Molecular Dynamics Simulations of Lipid Membranes

Biophysikpraktikum

Goethe-Universität Frankfurt
Max Planck Institute of Biophysics

Version 1.02

Ramachandra M. Bhaskara
Roberto Covino
Jürgen Köfinger
Martin Vögele

Frankfurt am Main, November 19, 2015
## Contents

1 Introduction 3

2 Molecular dynamics simulations 5
   2.1 Introduction ............................................. 5
   2.2 Integrating the equations of motions ....................... 5
   2.3 Boundary conditions ...................................... 6
   2.4 Cut-off radius and neighbor lists .......................... 7
   2.5 Thermodynamic ensembles .................................. 7

3 Force Fields 8
   3.1 Coarse-graining with MARTINI ............................. 11

4 Self-assembly of a lipid bilayer 13
   4.1 Overview of Gromacs file structure ....................... 13
   4.2 Preliminaries ............................................. 14
   4.3 System setup .............................................. 15
   4.4 Running the molecular dynamics simulation ............... 17
      4.4.1 Equilibration run ................................... 17
      4.4.2 Production run ....................................... 18
   4.5 Analysis ................................................ 18
      4.5.1 Visual analysis ....................................... 18
      4.5.2 Energy as a function of time .......................... 19
      4.5.3 Area per lipid ........................................ 19
      4.5.4 Bilayer thickness ..................................... 20
      4.5.5 Order parameter for self-assembly of the bilayer .... 21
      4.5.6 Lateral diffusion of lipids ............................ 22

Appendices 24

Appendix A Basic shell commands 24

Appendix B Plotting with gnuplot 24

Appendix C Visual Molecular Dynamics 25

Appendix D Gromacs tools used in this tutorial 25

Bibliography 26
1 Introduction

Nature employs membranes to partition and compartmentalize living systems. Biological membranes act as physical barriers to limit diffusion into and out of the compartment and regulate entry and exit of different types of molecules across membranes. Proteins embedded in membranes perform these essential gate keeping functions and sense chemicals, voltage, pH, mechanical stress, and other stimuli. Furthermore, the presence of proteins associated with energy conversion in almost all organisms is tightly associated with membranes indicating that these systems provided platforms for multiple chemical reactions during the origin of life.

Phospholipids are the primary components of biological membranes. The head groups can be chemically linked to various small molecules such as choline, ethanol amine etc. Tail groups differ in length and the degree of saturation. This repertoire of combinations of different head and tail groups generates enormous diversity in the chemical nature of the lipids and their specific interactions with other cellular components like proteins. At the same time these chemically distinct lipids share the the ability to form aggregates and higher order structures. The ability of phospholipids to aggregate and form micelles, vesicles, and bilayer structures is driven by the hydrophobic effect. The organization of individual lipid molecules within these aggregated structures such as a fluid membrane is generally dictated by their inter-molecular interactions and interactions with the solvent.

Depending on its lipid composition, a given membrane may prefer to remain flat, or it may spontaneously adopt a curved shape. It requires energy, i.e., force, to deviate from these preferred shapes. It turns out that a membrane just four or five nanometers thick has sufficient rigidity to resist thermal fluctuations, for example, which is important for maintenance of specific shapes of the cell membranes. On the other hand, this rigidity is small enough to enable biologically feasible forces, on the order of piconewtons, to generate curvature of the membranes.

The fundamental mechanism of curving a is quite simple. Curvature in bilayer is generated by an asymmetry of the two monolayers. If the two lipid monolayers that constitute the membrane have a similar structure then the lipid bilayer will remain flat. Changing the properties, molecular structure of lipids or structure of one monolayer will inevitably cause the membrane to bend. This can be achieved, for example, by binding proteins to one membrane surface but not the other. Even low-affinity binding between a protein and a membrane monolayer
can cause some curvature.

Molecular dynamics simulation allow us to study the formation of vesicles and bilayers, their structure and dynamics, and the dependence on lipid composition, for example. Many scientific questions relating to lipid membranes demand simulations on time and length scales that are inaccessible for fully atomistic simulations. Therefore, computationally efficient low resolution models of lipids, i.e., coarse-grained models like MARTINI, have been developed to study these system on current high performance computer clusters.

In this course, we will use molecular dynamics simulations to study the self-assembly of a lipid bilayer from a mixture of water and lipids. The files necessary to setup and run the simulation will be provided. A basic knowledge of unix systems (Linux), and shell command will be needed (see appendix). Help and support will be provided during the course.

This course is based on the tutorial of Prof. Dr. Siewert-Jan Marrink at the Groningen Biomolecular Sciences and Biotechnology Institute and Zernike Institute for Advanced Materials at the University of Groningen. See http://md.chem.rug.nl/cgmartini/index.php/bilayers.
2 Molecular dynamics simulations

2.1 Introduction

In molecular dynamics simulations [1, 3], the system to be simulated is considered to contain $N$ classical particles. These particles interact by mostly semi-empirical potentials $v_{ij} = v(\vec{r}_{ij})$. By differentiating these potentials, one obtains the forces acting on the particles:

$$\vec{f}_i = \sum_{i \neq j} \vec{f}_{ij} = -\sum_{i \neq j} \frac{\partial v(\vec{r}_{ij})}{\partial \vec{r}_i}.$$  

Because of this connection, we refer to interaction potential also as force fields. Their exact form shall be discussed in section 3. These forces enter Newton’s equations for each particle $i$, i.e.,

$$m_i \ddot{\vec{r}}_i = \vec{f}_i.$$

The result are trajectories for all particles, which can be used for calculating macroscopic properties (observables) that can be compared to experiments. More importantly, in simulations we obtain insight into the structure and dynamics of systems at an atomistic level, which is is hard to achieve in experiments.

2.2 Integrating the equations of motions

In general, Newton’s equations of motions cannot be solved analytically for many-body systems and we have to solve them numerically instead [3]. Here, we present the commonly used Velocity-Verlet algorithm. This algorithm is symplectic, i.e., it conserves phase space volume and it is time-reversible. Moreover, it is computationally efficient.

**Velocity Verlet algorithm**

In the Velocity Verlet algorithm, we obtain positions $\vec{r}$ and velocities $\vec{v}$ at a time $t + \Delta t$ from positions and velocities at time $t$ using

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

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

The accelerations $\vec{a}$ are calculated from the forces $\vec{a} = \frac{\vec{f}}{m}$. To calculate new positions and new velocities at a time $t + \Delta t$, we perform the following steps.
1. Calculate the position after one time step:
\[ \vec{r}(t + \Delta t) = \vec{r}(t) + \Delta t \cdot \vec{v}(t) + \frac{1}{2} \Delta t^2 \cdot \vec{a}(t) \]

2. Calculate the velocity after half the time step:
\[ \vec{v}(t + \frac{1}{2} \Delta t) = \vec{v}(t) + \frac{1}{2} \Delta t \cdot \vec{a}(t) \]

3. Calculate the forces and accelerations at time \( t + \Delta t \) according to the respective force function.

4. Calculate the velocity after one time step:
\[ \vec{v}(t + \Delta t) = \vec{v}(t + \frac{1}{2} \Delta t) + \frac{1}{2} \Delta t \cdot \vec{a}(t + \Delta t) \]

We repeat these steps to obtain a trajectory \( \vec{r}(t_i) \), i.e., the coordinates at times \( t_i \) of our system of interest.

### 2.3 Boundary conditions

The number of particles in a simulation is limited by computation time and computer memory. For such a system of finite size, we have to define boundary conditions. For small systems, the boundary conditions can have a large effect on the system properties. To avoid such finite-size effects, we aim to simulate systems as part of an infinitely large system.

![Figure 1: Schematic representation of periodic boundary conditions [1]](image)

To do so, we use periodic boundary conditions. A particle that leaves the simulation box on one side, enters the box again at the opposite side with the same velocity. These boundary conditions are equivalent to simulating an lattice of an infinite number of simulation boxes (see fig. 1).
For short range interactions, we apply the minimum image convention. For each particle, one only consider interactions with particles inside a region, which is as big as the simulation box but shifted such that the particle is in its center.

### 2.4 Cut-off radius and neighbor lists

In order to save computation resources, short-range interactions can generally be neglected for large distances. To do so, a so called cut-off radius \( r_{\text{cut}} \) is introduced, at which the interaction potential is truncated. Interactions of particles that are further apart than the cut-off radius do not have to be considered in each calculation step. Consequently, we can use neighbor lists, which contain for each particle all neighbors closer than a distance \( r_{\text{cut}} \). The runtime of the simulation will now be \( \propto N \) instead of \( \propto N^2 \), but neighbor lists have to be updated repeatedly.

### 2.5 Thermodynamic ensembles

The integration of the Newton equations of motion of a certain number of particles in a constant simulation box conserves the total energy and therefore corresponds to a microcanonical ensemble (NVE ensemble). Often, it is necessary to simulate the canonical ensemble (NVT) or the isothermal-isobaric (NPT) ensembles. Several methods have been developed to keep temperature and/or pressure constant using thermostats or barostats, respectively.

**Nosé-Hoover thermostat**

Nosé’s approach to do deterministic molecular dynamics simulations with conserved temperature was to use a modified Lagrangian. He added an additional coordinate \( s \) and an “effective” mass \( Q \) to the classical \( N \)-body Lagrangian given by

\[
L_{\text{Nosé}} = \sum_{i=1}^{N} \frac{m_i}{2} \dot{s}_i^2 - \frac{1}{2} \sum_{i,j,i\neq j} U(\vec{r}_i - \vec{r}_j) + \frac{Q}{2} s^2 - \frac{3N}{\beta} \ln s \tag{3}
\]

Using this Hamiltonian, we sample an ensemble of constant temperature that also conserves the canonical distribution. At the same time, the trajectories of the particles are deterministic and follow the corresponding equations of motion. These equations have later been simplified by Hoover.

However, the Nosé-Hoover thermostat only works correctly if there is only one conserved quantity in the system and if there are no external forces. By fixing
the center of mass possible external forces can be compensated. If there are more conserved quantities the Nosé-Hoover thermostat has to be coupled to a chain of thermostats, the so called Nosé-Hoover chains. [3]

Parrinello-Rahman barostat

Similar to the Nosé-Hoover thermostat, the Parrinello-Rahman barostat introduces an additional variable to the Hamiltonian, which serves to keep the pressure constant. The Parrinello-Rahman barostat extends this further by making each unit vector of the unit-cell independent so that both the volume and the shape of the box can change. The additional terms are therefore similar to the ones in the Nosé-Hoover thermostat, but more complex.

3 Force Fields

As mentioned in section 2.1 the force fields in a molecular dynamics simulation are interaction potentials from which the forces are calculated via

\[
\vec{f}_i = \sum_{i \neq j} \vec{f}_{ij} = -\sum_{i \neq j} \frac{\partial v(r_{ij})}{\partial \vec{r}_i}.
\]

In all-atom force fields, each atom is represented as a mass point moving in space. In so called “united-atom” force fields, hydrogen atoms that are not important for the effects to be focused on are incorporated into the heavier atom they are bound to. One can also go one step ahead and merge several atoms to one simulation bead or “effective atom”. This method is called coarse-graining (CG) and the aim of this work is to find such coarse-grained force fields of the MARTINI type as will be described in section 3.1.

Bonded interactions

Atoms that are bonded in a molecule are modeled by a special type of interaction. During the course of a simulation, bonded particles remain bonded. Consequently, chemical reactions cannot be modeled using such potentials.

- Covalent bonds are described by a harmonic potential

\[
V_b(\vec{r}_i, \vec{r}_j) = \frac{1}{2} k_b (r_{ij} - b)^2 \quad \text{with} \quad r_{ij} = \|\vec{r}_i - \vec{r}_j\| \quad (4)
\]

or by the so-called Morse potential

\[
V_{\text{Morse}}(\vec{r}_i, \vec{r}_j) = D_{ij} [1 - \exp (-\beta_{ij} (r_{ij} - b))]^2. \quad (5)
\]
The distance parameter $b$ and the force parameter $k_b$ differ for each interaction type.

- Covalent bond angles are in a similar manner described by a harmonic angular potential

$$V_a(\vec{r}_i, \vec{r}_j, \vec{r}_k) = \frac{1}{2} k_\theta (\theta - \theta_0)^2$$

\[V_a(\vec{r}_i, \vec{r}_j, \vec{r}_k) = \frac{1}{2} k_\theta (\theta - \theta_0)^2\quad(6)\]

or by a simpler potential, which is harmonic in the cosine of the bond angle

$$V_a(\vec{r}_i, \vec{r}_j, \vec{r}_k) = \frac{1}{2} k'_\theta (\cos \theta - \cos \theta_0)^2$$

\[V_a(\vec{r}_i, \vec{r}_j, \vec{r}_k) = \frac{1}{2} k'_\theta (\cos \theta - \cos \theta_0)^2\quad(8)\]

- Dihedral angles are defined as the angles between the normals $\vec{m}$ and $\vec{n}$ of two planes $i,j,k$ and $j,k,l$ that share two edges ($j$ and $k$).

$$\phi = \arccos \frac{\vec{n} \cdot \vec{m}}{nm}$$

\[\phi = \arccos \frac{\vec{n} \cdot \vec{m}}{nm}\quad(9)\]

with $\vec{n} = \vec{r}_{ij} \times \vec{r}_{kj}$ and $\vec{m} = \vec{r}_{jk} \times \vec{r}_{ik}$

\[\vec{n} = \vec{r}_{ij} \times \vec{r}_{kj} \quad \text{and} \quad \vec{m} = \vec{r}_{jk} \times \vec{r}_{ik}\quad(10)\]

The dihedral potential is then described either by a periodic function

$$V_d(\phi) = k_\phi (1 + \cos(n\phi - \phi_0))$$

\[V_d(\phi) = k_\phi (1 + \cos(n\phi - \phi_0))\quad(11)\]

for which all minima are equal and differences must be introduced by an extra interaction between atoms $i$ and $l$, or by a power series of cosine functions which is then called Ryckaert-Bellemans potential

$$V_{RB}(\phi) = \sum_{n=0}^{5} C_n \cos^n \phi$$

\[V_{RB}(\phi) = \sum_{n=0}^{5} C_n \cos^n \phi\quad(12)\]

- Improper dihedrals are defined as a harmonic restraint that keeps four atoms in one plane:

$$V_{\text{improper}}(\xi) = \frac{1}{2} k_\xi (\xi - \xi_0)^2 \cdot$$

\[V_{\text{improper}}(\xi) = \frac{1}{2} k_\xi (\xi - \xi_0)^2\quad(13)\]

Some bonded interactions (mostly hydrogen bonds) are stiff enough to produce only high frequency oscillations ($\nu \ll \frac{k_B T}{h}$). They can be replaced by constraints. These constraints can be realized either by resetting the coordinates after performing a conventional unconstrained step (SHAKE algorithm) or by rewriting
the equations of motion in a matrix notation using Lagrange multipliers, so they contain the constraints. Nowadays, the most efficient algorithm of these methods is the LINCS algorithm. It is faster, more robust, more accurate and easier to parallelize than SHAKE, but more complex. The SETTLE algorithm solves the constrained equations analytically and can be used in special applications, for example in water models. [2]

Non-bonded interactions

Those atoms that are not bonded in a molecule are assumed to interact only by pairwise additive terms of the following types.

- Lennard-Jones interactions include the van der Waals interaction, which is attractive and proportional to \(-r^{-6}\), as well as a repulsive term representing the Pauli exclusion principle, which is arbitrarily chosen as \(\propto r^{-12}\). The energy is usually written in one of the following forms:

\[
v_{\text{LJ}}(r) = \frac{C_{12}}{r^{12}} - \frac{C_6}{r^6} = 4\varepsilon \left(\left(\frac{\sigma}{r}\right)^{12} - \left(\frac{\sigma}{r}\right)^{6}\right)
\]

in which \(\sigma\) is the zero point of the function and \(\varepsilon\) is the depth of the minimum.

A more realistic but also more expensive alternative is the so called Buckingham potential which is using an exponential repulsion instead of the \(r^{-12}\) term:

\[
v_{\text{rep}} = A \exp(-B r).
\]

- Coulomb interactions are calculated between the atom centers that are given so called effective or partial charges \(q_i\):

\[
V_C(r) = f_{el} \cdot \frac{q_i q_j}{\varepsilon_r r}
\]

Finding suitable partial charges is highly non-trivial. Often they are derived from empirical dipole and quadrupole moments of small molecules. Charges can also be obtained from an analysis of orbital occupations. For force fields, it is best to use potential-derived charges that reproduce the electric potential around the molecule (for example RESP charges).

Coulomb-interactions are long-range interactions and one cannot apply a cut-off. In principle, one has to calculate all Coulomb interactions between
particles in all periodic images of the simulation box. To that end, methods like Ewald summation and Particle Mesh Ewald method (PME) have been developed. However, in this course we use a coarse-grained force field (MARTINI), where Coulomb interactions are not calculated explicitly but treated as effective short-range potentials. Therefore, we do not discuss these methods of great importance for atomistic simulations here.

Non-bonded interactions are not applied to pairs of atoms that are already form a covalent bond. Often also interactions with those atoms are excluded that are connected indirectly via two or more covalent bonds. Sometimes the latter ones are included but in a modified form, depending on the exact realization of the force field. [2]

## 3.1 Coarse-graining with MARTINI

Simulations considering each single atom are very expensive and in order to reach longer scales in space and time, coarse-graining (CG) can be used to decrease computational effort. Coarse-graining means merging several atoms to one simulation bead or “effective atom” so the number of particles to be simulated decreases. There are several CG approaches that differ widely in the accuracy of their mapping i.e. how many ”real” atoms are mapped to one bead. A CG Force field is usually parameterized by matching it to all-atom results. This is not trivial, because by merging the atoms, the potential energy landscape becomes smoother, which changes the dynamics of the simulation drastically. The conversion of the speeds of several effects on different scales is often different, so there is no unique relation between the timescales of all-atom simulations and coarse-grained simulations.

The coarse-grained model used in this work is the MARTINI model introduced by Marrink et. al [4]. It was originally designed for lipids and has been extended to biochemical compounds in general.

The mapping of the MARTINI model is basically four-to-one, which means that four heavy atoms are mapped to one interaction site. For ring-like structures, a finer mapping of two or three atoms to one site is chosen in order to represent their geometry more accurately. Liquid water fits into this scheme due to the tetrahedral coordination and four water molecules are represented by one CG bead. Also, ions including their first hydration shells are mapped to single beads.

The MARTINI model comprises four principal types of particles (abbreviated by capital letters): polar (P), non-polar (N), apolar (C), and charged (Q).
Figure 2: Examples for MARTINI mapping from the chemical structure to the coarse-grained model. The coarse grained bead types, which determine their relative hydrophilicity, are indicated. The prefix S denotes a special class of coarse-grained sites introduced to model rings. [4]

Hydrogen bonding capability is denoted by lowercase letters or zero: donor (d), acceptor (a), both (da), none (0, i.e., zero). Additionally, numbers indicate the degree of polarity from 1 (low polarity) to 5 (high polarity). All in all, we have 18 different "building blocks" available.

The non-bonded beads interact by a Lennard-Jones 12-6 potential. For the interaction parameters, see the original publication [4]. The charged groups of type Q are additionally assigned a charge $\pm e$. Their Coulomb interaction is screened by using a dielectric constant $\varepsilon_{\text{rel}} = 15$. All non-bonded interactions are cut-off at a distance $r_{\text{cut}} = 1.2 \text{ nm}$ and shifted from $r_{\text{shift}} = 0.9 \text{ nm}$ to $r_{\text{cut}}$.

In the MARTINI model the neighbor list searching radius $r_{\text{list}}$ is equal to the cut-off radius. This leads to a systematic error as atoms are taken into account, which have moved out of the respective area between two neighbor list updates and some are not taken into account that have moved in. Therefore the neighbor list algorithm (in the form as implemented in Gromacs) with updates every 10 steps is considered to be a part of the MARTINI model.

The bonded interactions are modeled by harmonic bond potential, angle potentials, proper dihedrals, and improper dihedrals.
4 Self-assembly of a lipid bilayer

The goal of this course is to learn how to setup, run, and analyze molecular dynamics simulations.

In molecular dynamics simulations, we calculate the evolution in time of a model system. For this purpose, we (1) need a model of the system, i.e. the coordinates of all particles, we (2) have to calculate the forces acting between the different components and (3) we need a molecular dynamics engine, that is an algorithm to solve Newton’s equations of motions.

In the following, we will simulate and study a lipid bilayer, the fundamental component of biological membranes. We will use the MARTINI force field, which is a coarse-grained force field specifically developed for the simulation of lipids, and more recently also extended for the simulation of proteins embedded in the lipid bilayer. We use the Gromacs program, which is one of the most popular molecular dynamics engines freely available on the web.

4.1 Overview of Gromacs file structure

To setup and run a Gromacs simulation, we have to prepare the configuration of the system we want to simulate, decide what type of simulation we want to perform, and specify the parameters. All files needed at this stage are text files, and can be modified with any text editor, e.g. vim or emacs. Doing so we obtain a binary file with extension .tpr. This file is not human readable anymore, but contains all information necessary to actually run the simulation.

In a simulation setup, we need a parameter file, a structure file, and a topology file. These different file types are explained below. While the format and names of these files vary between different molecular dynamics programs, the basic concept is essentially the same.

1. Parameter file (.mdp) – Input file containing all parameters of the simulation. An molecular dynamics simulation is a technically very complex object, and one has to specify many parameters. Open the minimization.mdp file and try to give a look to the different parameters (the file is heavily commented). It is not necessary and indeed impossible to understand the meaning of each of those parameters at the beginning, and only time and experience will bring some familiarity with most of them. Some of them, tough, are or primary importance.

   integrator – The numerical algorithm used to evolve the system.
dt – The time step to be used in the propagation of the system.

nsteps – Total number of time steps to be used, which determines the total duration of the simulation.

It is important to note and keep in mind that the MARTINI force field was developed and optimized to be used with a given set of parameters. For this reason, it is wise and safe not to change the parameters that are recommended by the authors of the force field, unless one is really sure of the modification she or he is going to apply.

2. **Structure file (.gro)** – The structure file contains the initial configuration of the system.

3. **Topology file (.top)** – In this file one has to specify which force field parameters have to be used and the type and number of molecules of different kind that are present in the .gro file. Depending on the system, the topology file can be created by using some scripts or has to be build up by hand. Note that the force field parameters are used by explicitly including the .itp files `martini_v2.1.itp` and `martini_v2.0.lipids.itp`. These are again text files that contain all the parameters of a given force field with a well defined format. The .itp files have to be contained either in a system folder of Gromacs, or in the current working folder.

These files (.mdp, .gro, .top) contain all information that is needed to run a simulation.

We will perform some simple simulations and learn step by step the key ingredients of a molecular dynamics simulation of a biomolecular system.

### 4.2 Preliminaries

For the present tutorial, some basic familiarity with LINUX operating systems and BASH commands is required. If you do not have any experience with any of these topics, please consult the appendix for short introductions.

You should have a folder containing the following files needed for setting up simulations

- water.gro
- dspc.top
- dspc_single.gro
• minimization.mdp
• martini_md.mdp
• martini_v2.0_lipids.itp
• martini_v2.1.itp

for analysis and visualization,
• axes.py - Show coordinate axes in pymol (run axes.py).
• cg_bonds.tcl - Tcl script to show MARTINI bonds in vmd.
• README_cg_bonds.txt - Explaining the usage of cg_bonds.tcl.
• do-order-multi.py - Python script to analyse calculate $P_2$ order parameter (see Eq. (17))

and the course description and gromacs manual
• manual-4.6.7.pdf
• Versuch13_Molecular_Dynamics.pdf

For your convenience, please add the gromacs path and vmd path to your search path, i.e.,

$ export PATH=’/usr/local/gromacs/bin:/$PATH’

and

$ export PATH=’/usr/local/vmd/bin:/$PATH’

### 4.3 System setup

To simulate the self-assembly of a lipid bilayer, we need a starting configuration of a mixture of lipids and water. Structures of a lipid molecule and a water molecule are provided by MARTINI. We put a number of lipid molecules randomly in a simulation box and fill up the remaining void space with water.

The structure of a single DSCP lipid is given in the file dscp_single.gro. Files with the extension ‘.gro’ are text files of fixed format and contain all information necessary to determine the structure of a molecule.

• Take a look at the file (open it in a text-editor) and try to understand the content.
Now let us visualize the lipid, to have an idea of how it looks like. In order to visualize molecules, we will use the program “Visual Molecular Dynamics” or VMD for short, e.g.,

$ vmd dspc_single.gro

We use this structure to build our simulation box, which contains all elements of the system we want to simulate. For simulations with Gromacs, we need a structure file (.gro) of the simulation box. We do not have to create such a file by hand, which would be a tedious and prone to error. Indeed, most of these mundane but frequent tasks can be carried out by using scripts and programs provided by Gromacs or other sources. These programs are executed in a shell and usually have multiple command-line parameters.

To set up the simulation box, we use the program genbox. To read the help text of this program, execute the following command in a shell. The “$” sign indicates the so-called command prompt, and must not be typed.

$ genbox

• What does the following line do?

$ genbox −ci dspc_single.gro −nmol 128 −box 7.5 7.5 7.5 −try 500 −o 128_noW.gro

Filling the simulation box randomly with molecules will lead to high energy configurations, i.e., molecules that are very close to each other. In order to relax such high energy configurations, we use an optimization algorithm (e.g., steepest decent) to minimize the energy of the system. Performing an energy minimization follows the same steps as performing a molecular dynamics simulations. The difference between these two modes of operation is that we either have to specify a minimization algorithm or an integrator.

The program grompp will combine these files in a single .tpr file, i.e.,

$ grompp −f minimization.mdp −c 128_noW.gro −p dspc.top −maxwarn 10 −o dspc−min−init.tpr

The molecular dynamics engine of Gromacs, the program that is actually performing the simulation, is mdrun. To run the energy minimization, type

$ mdrun −deffnm dspc−min−init −v −c 128_minimised.gro

The minimization should be very quick and should remove the most severe clashes between lipid molecules.
So far our system contains only lipids, but the self-assembly of a bilayer is an entropic driven phenomenon that is possible only in presence of water. We will use the genbox tool again to add MARTINI water, that is a coarse-grained water model where one bead corresponds to four atomistic water molecules:

\[
\text{genbox} -c p 128\_minimised\_gro -c s water\_gro -o waterbox\_gro -m axsol 768 -v dwd 0.21
\]

- From the output of the program, how are the water molecules put into the box?
- Why do we have to use the \texttt{-vdwd} flag? What happens if we do not use it?

Once we have added water to the box, we have to minimize the potential energy of the structure again. Before using \texttt{grompp}, open the topology file in an editor and modify it correspondingly to reflect the presence of the water molecules (hint: “;” indicates a comment). Again, we minimize the energy of the structure using

\[
\text{grompp} -f \text{minimization}\_mdp -c waterbox\_gro -p dspx\_top -m axwarn 10 -o dspx\_min\_solvent\_tpr
\]

\[
\text{mdrun} -d effnm dspx\_min\_solvent -v -c minimised\_gro
\]

### 4.4 Running the molecular dynamics simulation

We are ready to run a molecular dynamics simulation of the system in the canonical ensemble starting from the energy minimized configuration of lipids and water. We will perform two simulations. First we will equilibrate the system and then we will perform a production run of the formed bilayer.

#### 4.4.1 Equilibration run

Such a simulation is different compared to a simple energy minimization, and consequently the input file has to change. To prepare the new \texttt{.tpr} file, type

\[
\text{grompp} -f \text{martini}\_md\_mdp -c minimised\_gro -p dspx\_top -m axwarn 10 -o dspx\_md\_tpr
\]

Have a look at the \texttt{martini}\_md\_mdp file:

- How long will we simulate the system?
- What are the two most important differences compared to the previous parameter files?
Now we can run the simulation, by typing

```
$ mdrun -deffnm dspc -md -v
```

This step can be quite lengthy, and depending on the amount of available cores it might take around one hour. (Gromacs should provide an estimate of the required time).

### 4.4.2 Production run

In the first run a bilayer should form. The bilayer might not be formed perfectly. For this reason, and because we do not want our analysis to be influenced by the equilibration phase, we use the end point of our first simulation, the **equilibration run**, for a new simulation, i.e., the **production run**. The resulting trajectory will be used to analyse the properties of the bilayer.

```
tpbconv -s previous.tpr -extend timetoextend by -o next.tpr
```

'previous.tpr' is your old .tpr file and 'timetoextend by' is the time in ps of your new run. This command will generate a new .tpr file called 'next.tpr' here, which will be used for the simulation via

```
mdrun -s next.tpr -cpi previous.cpt -noappend
```

where 'previous.cpt' has been generated by your first simulation and contains the state of your system at the last frame. File names have to be adopted accordingly.

### 4.5 Analysis

#### 4.5.1 Visual analysis

While performing a simulation, one should check the trajectory visually, for instance using VMD. We want to make sure that the evolution of the system looks reasonable and we check for pathological behavior, which would indicate technical problems or mistakes in the input file. For your first simulation,

- Does the bilayer form? If yes, when (after how many timesteps)? Is the possible bilayer stable during the simulation?

- How is the bilayer oriented? Note the direction normal to the bilayer plane.

- Some molecules are jumping across the boundaries of the box. What is the reason for that?
4.5.2 Energy as a function of time

The Gromacs tool `g_energy` can read and extract many parameters of the simulation from the binary `.edr` file. Using

```
$ g_energy -f dspc-md.edr -o total_energy.xvg
```

we can choose, which quantities should be given out. `g_energy` produces an `xmgrace` file with the extension `.xvg`, which can be viewed using

```
$ xmgrace total_energy.xvg
```

or using `gnuplot`.

- Write out the total energy as a function of time (Option 7 Total-Energy) for both simulations?
- Do you see any correlations of the total energy with the formation of the lipid bilayer?

For the following analysis, only the production run should be used.

4.5.3 Area per lipid

An important parameter of lipid membranes is the area per lipid, quantifying how densely the lipids are packed. Approximately half of the lipids should be located in each leaflet of the bilayer. Consequently, the area per lipid can be estimated using the total number of lipids and the area of the membrane. The latter is obtained from the box dimensions. The parameters that have been calculated and stored during the simulation are shown.

- Use `g_energy` to obtain the average values of two box dimensions, in which the lipid bilayer has formed.
- What is the average area per lipid?

A time average along the whole trajectory will be returned. This might be not very accurate, because we should perform the average only in the segment of the trajectory where the bilayer is formed. We could thus use the option `-b` to specify the first frame from which to start to read from the trajectory. This will be in general true for all of the following analyses. To have an idea of the magnitude of the error we make by considering the whole trajectory, we can give a look at the time series the box dimensions, that were recovered by `g_energy` and printed in the `.xvg` file. Note, that if you do not specify an output file using the `o` option, the default file will be called `energ.xvg`.  

19
4.5.4 Bilayer thickness

Another important physical property of the bilayer is the thickness. How can we define it? A possible solution is to take the difference between the average position of some heavy atom in the head groups of the lipids, for instance the phosphate groups represented by beads with names starting with 'P'. We must therefore create and index file containing the atom numbers corresponding to P atoms. This can be done by using the `make_ndx` Gromacs tool. Execute

```
$ make_ndx -f dspc-md.gro
```

and select only the P beads in the configuration by choosing 'atom' and all phosphate beads by typing

```
$ a P*
```

The tool should find 128 atom types of this kind. Type `q` to save and exit the program. Now we will invoke `g_density` to calculate the density of the P beads along the axis normal to the bilayer. In the following we assume that the membrane normal is in direction of the z-axis. Before using this tool check carefully the orientation of the membrane normal and adapt the commands below accordingly.

To calculate the density along the z-axis, execute

```
$ g_density -f dspc-md.xtc -n index.ndx -s dspc-md.tpr -d Z -b 5000
```

The 'P*' group should appear among the possible choices, select it and let the program calculate the desired density. This command will produce a `density.xvg` file, which can be plotted for instance with gnuplot, by typing

```
$ p 'density.xvg' u 1:2 w l
```

Two peaks should be clearly visible, and the bilayer thickness can be estimated from their location.

- Which kind of density did we plot? What are the units of measure?
- Calculate the density along the membrane normal for water and carbon beads. Which densities are more meaningful and should be plotted in the same figure?
- Do the densities overlap? What does this mean?
4.5.5 Order parameter for self-assembly of the bilayer

We want now to calculate a quantity that measures in a the progression of the system from a disordered phase, where lipids and water are randomly mixed, to an ordered phase, where the bilayer has formed. Such a quantity is usually known as an order parameter. How can we follow the formation of the bilayer? What could we measure to monitor the increasing order of the bilayer in time? One possibility is to use the so called $P_2$ second-rank order parameter, that measures the alignment with a specified direction. It is defined as

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

(17)

where $\theta$ is the angle formed by a given bond stretching between two atoms and the normal to the plane of the bilayer. Angle brackets indicate the average over all lipids for a single frame.

- Can you explain how this quantity behaves in different situations? What is the value of $P_2$ for a bond that is parallel/orthogonal to the bilayer normal?

- What characterizes a good order parameter and is $P_2$ a good order parameter to monitor the formation of the bilayer?

Let us visualize again the structure of one single lipid molecule by using VMD, by typing

$\texttt{vmd dspc single.gro}$

- Which bond would you choose to calculate $P_2$?

In order to calculate the value of $P_2$ on all bonds along the simulated trajectory, we will use the do-order-multi.py script. For this purpose, create an analysis folder, and copy there the trajectory as trajectory.xtc and the .tpr file of the molecular dynamics simulation as topol.tpr (the script accepts only the files with these names). The command to run the analysis is given by

$\texttt{python do-order-multi.py traj.xtc 0 1500000 20 0 0 1 128 DSPC}$

where we have to specify

- the trajectory file,
- the initial time in the trajectory to start the analysis (in ps),
• the time where to stop the analysis (in ps),
• the number of frames to skip along the trajectory,
• the axis normal to the bilayer (1 0 0 X, 0 1 0 Y, 0 0 1 Z, ),
• the number of lipid molecules,
• the type of lipid molecules.

The script produces a order.dat file. The file contains a table with the values of $P_2$ as function of time for each bond in the specific lipid molecule.

• Plot the time series of $P_2$ for some bonds you think may be important for our problem. What can we learn from these plots?

4.5.6 Lateral diffusion of lipids

The bilayer is a sort of liquid and the lipids are free to diffuse within each leaflet. The mean square deviation, is given by

$$MSD(\tau) = \langle [\vec{r}(\tau) - \vec{r}(0)]^2 \rangle,$$

(18)

where $\vec{r}$ is the position vector of a lipid. According to Einstein’s classical result, $MSD(\tau)$ is proportional to the diffusion constant, i.e.,

$$MSD(\tau) = 4D\tau,$$

(19)

where the factor 4 accounts for the diffusion in two dimensions. The diffusion constant can be determined from the slope of a linear fit to $MSD(\tau)$.

• Remove the periodic boundary conditions by using ”unwrapping” the motion of each lipid molecule:

```
$ trjconv -f traj.xtc -s topol.tpr -pbc nojump
    -o traj_nj.gro
```

• Calculate the MSD using

```
$ g_msd -f traj_nj.gro -s topol.tpr -rmcomm
```

Why do we need to remove the motion of the center of mass of the system?

• This script returns a file msd.xvg, and by plotting its content we should find a straight line. Is the plot a straight line over all its domain? What could be a possible reason for any deviation from a straight line?

22
• Fit Eq. 19 to the linear regime of the plot. This can be done with gnuplot or with \texttt{g.msd} (that would do some supplementary statistics, too).

\$ \texttt{g.msd -f traj.nj.gro -s topol.tpr -rmcomm -lateral z} \$

where \( z \) is the membrane normal.
Appendix A  Basic shell commands

Various different shells exist, which allow to execute commands from a terminal window. For this course bash is recommended, which you can start by typing

$ bash

int a terminal window. In the following, a short list of the most useful commands is presented.

- **ls** - Show lists of files or information on the files.
  - `ls file` Does the file exist?
  - `ls -l file` Show information about the file.
  - `ls *.txt` Show all files ending in .txt.
  - `ls dir` Show contents of directory.

- **cd** - Change current directory.
  - `cd` Go to home directory.
  - `cd ~/papers` Go to /home/user/papers.
  - `cd dir` Go to directory ‘dir’ (relative).
  - `cd -` Go to last directory you were in.

- **pwd** - Show current working directory.

Appendix B  Plotting with gnuplot

Plotting with gnuplot is easy and fun! Be sure to have a data file with numbers arranged in columns, for instance for a time series $x(t)$, a first column with the values of $t$ and a second column with the values of $x$. Type

`gnuplot`

to access the program. To plot the curve using columns 1 and 2 of the data file use

`plot 'filename' using 1:2 with lines`

Gnuplot has been designed for quick plotting, and key words can be contracted unless there is any ambiguity. So the previous command becomes

`p 'filename' u 1:2 w l`

If we have the sightliest more complicated situation of having $t$, $x(t)$ and $y(t)$ as column 1, 2 and 3 in the same file, the plotting command would look like

`p 'filename' u 1:2 w 1 t "x(t)" , "" u 1:3 w 1 t "y(t)"`
where \( t \) stands for \texttt{title}, which adds a label to a curve and which increases the readability of plots with many curves. We recommend to follow some of the many excellent tutorials that can be found online in order to learn more advanced features. For example, \url{http://www.mathematik.hu-berlin.de/~ccafm/lehre_BZQ_Numerik/allg/gnuplotkurs.html}.

**Appendix C  Visual Molecular Dynamics**

The program Visual Molecular Dynamics or VMD is freely available for various operating systems at \url{http://www.ks.uiuc.edu/Research/vmd/}. For a tutorial see \url{http://www.ks.uiuc.edu/Training/Tutorials/vmd/tutorial-html/}.

VMD will be briefly introduced in the course.

**Appendix D  Gromacs tools used in this tutorial**

Gromacs is freely available and can be downloaded from \url{http://www.gromacs.org/Downloads}.

- \texttt{genbox} Generates and modifies a simulation box.
- \texttt{grompp} Gromacs preprocessor. Merges text file containing input parameters and initial configuration in a runnable binary file.
- \texttt{mdrun} Gromacs main MD engine.
- \texttt{g_energy} Extract quantities stored in the .edr file during the simulation.
- \texttt{makendx} Makes a Gromacs index file, i.e. a file with a label associated to a a group of atoms.
- \texttt{g_density} Calculates density of a given group along a direction.
- \texttt{trjconv} Converts Gromacs trajectories between different formats, changes the number of frames, removes PBC conditions and much more.
- \texttt{g_msd} Calculates MSD and estimates linear and lateral diffusion of a given type of atoms or molecules.
References


