compiling the standard version first. This will help isolate issues with missing or mismatched libraries that may arise during compilation. After successfully compiling the standard GROMOS engine, you are ready to compile the NN-enabled version. To do that, you first need to have installed SchNetPack and the pybind11 library [68]. The interface uses the pybind11 library to call the model-prediction functions in the Python code of SchNetPack. To install SchNetPack and the pybind11 library on your system, you can use the YAML file, which can be conveniently installed using conda [69]

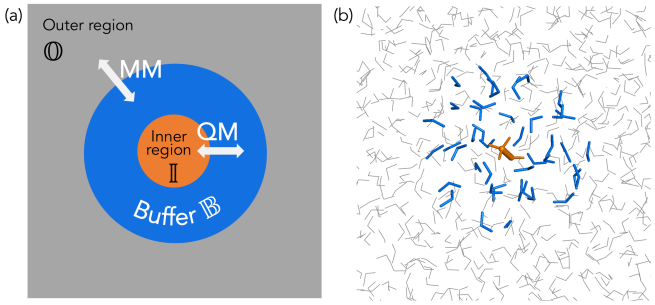

Figure 11. (a) BuRNN scheme with its three regions: Inner region (orange) described by quantum mechanics (QM), buffer region (blue) described by QM and molecular mechanics (MM), outer region (gray) described by MM. (b) Test system: methanol solvated in SPC water. Methanol is the inner region, approximately two shells of water are the buffer region, and the rest of the water box builds the outer region.

```
conda create -f spk_gromos.yml
```

For a system-wide installation without using the environment, please follow the installation instructions of the respective packages. The tested version of SchNetPack is v1.0.1 and pybind11 v2.6.2.

GPU acceleration

This part is optional to improve the efficiency of the simulations. The tasks in this tutorial are small enough to be run CPU only in a reasonable time. For the production runs this is inefficient since the training and evaluation of models on a GPU is usually manifold faster. NN/MM MD simulations in GROMOS benefit from this as well. Setting up your system to support the GPU acceleration is strongly architecture specific and is dependent on the installed CUDA and Pytorch versions. Please follow the installation instructions for CUDA and Pytorch to run tasks with the GPU acceleration.

GROMOS compilation

The NN feature of GROMOS is activated with the `--enable-schnetpack` flag in the configuration step. The `configure` script automatically looks for the pybind11 library; however, depending on the specific versions of the packages, architecture and configuration, it might not be found. In that case, you should provide the `PYFLAGS` and `PYLDFLAGS` flags to the `configure` script. The `PYFLAGS` variable holds the C preprocessor flags, which can be obtained by calling

https://doi.org/10.33011/livecoms.6.1.3485
Living J. Comp. Mol. Sci. 2025, 6(1), 3485

```
python3 -m pybind11 --includes
```

The PYLDFLAGS are the linker flags. Depending on the Python version, they can be generated by calling either

```
python3-config --ldflags
```

for Python version older than 3.8, or

```
python3-config --ldflags --embed
```

for newer versions. If you have installed the packages using the provided conda environment, we recommend installing GROMOS to the specific environment it depends on using the --prefix flag. An example of the configure script usage then could be:

```
$ mkdir build
$ cd build
$ ../configure --enable-schnetpack --prefix=
    $CONDA_PREFIX PYFLAGS='-I/path/to/
    include/python3.7 -I/path/to/lib/python3
    .7/site-packages/pybind11/include'
    PYLDFLAGS='-L/path/to/lib/python3.7/
    config -L/path/to/lib -lpython3.7 -
    lcrypt -lpthread -ldl -lutil -lrt -lm'
```

Further steps follow the standard compilation:

```
$ make
$ make install
```

3.6.2 Training dataset generation and model training

The BuRNN approach replaces expensive QM calculations with NN predictions during MD simulations. In order to train NN models, QM data points for building a training dataset have to be generated first. In this tutorial, an example QM dataset has been generated using the semi-empirical program MOPAC [70, 71]. In practice, the choice of the QM software and the reference method to be used is entirely up to the user, depending on the system and accuracy requirements. However, in order to prepare the training dataset in a reasonable time, it is necessary to automate the QM calculations. In addition, it takes a certain amount of time to perform the QM calculations for all the snapshots generated by MD. Therefore, a small QM dataset has been prepared in advance (train_dataset_tutorial/QM_dataset_example). The initial dataset uses molecular configurations generated from a purely classical MD simulation. The entire process is described in the following section.

This tutorial is accompanied by a Jupyter notebook (train_dataset_tutorial/tutorial.ipynb) to give a hands-on example of building the training database. It uses an in-house-written Python module additional_spk_utils.py for building the training dataset.

Extracting the QM regions from the MD trajectory

The resulting QM dataset will contain the inner-plus-buffer (IPB) region and the buffer region for each training configuration. Our protocol for creating such a dataset is as follows:

1. Classical MD simulation of the system of interest.
2. Extraction of the IPB regions from the MD trajectory.
3. QM energy minimization for all (or selected) extracted IPB regions.
4. Include all configurations along the minimization process into the dataset.
5. Extraction of the buffer regions from all the data points.
6. Single-point QM calculations for the IPB and buffer regions separately.
7. Create the training database

Classical MD simulations are performed as described in Tutorial 1. QM regions (inner-plus-buffer regions) are extracted from each snapshot of the MD trajectory using the GROMOS++ program filter. The program writes out coordinate trajectories for atoms that are within a specified distance of a certain part of the system. All coordinates are written into one configuration file (filter_output_example.cnf) in the train_dataset_tutorial/filter directory.

For methanol, we specify all methanol atoms as filtering centers (@atoms 1:a) and filter with a charge-group-based cut-off (@pairlist CHARGEGROUP) of 0.5 nm radius (@cutoff 0.5). The optimal/reasonable size for the buffer region can be determined from an rdf analysis. It should be as small as possible, since we want to save computational effort for the following QM calculations. The chargegroup cut-off scheme is important as we want to include full water molecules in the QM calculations. Then we extract individual QM regions into separate configuration files using an in-house-written Python script. The arguments we use for filter are as follows:

```
@topo       <topology file>
@pbc        r cog
@atoms      1:a
@pairlist   CHARGEGROUP
@cutoff     0.5
@outformat  cnf

@traj       <trajectory file(s)>
```

We provide an example of two configuration files (in train_dataset_tutorial/separated_cnf_files) which will be further used in this tutorial. An example of the argument file for filter is provided in filter/filter_meoh_example.arg.

QM calculations

In this part, we mainly focus on the concept of QM geometry optimization and single-point calculations in the context

https://doi.org/10.33011/livecoms.6.1.3485
Living J. Comp. Mol. Sci. 2025, 6(1), 3485

of BuRNN. Once we have the coordinates of the IPB region, geometry optimization will optimize the atomic positions to find the local minimum according to the QM potential energy surface. This calculations will not only find the local energy minimum but also connect the classical snapshots with the optimal QM conformations and thus represents a sufficient part of the conformational space of the system. Therefore, we include all the optimization steps in the training dataset, but we have to be aware that the training dataset will be significantly increased in size. Many similar structures will be present at the end of the minimization and thus clustering before proceeding with the next step is recommended. Alternatively, the minimization can be stopped well before reaching a strict energy minimum. We use MOPAC with the following keywords to perform the GO:

```
PM7 GRAD AUX(PRECISION = 9, XP, XS, XW)
    CHARGE=0
```

For each (remaining) configuration in the optimization procedure, a single-point QM calculation is required to obtain the associated QM energy and gradients (forces). The single-point calculation is not only performed for the IPB region but also for the buffer region alone. The coordinates of the buffer region are identical for the two calculations. The number of atoms for the inner region is required, as these atoms need to be deleted for the second calculation (buffer region only). Furthermore, the geometry of water molecules in the buffer region has to be preserved as in the following BuRNN simulations, the bonds will be constrained by the SHAKE algorithm [72]. The following MOPAC keywords are used for the single-point calculations:

```
PM7 GRAD AUX(PRECISION = 9, XP, XS, XW) 1SCF
     CHARGE=0
```

The whole process of geometry optimization and subsequent single-point calculations is automated by our Python module. The examples of the geometry optimization outputs (`MOPAC_minimization_files`) and single-point outputs (in `QM_dataset_example`) are provided. Geometry optimization was performed for two previously mentioned MD configurations. The size of the dataset increased to 860 configurations after including the geometry optimization steps.

Building a training database

In this chapter, we will explain how to build the ASE (atomic simulation environment) [73] database from the previously calculated energies and forces of the configurations. The procedure is described in more detail in the executable `tutorial.ipynb` Jupyter notebook. The `additional_spk_utils` module will be used for parsing and storing the relevant properties and metadata in the ASE database.

Model training

The last step before running BuRNN simulations is the NN model training. In the previous parts, the ASE database was created based on structures from QM geometry optimization and single-point calculations. We will now use SchNetPack for training of the example NN model. Due to the significant computational requirements of NN model training, we prepared fully trained NN models in advance, which can all be found in the `models` directory.

The SchNet model [66] is a convolutional neural network (CNN) with a continuous filter. It is very similar to the common CNNs used in image recognition. In contrast to images, molecules can not be described by the discrete matrix of pixels and thus a continuous filter is applied instead of a discrete one. The SchNet model is composed of two NNs. The main one is responsible for the prediction of the given property itself and uses the vector of atomic numbers for the given structure as input. The second one generates the filter for the convolution using the atomic positions as the input. The main NN is divided into three blocks. The first is the embedding layer which creates the feature vectors for the individual atoms within the structure. Therefore, the whole structure is represented by the 2D matrix of shape: number of atom-wise features $\times$ number of atoms. The second part of the model is a series of interaction blocks. One interaction block contains one convolutional layer. This block is responsible for creating the representation of the system. The last part is the output module which predicts the desired property of the structure, which is in our case the energy.

For model training, we use two scripts, the `spk_run.py` script and the `train.sh` bash script, which runs the Python script with all the specified arguments. The Python script loads the data from the previously prepared ASE database (the `datapath` variable) and splits it into a training, validation and test set. The split is controlled by the `-split` argument. We will use a split of 80/10/10%. The NN model is built by fitting the hyperparameters. It is reasonable to use the default values to start with the training and then optimize them later. The number of atom-wise features and the number of interaction blocks are the most important hyperparameters of the SchNet architecture, as they define the model complexity and therefore the precision and computational costs. In our case, we will train very small models with only a few features and interactions (arguments `features` and `interactions`), and we will train it only for two epochs (argument `n_epochs`). The complete list of arguments (hyperparameters) for the `spk_run.py` is listed in the `tutorial.ipynb`. The following parameters are defined in our example training:

https://doi.org/10.33011/livecoms.6.1.3485
Living J. Comp. Mol. Sci. 2025, 6(1), 3485

- property = property to be predicted by the model (default energy)
- derivative = derivative of the property to be predicted
- rho = weight factor for the property and derivative, respectively
- split = determines how many training, validation and test data will be selected, the first value for training data size, the second for validation data size, and the rest of the dataset is used for the model evaluation (default (None, None))
- batch_size = batch size (default 8)
- n_epochs = maximum number of training epochs (default 5000)
- lr = learning rate (default 0.0001)
- lr_patience = learning rate will be decreased after a certain number of epochs without improvement of the model (default 25)
- lr_min = minimal learning rate, when the value is reached, the training process is stopped (default 1e-06)
- cutoff = cutoff for long-range interactions (default 5.0)
- num_gaussians = number of Gaussians to describe the local environment (default 50)
- features = number of atom-wise features (default 128)
- interactions = number of interaction blocks (default 3)

The main goal of the training is to obtain a potential-energy function that is as close as possible to the reference one. Since, besides the energies, we also have the analytical gradients or negative forces, it is beneficial to provide them for the training as well. They carry essential information about the shape of the function at the provided data points, which significantly improves the accuracy of the generated model. For that purpose, SchNetPack has a built-in functionality, which uses a loss function combining energies and forces. A loss weight (the argument `-rho`) is introduced to control the trade-off between the energy loss and the force loss. In this tutorial, we provide the optimized weights of 0.01 for the energies and 0.99 for the gradients. Please note that the loss function mixes units of energies and forces and therefore the weights are unit-dependent. The gradients are not trained as a separate property; instead, they are used to improve the representation of the energy landscape by the model. In production runs, the model then provides the analytical gradients, which is essential for the conservation of energy in MD simulations.

The `train.sh` bash script, which calls the Python script for training, is written as follows:

```
#!/bin/bash

datapath= your path to the ASE database
modelpath= your path where the model will be
    stored

# model training
python spk_run.py train schnet custom
    $datapath $modelpath --property energy
    --derivative forces --rho property=0.01
    derivative=0.99 --split 688 86 --
    batch_size 8 --n_epochs 2 --lr 0.0001 --
    lr_patience 10 --lr_min 1e-06 --cutoff
    100.0 --num_gaussians 50 --features 32
    --interactions 1
```

The script also calls the `spk_run.py` in the evaluation mode, which evaluates the model against the test data:

```
# model evaluation
python spk_run.py eval $datapath $modelpath
    --split test --batch_size 1
```

Run the script to train and evaluate the example model:

```
$ ./train.sh
```

The training will produce several files in the `model-path` directory. These include the model, the evaluation file, a training log file, a file with the random split, a checkpoint file and a JSON file with the model settings. The checkpoint file is written periodically and can be used to restart the training. For BuRNN simulations, we train at least two NN models, as we use the second model to evaluate the predictions of the first model.

3.6.3 BuRNN simulation

For the BuRNN simulation, we have to specify which atoms are in the inner and buffer regions. GROMOS allows for an adaptive buffer region, i.e. the atoms are allowed to move freely between the outer and buffer region. The atoms of the inner region are not allowed to change. This information, together with other specifications are specified in an additional input file (`meoh.qmm`) in the `md_burnn` directory:

```
TITLE
qmmm specification file for BuRNN
END
QMUNIT
# QMULEN     QMUENE     QMUFOR     QMUCHR
     0.1      4.184     -41.84        1.0
END
```

The `QMUNIT` block may convert units of the model by a conversion factor for distances (`QMULEN`), energies (`QMUENE`), forces (`QMUFOR`), and charges(`QMUCHR`).

```
NNDEVICE
auto
```


https://doi.org/10.33011/livecoms.6.1.3485
Living J. Comp. Mol. Sci. 2025, 6(1), 3485

```
END
NNMODEL
/path/to/best_model
# MT LT
   0  1
END
NNVALID
/path/to/best_model2
# NUMSTP THRSH FCON
       1 4.184  0.0
END
```

The NNDEVICE block specifies the architecture for model execution. Allowed options are auto, cuda or cpu. Selecting auto instructs the program to run on GPU if available, or else to fall back to CPU.

In the next block, NNMODEL, the path to the prediction model is specified. The model type (MT) is set to 0, which means that BuRNN will run with a single NN evaluation on the previously trained energy differences. It would also be possible to calculate differences directly during the simulation in a combined model or even calculate it with QM on the fly by setting it to 1. The learning type (LT) is set to 1 for a standard calculation.

The NNVALID block is optional and is used if one or more validation model(s) are evaluated during the simulation. In this case, we use a validation model that also predicts the energy of the given configuration. The predictions of the two individually trained models are compared to estimate the robustness of the prediction. The models are trained with the same hyperparameters on the same dataset but with a different random split.

The models will be compared at every *n*th step (NUMSTP), and the energy difference will be deemed accurate if it is below the threshold (THRSH). If it is above the threshold, the force constant (FCON) can be set to a positive value, to push back the atoms to a configuration with a better agreement between the models (This option is not recommended).

We set the threshold to 4.184 kJ/mol, which reflects the chemical accuracy of 1 kcal/mol. The difference between the models is calculated at each time step (NUMSTP=1). This is convenient if we need to generate coordinates that are not included in the training set or if the simulation is not stable yet. Once we have a stable simulation, we can e.g. check the differences only every 100th step.

```
QMZONE
# CHG MULT
    0    1
# RESIDUE   ATOM     QMI   QMZ   QMLI
   1 MeOH   CA         1     6      0
   1 MeOH   OB         2     8      0
   1 MeOH   HA1        3     1      0
   1 MeOH   HA2        4     1      0
   1 MeOH   HA3        5     1      0
   1 MeOH   HB1        6     1      0
END
BUFFERZONE
# CHG  MULT  CUTOFF
    0     1    0.5
# RESIDUE   ATOM     QMI   QMZ   QMLI
   1 SOLV   OW        7     8     0
   1 SOLV   HW1       8     1     0
   1 SOLV   HW2       9     1     0
   2 SOLV   OW       10     8     0
   2 SOLV   HW1      11     1     0
   2 SOLV   HW2      12     1     0
   ...
END
```

These blocks finally contain the atoms that belong to the inner region (QMZONE) and that are allowed to be a part of the buffer region (BUFFERZONE). In the first line of both blocks, we specify the charge and multiplicity of the given zone. In the BUFFERZONE block we also define the cutoff which we choose to be 0.5 nm. The atoms specified in the BUFFERZONE block are considered as buffer region as soon as they come within the cutoff of any atom of the inner region. We use a group-based cutoff, to ensure that only complete water molecules (residue) enter the buffer region.

In addition, the QMMM block must be specified in the md.imd file:

```
QMMM
# NTQMMM NTQMSW RCUTQM NTWQMMM QMLJ QMCON
 MMSCALE
      -1      5    1.4        0    0     0
    -1.0
END
```

Most of the parameters are only relevant for conventional QM/MM simulations and are therefore not explained in detail.

QM/MM hybrid simulations (in mechanical embedding with constant charges) are enabled by setting the first parameter NTQMMM=-1. The interactions between the inner region and the outer region are treated at the mechanical embedding level. NTQMSW determines which software will be used for the QM calculation. SchNetPack corresponds to NTQMSW=5. The cutoff determined by RCUTQM is only relevant for electrostatic or polarizable embeddings, while for mechanical embedding (NTQMMM=+-1) it is ignored. NTWQMMM can be used if QM/MM related data should be written to a separate trajectory every *n*th step. For BuRNN simulations LJ interactions are not calculated for pairs of atoms within

https://doi.org/10.33011/livecoms.6.1.3485
Living J. Comp. Mol. Sci. 2025, 6(1), 3485

the inner region `QMLJ=0`. Constraining the bond lengths in the QM zone is optional (`QMCON`). With `MMSCALE` we can scale MM charges (by a scaling factor), but if `MMSCALE<0` no scaling is applied.

In analogy to Tutorial 1, job scripts can be generated with the `mk_script` program. The argument file (`md_mk_script.arg`) is provided. In the `env.sh` script, you might need to specify the name of the environment, the number of OpenMP processes to use and the full path to the conda command. You might also need to specify the correct full paths to `@bin` and `@dir` in the `md_mk_script.arg` file. Finally, you can execute the run files.

3.6.4 BuRNN (example) analysis

The trajectory files produced by the BuRNN simulation are the same as those produced by the standard MD simulation in GROMOS, which means that we can perform any analysis. Here we decided to show hydrogen bonds and the radial distribution function (rdf) between the methanol treated at the QM level and the surrounding MM water molecules. The argument files (`hbond_meoh.arg` and `rdf_ob_ow.arg`) for the GROMOS programs `hbond` and `rdf` are provided in the `ana` directory. The results can be seen in Figure 12.

3.6.5 Advanced options

Charge model

For running BuRNN simulations with dynamic QM charges, charges need to be calculated with the QM method of choice and included in the training dataset. A separate NN must be trained for the charges. The adapted version can be found in this repository. To use charge models, the `NNCHARGE` block has to be added to the `charge.qmm` file and `NTQMMM` must be set to +1.

```
NNCHARGE
/path/to/best/charge_model
# NUMSTP
        1
END
```

This block contains the path to the NN charge model and how often the charges are updated (every `NUMSTP` steps).

Including a charge model was, for example, relevant for the hexa-aqua iron, the model system of the original BuRNN paper [62]. In this case, a considerable amount of charge transfer between the inner and buffer regions was observed, which needed to be reflected in their interactions with the outer region.

3.6.6 Common errors

Buffer region size

The buffer region is defined by a cutoff during atom selection with the `filter` program and determines which atoms

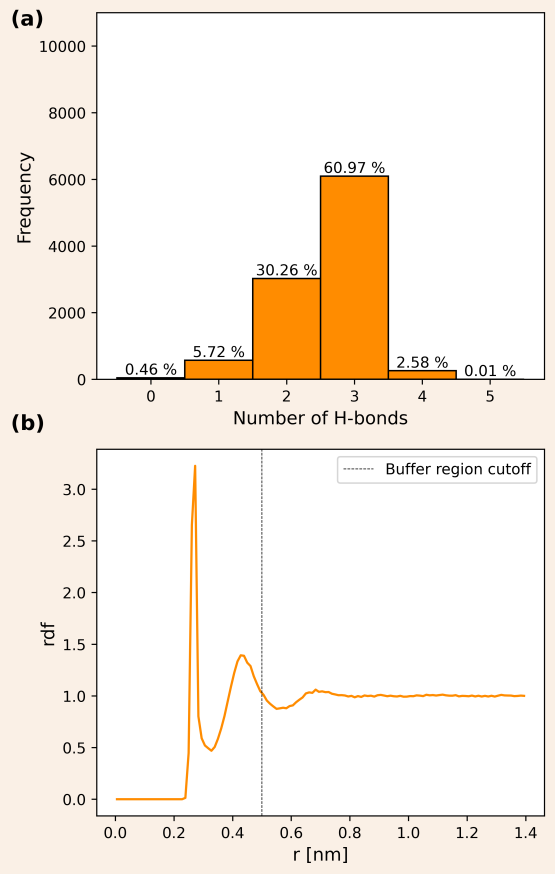

Figure 12. BuRNN simulation analysis. (a) The number of hydrogen bonds formed between methanol and water for each snapshot of the trajectory. **(b)** Radial distribution function (rdf) between the methanol's oxygen and all oxygen atoms of the water molecules. The results are based on a 2 ns BuRNN simulation of the methanol in water.

are included in QM calculations and the subsequent neural network (NN) training. Incorrect buffer region handling can cause major issues:

- Inconsistent cutoff use: The same cutoff size must be applied in the BuRNN simulations as in the training dataset preparation. The neural networks predict reliable energies and forces only in the trained region, extrapolation beyond the trained buffer size can result in unphysical energies and forces which will be detected with the validation model.
- Poor cutoff placement: The cutoff should not lie in structurally or chemically ambiguous regions such as peaks in the solvation shells. We recommend analyzing radial distribution functions (rdfs) to ensure the cutoff falls outside these regions.
- Buffer sizing: An excessively large buffer increases QM calculation and NN training costs. If the buffer is too small, relevant environmental effects (e.g. polarization,

https://doi.org/10.33011/livecoms.6.1.3485
Living J. Comp. Mol. Sci. 2025, 6(1), 3485

charge redistribution or steric constraints) may be missed, leading to an incomplete description and inaccuracies.

Charged molecules

Simulation of charged molecules with BuRNN will most likely require the optional charge model. If atom crowding or depletion at the buffer cutoff occurs, this can be detected with an rdf analysis. If artifacts occur directly at the cutoff, it is strongly recommended to prepare a charge model by training on the partial charges of inner-plus-buffer region.

Hyperparameter settings

Neural network training requires careful selection of hyperparameters. When applying the model to a new system, these parameters should be adjusted accordingly to avoid common issues:

- Low accuracy: Too few features or interaction blocks (`features`, `interactions`) can lead to underfitting, reducing the precision of predicted energies and forces.
- Overfitting: Too many features or interaction blocks (`features`, `interactions`) can lead to overfitting, especially with limited training data.
- Missed interactions: A small cutoff (`cutoff`) may ignore relevant long-range interactions, especially in larger or charged systems.
- Inefficient training: Too few epochs or a poorly chosen learning rate (`n_epochs`, `lr`, `lr_patience`, `lr_min`) can prevent convergence or result in slow learning.
- Memory problems: An overly large batch size (`batch_size`) can exceed available GPU memory, causing crashes or slowdowns.
- Insufficiently robust dataset: Inappropriate data splits (`split`) may cause weak performance on unseen data.
- Unresolved environments: Too few Gaussians (`num_gaussians`) may not capture chemical detail in the local atomic environment.

In order to assess model quality, it is recommended to investigate the training and validation loss.

Incorrect unit conversion

BuRNN simulations require external quantum chemical software, which leads to frequent errors from incorrect unit conversion between software packages. When coupling QM and MM components, the unit conversion block `QMUNIT` must be correctly configured. Although GROMOS is unit independent, the use of *Standard International (SI)* units (e.g. kJ/mol for energy and nm for length) is recommended.

While this tutorial uses MOPAC (which uses kcal/mol and Å), other QM software typically operate in atomic units (Hartree and Bohr). If incorrect unit conversion factors are applied, the simulations will usually fail, as indicated by:

- Exploding or collapsing structures due to extreme force magnitudes.
- Non-physical forces and dynamics that lead to distorted structures (e.g. the optimized geometry changes rapidly).
- Unrealistic energies, e.g., large energy differences between steps or replicates.

4 Author Contributions

JG and NH prepared Tutorial 1, BL, CÖ, AdR and CO prepared Tutorials 2 and 3. OGC prepared Tutorial 4. BB prepared Tutorial 5. BL, RC, and PP prepared Tutorial 6. MBMS tested the tutorials. All authors contributed to manuscript writing.

5 Other Contributions

This work has profited from the experiences of various members of the research groups of WFvG, NH and CO using the methods explained in this tutorial.

6 Potentially Conflicting Interests

WFvG, CO and NH and their research groups are developers of the GROMOS simulation package.

7 Funding Information

NH acknowledges funding by the Deutsche Forschungsgemeinschaft (DFG, German Research Foundation) under Germany's Excellence Strategy – EXC 2075 – 390740016. CO acknowledges the Austrian Federal Ministry for Digital and Economic Affairs, The National Foundation for Research, Technology and Development and the Christian Doppler Research Association for funding.

Author Information

ORCID:

Bettina Lier: 0000-0002-8032-0084
Benedict Braunsfeld: 0000-0002-0286-8239
Radek Crha: 0000-0001-9293-8562
Oriol Gracia Carmona: 0000-0001-6560-9106
Julia Gebhardt: 0000-0002-5529-7337
Christoph Öhlknecht: 0000-0003-1847-1719
Peter Poliak: 0000-0002-6272-4907
Peter Fraško: 0009-0009-9491-2732
Anita de Ruiter: 0000-0003-3046-8969
Marcelle B. M. Spera: 0000-0001-7841-0489
Michael Gillhofer: 0000-0001-8081-6373
Wilfred F. van Gunsteren: 0000-0002-9583-7019
Chris Oostenbrink: 0000-0002-4232-2556
Niels Hansen: 0000-0003-4366-6120

https://doi.org/10.33011/livecoms.6.1.3485
Living J. Comp. Mol. Sci. 2025, 6(1), 3485

References

[1] **van Gunsteren WF**, Dolenc J. Thirty-Five Years of Biomolecular Simulation: Development of Methodology, Force Fields, and Software. Mol Sim. 2012; 38:1271–1281. https://doi.org/10.1080/08927022.2012.701744.

[2] **Schmid N**, Christ CD, Christen M, Eichenberger AP, van Gunsteren WF. Architecture, Implementation and Parallelization of the GROMOS Software for Biomolecular Simulation. Comput Phys Commun. 2012; 183:890–903. https://doi.org/10.1016/j.cpc.2011.12.014.

[3] **van Gunsteren WF et al.**, The GROMOS Software for (Bio)Molecular Simulation. Volumes 1–9; 2011. http://www.gromos.net/. Accessed 31 May 2020.

[4] **Riniker S**. Fixed-Charge Atomistic Force Fields for Molecular Dynamics Simulations in the Condensed Phase: An Overview. J Chem Inf Model. 2018; 58:565–578. https://doi.org/10.1021/acs.jcim.8b00042.

[5] **Diem M**, Oostenbrink C. The Effect of Using a Twin-Range Cutoff Scheme for Nonbonded Interactions: Implications for Force-Field Parametrization? J Chem Theory Comput. 2020; 16:5985–5990. https://doi.org/10.1021/acs.jctc.0c00509.

[6] **Diem M**, Oostenbrink C. The effect of different cutoff schemes in molecular simulations of proteins. J Comput Chem. 2020; 41(32):2740–2749. https://doi.org/https://doi.org/10.1002/jcc.26426.

[7] **van Gunsteren WF et al.**, The GROMOS Software for (Bio)Molecular Simulation. Volume 7: Tutorial with examples; 2011. http://www.gromos.net/. Accessed 31 May 2020.

[8] **van Gunsteren WF et al.**, The GROMOS Software for (Bio)Molecular Simulation. Volume 5: Program Library Manual; 2011. http://www.gromos.net/. Accessed 31 May 2020.

[9] The PyMOL Molecular Graphics System. Schrödinger, LLC.; 2015. https://pymol.org/.

[10] **Humphrey W**, Dalke A, Schulten K. VMD – Visual Molecular Dynamics. J Molec Graphics. 1996; 14:33–38. https://doi.org/10.1016/0263-7855(96)00018-5.

[11] **van Gunsteren WF et al.**, The GROMOS Software for (Bio)Molecular Simulation. Volume 6: Technical Details; 2011. http://www.gromos.net/. Accessed 31 May 2020.

[12] **van Gunsteren WF et al.**, The GROMOS Software for (Bio)Molecular Simulation. Volume 2: Algorithms and Formulae for Modelling of Molecular Systems; 2011. http://www.gromos.net/. Accessed 31 May 2020.

[13] **van Gunsteren WF et al.**, The GROMOS Software for (Bio)Molecular Simulation. Volume 8: Installation Guide; 2011. http://www.gromos.net/. Accessed 31 May 2020.

[14] **Hansen N**, Heller F, Schmid N, van Gunsteren WF. Time-Averaged Order Parameter Restraints in Molecular Dynamics Simulations. J Biomol NMR. 2014; 60(2-3):169–187. https://doi.org/10.1007/s10858-014-9866-7.

[15] **Smith LJ**, van Gunsteren WF, Hansen N. Interpretation of Seemingly Contradictory Data: Low NMR S^2 Order Parameters Observed in Helices and High NMR S^2 Order Parameters in Disordered Loops of the Protein hGH at Low pH. Chem Eur J. 2017; 23:9585–9591. https://doi.org/10.1002/chem.201700896.

[16] **Smith LJ**, van Gunsteren WF, Hansen N. On the Use of Side-Chain NMR Relaxation Data to Derive Structural and Dynamical Information on Proteins: A Case Study using Hen Lysozyme. ChemBioChem. 2021; 22:1049–1064. https://doi.org/10.1002/cbic.202000674.

[17] **Ulmer TS**, Ramirez BE, Delaglio F, Bax A. Evaluation of Backbone Proton Positions and Dynamics in a Small Protein by Liquid Crystal NMR Spectroscopy. J Am Chem Soc. 2003; 125(30):9179–9191. https://doi.org/10.1021/ja0350684.

[18] **Berendsen HJC**, Postma JPM, van Gunsteren WF, Hermans J. Interaction Models for Water in Relation to Protein Hydration. In: Pullmann B, editor. *Intermolecular Forces* Dordrecht, the Netherlands: Reidel; 1981.p. 331–342. https://doi.org/10.1007/978-94-015-7658-1_21.

[19] **Hall JB**, Fushman D. Characterization of the Overall and Local Dynamics of a Protein with Intermediate Rotational Anisotropy: Differentiating Between Conformational Exchange and Anisotropic Diffusion in the B3 Domain of Protein G. J Biomol NMR. 2003; 27(3):261–275. https://doi.org/10.1023/A:1025467918856.

[20] **Gilson MK**, Given JA, Bush BL, McCammon JA. The Statistical-Thermodynamic Basis for Computation of Binding Affinities: A Critical Review. Biophys J. 1997; 72(3):1047–1069. https://doi.org/10.1016/S0006-3495(97)78756-3.

[21] **Ruiter AD**, Oostenbrink C. Extended Thermodynamic Integration: Efficient Prediction of Lambda Derivatives at Nonsimulated Points. J Chem Theory Comput. 2016; 12:4476–4486. https://doi.org/10.1021/acs.jctc.6b00458.

[22] **Öhlknecht C**, Petrov D, Engele P, Krößß C, Sprenger B, Fischer A, Lingg N, Schneider R, Oostenbrink C. Enhancing the Promiscuity of a Member of the Caspase Protease Family by Rational Design. Proteins. 2020; 88:1303–1318. https://doi.org/10.1002/prot.25950.

[23] **Singh RK**, Ethayathulla AS, Jabeen T, Sharma S, Kaur P, Singh TP. Aspirin Induces its Anti-Inflammatory Effects Through its Specific Binding to Phospholipase A2: Crystal Structure of the Complex Formed Between Phospholipase A2 and Aspirin at 1.9 Å Resolution. J Drug Target. 2005; 13(2):113–119. https://doi.org/10.1080/10611860400024078.

[24] **Christen M**, Kunz APE, van Gunsteren WF. Sampling of Rare Events Using Hidden Restraints. J Phys Chem B. 2006; 110(16):8488–8498. https://doi.org/10.1021/jp0604948.

[25] **Kirkwood JG**. Statistical Mechanics of Fluid Mixtures. J Chem Phys. 1935; 3:300–313. https://doi.org/10.1063/1.1749657.

[26] **Tironi IG**, Sperb R, Smith PE, van Gunsteren WF. A Generalized Reaction Field Method for Molecular-Dynamics Simulations. J Chem Phys. 1995; 102:5451–5459. https://doi.org/10.1063/1.469273.

https://doi.org/10.33011/livecoms.6.1.3485
Living J. Comp. Mol. Sci. 2025, 6(1), 3485

[27] **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. Chem Phys Lett. 1994; 222:529–539. https://doi.org/10.1016/0009-2614(94)00397-1.

[28] **Bennett CH**. Efficient Estimation of Free Energy Differences from Monte Carlo Data. J Comput Phys. 1976; 22(2):245 – 268. https://doi.org/10.1016/0021-9991(76)90078-4.

[29] **Maurer M**, Hansen N, Oostenbrink C. Comparison of Free-Energy Methods using a Tripeptide-Water Model System. J Comput Chem. 2018; 39(26):2226–2242. https://doi.org/10.1002/jcc.25537.

[30] **Roux B**, Nina M, Pomès R, Smith JC. Thermodynamic Stability of Water Molecules in the Bacteriorhodopsin Proton Channel: A Molecular Dynamics Free Energy Perturbation Study. Biophys J. 1996; 71:670–681. https://doi.org/10.1016/S0006-3495(96)79267-6.

[31] **Boresch S**, Tettinger F, Leitgeb M. Absolute Binding Free Energies: A Quantitative Approach for Their Calculation. J Phys Chem B. 2003; 107(35):9535–9551. https://doi.org/10.1021/jp0217839.

[32] **Gebhardt J**, Hansen N. Calculation of Binding Affinities for Linear Alcohols to α-Cyclodextrin by Twin-System Enveloping Distributions Sampling Simulations. Fluid Phase Equilib. 2016; 422:1–17. https://doi.org/10.1016/j.fluid.2016.02.001.

[33] **Reif MM**, Oostenbrink C. Net Charge Changes in the Calculation of Relative Ligand-Binding Free Energies via Classical Atomistic Dynamics Simulation. J Comput Chem. 2014; 35(3):227–243. https://doi.org/10.1002/jcc.23490.

[34] **Öhlknecht C**, Lier B, Petrov D, Fuchs J, Oostenbrink C. Correcting Electrostatic Artifacts due to Net-Charge Changes in the Calculation of Ligand Binding Free Energies. J Comput Chem. 2020; 41:986–999. https://doi.org/10.1002/jcc.26143.

[35] **Hünenberger PH**, Reif MM. Single-Ion Solvation. Theoretical and Computational Chemistry Series, The Royal Society of Chemistry; 2011. https://doi.org/10.1039/9781849732222.

[36] **Kastenholz MA**, Hünenberger PH. Computation of Methodology-Independent Ionic Solvation Free Energies from Molecular Simulations. I. The Electrostatic Potential in Molecular Liquids. J Chem Phys. 2006; 124(12):124106. https://doi.org/10.1063/1.2172593.

[37] **Berendsen HJC**, Grigera JR, Straatsma TP. The Missing Term in Effective Pair Potentials. J Phys Chem. 1987; 91(24):6269–6271. https://doi.org/10.1021/j100308a038.

[38] **Jorgensen WL**, Chandrasekhar J, Madura JD, Impey RW, Klein ML. Comparison of Simple Potential Functions for Simulating Liquid Water. J Chem Phys. 1983; 79(2):926–935. https://doi.org/10.1063/1.445869.

[39] **Mahoney MW**, Jorgensen WL. A Five-Site Model for Liquid Water and the Reproduction of the Density Anomaly by Rigid, Nonpolarizable Potential Functions. J Chem Phys. 2000; 112(20):8910–8922. https://doi.org/10.1063/1.481505.

[40] **Stillinger FH**, Rahman A. Improved Simulation of Liquid Water by Molecular Dynamics. J Chem Phys. 1974; 60(4):1545–1557. https://doi.org/10.1063/1.1681229.

[41] **de Ruiter A**, Oostenbrink C. Protein–Ligand Binding from Distancefield Distances and Hamiltonian Replica Exchange Simulations. J Chem Theory Comput. 2013; 9(2):883–892. https://doi.org/10.1021/ct300967a.

[42] **Nagy G**, Oostenbrink C, Hritz J. Exploring the Binding Pathways of the 14-3-3ζ Protein: Structural and Free-Energy Profiles Revealed by Hamiltonian Replica Exchange Molecular Dynamics with Distancefield Distance Restraints. PloS ONE. 2017; 12:e0180633. https://doi.org/10.1371/journal.pone.0180633.

[43] **Hritz J**, Oostenbrink C. Optimization of Replica Exchange Molecular Dynamics by fast Mimicking. J Chem Phys. 2007; 127(20):204104. https://doi.org/10.1063/1.2790427.

[44] **Trzesniak D**, Kunz APE, van Gunsteren WF. A Comparison of Methods to Compute the Potential of Mean Force. ChemPhysChem. 2007; 8:162–169. https://doi.org/10.1002/cphc.200600527.

[45] **Hansson T**, Oostenbrink C, van Gunsteren WF. Molecular dynamics simulations. Curr Opin Struct Biol. 2002; 12(2):190–196. https://doi.org/10.1016/S0959-440X(02)00308-1.

[46] **Volkhardt A**, Grubmüller H. Estimating ruggedness of free-energy landscapes of small globular proteins from principal component analysis of molecular dynamics trajectories. Phys Rev E. 2022; 105(4):044404. https://doi.org/10.1103/PhysRevE.105.044404.

[47] **Yang YI**, Shao Q, Zhang J, Yang L, Gao YQ. Enhanced sampling in molecular dynamics. J Chem Phys. 2019; 151(7):070902. https://doi.org/10.1063/1.5109531.

[48] **Miao Y**, Feher VA, McCammon JA. Gaussian accelerated molecular dynamics: Unconstrained enhanced sampling and free energy calculation. J Chem Theory Comput. 2015; 11(8):3584–3595. https://doi.org/10.1021/acs.jctc.5b00436.

[49] **Miao Y**, Sinko W, Pierce L, Bucher D, Walker RC, McCammon JA. Improved reweighting of accelerated molecular dynamics simulations for free energy calculation. J Chem Theory Comput. 2014; 10(7):2677–2689. https://doi.org/10.1021/ct500090q.

[50] **Miao Y**, Bhattarai A, Wang J. Ligand Gaussian accelerated molecular dynamics (LiGaMD): Characterization of ligand binding thermodynamics and kinetics. J Chem Theory Comput. 2020; 16(9):5526–5547. https://doi.org/10.1021/acs.jctc.0c00395.

[51] **Wang J**, Miao Y. Peptide Gaussian accelerated molecular dynamics (Pep-GaMD): Enhanced sampling and free energy and kinetics calculations of peptide binding. J Chem Phys. 2020; 153(15):154109. https://doi.org/10.1063/5.0021399.

[52] **Wang J**, Miao Y. Protein–protein interaction-Gaussian accelerated molecular dynamics (PPI-GaMD): Characterization of protein binding thermodynamics and kinetics. J Chem Theory Comput. 2022; 18(3):1275–1285. https://doi.org/10.1021/acs.jctc.1c00974.

https://doi.org/10.33011/livecoms.6.1.3485
Living J. Comp. Mol. Sci. 2025, 6(1), 3485

[53] **Wang J**, Miao Y. Ligand Gaussian accelerated molecular dynamics 2 (LiGaMD2): Improved calculations of ligand binding thermodynamics and kinetics with closed protein pocket. J Chem Theory Comput. 2023; 19(3):733–745. https://doi.org/10.1021/acs.jctc.2c01194.

[54] **Pang YT**, Miao Y, Wang Y, McCammon JA. Gaussian accelerated molecular dynamics in NAMD. J Chem Theory Comput. 2017; 13(1):9–19. https://doi.org/10.1021/acs.jctc.6b00931.

[55] **Copeland MM**, Do HN, Votapka L, Joshi K, Wang J, Amaro RE, Miao Y. Gaussian accelerated molecular dynamics in OpenMM. J Phys Chem B. 2022; 126(31):5810–5820. https://doi.org/10.1021/acs.jpcb.2c03765.

[56] **Berg BA**. Multicanonical simulations step by step. Comput Phys Commun. 2003; 153(3):397–406. https://doi.org/10.1016/S0010-4655(03)00245-5.

[57] **Zwanzig RW**. High-Temperature Equation of State by a Perturbation Method. I. Nonpolar Gases. J Chem Phys. 1954; 22(8):1420–1426. https://doi.org/10.1063/1.1740409.

[58] **Perthold JW**, Oostenbrink C. Accelerated Enveloping Distribution Sampling: Enabling Sampling of Multiple End States while Preserving Local Energy Minima. J Phys Chem B. 2018; 122(19):5030–5037. https://doi.org/10.1021/acs.jpcb.8b02725.

[59] **Perthold JW**, Petrov D, Oostenbrink C. Toward Automated Free Energy Calculation with Accelerated Enveloping Distribution Sampling (A-EDS). J Chem Inf Model. 2020; 60(11):5395–5406. https://doi.org/10.1021/acs.jcim.0c00456.

[60] **Gracia Carmona O**, Gillhofer M, Tomasiak L, De Ruiter A, Oostenbrink C. Accelerated Enveloping Distribution Sampling to Probe the Presence of Water Molecules. J Chem Theory Comput. 2023; 19(11):3379–3390. https://doi.org/10.1021/acs.jctc.3c00109.

[61] **Woodhead AJ**, Angove H, Carr MG, Chessari G, Congreve M, Coyle JE, Cosme J, Graham B, Day PJ, Downham R, Fazal L, Feltell R, Figueroa E, Frederickson M, Lewis J, McMenamin R, Murray CW, O'Brien MA, Parra L, Patel S, et al. Discovery of (2,4-Dihydroxy-5-isopropylphenyl)-[5-(4-methylpiperazin-1-ylmethyl)-1,3-dihydroisoindol-2-yl]methanone (AT13387), a Novel Inhibitor of the Molecular Chaperone Hsp90 by Fragment Based Drug Design. J Med Chem. 2010; 53(16):5956–5969. https://doi.org/10.1021/jm100060b.

[62] **Lier B**, Poliak P, Marquetand P, Westermayr J, Oostenbrink C. BuRNN: Buffer Region Neural Network Approach for Polarizable-Embedding Neural Network/Molecular Mechanics Simulations. J Phys Chem Lett. 2022; 13(17):3812–3818. https://doi.org/10.1021/acs.jpclett.2c00654.

[63] **Warshel A**, Levitt M. Theoretical studies of enzymic reactions: Dielectric, electrostatic and steric stabilization of the carbonium ion in the reaction of lysozyme. Journal of Molecular Biology. 1976; 103(2):227 – 249. https://doi.org/10.1016/0022-2836(76)90311-9.

[64] **Senn HM**, Thiel W. QM/MM Methods for Biomolecular Systems. Angewandte Chemie International Edition. 2009; 48(7):1198–1229. https://doi.org/10.1002/anie.200802019.

[65] **Schütt K**, Kindermans PJ, Sauceda Felix HE, Chmiela S, Tkatchenko A, Müller KR. SchNet: A continuous-filter convolutional neural network for modeling quantum interactions. Advances in Neural Information Processing Systems 30. 2017; p. 991–1001.

[66] **Schütt KT**, Sauceda HE, Kindermans PJ, Tkatchenko A, Müller KR. SchNet – A deep learning architecture for molecules and materials. The Journal of Chemical Physics. 2018; 148(24):241722. https://doi.org/10.1063/1.5019779.

[67] **Schütt KT**, Kessel P, Gastegger M, Nicoli KA, Tkatchenko A, Müller KR. SchNetPack: A Deep Learning Toolbox For Atomistic Systems. Journal of Chemical Theory and Computation. 2019; 15(1):448–455. https://doi.org/10.1021/acs.jctc.8b00908.

[68] **Jakob W**, Rhinelander J, Moldovan D, pybind11 – Seamless operability between C++11 and Python; 2017. https://github.com/pybind/pybind11.

[69] Anaconda Software Distribution. Anaconda Inc.; 2016. https://docs.anaconda.com/.

[70] **Setwart JJP**. MOPAC: A General Molecular Orbital Package. J Computer-Aided Mol Des. 1990; 4:1–103. https://doi.org/10.1007/BF00128336.

[71] **Setwart JJP**. Optimization of parameters for semiempirical methods VI: more modifications to the NDDO approximations and re-optimization of parameters. Journal of Molecular Modeling. 2013; 19:1–32. https://doi.org/10.1007/s00894-012-1667-x.

[72] **Ryckaert JP**, Ciccotti G, Berendsen HJC. Numerical integration of the cartesian equations of motion of a system with constraints: molecular dynamics of n-alkanes. Journal of Computational Physics. 1977; 23(3):327–341. https://doi.org/10.1016/0021-9991(77)90098-5.

[73] **Larsen AH**, Mortensen JJ, Blomqvist J, Castelli IE, Christensen R, Dułak M, Friis J, Groves MN, Hammer B, Hargus C, et al. The atomic simulation environment—a Python library for working with atoms. Journal of Physics: Condensed Matter. 2017; 29(27):273002. https://doi.org/10.1088/1361-648X/aa680e.