Advanced physics practical / Fortgeschrittenen-Praktikum

Molecular Dynamics Simulations of the SARS-CoV-2 Fusion Peptide

Prof. Dr. Jochen Hub
Computational Biophysics Group, Saarland University

This practical was designed by Chetan Poojari, Katharina Scherer, and Jochen Hub, motivated by and partly based upon a Bachelor thesis by Tobias Bommer.

1 Molecular dynamics simulations

Although today we have a good understanding of the fundamental principles and laws of nature and chemistry, application of these theories to systems of interest, such as biological macromolecules, remains a tough challenge, as no analytic solutions exist for the relevant equations of motions. However, using numerical methods, it is possible to compute approximate solutions to fundamental equations and to approximately compute expectation values of thermodynamics quantities, which can be compared with experimental data. Molecular dynamics (MD) simulations are a computational method that solve physical equations of motion (typically Newton's equations of motion) numerically in order to study the dynamics of atoms, molecules, liquids, polymers, or biomolecules.

1.1 Levels of Approximation in MD simulations

The physical equation serving as a starting point for MD simulations is the time-dependent Schrödinger equation. Yet, due to limitations of computational power, several approximations are required if one desires to study large systems within reasonable computation time. In the following section, we will briefly discuss the most prominent levels of approximation used in MD simulations.

1.1.1 Born-Oppenheimer approximation

The Born-Oppenheimer approximation is used to simplify the Schrödinger equation for molecules, thus enabling a speed-up of the calculation of molecular wave functions or of the time evolution of a molecule. Since the mass of the electron is much smaller than the mass of an atomic nucleus, the motions of electrons inside a molecule happen on a much smaller timescale as compared to the motion of the atomic nuclei. The Born-Oppenheimer approximation makes use of this difference in timescales, as it allows us to treat the electronic and nucleic dynamics separately [1].

Let $\mathbf{r}$ and $\mathbf{R}$ denote the positions of the electrons and nuclei, respectively. In the approximation, the molecular wave function $\Psi(\mathbf{r},\mathbf{R})$ can be written as the product of a wave function describing the electrons $\Psi_e(\mathbf{r})$ and a wave function describing the atomic nuclei $\Psi_n(\mathbf{R})$,

$$\Psi(\mathbf{r},\mathbf{R}) = \Psi_e(\mathbf{r}) \cdot \Psi_n(\mathbf{R})$$

From the point of view of the electrons, the atomic nuclei appear to almost stand still, so their kinetic energy can be approximated to zero and their positions can be considered fixed. Hence, from the Hamilton operator H of the whole molecule

$$H = T_e + T_n + V_{ee} + V_{nn} + V_{en}$$

we can extract the electronic Hamilton operator

$$H_e = T_e + V_{ee} + V_{nn} + V_{en}$$

where T_e is the kinetic energy of the electrons, T_n is the kinetic energy of the nuclei, V_{ee} is the potential energy of the Coulomb interaction between the electrons, V_{nn} is the potential energy of the Coulomb interaction between the nuclei and finally, V_{en} is the potential energy of the Coulomb interaction between electrons and nuclei. We neglect any further interactions such as spin coupling.

In the electronic Hamilton operator, the chosen nuclei positions R are treated as fixed parameters, so the electron positions $\mathbf{r}$ are the only variables left entering the electronic Schrödinger equation. By solving the eigenvalue problem of the electronic Hamilton operator

$$H_e\psi_e(\mathbf{r};\mathbf{R}) = E_e(\mathbf{R})\psi_e(\mathbf{r};\mathbf{R})$$

for different sets of nuclei positions R then enables expressing the ground-state total energy of the electrons $E_e(\mathbf{R})$ as a function of the nuclei positions $\mathbf{R}$. This function $E_e(\mathbf{R})$ is often referred to as the *potential energy surface* [2].

From the point of view of the nuclei, the electrons move so fast that they seem to instantaneously adapt to any change of the nuclei positions. Hence, the motion of the nuclei does not depend on the electron positions $\mathbf{r}$; however, the nuclei still feel the overall presence of the electrons as manifested in the potential energy surface E_e. We obtain

$$i\hbar\partial_t\Psi_n(\mathbf{R}) = H_n\Psi_n(\mathbf{R})$$

as the time-dependent Schrödinger equation describing the motion of the nuclei, with the nuclear Hamilton operator

$$H_n = T_n + E_e(\mathbf{R})$$

The reason why the Born-Oppenheimer approximation provides a speed-up in calculations is that the Schrödinger equation imposes an eigenvalue problem; and the computational cost required to solve an eigenvalue problem increases faster than the square of the number of coordinates [3]. It is hence more efficient to solve an electronic Schrödinger equation (that only depends on the electron positions $\mathbf{r}$) and a nuclear Schrödinger equation (that only depends on the nuclei positions $\mathbf{R}$) separately, instead of solving the molecular Schrödinger equation that depends on both the nuclei and electron positions.

1.1.2 Classical mechanics and numerical integration

Solvated molecules (i.e., molecules in water) often behave largely classically because of the strong interactions with the solvent environment. Hence, to further simplify and speed up calculations, nuclear motions are often described using classical mechanics.

This approach is only invalid for systems whose dynamics are characterized by quantum effects such as hydrogen tunneling or chemical reactions; however, sophisticated hybrid methods combining quantum and classical mechanics have been successfully applied to also model such systems. These models use quantum mechanics to describe parts of a system where the precision is needed, but still benefit from the fast classical calculations applied to the remaining part of the system that might not require quantum mechanical precision [4]. For many biomolecular systems there is no need for direct treatment of quantum effects.

If one uses classical mechanics to describe the nuclear and thus atomic motion, the atomic nuclei are treated as point particles and the Newtonian equation of motion of the i-th atom is given by

$$M_i\ddot{\mathbf{R}}_i = -\nabla_i E_e(\mathbf{R})$$

where M_i is the mass of the i-th atom, $\mathbf{R}_i$ is the position of the i-th atom, $E_e(\mathbf{R})$ is the potential energy surface.

Equivalent to the above Newtonian equation of motion, which is a second-order ordinary differential equation, are the two first-order ordinary differential equations

$$M_i \dot{\mathbf{v}}_i = -\nabla_i E_e(\mathbf{R})$$
$$\dot{\mathbf{R}}_i = \mathbf{v}_i$$

These equations can be solved numerically using leapfrog integration:

$$\mathbf{v}_i(t + \frac{\Delta t}{2}) = \mathbf{v}_i(t - \frac{\Delta t}{2}) + \mathbf{a}_i \Delta t$$
$$\mathbf{R}_i(t + \Delta t) = \mathbf{R}_i(t) + \mathbf{v}_i(t + \frac{\Delta t}{2})\Delta t$$

with the acceleration of atom i

$$\mathbf{a}_i = \frac{-\nabla_i E_e(\mathbf{R}(t))}{M_i}$$

and the integration timestep Δt [5].

The error of the leapfrog integration vanishes as Δt approaches zero, so Δt has to be small to achieve good accuracy. However, choosing a smaller timestep means more steps need to be calculated to simulate a given period of time, thus more computational power is required. Hence, we aim to choose the integration time step Δt as large as possible, limited by the necessity of sufficient accuracy. In practice, this limit is usually imposed by bond vibrations, as about 20 time steps per period of these fast vibrations are required for sufficient accuracy. One therefore often makes use of algorithms like LINCS [6] that enable constraining of bond lengths, thereby allowing larger integration time steps. Typical choices for the integration timestep Δt used in MD simulations are 2 to 4 femtoseconds.

1.1.3 Empirical force fields

Instead of calculating the potential energy surface $E_e(\mathbf{R})$ using computationally expensive, quantum mechanical methods, one often uses so-called empirical force fields instead, which try to describe the interatomic interactions using empirical formulae of the potential energy. The parameters occurring in these energy terms are fitted against experimental data and results of quantum mechanical calculations, with the aim to capture the true physical interactions as accurately as possible [7, 8, 9, 10]. Many such empirical force fields have been developed in the last decades; they often are specialized for certain systems and their characteristic interactions. In any case, testing and further development of force fields imposes a continuously ongoing challenge and process [11, 12, 13, 14, 15].

Despite a huge diversity of force fields available today, they often use similar potential terms to model interatomic interactions. These interactions can be separated into bonded and non-bonded interactions. Bonded interactions constitute potential energy terms emerging from covalently bound atoms deviating

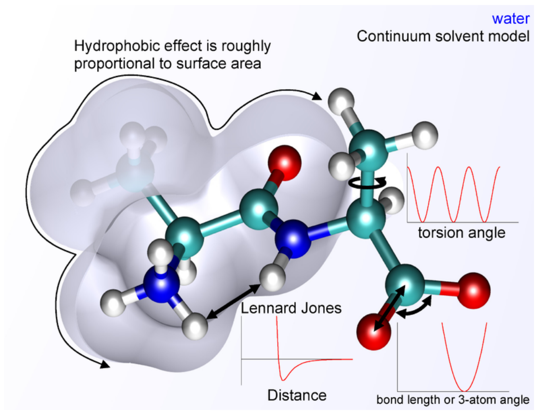

water
Continuum solvent model
Hydrophobic effect is roughly proportional to surface area
torsion angle
Lennard Jones
Distance
bond length or 3-atom angle

Figure 1: Illustration of bonded and non-bonded interactions. (Figure taken from Wikipedia, Creative Commons Attribution-Share Alike 3.0 Unported license.)

from their equilibrium position of bond length, angles, or torsion of bonds.

$$\begin{aligned} V^b(\mathbf{R}) = &\sum_{\text{bonds}} \frac{1}{2} K_l (l(\mathbf{R}) - l_0)^2 \\ &+ \sum_{\text{bond angles}} \frac{1}{2} K_\phi (\phi(\mathbf{R}) - \phi_0)^2 \\ &+ \sum_{\text{proper dihedrals}} K_\alpha (1 + \cos(n\alpha(\mathbf{R}) - \delta)) \\ &+ \sum_{\text{extraplanar angles}} \frac{1}{2} K_\theta (\theta(\mathbf{R}) - \theta_0)^2 \end{aligned}$$

where $K_l, l_0, K_\phi, \phi_0, K_\alpha, n, \delta, K_\theta, \theta_0$, are parameters, and parameters indexed 0 represent the respective equilibrium length or angle. Note that these parameters depend on the specific kind of bond and atoms involved; and need to be defined for every bond and atom in the molecule [16].

The non-bonded interactions include electrostatic and Van-der-Waals forces as well as short-range repulsion due to Pauli's principle, described by the Coulomb and the Lennard-Jones potentials, respectively:

$$V^{nb}(R) = \sum_{\text{pairs (i,j)}} \left(\frac{q_i q_j}{4\pi\epsilon_0\epsilon_r r_{ij}(R)} + \frac{C_{12}(i,j)}{r_{ij}^{12}(R)} - \frac{C_6(i,j)}{r_{ij}^6(R)} \right)$$

with $q_i, q_j, C_{12}(i,j), C_6(i,j)$ being parameters that depend on the specific type of atoms interacting, ϵ_0 denoting the vacuum permeability, the parameter ϵ_r models the relative permeability and r_{ij} represents the distance between atom i and atom j [2, 16].

The total potential energy V is then used to approximate the potential energy surface,

$$E_e(\mathbf{R}) \approx V(\mathbf{R}) = V^b(\mathbf{R}) + V^{nb}(\mathbf{R})$$

Evaluating the force field energy $V(\mathbf{R})$ is orders of magnitudes cheaper as compared to computing the

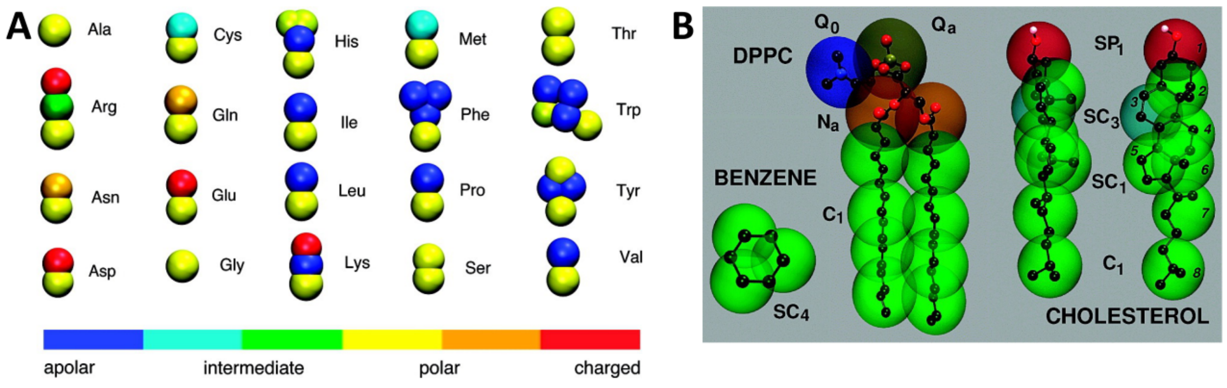


Figure 2: (**A**) Amino acids represented by coarse-grained beads and are colored by different particle type (**B**) Coarse grained beads mapped on the atomistic structure for benzene, DPPC and cholesterol molecules. *(Figure 1 A) Reprinted with permission from Monticelli, et al. The MARTINI Coarse-Grained Force Field: Extension to Proteins. J. Chem. Theory and Comput. 2008, 4, 5, 819-834. (Figure 1 B) Reprinted with permission from Marrink, et al. The MARTINI Force Field: Coarse Grained Model for Biomolecular Simulations. J. Phys. Chem. B 2007, 111, 27, 7812-7824. Copyright 2007 American Chemical Society.*

eigenvalues $E_e(\mathbf{R})$.

1.2 Efficient evaluation of the force field energy

Since each atom can only covalently bind to very few other atoms, the computational cost required to calculate the bonded interactions potential energy V^b increases only in linear order with the total number of atoms. In contrast, non-bonded interactions exist between any pair of atoms; thus, the computational cost required to calculate the non-bonded interactions potential energy V^{nb} increases with the square of the total number of atoms.

To reduce the number of calculations, one can make use of a cutoff radius. Then, the nonbonded interactions beyond the cutoff radius are approximated to zero. While such simple cutoff methods are effective in decreasing the required computational power, they are also very prone to produce artifacts, especially when applied to the calculation of Coulomb potentials. Hence, more sophisticated methods and further approximations are often used, such as potential shifts, particle-mesh Ewald summation, reaction field algorithms, or neighbor lists [2, 17]. In addition, neighbor lists are used to avoid that the distance between atoms pairs are computed every integration step.

To enable simulations of systems comprising many particles or simulations for prolonged periods of time, some force fields use united-atom or coarse-grained methods, meaning that specific functional groups or parts of molecules are represented and described by one single pseudo particle [18, 16]. In tandem with what we discussed above, this consistent reduction of the number of particles is accompanied by a significant reduction of the computational cost, but it may also impose quite severe approximations. An example of such a coarse-grained force field is MARTINI [8], which is discussed in some detail in the next section; it is the force field which will be used in this practical along with some all-atom simulations.

1.3 Coarse-grained simulations: MARTINI force field

The coarse-grained force field MARTINI mostly uses a four-to-one mapping, meaning that approximately four physical heavy atoms constitute one pseudo particle interaction center; the term heavy atom indicates non-hydrogen atoms. However, to properly reflect the geometry of ring structures, less coarse two-to-one or three-to-one mappings are used; as an example, the six atoms of a benzene ring are represented by three coarse-grained (CG) particles [8, 19].

While initially developed for lipids, MARTINI aims for a broad range of applications today; and thus CG representations are available for many different molecules including proteins. Hence, there also exist many different types of CG interaction sites in order to describe the chemical nature of the underlying atomic structure with reasonable accuracy [8, 19].

MARTINI has been parametrized to match thermodynamic experimental data such as solvation free energies or water/oil partition coefficients. The force field thus appears to be a reasonable choice for the simulations performed in this practical.

Notably, the dynamics observed with CG models like MARTINI are faster in comparison to more accurate, atomistic models. This is a direct consequence of the smoothened energy landscape resulting from increased particle size and missing friction owing to missing atomic details [8]. For example, lipid diffusion is speed up by a factor of four in MARTINI simulations [20], while sampling of the configurational space has been shown to happen up to ten times faster than in atomistic simulations [18, 21]. While fast sampling is advantageous for the statistic analysis performed in this tutorial, this also shows that CG approximations cause artifacts. Hence, caution is indispensable for the evaluation of data and results of CG simulations. However, we expect MARTINI to be able to accurately show *trends* and provide at least qualitative information.

2 Free energies and binding affinities

Biological processes such as virus entry is a stochastic event [22] – an unsurprising consequence of the immense complexity of biological systems. The topic 3 of this tutorial uses methods and terms of statistical physics to analyze and quantify the affinity of a viral fusion peptide binding to a lipid bilayer.

First, we will have a look at a simple example, to allow an easy introduction of the quantities needed to describe a system in the context of statistical mechanics. We then generalize the ideas of the description of this example to make them applicable to the systems we are interested in.

2.1 Helmholtz free energy – a simple introduction

Consider a system that has a fixed number of particles, a fixed volume, and can exchange heat with some reservoir. A system that satisfies these requirements is described by a NVT ensemble or canonical ensemble [23]. Furthermore, we assume that this system can be described classically and has a countable spectrum of states, i.e.: Let $n \in \mathbb{N}$ be the finite number of states, X_i denote some states of the system with $i \in \{1, ..., n\}$, and

$$S := \{X_i : i \leq n\}$$

the set of all possible states, also called the spectrum of states of the system. The partition function $Z(S)$ of the system is then given by

$$Z(S) = \sum_{i=1}^{n} \exp\left(-\frac{E(X_i)}{kT}\right) \tag{1}$$

where $E(X_i)$ is the energy of the state X_i, k is the Boltzmann constant and T is the absolute Temperature [23].

We define the Helmholtz free energy $F(S)$ of the system by

$$F(S) := -kT \ln(Z(S)) \tag{2}$$

Note that in the literature $F(S)$ is usually defined in a different manner, and Eq. 2 is then derived as a result [24]. In any case, one may wonder why we are interested in such a quantity, especially since the definition given here does not provide any obvious intuition – yet. The point is, that the quantity satisfying this expression yields a very useful, desirable property of high experimental and biological relevance, which we will be able to write down after introducing just one new quantity.

2.2 Reaction coordinates and the Helmholtz free energy profile

A reaction coordinate $f : S \to f(S) \subset \mathbb{R}, d \in \mathbb{N}$ is a function that maps each possible state of the system to a real number. To give an easy example, f could map the state X_i to the Cartesian z-coordinate of the center of mass of a protein Z^{prot} when the system is in the state X_i. Alternatively, f may describe the length of an important molecular distance or a degree of bending of a membrane — whatever quantity is biologically relevant.

So while there is a very graphic and intuitive interpretation for a reaction coordinate, we can also think about it in a more abstract manner (which is why we gave a rigorous definition in the first place): Since f is a function, it associates to every state X_i exactly one value $f(X_i) \in f(S)$. This implies that if $a, b \in f(S), a \neq b$ are two unequal values of f, then $f^{-1}(\{a\})$ and $f^{-1}(\{b\})$ define two disjoint, nonempty subsets of S. So f defines – and divides the system it operates on into – disjoint subsystems with respect to f, where each subsystem $f^{-1}(\{r\}) \subset S$ is associated to a specific value $r \in f(S)$ of the reaction coordinate f. From now on, we will just write $f^{-1}(r)$ instead of $f^{-1}(\{r\})$, to keep the notation simple.

As promised, the reaction coordinate bridges the gap between the abstract definition and graphic visualization and paves the way to derive a very powerful formula, justifying the formal definition of the Helmholtz free energy. Let $f : S \to f(S)$ be a reaction coordinate, $r \in f(S)$. We call

$$F(r) := F(f^{-1}(r)) = kT \ln[Z(f^{-1}(r))] \tag{3}$$

the Helmholtz free energy of the reaction coordinate f in r. Here,

$$Z(f^{-1}(r)) = \sum\nolimits_{X_i \in f^{-1}(r)} \exp\left(-\frac{E(X_i)}{kT}\right) \tag{4}$$

involves only the sum over states in $f^{-1}(r)$, in contrast to Eq. 1. Note that $F(f^{-1}(r))$ is the Helmholtz free energy of the subsystem $f^{-1}(r) \subset S$, that is, of the set of all states X_i that satisfy $f(X_i) = r$. In the example above, this would involve only the states in which the protein center of mass Z^{prot} is set to one specific fixed value.

Recall from thermodynamics that the Boltzmann distribution [23] provides the *probability* for finding the system in thermodynamic equilibrium in state X_i:

$$p(X_i) = Z(S)^{-1} \exp\left(-\frac{E(X_i)}{kT}\right)$$

Now, let $p(r)$ be the probability that the system is in *any* of the states $X_i \in f^{-1}(r)$. Because the probabilities simply add up, we find

$$p(r) = \sum_{X_i \in f^{-1}(r)} p(X_i) = \frac{\sum_{X_i \in f^{-1}(r)} \exp\left(-\frac{E(X_i)}{kT}\right)}{Z(S)} \overset{\text{(Eq. 4)}}{=} \frac{Z(f^{-1}(r))}{Z(S)}$$

and, with Eq. 3, we find the important result

$$p(r) = \frac{1}{Z(S)} \exp\left(-\frac{F(r)}{kT}\right) \tag{5}$$

However, since $Z(S)$ is just some constant for our given system, this means that the probability $p(r)$ is directly proportional to $\exp(-\beta F(r))$, where $\beta = (kT)^{-1}$. This way, we can interpret $F(r)$ as a *measure of the probability* $p(r)$, where higher Helmholtz free energy $F(r)$ along the reaction coordinate r corresponds to a lower probability $p(r)$ of the system to be in any state that returns the value r of the reaction coordinate.

Furthermore, $F(r)$ defines a function – it assigns each value r of the reaction coordinate f its respective Helmholtz free energy. We call this function the *Helmholtz free energy profile* along the respective reaction coordinate f. Alternatively, this function is often called *potential of mean force (PMF).*

Now, let $r_1, r_2 \in f(S)$ be two different values of an arbitrary reaction coordinate f. Notably, we then obtain the relative probability

$$\frac{p(r_1)}{p(r_2)} = \exp\left(-\frac{F(r_1) - F(r_2)}{kT}\right) \equiv \exp\left(-\frac{\Delta F}{kT}\right) \tag{6}$$

without having to calculate the partition function $Z(S)$ of the whole system. So if we are only interested in how likely it is for the system to be in a state corresponding to the value r_1 of the reaction coordinate relative to the probability of being in a state corresponding to the value r_2 of the reaction coordinate, we don't need to know $Z(S)$, since the relative probability depends only on the *difference* of the Helmholtz free energies $\Delta F = F(r_1) - F(r_2)$ [25].

This means that knowing the Helmholtz free energy profile along a reaction coordinate allows us to compare probabilities of our system to return a specific value of the respective reaction coordinate – providing a precise and quantitative description of what we may intuitively refer to as "favorable" or "unfavorable" states of the system.

Furthermore, we only need to know the Helmholtz free energy profile up to an arbitrary additive constant, as this constant will cancel out when calculating free energy differences; in other words, we are free to choose some reference value of the reaction coordinate, where we can define the associated Helmholtz free energy to any reference value such as zero.

2.3 Gibbs free energy

Biomolecular systems are typically not investigated at constant volume but instead at constant pressure. The ensemble taken at temperature and pressure coupling and fixed number of particles is called *NpT-ensemble* [26]. It is easy to generalize the discussion above to a different ensemble. The obtained free energy profile is then called *Gibbs free energy profile.* Because the volume of biomolecular systems remains mostly constant, the Helmholtz and Gibbs free energies are nearly equal. Hence, we often simply talk of

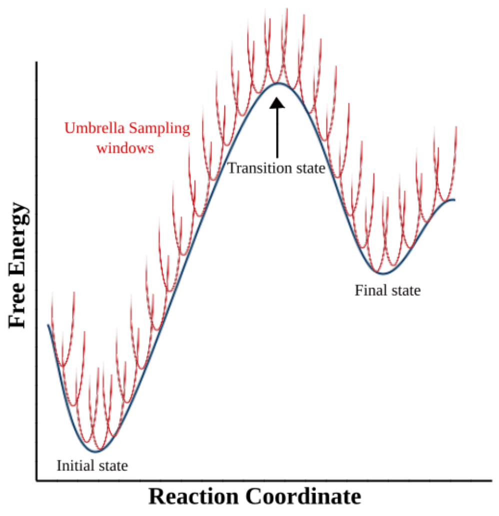


Figure 3: Free energy profile (or potential of mean force, PMF) along a hypothetical reaction coordinate (black line). Umbrella potentials (red lines) restrain the system along various positions along the reaction coordinate, thereby enforcing proper sampling of the complete reaction.

a “free energy”, but we strictly speaking mean the Gibbs free energy.

Let S be the spectrum of states of a NpT-ensemble, $Z(S)$ be the respective partition function, $\xi : S \to \xi(S)$ be a reaction coordinate, $r \in \xi(S)$ be a value of ξ and $\xi^{-1}(r)$ be the associated subsystem with respect to ξ. The quantity

$$G(r) := -kT \ln(Z(\xi^{-1}(r))) \tag{7}$$

is called the Gibbs free energy of ξ in r [27]. Any result we derived for the Helmholtz free energy in the NVT-ensemble stays true in perfect analogy for the Gibbs free energy in the NpT-ensemble [24].

2.4 Umbrella sampling

We now carry out a shift in perspective: So far, we motivated the concept of free energies by being able to derive probabilities from them, while the free energies themselves were given by abstract quantities like the partition function, which are hard to calculate in practice. Now, instead, we are interested in the free energy itself as a measure of binding affinities, such as the binding affinity of a protein to a lipid membrane. The free energy difference between the bound and the unbound state quantifies how much more "favorable" the bound state is compared to the unbound state. Instead of deriving probabilities from the free energy, we now use these probabilities to calculate the free energy:

$$G(r) = G(r_{\text{ref}}) - kT \ln\left(\frac{p(r)}{p(r_{\text{ref}})}\right) \tag{8}$$

Equation 8 is of great practical use since we can approximate the *relative probability*

$$\frac{p(r)}{p(r_{\text{ref}})}$$

by the *relative frequency of occurrence* of simulation frames yielding the respective values of the reaction coordinate; these relative frequencies can be taken from an MD simulation [25].

In practice, however, energy barriers along $G(r)$ often prohibit proper sampling of $p(r)$ within reasonable simulation times, since large energy barriers cannot be overcome easily by thermal fluctuations. To overcome such sampling problems, you will use a method called "umbrella sampling" in this practical. The key trick of umbrella sampling is to apply harmonic potentials ("restraints") along the reaction coordinate to ensure that all positions along the reaction coordinate are well sampled (see Fig. 3)

To perform umbrella sampling in practice, one first forces the system to move slowly along the reaction coordinate ξ by applying an artificial force along ξ. Then, a set of N_w snapshots is taken from the pulling trajectory at various position ξ_i $(i = 1, \ldots, N_w)$. Finally, N_w independent simulations are carried out, with restraints

$$V_i(\xi) = \frac{1}{2}k\left(\xi(\mathbf{r}) - \xi_{\mathrm{ref}}^{(i)}\right)^2 \tag{9}$$

where k is a force constant (acting along ξ) and the $\xi_{\mathrm{ref}}^{(i)}$ is the reference position of simulation i. These N_w independent simulations are called "umbrella windows" [24, 28]. After the N_w simulations have finished, the positions of the N_w simulations are gathered into N_w umbrella histograms, which can further be used to reconstruct the underlying free energy profile using a method called "Weighted histogram analysis method" (WHAM) [29].

Notably, the name "umbrella sampling" was chosen because the harmonic restraining potentials look like inverted umbrellas.

3 Outline of this practical on the SARS-CoV-2 fusion peptide

Our starting point is the fusion peptide (FP) of the severe acute respiratory syndrome coronavirus 2 (SARS-CoV-2). The FP is located at the tip of the SARS-Cov-2 fusion protein (also called "Spike protein"), where it plays a central role in the early stages of virus fusion with the host cell membrane. The FP is intrinsically disordered in water, but adopts a fold of three helices when bound to cell membrane [30]. In this practical we simulate only the FP, as it is a small peptide (42 amino acids); simulating the complete postfusion SARS-CoV-2 protein would take too long within our practical.

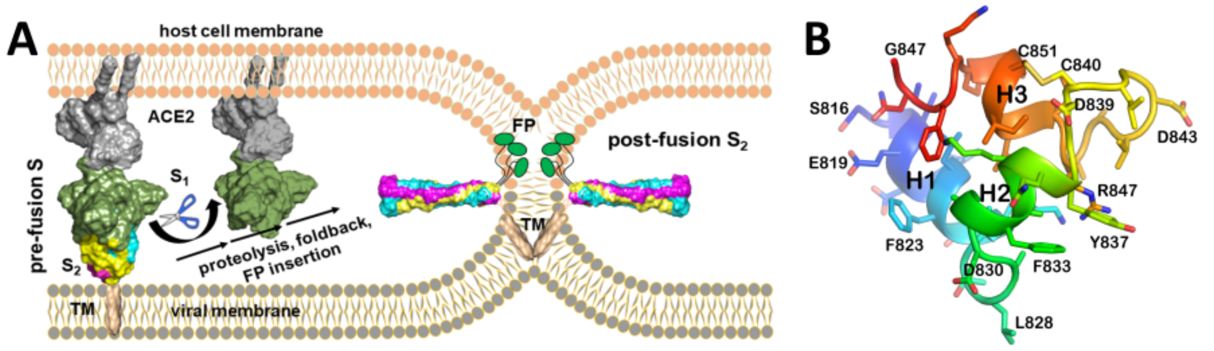


Figure 4: (**A**) Structural transformation of SARS-Cov2 spike protein from the pre-fusion S to the post-fusion S_2 state exposes the fusion peptide (FP) to the host membrane. (**B**) The fusion peptide of SARS-Cov2 (PDB ID: 7MY8) with three helices labelled H1, H2 and H3. *Reprinted with permission from Koppisetti, et al. Fusion Peptide of SARS-CoV-2 Spike Rearranges into a Wedge Inserted in Bilayered Micelles. J. Am. Chem. Soc. 2021, 143, 13205-13211. Copyright 2021 American Chemical Society.*

3.1 Outline

In this practical you will use MD simulations to study the following three topics. You will use the popular GROMACS simulation software.

1. You will investigate the structure and dynamics of the SARS-CoV-2 fusion peptide in water.
2. You will study the effect of cholesterol on properties of lipid membranes.
3. You will compute the free energy for binding of the SARS-CoV-2 fusion peptide to a lipid membrane.

3.2 Files

All the input parameter files and scripts required to setup simulations can be downloaded from the respective directories. The purpose of each file is explained below. Many tools also provide help text with the option `-h`, such as `insane.py -h`

- SARS-Cov-2 FP structure file in PDB format: `sars_cov2_fp_model1.pdb` (PDB ID: 7MY8)
- Input parameter file for minimization: `em.mdp`
- Input parameter file for equilibration: `eq.mdp`
- Input parameter file for production MD: `prod.mdp`
- Python script to setup coarse-grained membranes: `insane.py`
- Python script to convert all-atom to coarse-grained model: `martinize`
- Python script to analyze lipid order parameters: `do-order-gmx5.py`
- Bash script to setup umbrella sampling simulations: `setup-umbrella-sims.sh`

3.3 Simulation and Visualization tools

These tools are freely available and installed on your desktop computer. To use the tools form your terminal you need to source the software.bash each time you open a new terminal with

```
source /home/pract/opt/software.bash
```

GROMACS is a popular, powerful, and highly optimized MD simulation software. Gromacs uses a single command line executable `gmx`, always followed by a module name and further command line options. For example, `gmx mdrun -s topol.tpr` reads a so-called run-input-file (tpr file) and runs and MD simulation. The simulation makes, by default, use of all CPU cores and GPUs that it finds on your workstation. You find help with `-h`, e.g. `gmx mdrun -h`. The command that you will mostly use are

```
gmx pdb2gmx    # Read a protein structure, add hydrogen atoms, and write the topology file
               # topol.top, which defines all the atomic interactions (the force field)
gmx editconf   # Create a simulation box, or several other box manipulations
gmx solvate    # Add water to the box
gmx genion     # Add counter ions to neutralize the box
gmx grompp     # The Gromacs pre-processor. Read the topology topol.top, the MD parameters (mdp file)
gmx mdrun      # Run the MD simulation

You will use several analysis modules, such as
gmx rms, gmx rmsf, gmx hbonds, etc.
```

GROMACS Input and Output files Here we list some common input and output files you require to setup and run MD simulations.

Input files:

- Protein structure files - You can download structures from the Protein Data Bank (PDB) `https://www.rcsb.org/`. The PDB file includes information describing the three-dimensional (3D) structures of your biomolecule. You can view the contents by opening the file using any text editor and visualize the 3D structure using VMD or any other viewing program. You can find more details on the PDB file in this link: `https://pdb101.rcsb.org/learn/guide-to-understanding-pdb-data/introduction`.
- Gromacs uses either `.pdb` files or `.gro` files to store structures or proteins or complete simulations systems (protein and solvent). They contain the 3D Cartesian coordinates of all atoms, but also information on the atom names, atom numbers, amino acid names (also called residue names) and numbers, and the simulation box size. `pdb` and `gro` files can be used interchangeably and be converted in each with with `gmx editconf`.
- The topology file `topol.top` is written by the `gmx pdb2gmx` command based on your input protein structure file. The file `topol.top` contains information on force field parameters chosen to describe your simulation system and the `*.itp` files contains information on atom types, charges, bonds, angles, and dihedrals.

  ```
  #include "path-to-your-forcefield-of-choice/forcefield.itp"
  #include "path-to-your-forcefield-of-choice/TIP3P.itp"
  #include "path-to-your-forcefield-of-choice/ions.itp"
  #include "path-to-your-forcefield-of-choice/lipid.itp"
  ```

At the end of the topol.top file you will find information defining the molecules in your simulation system. That atoms in your PDB and topol.top files must be in the same order.

```
[ system ]
; Name
SARS-COV2

[ molecules ]
; Compound        #mols
Protein_A           1
POPC               128
TIP3P             30000
NA                  10
CL                  10
```

- Run input file (*.tpr) — This binary file contains all the information such as the starting structure, topology and simulation paramters. You will be using the `gmx grompp` command (GROMacs Pre Processor), which will collect all the input parameters and generate a `.tpr` file to be passed to `gmx mdrun` for running the MD simulation.
- Molecular dynamics parameter file (*.mdp) — This file contains various MD parameters such as the integration time step, the number of steps, settings for computing Lennard-Jones and electrostatic interactions interactions, settings for temperature and pressure coupling and constraints, and many more. A explanation on the `*.mdp` file settings can be found on GROMACS manual. For this practical, only few mdp options will be important such as

```
dt         = 0.002   ; the time step in picoseconds, here 2 femtoseconds
nsteps     = 5000000 ; number of steps, summing up to 10 nanoseconds of simulation
ref-t      = 300     ; temperature
```

  For real research projects, it is important to use correct mdp options, for instance because different force fields were parameterized for the use with different cutoff settings. In addition, reaction coordinates (such as center-of-mass distances) and restraints (so-called pulling) are defined in the mdp file, see the `pull-...` parameters.
- Index files (`index.ndx`) allow the user to define groups of atoms, such as atoms of water, protein, protein backbone etc. If no index files are provided, many GROMACS tools create default groups similar to:

```
  0 System              :  9361 atoms
  1 Protein             :   641 atoms
  2 Protein-H           :   323 atoms
  3 C-alpha             :    42 atoms
  4 Backbone            :   126 atoms
  5 MainChain           :   167 atoms
  6 MainChain+Cb        :   205 atoms
  7 MainChain+H         :   211 atoms
  8 SideChain           :   430 atoms
  9 SideChain-H         :   156 atoms
 10 Prot-Masses         :   641 atoms
 11 non-Protein         :  8720 atoms
```

```
12 Water              :  8718 atoms
13 SOL                :  8718 atoms
14 non-Water          :   643 atoms
15 Ion                :     2 atoms
16 NA                 :     2 atoms
17 Water_and_ions     :  8720 atoms
```

If you want to analyze groups that are not written by default, then you can generate groups using the `gmx make_ndx` command. For example, if we are interested in analyzing the hydrogen-bond interactions between residues 1 and 25 as function of time, we can create the groups as follows:

```
gmx make_ndx -f md.tpr -n index.ndx -o index.ndx
```

and then enter:

`1 & ri 1`

press enter button

`1 & ri 25`

press enter button twice (`ri` stands for "residue index").

You will see the two new groups at the end and by entering `q`, you will save all the changes with two new groups included in the `index.ndx` file. You will become more familiar with the usage of `gmx make_ndx` tool during the tutorial.

`gmx mdrun` output Files:
Over a simulation, `gmx mdrun` writes the following output files:

- trajectory file (`*.xtc`) - compressed trajectory file with atomic coordinates and box vectors vs. time. This is read by many GROMACS analysis tools.
- log file (`*.log`) - ASCII-text file with information on simulation process, CPU time, M-FLOPs etc.
- energy file (`*.edr`) - Binary energy file with energy contributions and box dimensions saved from the simulation, to be analyzed with `gmx energy`
- trajectory file (`*.trr`) - full-precision trajectory file with coordinates, velocities and forces information. This is normally not used.
- structure file (`confout.gro`) - final structure file written in `.gro` format. Take a look with `less` or `more` and with a viewer such as `PyMol` or `VMD`
- pull coordinate file (`pullx.xvg`) - ASCII-text file reporting the *positions* of reaction coordinates. This file is written only when pull options are activated, e.g. during umbrella sampling. The reaction coordinates and reference positions for umbrella sampling defined in the mdp file, see options starting with `pull-...`
- pull forces (`pullf.xvg`) - ASCII-text file reporting the pull force vs. time. This file is written only when pull options are activated in the mdp file.

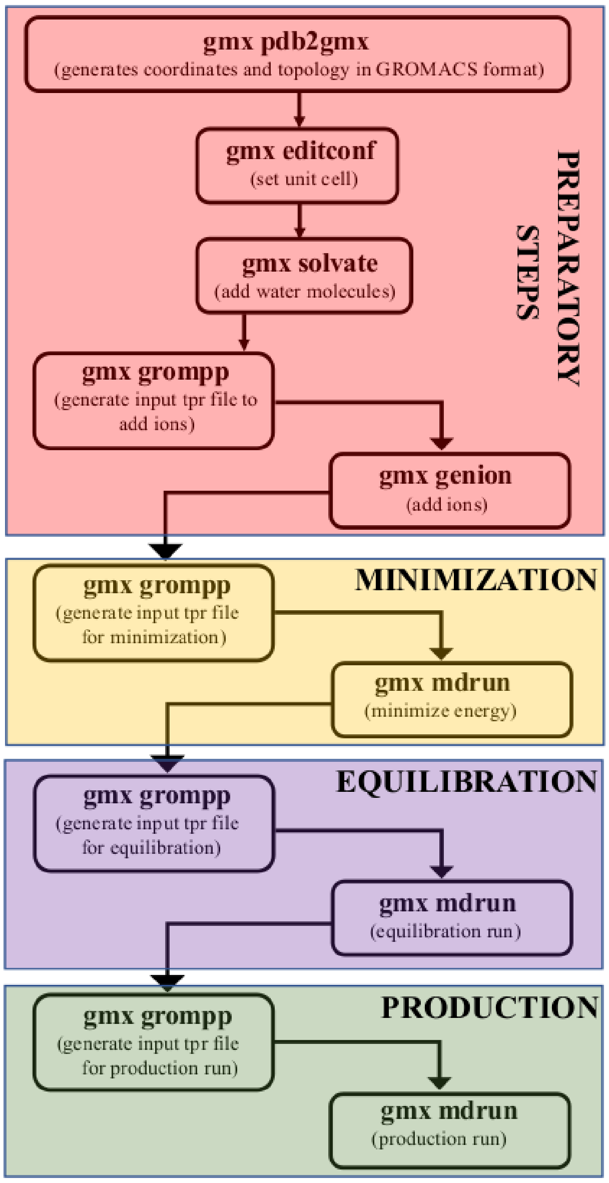
VMD, Visual Molecular Dynamics is a tool to visualize proteins, membranes etc, but it also supports various manipulation and analysis options. You will be introduced to the usage of VMD during practial course and hand-outs will be provided to get started with loading and rendering molecules.

xmgrace is a fast and simple what-you-see-is-what-you-get (WYSIWYG) 2D plotting tool. Double-click on axes and data sets if you want to change their representation (axis labels, spacing, line thickness and colors etc.) The position of the legend can be adjusted at Plot→Graph appearance→Leg. Box or Legends.

bash You will work in the command line, using the bash. If you are not yet familiar with the bash, play with the following most important commands: `ls` (`ls -l`, `ls -lrt`), `cp` (`cp -r`), `mkdir`, `mv`, `rm` (`rm -r`), `pwd`. Understand that the dot (`.`) and double-dot (`..`) denote the present and the next upper directories, respectively. You will be provided with the **bash cheat sheet** during the practicals.

Hint Organize your practical is several (sub-)directories. Run every simulation in a separate directory, otherwise you will run into a massive mess. Give your directories meaningful names and, optionally, add a small README file in each directory that documents what you have done.

3.4 This flowchart gives an overview of a MD simulations workflow with GROMACS



4 Questions

Please prepare answers to the following questions before attending to the practical course.

Biological Background

The following questions were not discussed in this protocol, but should be easy to answer by looking at cell biology book, or via you favorite internet search engine. Answering these questions will allow you to understand the biological significance of this practical.

1. Describe the structure and building blocks of proteins. What is meant with primary, secondary, and tertiary structure of a protein?
2. What is meant with "protein folding"? Speculate, why the fold of a protein is important for its function?
3. What is the main function of a biological membrane?
4. How does a lipid molecule look like? Why do lipids assemble to membranes?
5. What are the main steps in membrane fusion? Where is membrane fusion found in biology?
6. Why does nature use fusion proteins to drive membrane fusion?

MD methods

These question should help you to crosscheck if you understood the main ideas introduced in the sections above on MD simulations and the theoretical background of the methods used in this practical.

1. What is a force field? How does the simulation get from the force field $V(\mathbf{r})$ to the trajectory of the atoms $\mathbf{r}_i(t)$?
2. Mention one difference between bounded and nonbounded interactions, for instance regarding their computational cost?
3. What is the difference between all-atom force fields and coarse-grained force fields? What are their advantages and disadvantages?
4. How to find a good reaction coordinate: Imagine a protein with two domains that are connected by a flexible hinge, allowing the protein to open and close. What could be a suitable reaction coordinate to describe the opening/closing process?
5. Why are we using Umbrella sampling? Can't we get the probabilities for the bound and unbound state simply by counting the populations of the bound/unbound states in a normal simulation?

5 Tutorial 1: Structure and dynamics of SARS-CoV-2 FP in water

In this tutorial you will learn to perform classical all-atom simulations of SARS-CoV-2 fusion peptide in water.

Step 1) Processing starting PDB file

You will use the file "`sars_cov2_fp_model1.pdb`" from the "`PRACT_1_AA_FP_WATER`" folder of this tutorial. The fusion peptide is one part of the complete fusion protein consisting of 42 residues and spans from residue serine-816 to glycine-857. Below is the sequence of the fusion peptide which will be used in the tutorial.

```
  SFIED LLFNKVTLAD AGFIKQYGDC LGDIAARDLI CAQKFNG
816                        840    844        851      857
```

Figure 5: Single letter code sequence of SARS-Cov2 fusion peptide spanning from residue Ser 816 to Gly 857. Cysteine residues involved in disulfide bond are marked in green and position of residue mutated in experiments is marked in red.

Visualize the peptide in VMD to see the conformation and three helices of the peptide. Now process your pdb file using

```
gmx pdb2gmx -f sars_cov2_fp_model1.pdb -o sars_cov2_fp_model1.gro -ignh
```

When prompted, select (by corresponding numbers) the CHARMM36 all-atom force field and for the water model choose TIP3P water.

All Gromacs commands starts with the executable name `gmx`, followed by the module name (here `pdb2gmx`). Extensive help text is available via `-h`, e.g. `gmx pdb2gmx -h`.

The above `pdb2gmx` command generates three files, output structure file (`sars_cov2_fp_model1.gro`), the topology (`topol.top`) and a "include topology file" `posre.itp` that defines position restraints. Use VMD to visualize the `sars_cov2_fp_model1.gro` file:

```
vmd sars_cov2_fp_model1.gro
```

Represent the peptide as lines or licorice (see VMD tutorial) and check whether a disulfide bond has been added between the two cystein residues. You should see a covalent bond such as -C-S-S-C-.

Step 2) Setting the unit cell or the simulation box

The next step is to define a unit cell for our peptide which will be filled with solvent. Gromacs provides options to choose various boxes: triclinic, cubic, dodecahedron and octahedron. For this tutorial, we will use dodecahedron box which reduces the box volume by ~71% relative to a cubic box given the same distance between peptide and box surface. Hence, using a dodecahedron box will lead to fewer water molecules and some simulation speedup. The following command centers the peptide in the dodecahedron box with a minimum distance of 1.0 nm (-d) from the box surface.

```
gmx editconf -f sars_cov2_fp_model1.gro -o sars_cov2_fp_model1_newbox.gro -bt dodecahedron -d 1.0
```

Look at the box dimensions printed at the end of the sars_cov2_fp_model1_newbox.gro file:

```
tail sars_cov2_fp_model1_newbox.gro
```

To visualize the simulation box, open the gro file with VMD and type `pbc box` into the VMD terminal.

Alternatively, use pymol via `pymol sars_cov2_fp_model1_newbox.gro`. Press, on the right side of the Pymol window the little buttons “S” (show), “H” (hide) and “L” (labels) and play with different molecular representations (cartoon, sticks, spheres ect) or labels.

Step 3) Solvation: adding water

Now we fill the box with explicit water molecules using the command

```
gmx solvate -cp sars_cov2_fp_model1_newbox.gro -cs spc216.gro -o sars_cov2_fp_model1_solvated.gro
-p topol.top
```

The number of inserted water molecules will be printed on your terminal. The `[ molecules ]` section at the end of your topology file (`topol.top`) will reflect the new number of water molecules. Alternatively, you can count the number of water oxygen atoms using

```
grep -c OW sars_cov2_fp_model1_solvated.gro
```

and then check if the correct number of water molecules below the `[ molecules ]` section of topol.top is updated, such as

```
[ molecules ]
; Compound        #mols
Protein_A           1
SOL               5125
```

Have a look at `sars_cov2_fp_model1_solvated.gro` in VMD.

Step 4) Adding Ions

The chosen peptide might carry a net positive or negative charge; thus, the system must be neutralized by adding appropriate ions. Here you will add appropriate number of either Na or Cl counterions to neutralize the system. In addition to neutralizing ions, you can also add excess ions to mimic physiological salt concentration. If you had carefully looked into the output message on your terminal after running `gmx pdb2gmx`, the charge of your system was given. If you missed it, charge can be looked up at the last line of the `[atoms]` of `topol.top`, see the value next to `qtot`.

For adding ions, we must first create a dummy `tpr` file, but we can use an empty mdp file for this. Use

```
rm -f empty.mdp; touch empty.mdp
```

to create such a empty mdp file. The mdp file along with input gro/pdb file is processed by gmx grompp command to output a *.tpr file, which is used by gmx genion command to add ions. Below are the commands to add ions:

```
gmx grompp -f empty.mdp -c sars_cov2_fp_model1_solvated.gro -r sars_cov2_fp_model1_solvated.gro
-p topol.top -o ions.tpr -maxwarn 2
```

`gmx grompp` also reports the charge of your system. If the peptide carries a charge of -2, you would add two positively charged sodium Na^+ ions to neutralize the system:

```
gmx genion -s ions.tpr -o sars_cov2_fp_model1_solvated_ions.gro -p topol.top -pname NA
-nname CL -np 2
```

Choose water (`SOL`) to replace two water molecules with ions. Again, look at end of the `topol.top` file (e.g., with `tail topol.top`) to check the result.

Step 5) Energy Minimization

Addition of water and ions to the simulation box typically creates overlaps/clashes with the peptide atoms. To remove clashes, the system will be energy minimized using steepest descent minimization algorithm. You will now generate a `.tpr` file using the provided `em.mdp` parameter file and run the minimization using `gmx mdrun`.

```
  gmx grompp -f em.mdp -c sars_cov2_fp_model1_solvated_ions.gro
       -r sars_cov2_fp_model1_solvated_ions.gro -p topol.top -o em.tpr
 gmx mdrun -v -deffnm em
```

The energy-minized structure is written to `em.gro`. Also look at the `gmx mdrun` terminal output to observe how the potential energy decreases during the minimization. The mdrun command will generate a number of output files, among them the energy file em.edr stores information on different energy terms which you can extract use the below command:

```
  gmx energy -f em.edr -o em_Epot.xvg
```

when prompted, type "11 0" to select potential. you can plot the data using xmgrace:

```
  xmgrace em_Epot.xvg
```

You could already write out a pdb file with only protein atoms for umbrella sampling simulations which you will perform later.

```
  echo 1 | gmx trjconv -s em.tpr -f em.gro -o sars_cov2_fp_model1_minimized.pdb -pbc mol
-ur compact
```

Step 6) Equilibration

To further relax the system and to allow water molecules to rearrange around the peptide, you will perform short a equilibration (also called thermalization) with the peptide backbone restrained to its initial position. The restraints can be applied by setting the line

```
 define = -DPOSRES
```

in the mdp file, as already present at the provided `eq.mdp` file. This way, the position restraint definitions in `posre.itp` will be applied.

First, we create an index file with a group of water and ions:

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

```
...
> 13|16
> q
```

Next, the equilibration is done in the NPT ensemble (constant number of particles, pressure and temperature) for 100 ps. The index file you created before is needed here to apply the pressure and temperature coupling for the protein and the solvent as separated groups. With the following commands you run the equilibration.

```
  gmx grompp -f eq.mdp -c em.gro -r em.gro -p topol.top -o eq.tpr -n index.ndx
 gmx mdrun -v -deffnm eq
```

Have look what files are created by the mdrun, e.g. an energy file *.edr will be written out. With this file you can analyze the temperature and pressure of the system during the equilibration by using:

```
  gmx energy -f eq.edr -o eq_temp.xvg
```

and selecting "temperature". Plot the data using xmgrace:

```
xmgrace eq_temp.xvg
```

Analogously, for the pressure analyze of the system:

```
gmx energy -f eq.edr -o eq_press.xvg
xmgrace eq_press.xvg
```

Step7) Production MD run

The final production MD simulations is performed without any position restraints. Compare the mdp files from equilibration run and production runs. We want to run the simulation for 10 ns. Look at the mdp options `dt = ...` and `nsteps = ...` in `prod.mdp` to check if the simulation time is indeed 10 ns, and correct `nsteps` if needed.

```
gmx grompp -f prod.mdp -c eq.gro -r eq.gro -t eq.cpt -p topol.top -n index.ndx -o prod.tpr
gmx mdrun -v -deffnm prod
```

Depending on your workstation the simulation might take few minutes up to one hour to finish. We have run the simulation on our computing cluster for 100 ns and the trajectory file (*.xtc) is provided for you to analyze how the structure changes from 10 ns onwards. Therefor, use the provided 100 ns production run trajectory (`prod.xtc`) for the analysis.

6 Analysis

The analysis of an MD simulation typically starts with a visualization of the simulation. Does the protein behave as expected? Or does it unfold? Is some interesting transition visible? Further quantitative analysis strongly depends on the biological question that you want to answer. Below, we suggest a few very standard types of analysis.

1) Trajectory conversion and visualization in vmd

Take a look at the simulation in VMD:

```
vmd eq.gro prod.xtc
```

What do you see? Can you explain the strange output?

Before further analysis, we need to make sure that the molecules are whole, that is, that the molecules are not broken over the periodic boundaries. The trajectory (`xtc` file) can be processed with the below `gmx trjconv` command. Select 1 (Protein) for centering and 0 (System) for output

```
gmx trjconv -s prod.tpr -f prod.xtc -o fp_centered.xtc -pbc mol -ur compact -center
```

To view the trajectory in VMD, along with the above generated processed xtc file, you also need a starting structure which can be written using the below command. Select again 1 (Protein) for centering and 0 (System) for output.

```
gmx trjconv -s prod.tpr -f prod.xtc -o fp_centered.pdb -pbc mol -ur compact -center -dump 0
```

Now open the fp_centered.pdb file in vmd and load the fp_centered.xtc file to view the structure and dynamics of fusion peptide in water:

```
vmd fp_centered.pdb fp_centered.xtc
```

What to you observe now? Can you easily observe the *internal* dynamics of the peptide? To better observe the internal peptide dynamics use a mean-square fit with

```
gmx trjconv -s prod.tpr -f prod.xtc -fit rot+trans -o fp_fitted.xtc ...
```

and think yourself which options are needed to write only the peptide to the xtc file.

2) Root mean square deviation (RMSD)

To quantify the flexibility of the peptide, it is common to compute the RMSD. Use

```
gmx rms -f fp_centered.xtc -s fp_centered.pdb -o rmsd.xvg -tu ns
```

and select the Backbone. Visualize the RMSD with `xmgrace`. Stable proteins typically have an RMSD of 0.1 to 0.2 nm. And RMSD of 0.3 or even 0.4 nm indicates larger flexibility. What do you observe?

3) Root mean square fluctuation (RMSF)

To compute the residues contributing to structural fluctuations quantified by the RMSD, fluctuation of each residue with respect to the starting frame can be calculated using the command below. Select 4 (backbone atoms).

```
gmx rmsf -f fp_centered.xtc -s fp_centered.pdb -o rmsf.xvg -res
```

Which regions of the peptide are most flexible? To get a better idea of the position of different residues, open the peptide in VMD (Hint: you can select single residues in VMD)

```
vmd fp_centered.pdb
```

4) Secondary structure analysis

The by far most important secondary structure motifs in proteins are called α-helix and β-sheet. Take a look at the Wikipedia article on protein secondary structure. To observe the changes in secondary structure during simulations, you can generate secondary structure plot per residue versus time using the DSSP tool. Choose 5 (Mainchain).

```
gmx do_dssp -f fp_centered.xtc -s fp_centered.pdb -o ss.xpm -tu ns
gmx xpm2ps -f ss.xpm -o ss.eps -bx 0.5 -by 4
ps2pdf ss.eps ss.pdf
```

Is the secondary structure largely stable? Is this compatible with the RMSD?

5) Hydrogen Bonds (H-bonds)

The secondary structures α-helix and β-sheet are stabilized by H-bonds between main chain carbonyl oxygen and amide nitrogen. Loss in secondary structure correlates with reduced intrapeptide H-bonds. We can extract the H-bonds within the peptide or between water and peptide using `gmx hbond`.
To measure intrapeptide H-bonds, select 1 (Protein) twice.

```
gmx hbond -f fp_centered.xtc -s prod.tpr -num hbnum_Pep-Pep.xvg
```

To measure H-bonds between peptide and solvent, select 1 (Protein) and SOL.

```
gmx hbond -f fp_centered.xtc -s prod.tpr -num hbnum_Pep_Sol.xvg
```

Take a look at the time development of H-bonds. Is the result compatible with the secondary structure analyzed above.

Optional: Repeat the production simulation at higher temperature such as 370 K in a different directory, starting again from `eq.gro`. How do the dynamics change?

7 Tutorial 2: Effect of Cholesterol on membrane properties

In this tutorial, you will learn to perform coarse-grained MD simulations using Martini force field. You will build membranes with varying cholesterol concentration to study the effect of cholesterol on membrane properties. If you have not heard about cholesterol before (German: Cholesterin), take a brief look at the Wikipedia. Lipid membranes of animal cells contain up to 50% cholesterol. Cholesterol plays an important role in maintaining the correct stability and fluidity of membranes.

Step 1) Preparation

To setup a coarse-grained model of a lipid membrane, we use the Python script `insane.py`:

```
python2.7 insane.py -l DOPC:1 -x 9 -y 9 -z 8 -pbc cubic -sol W -o dopc.gro -p topol.top
```

The command builds a membrane composed of DOPC lipids in a cubic box of dimension $9{\times}9{\times}8\,\mathrm{nm}^3$ and solvated with a coarse-grained water model (-sol W). For more details on the functionality of the script, use the help command: `python2.7 insane.py -h`

The command outputs a structure file `dopc.gro` consisting of DOPC lipids and water molecules, and a corresponding rudimentary `topol.top` file listing force field parameter file (itp) and molecule names with their correct numbers. You are required to change the #include statement in the `topol.top` from:

```
#include "martini.itp"
```

```
to:
```

```
#include "toppar/martini_v2.2.itp"
#include "toppar/martini_v2.0_lipids_all_201506.itp"
#include "toppar/martini_v2.0_CHOL_02.itp"
#include "toppar/martini_v2.0_ions.itp"
```

All the itp files are provided. Copy the `toppar` folder containing itp files into your working directory.

Step 2) Minimization, equilibration and production run

Similar to the protocol in Topic 1, run an energy minimization and an equilibration followed by a production simulation. You find mdp files in the folder `PRACT_2_CG_MEMB`.

Step 3) Effect of cholesterol

Repeat the setup with `insane.py` using an increasing amount of cholesterol of 10%, 20%, 30%, 40%. Create separate sub-folders named DOPC_CHOL10, DOPC_CHOL20 etc. and repeat steps 1 and 2. Here, prepare an index group (`gmx make_ndx ...`) containing both DOPC and cholesterol and replace group names in the mdp file `prod.mdp` accordingly.

Optional: If you are familiar with bash scripting, try to write a bash script that, taking the cholesterol fraction as command line argument, sets up the directory with the correct name, runs insane.py, the energy minimiation, and production. This gives you a first flavor of how scripting gives you the power of doing complex simulation pipelines in a high-throughput manner.

8 Analysis

Before doing the analysis, you are required to process your trajectory file to remove jumps over the box edges.

```
gmx trjconv -f prod.xtc -s prod.tpr -pbc mol -ur compact -o nojump.xtc
```

Dump a starting structure to be used later in the analysis:

```
gmx trjconv -f nojump.xtc -s prod.tpr -b 0 -dump 0 -o 0ns.gro
```

1) Area per lipid

The area per lipid (APL) for pure membranes such as POPC, DOPC can be easily quantified by simply dividing the area of the simulation box by half the number of lipids: (Box-X * Box-Y)/(number of lipids per leaflet). The box-sizes in X and Y plane can be easily extracted from the output energy file (.edr) file using gromacs tool `gmx energy`. However, such conventional approach cannot be used for mixed membranes (DOPC-CHOL) and for membranes with proteins. Here you will calculate the APL using a software tool called FATSLiM [31], which makes use of Voronoi tessellation. The FATSLiM tool is installed on your computer. An index group for the DOPC and CHOL headgroup beads (named `PO4` and `ROH`, respectively) can be created as follows:

```
gmx make_ndx -f prod.tpr -o index_groups.ndx
```

```
...
> 2 & a PO4
> name 5 DOPC_PO4
> 3 & a ROH
> name 6 CHOL_ROH
> 5|6
> q
```

Using the index file index_groups.ndx, the APL can be calculated for the last 150 ns of the trajectory using FATSLiM as:

```
fatslim apl -c 0ns.gro -t nojump.xtc -n index_groups.ndx -b 50000 -e 200000
--plot-apl dopc_chol_10_bilayer_apl.xvg --plot-area dopc_chol_10_bilayer_area.xvg
--hg-group DOPC_PO4_CHOL_ROH --apl-by-type
```

2) Bilayer thickness

Now estimate the thickness of the bilayer by measuring phosphate-to-phosphate distance.

```
fatslim thickness -c 0ns.gro -t nojump.xtc -n index_groups.ndx -b 50000 -e 200000
--plot-thickness dopc_chol_10_bilayer_thick.xvg --hg-group DOPC_PO4
```

3) Lateral Diffusion

The tool `gmx msd` (mean-square displacement) computes the lateral diffusion coefficient of lipids in a plane. As a reference group choose the PO4 headgroup beads of the DOPC lipids.

```
gmx msd -s prod.tpr -f nojump.xtc -n index_groups.ndx -lateral z
```

4) Order parameters

Order parameters quantify the average tilting of lipid tails relative to the membrane normal. This quantity is important because it can be measured by NMR spectroscopy, thereby allowing quantitative validation of the simulated membrane against experimental data.

The order parameter is defined as

$$P_2 = \frac{1}{2}\langle 3\cos^2(\theta) - 1\rangle$$

where θ is the time dependent angle between the the C-H bond vectors in the lipid tails and the bilayer normal (z-axis). The brackets $\langle\cdot\rangle$ denote the time or ensemble average. The script for calculating the order parameter (`do-order-gmx5.py`) is provided on your workstation:

```
python2.7 do-order-gmx5.py nojump.xtc prod.tpr 50000 200000 10 0 0 1 258 DOPC
```

The command reads the trajectory from simulation times 50000 ps to 200000 ps, analyses every 10th frame relative to the z-axis (`0 0 1`), assumes that the membrane consists of 258 DOPC lipids. The output `S-profile.dat` is the order parameter profile which can compared with atomistic simulations or to experimental data.

Task: Now carry out the analysis 1–4 for lipid membranes with cholesterol content between 0% and 40%. Plot the relevant quantities as function of cholesterol content. From your analysis, describe the effect of cholesterol on the structure and dynamics of lipid membranes. In addition, compare the areas per lipid with the experimental results reported in Chakraborty *et al.*, PNAS (2020)) [32] In addition, compare the diffusion coefficients with Khana and Schwille, J Fluorescence (2006) [33]. Do you find reasonable agreement? If not, can you explain why?

9 Tutorial 3: Binding energy of SARS-CoV-2 fusion peptide (FP) to lipid membrane

Having finished the above simulations, you are now ready to carry out more advanced techniques: your fist free energy calculation. This exercise will guide you to set up and run umbrella sampling (US) simulations to calculate the binding energy of the SARS-CoV-2 FP to a lipid membrane. Pay special attention to the `pull-...` definition lines in the input parameter mdp files.

To calculate the binding energy ΔG_{bind}, a suitable reaction coordinate ξ (also called collective variable) must be chosen. Here, we take the center-of-mass distance between peptide and membrane projected onto the membrane normal (z axis). Then, a series of umbrella sampling simulations is performed along the reaction coordinate. Each simulation starts with the peptide harmonically restrained via an umbrella biasing potential at increasing distance from the membrane surface (Eq. 9). The applied harmonic restraints allows the peptide to sample the configurational space within a defined region/window (Fig. 3). From these simulations, we can extract the potential of mean force (PMF, or free energy profile) using `gmx wham` code which will give ΔG_{bind} of peptide binding to membrane. The following steps will guide you to construct PMF curve.

1) Setup of peptide-membrane system

Step 1) Setup of peptide-membrane system

Since the US simulations are performed using the Martini coarse-grained (CG) force field, we first convert the all-atom structure of the SARS-CoV-2 fusion peptide to coarse-grained resolution using `martinize` script, see `python2.7 martinize -h` for details. Use:

```
python2.7 martinize -f sars_cov2_fp_model1_minimized.pdb -o cg_system.top -x cg_protein.pdb
-dssp /home/pract/opt/dssp/dssp -p backbone -elastic -ef 500 -el 0.5 -eu 0.9 -ea 0 -ep 0
-cys auto -nt
```

The `martinize` script reads the structure in `sars_cov2_fp_model1_minimized.pdb` obtained in Tutorial 1 and converts it to CG resolution (`cg_protein.pdb`). Since Martini is too simple to maintain the secondary structure (why?), CG topology (`cg_system.top`) contains a elastic network between the C_α beads that maintains the secondary structure. The script also writes peptide parameter file `Protein.itp`.

Next, use `insane.py` to (i) build a membrane composed of 1-Palmitoyl-2-linoleoyl-glycero-3-phosphocholine (PIPC, 60%) and cholesterol (CHOL, 40%) and (ii) to position the peptide approximately 1 nm above the membrane. Notably, PIPC is a poly-unsaturated lipid.

```
python2.7 insane.py -f cg_protein.pdb -o system.gro -pbc square -box 8,8,12 -l PIPC:6.0
-l CHOL:4.0 -sol W -dm -4.5 -p topol.top
```

Adjust the `-dm` option to position the peptide as needed. To this end, look at `system.gro` in VMD and measure the distance by pressing “2” followed by clicking on two atoms. Take a look at the Martini website to get an overview of available lipids and take a look at the models for PIPC `http://cgmartini.nl/index.php/force-field-parameters/lipids` If you want to know the chemical structure, search for 1-Palmitoyl-2-linoleoyl-glycero-3-phosphocholine in PubChem.

The command outputs not only the structure file `system.gro` consisting of PIPC, CHOL and water molecules, but also a corresponding `topol.top` file listing force field parameter file (itp) and molecule names with their correct numbers. The itp files describing the martini force field, lipids, ions and solvent are provided. Copy the `toppar` folder containing itp files into your working directory. In addition, also copy the `Protein.itp` file generated using `martinize` script into `toppar` folder. You are then required to change the #include statement in the `topol.top` from:

```
#include "martini.itp"
```

```
to:
```

```
#include "toppar/martini_v2.2.itp"
#include "toppar/martini_v2.0_lipids_all_201506.itp"
#include "toppar/martini_v2.0_CHOL_02.itp"
#include "toppar/martini_v2.0_ions.itp"
#define RUBBER_BANDS
#include "toppar/Protein.itp"
```

Make sure the protein molecule name under [`moleculetype`] section in `Protein.itp` and [`molecules`] `topol.top` are the same.

2) Energy minimization and free MD simulation

As detailed in Tutorials 1 and 2, neutralize the system by adding the correct number of ions, perform the energy minimization and equilibration and run a production simulation. Use the provided mdp parameter files.

Process the trajectory file to remove jumps across the periodic boundary and use VMD to observe how the peptide associates with the membrane.

3) Steered molecular dynamics simulations

To generate starting conformations for US simulations, we use steered MD simulations (SMD) to first slowly push the peptide slightly deeper into the membrane and then to pull it slowly fully out of the membrane. In SMD, a harmonic potential is applied to the reaction coordinate ξ, while the mininum of the harmonic potential moves with constant velocity v_{ref}:

$$\begin{aligned} V_{\mathrm{SMD}}(\mathbf{R},t) &= \frac{1}{2}k[\xi(\mathbf{R}) - \xi^{\mathrm{ref}}(t)]^2 \\ \xi_{\mathrm{ref}}(t) &= \xi_0^{\mathrm{ref}} + v_{\mathrm{ref}}t \end{aligned}$$

Gromacs supports a powerful set of options for pulling along the center of mass of atomic groups, which are defined in the `*.mdp` file. For this practical, the important parameters are:

```
pull                  = yes
pull-group1-name      = ...    ; reference group
pull-group2-name      = ...    ; group to pull on
pull-coord1-type      = umbrella
pull-coord1-geometry  = cylinder
pull-coord1-groups    = 1 2    ; Use pull group1 and group2
pull-coord1-dim       = N N Y  ; pull only along Z direction (not along X or Y)
pull-coord1-vec       = 0 0 1  ; vector along which the reference position is moved
pull-coord1-init      = ...    ; reaction coordinate at t=0, $\xi_0^{\mathrm{ref}}$
pull-coord1-rate      = ...    ; pull velocity in nm/ps, $v_{\mathrm{ref}}$
pull-coord1-k         = ...    ; force constant $k$
```

Have a look at the provided `martini_pull-inside.mdp` `martini_pull-away.mdp`.

To perform the SMD simulations, you need a starting conformation where the peptide is bound to the membrane (dump a final frame using `gmx trjconv` from production simulation):

```
gmx trjconv -f nojump.xtc -s prod.tpr -b 500000 -dump 500000 -o 500ns.gro
```

Next, you need to update the index file with the reference group (Look at the mdp option `pull-group1-name = ...`) for pulling the peptide. Depending on whether the peptide binds to the upper or lower leaflet of the membrane, the reference group will be either PO4 headgroup beads from upper (upper_PO4) or lower leaflet (lower_PO4). If you are not sure on which side of the membrane the peptide is bound, open the structure dumped at 500ns.gro in VMD and update the `pull-group1-name = ...` in the `*.mdp` files accordingly. To update the index file, run the provided script (splitleaflets.py needs to be in your working directory for this):

```
./make_ndx.sh
```

The `make_ndx.sh` script reads the structure file `500ns.gro` and splits the membrane into upper and lower leaflets. The generated upper_PO4 and lower_PO4 groups, along with other groups created in step 4 are written to a new index file: `index_pull.ndx`.

Additionally, you are required to update `pull-group1-pbcatom = ...` and `pull-group2-pbcatom = ...` in the `*.mdp` files.This is needed to take care of the periodic boundary conditions (pbc) when calculating the center of mass (COM). Therefore, you have to choose on reference atom per group, one at the center of the peptide and one P04 atom of the membrane close to the peptide. Use VMD to find suitable atoms and add the residue number in the mdp files.

With this you are ready to perform the two stages of SMD simulations (pulling the peptide in and out of the membrane):

- Pull the peptide by 0.5 nm *into* the membrane over 50 ns of simulation. Use the provided mdp parameter files. Take a final close look at the pull options inside, and think about the correct sign and value of `pull-coord1-rate`. As usual, use `gmx grompp` and `gmx mdrun` to prepare and to run the simulations.

  ```
   gmx grompp -f martini_pull-inside.mdp -c 500ns.gro -r 500ns.gro -p topol.top
  -n index_pull.ndx -o pull_inside.tpr
   gmx mdrun -v -deffnm pull_inside -cpi -cpt 1
  ```

- Pull the peptide away from the membrane surface by 3.5 nm over 250 ns. Take a close look at the pull options, and think about the correct sign and value of `pull-coord1-rate`

  ```
   gmx grompp -f martini_pull-away.mdp -c pull_inside.gro -r pull_inside.gro -p topol.top
  -n index_pull.ndx -o pull_away.tpr
   gmx mdrun -v -deffnm pull_away -cpi -cpt 1
  ```

You can follow the pulling process for both the stages by plotting the pull positions in `pullx.xvg` with `xmgrace`. In addition, observe the simulation in VMD and judge whether the pulling has worked as expected.

4) Umbrella sampling simulations

To perform umbrella sampling (US), the trajectory from the pulling away SMD simulation is used. By splitting this trajectory into equally spaced frames, a series of initial configurations along the reaction coordinate is generated. With this we get one initial conformation for each umbrella window. In order to run simulations in each window, mdp files have to be prepared in such way that the position of the peptide is updated inside the mdp file (`pull-coord1-init`) for each frame. Using these frames and the corresponding mdp files, tpr files for each window can be generated with `gmx grompp`. And finally, one can run the simulation with `gmx mdrun` for all tpr files.

Setting up these steps for all windows is tedious, error-prone, and hardly reproducible. Therefore, such setups should be carried out with scripts that automate the process in a quick and reproducible manner. You are provided with the script `do-setup-umbrella.sh`, which generates the mdp and tpr files sorted in numbered subfolders for each umbrella window automatically. Have a look at the script and try to understand what it does. You have to check if the script includes the right path to your files from the pulling away SMD simulation. If not you have to adjust the path and file names in the script. Additionally check the provided `martini_umb.mdp` file and copy it to your working directory. Does this mdp file include the right reference group, pbcatoms and is the sign of the pull rate correct? If not adjust the parameter like you did before for the SMD simulations. After checking these things, you can run the script with:

```
./do-setup-umbrella.sh
```

With this script the folder `umb_t50ns` and `run-umb_t50ns` will be generated. Have a look at the created folders and the files inside. For instance, check inside the `umb_t50ns` folder with `grep pull-coord1-init *.mdp` how the position of the peptide is updated in the mdp file for each window.

Now that all tpr files are prepared inside `run-umb_t50ns`, you can run the simulations in each window using the provided script:

```
./submit_mdrun.sh
```

The script will loop into each subfolder and execute the `gmx mdrun` command.

It will take a while until all the runs are finished. Due to the limited time for this FoPra, you will be simulating each window only for 50 ns. However, peptide membrane binding events happen on longer timescales from nanoseconds to mircoseconds. Therefore, in order to analyze a converged data set, we will provide you with files from US simulation done for 250 ns. Please use the provides files for further analysis.

5) Extract the free energy profile / potential of mean force (PMF)

Each US simulation will produce `pullx.xvg` and `pullf.xvg files`. Take a look at a few `pullx.xvg` files with `xmgrace` - do they look as expected?

Now carry out the Weighted histogram analysis method (WHAM) with `gmx wham` to compute the PMF, from which you will obtain the ΔG_{bind} for peptide binding to the membrane. As input for `gmx wham`, create a file `tpr-files.dat` that contains the path and file names of all `topol.tpr` file:

```
ls */topol.tpr > tpr-files.dat
```

For example, the `tpr-files.dat` file looks similar to:

```
000/topol.tpr
001/topol.tpr
...
040/topol.tpr
```

In a similar manner, prepare `pullf-files.dat` file:

```
000/pullf.xvg
001/pullf.xvg
...
```

```
040/pullf.xvg
```

Run `gmx wham` as:

```
gmx wham -it tpr-files.dat -if pullf-files.dat -bins 200
```

First check that the umbrella histograms well overlap — if not, WHAM could not compute the PMF:

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

As the key result of the practical, plot the PMF (profile.xvg).

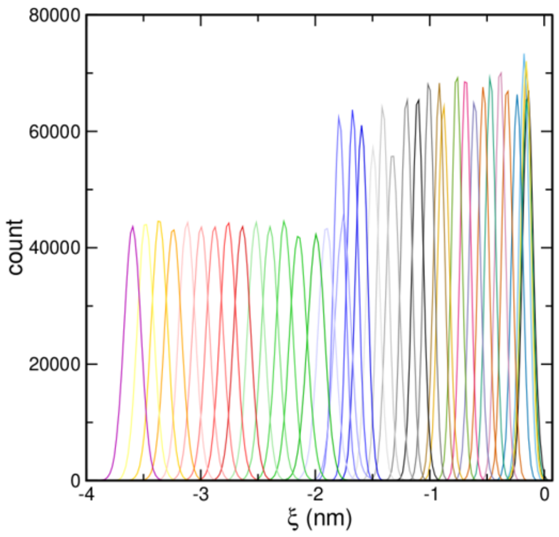


Figure 6: Histogram showing good overlap between umbrella sampling windows.

Step 10) Error estimate of the PMF by bootstrapping

Now you will estimate the statistical uncertainty of the PMF using bootstrapping. This is a widely used resampling techniques that typically simulates the measurement of a new set of data by drawing *random samples with replacement* from the given data set. In gmx wham, bootstrapping is carried out in a slightly different manner, namely by assigning random weights to the umbrella histograms [28]. Use `gmx wham -it -if -bins 200 -nBootstrap 50 -zprof0 XXX` and replace `XXX` with a position of the reaction coordinate where PMF is flat, corresponding to the region where peptide has dissociated from the membrane. Check the bootstrapped PMFs (bsProfs.xvg) and the final PMF with estimated errors bsResult.xvg.

Extract the binding free energy ΔG_{bind} from the PMF.

7) Optional: How does ΔG_{bind} depend on the lipid composition?

Experiments and previous simulations suggesting that the free energy of binding of viral fusion proteins to membranes strongly depends on the lipid composition [34]. Since membranes from different organisms or cell types (e.g., bacterial vs. animal cells) exhibit different lipid compositions, a lipid-dependent ΔG_{bind} may help viruses to target their host cells.

Investigate the effect of lipid composition on ΔG_{bind} using different lipid compositions. For example you may try (i) membranes with out cholesterol; (ii) POPE:POPG 75:25 mixtures to model bacterial membranes; or (iii) POPC:DPSM:CHOL 35:25:40 to model the outer leaflet of a typical human plasma membrane, where DPSM is sphingomyelin.

10 Summary

Congratulations, you went quite far with learning both the basics and advanced topics of MD simulations! You have now conducted MD simulations of three different systems with GROMACS, using either atomistic or coarse-grained models. You learned that proteins are flexible little machines that dynamically interact with their environments composed of water, membranes, and a lot of other proteins (not considered here).

The simulation procedures and mdp files provided in this tutorial serve as examples only. Suitable simulation procedures and mdp settings will vary depending on the biological system and on the simulation technique. Therefore, for a more extensive project such as a Bachelor or Master project, reconsider with experienced experts to modify the setups and parameters.

If you have suggestions or questions, please feel free to email us.

Happy simulating!

11 Acknowledgements

This practical was designed by Dr. Chetan Poojari, based by the Bachelor thesis by Tobias Bommer.

References

[1] Born M, Oppenheimer R (1927) Zur quantentheorie der molekeln. *Annalen der physik* 389(20):457–484.

[2] Leach AR, Leach AR (2001) *Molecular modelling: principles and applications.* (Pearson education).

[3] Cormen TH, Leiserson CE, Rivest RL, Stein C (2009) *Introduction to algorithms.* (MIT press).

[4] Billeter SR, Webb SP, Agarwal PK, Iordanov T, Hammes-Schiffer S (2001) Hydride transfer in liver alcohol dehydrogenase: quantum dynamics, kinetic isotope effects, and role of enzyme motion. *Journal of the American Chemical Society* 123(45):11262–11272.

[5] Birdsall CK, Langdon AB (2004) *Plasma physics via computer simulation.* (CRC press).

[6] Hess B, Bekker H, Berendsen HJ, Fraaije JG (1997) Lincs: a linear constraint solver for molecular simulations. *Journal of computational chemistry* 18(12):1463–1472.

[7] Huang L, Roux B (2013) Automated force field parameterization for nonpolarizable and polarizable atomic models based on ab initio target data. *Journal of chemical theory and computation* 9(8):3543–3556.

[8] Marrink SJ, Risselada HJ, Yefimov S, Tieleman DP, De Vries AH (2007) The martini force field: coarse grained model for biomolecular simulations. *The journal of physical chemistry B* 111(27):7812–7824.

[9] Hopfinger A, Pearlstein R (1984) Molecular mechanics force-field parameterization procedures. *Journal of computational chemistry* 5(5):486–499.

[10] Mayne CG, Saam J, Schulten K, Tajkhorshid E, Gumbart JC (2013) Rapid parameterization of small molecules using the force field toolkit. *Journal of computational chemistry* 34(32):2757–2770.

[11] Senftle TP, et al. (2016) The reaxff reactive force-field: development, applications and future directions. *npj Computational Materials* 2(1):1–14.

[12] Varshney V, et al. (2010) Md simulations of molybdenum disulphide (mos2): Force-field parameterization and thermal transport behavior. *Computational Materials Science* 48(1):101–108.

[13] Lindorff-Larsen K, et al. (2010) Improved side-chain torsion potentials for the amber ff99sb protein force field. *Proteins: Structure, Function, and Bioinformatics* 78(8):1950–1958.

[14] MacKerell Jr AD, Banavali N, Foloppe N (2000) Development and current status of the charmm force field for nucleic acids. *Biopolymers: Original Research on Biomolecules* 56(4):257–265.

[15] Borodin O, Smith GD, Douglas R (2003) Force field development and md simulations of poly (ethylene oxide)/libf4 polymer electrolytes. *The Journal of Physical Chemistry B* 107(28):6824–6837.

[16] Strodel B, Barz B (2020) Progress in molecular biology and translational science computational approaches for understanding dynamical systems: Protein folding and assembly preface.

[17] Abraham M, van der Spoel D, Lindahl E, Hess B, , et al. (2018) Gromacs user manual version 2018.

[18] Baron R, de Vries AH, Hünenberger PH, van Gunsteren WF (2006) Comparison of atomic-level and coarse-grained models for liquid hydrocarbons from molecular dynamics configurational entropy estimates. *The Journal of Physical Chemistry B* 110(16):8464–8473.

[19] Luca Monticelli, Kandasamy SK, Periole X, Ronald G. Larson DPT, Marrink SJ (2008) The martini coarse-grained force field: Extension to proteins. *J. Chem. Theory and Comput.* 4:819–834.

[20] Marrink SJ, De Vries AH, Mark AE (2004) Coarse grained model for semiquantitative lipid simulations. *The Journal of Physical Chemistry B* 108(2):750–760.

[21] Baron R, De Vries AH, Hünenberger PH, van Gunsteren WF (2006) Configurational entropies of lipids in pure and mixed bilayers from atomic-level and coarse-grained molecular dynamics simulations. *The Journal of Physical Chemistry B* 110(31):15602–15614.

[22] Lorizate M, Kräusslich HG (2011) Role of lipids in virus replication. *Cold Spring Harbor perspectives in biology* 3(10):a004820.

[23] Perrot P (1998) *A to Z of Thermodynamics.* (Oxford University Press on Demand).

[24] Kästner J (2011) Umbrella sampling. *Wiley Interdisciplinary Reviews: Computational Molecular Science* 1(6):932–942.

[25] Michel J, Essex JW (2010) Prediction of protein–ligand binding affinity by free energy simulations: assumptions, pitfalls and expectations. *Journal of computer-aided molecular design* 24(8):639–658.

[26] McDonald I (1972) Npt-ensemble monte carlo calculations for binary liquid mixtures. *Molecular Physics* 23(1):41–58.

[27] Attard P (1995) On the density of volume states in the isobaric ensemble. *The Journal of chemical physics* 103(22):9884–9885.

[28] Hub JS, De Groot BL, Van Der Spoel D (2010) g_wham a free weighted histogram analysis implementation including robust error and autocorrelation estimates. *Journal of chemical theory and computation* 6(12):3713–3720.

[29] Kumar S, Rosenberg JM, Bouzida D, Swendsen RH, Kollman PA (1992) The weighted histogram analysis method for free-energy calculations on biomolecules. i. the method. *Journal of computational chemistry* 13(8):1011–1021.

[30] Koppisetti RK, Fulcher YG, Doren SRV (2021) Fusion peptide of sars-cov-2 spike rearranges into a wedge inserted in bilayered micelles. *J. Am. Chem. Soc.* 143:13205–13211.

[31] Buchoux S (2017) Fatslim: a fast and robust software to analyze md simulations of membranes. *Bioinformatics* 33:133–134.

[32] Chakraborty S, et al. (2020) How cholesterol stiffens unsaturated lipid membranes. *Proc. Natl. Acad. Sci. U.S.A.* 117(36):21896–21905.

[33] Kahya N, Schwille P (2006) How Phospholipid-Cholesterol Interactions Modulate Lipid Lateral Diffusion, as Revealed by Fluorescence Correlation Spectroscopy. *J Fluoresc* 16(5):671–678.

[34] Guardado-Calvo P, et al. (2017) A glycerophospholipid-specific pocket in the RVFV class II fusion protein drives target membrane insertion. *Science* 358(6363):663–667.