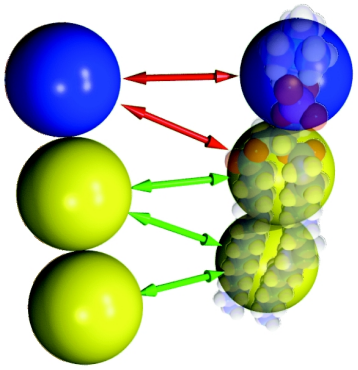
# Mesoscopic membrane simulations with mbtools

Tristan Bereau, Mingyang Hu, Patrick Diggins, and Markus Deserno
Department of Physics
Carnegie Mellon University
Pittsburgh, PA USA

October 4, 2012

## 1 Introduction

This tutorial introduces the `mbtools` package contained in Espresso. It implements the mesoscopic membrane model of Cooke *et al.*[1] While of low resolution, this three-bead-lipid, solvent-free coarse-grained model can self-assemble fluid bilayers as well as reproduce physically meaningful large-scale parameters of the membrane.[2] This tutorial will illustrate some of the capabilities of the coarse-grained model: bilayer self-assembly, bending modulus and line tension coefficients determination, and energetics of spontaneous vesicle formation.



[1] Cooke, I. R., Kremer, K. and Deserno, M. (2005): Tuneable, generic model for fluid bilayer membranes. *Phys. Rev. E.* 72 - 011506

[2] For a review, see Deserno, M. (2009): Mesoscopic Membrane Physics: Concepts, Simulations, and Selected Applications. *Macromol. Rapid Commun.* 30 - 752-771

# 2 Background

Lipid membranes form the barriers that compartmentalize the cell. They consist of two thin layers of lipids. These amphiphilic molecules (i.e. possessing both hydrophilic and hydrophobic properties) can self-assemble into large structures in order to minimize the exposition of their hydrophobic tails to the (polar) solvent. The main structures that lipids form in solution include micelles (spherical shape where all tails point to the center), bilayer sheets, and vesicles.

At a mesoscopic level ($\sim 10\,\mathrm{nm} - 10\,\mu\mathrm{m}$) the structure and energetics of membranes are often modeled using continuum elastic theory. To bridge the gap between continuum theory and molecular details, coarse-grained simulations of lipids have become increasingly popular. Here we will use a model that coarse-grains away most of the chemical details of a lipid, but, as will be shown in the tutorial, can self-assemble into membranes as well as reproduce physically meaningful large-scale parameters.

One molecule is represented by only three beads. This allows us to define two different types of beads (i.e. hydrophobic head beads, hydrophilic tail beads), as well as to reproduce a fair lipid aspect ratio. The model does not contain any explicit solvent. While this largely increases the computational efficiency of the model, it requires some care when parametrizing the interactions between the lipid beads. Hydrophobicity in an implicit solvent model can be represented by an effective attraction between nonpolar beads. While the use of Lennard-Jones interactions allows the formation of lipid bilayers under proper conditions, the stabilized system consists of a solid (gel) phase, rather than a (biologically relevant) fluid bilayer. In the model considered here, lipid tails interact via potentials that have a longer attractive range. Tuning the range of this interaction as well as the temperature allows for the self-assembly of a fluid bilayer. Additionally, the model is capable of reproducing important energetic aspects of continuum theory, such as bending and stretching moduli. In this tutorial, two of these large-scale parameters will be calculated by simulating simple systems.

## 2.1 Outline

The tutorial presented here will first introduce the main property of the force-field: the ability to self-assemble lipids into a stable bilayer. Then the line tension and the bending modulus—two important large-scale parameters of the membrane—will be extracted by simulating a semi-periodic bilayer and an edge-less cylindrical vesicle, respectively. Finally, these parameters will be applied in the context of spontaneous vesicle formation from curved bilayers, which is the essential part of a method to measure the Gaussian bending modulus of the bilayer.

# 3 Getting started

This tutorial assumes that the reader is already familiar with the basics of molecular dynamics (MD) simulations. We will be using the ESPResSo [3] [4] MD package for our simulations, where the membrane package `mbtools` has already been implemented. In addition, VMD[5] will be used for visualisation.

There are two ways of using ESPResSo : interactive and noninteractive. The interactive mode works as a tcl shell, which can be initiated by simply calling ESPResSo from command line without any arguments:

```
Espresso
```

Under this mode, one can check the version and features of ESPResSo on the current machine by command

```
code_info
```

Ideally, all the necessary features for this tutorial have been activated on the machines provided by the workshop. For those who also want to use ESPResSo later, please see Section A for more details on ESPResSo configuration.

For the purpose of this tutorial, the noninteractive mode of ESPResSo will be used. Making `Espresso` the shorthand notation of the ESPResSo binary, the basic syntax is

```
Espresso main.tcl parameter.tcl
```

where `main.tcl` is the main script to feed to ESPResSo. One can find it in the `scripts` subdirectory of the tutorial package. Throughout the tutorial, there will be NO need to change this main script.

`parameter.tcl` contains the specific parameters for each simulation and will be sourced by `main.tcl`. If MPI is correctly installed, a parallel version of ESPResSo can be used as

```
mpirun -np N Espresso main.tcl -n N parameter.tcl
```

where `N` is the number of CPUs one would like to use.

After a simulation is finished, one can `cd` into the result directory and visualize the result with VMD. Make sure a plugin `vmd_plg.tcl` can be found by VMD (See in Section A.3).

Through out the tutorial, participants are highly encouraged to write their own scripts based on the example provided in Section 4. More detailed sample codes can also be found in the tutorial materials for reference.

[3] Hans-Jörg Limbach, Axel Arnold, Bernward A. Mann and Christian Holm. "ESPResSo - An Extensible Simulation Package for Research on Soft Matter Systems". *Comput. Phys. Commun.* 174(9) (704-727), 2006.

[4] `http://espressomd.org`

[5] `http://www.ks.uiuc.edu/Research/vmd/`

# 4 Self assembly

The first exercise will consist of simulating 320 randomly-placed lipids and watching them self-assemble into a lipid bilayer. The following parameter script describes all the necessary commands to setup such a system:

```
############################
set ident "self_assembly320"; # job name
set tabledir "./forcetables/"; # forcetable directory (unused)
set outputdir "./$ident/"; # folder name
set topofile "$ident.top"; # topology name

set use_vmd "offline"; # interactive VMD?

set moltypes [list { 0 lipid { 0 1 1 } { 0 1 } } ]; # molecule topology

set geometry { geometry random }; # define geometry

set n_molslist { n_molslist {  { 0 320 } } }; # number of molecules

# Add the bilayer to the total system
lappend system_specs [list $geometry $n_molslist]

set setbox_l  { 14. 14. 14. }; # system size

# ------ Warmup parameters ----------------------#
set warm_time_step 0.002
set free_warmsteps 0
set free_warmtimes 1

# ------ Integration parameters -----------------#
set main_time_step 0.01         ;# integration time step
set verlet_skin 0.4             ;# verlet skin
set systemtemp 1.1              ;# temperature
set thermo "DPD"                ;# DPD thermostat
set dpd_gamma 1.0               ;# DPD thermostat parameter "gamma"
set dpd_r_cut [expr 1.12 + 1.8] ;# D_Rc should be > Rc + wc

set int_steps   200             ;# number of integration steps per cycle
set int_n_times 1000            ;# number of integration cycles
set write_frequency 1           ;# frequency of config files
set analysis_write_frequency 1  ;# frequency of analysis output

# Bonded and bending Potentials
lappend bonded_parms [list 0 FENE 30 1.5 ]
lappend bonded_parms [list 1 harmonic 10.0 4.0 ]

# Non Bonded Potentials
set lj_eps 1.0; set lj_cutoff 2.5; set ljshift 0.0;
set ljoffset 0.0; set lj_sigmah 0.95; set lj_sigma 1.0
```

```
# Define the interaction matrix
lappend nb_interactions [list 0 0 lennard-jones $lj_eps $lj_sigmah \
  [expr 1.1225*$lj_sigmah] [expr 0.25*$lj_eps] $ljoffset ]
lappend nb_interactions [list 0 1 lennard-jones $lj_eps $lj_sigmah \
  [expr 1.1225*$lj_sigmah] [expr 0.25*$lj_eps] $ljoffset ]
lappend nb_interactions [list 1 1 lj-cos2 $lj_eps $lj_sigma $ljoffset 1.6 ]

# Analysis Parameters
lappend analysis_flags energy ;# calculate energy
############################
```

Any such parameter script must be fed into the `main.tcl` script of `mbtools`, a sample file can be found in `scripts/`. This `mbtools` script is itself fed into Espresso, such that one can run the simulation by simply invoking

```
Espresso main.tcl self_assembly.tcl
```

assuming all files are accessible from the current directory (and the parameter file was called `self_assembly.tcl`).

While running the simulation, trajectory files are regularly being output, and you can start visualizing what has already been produced. To do so, go to the newly created directory `self_assembly320` and type

```
vmd -e vmd_animation.script
```

Initial and final conformations of a system consisting of 320 randomly-placed lipids is shown on Fig. 1. The box dimensions were set to $14\sigma \times 14\sigma \times 14\sigma$.

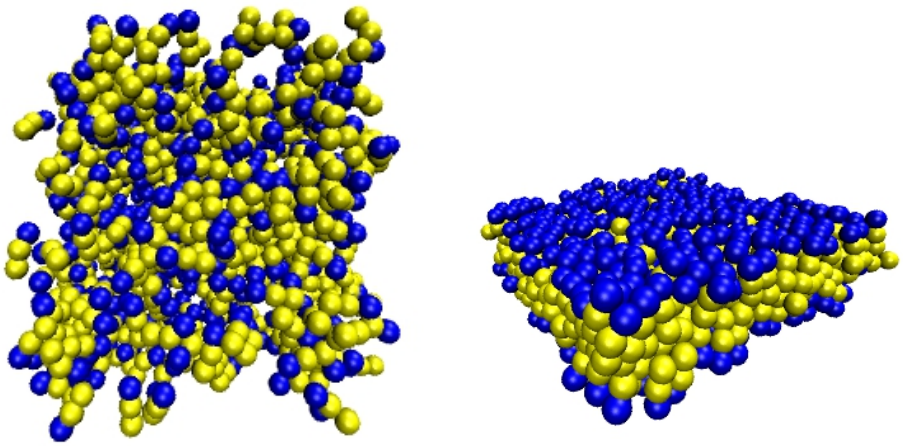


Figure 1: Initial (left) and final (right) conformations of a 320-randomly-placed-lipid simulation.

The simulation time required to self-assemble the system may vary depending on the initial conformation and the overall kinetics. For example, simulating

320 lipids for $2000\,\tau$ as in our sample code will take around 15 minutes with a single CPU. If at the end of the simulation, the lipids are stuck in some intermediate global structure due to the periodic boundary conditions, check to see if the local structure resembles a bilayer. To increase the probability that a bilayer forms during the simulation, double the length of one axis of the box while leaving the other two axis unchanged.

If interested, you may check how the simulation time depends on system size by changing the number of lipids. Note that changing the number of lipids will require you also to change the box size, such that the new bilayer is still neither too compressed nor too stretched across the periodic system. This can be tuned by keeping the *area per lipid* constant (i.e. box length squared divided by number of lipids *in one leaflet*, assuming all box sides are equally long). A DPD thermostat was used here to speed up the dynamics.

# 5 Line tension measurement

In this section, we will extract the line tension of a semi-periodic flat bilayer at constant temperature and area. The simulation will consist of a bilayer spanning half of the $x - y$ plane, such that the bilayer is periodic only along the $y$ axis (see Fig. 2). The open edges of the bilayer will give rise to a pulling force that is precisely imbalanced by twice the line tension of the system:

$$\gamma = \frac{1}{2}(\sigma_{xx} - \sigma_{yy})L_x L_z, \tag{1}$$

where $\sigma_{xx}$ and $\sigma_{yy}$ are the diagonal components of the stress tensor along $x$ and $y$, respectively; $L_x$ and $L_y$ are the box dimensions along the axes, and the factor of $1/2$ accounts for the two open-edges of the system.

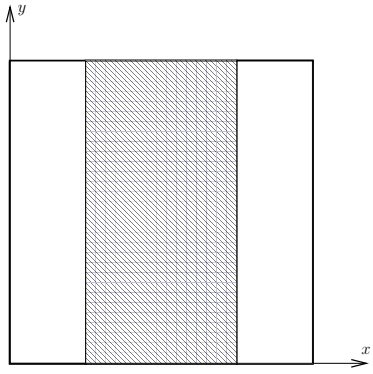



Figure 2: Top view of semi-periodic bilayer, lipids are only placed whithin the gray region. The open edges along the $y$ axis give rise to a pulling force.

Adapt the script given in Section 4 to create the system represented in Fig. 2 using:

- 320 lipids, but a large box size of $20\sigma \times 20\sigma \times 20$ (Remember that we need open edges, so a larger box that the lipids cannot cover completely.)
- the correct initial geometry can be set using the following command

  ```
  set geometry { geometry "flat -fixz -half" }
  ```

- additional warmup to avoid initial steric clashes by adding

  ```
  set warmsteps 100
  set warmtimes 20
  ```

- a reduced main time step to accurately measure components of the stress tensor (use $\delta t = 0.002$ in time units of the system)
- this time we'll use a Langevin thermostat. Remove the command `thermo "DPD"` and the variables `dpd_gamma` and `dpd_r_cut` and replace them with

  ```
  set langevin_gamma 1.0
  set systemtemp 1.1
  ```

- any observable that is analyzed during the simulation will be measured every `int_steps` * `main_time_step`. Given the aforementioned value of the time step, set `int_steps` such that a measurement is output every $2\,\tau$, where $\tau$ is the unit of time of the system (`main_time_step` is expressed in units of $\tau$).
- add `stress_tensor` to the list of `analysis_flags`.

Before measuring the line tension $\gamma$, make sure the system that you are simulating corresponds to what you expect by inspecting it with VMD.

Equation 1 requires the measurement of the diagonal components of the stress tensor along $x$ and $y$. The absence of off-diagonal components is due to the symmetry of the system. There will be no stress along the $x$ and $z$ directions because in $x$ direction there are open edges and in $z$ direction the bilayer is not periodic. Eq. 1 involves the difference between $\sigma_{yy}$ and $\sigma_{xx}$ in order to subtract the stress due to the solvent, absent in this model.

You will need to take measurements in order to gather statistics of the stress tensor components. These measurements should be taken only after the system has had time to equilibrate. Estimating the equilibration time is always difficult and somewhat arbitray. One can plot the time dependence of the energy `time_vs_energy_tmp` [6] and start measuring observables sufficiently long after

[6] See Section A.4 for some basic uses of Gnuplot, such as plotting a time sequence and fitting a function, in case your favorite maths software is not installed here in the cluster.

the energy has relaxed. Bear in mind this does not assure equilibrium in any way (in fact it can only tell you the opposite: a signal that is still strongly varying is out-of-equilibrium). The nine components of the stress tensor are stored in `time_vs_stress_tensor_tmp`. The first column is time, and the other ones are the $\sigma_{ij}$ components as described at the top of the file. For example, the second column is the diagonal term $\sigma_{xx}$. and the sixth column is the diagonal term $\sigma_{yy}$. Plot the time dependence of $\sigma_{xx}$, $\sigma_{yy}$, and $\sigma_{zz}$. Calculate the average and the error of the mean for the *equilibrated* part of the simulation only. The error of the mean is $\sigma/\sqrt{n}$, where $\sigma$ is the standard deviation of the distribution and $n$ is the sample size.[7] First make sure that $\sigma_{xx}$ and $\sigma_{zz}$ average to 0 (up to error bars), and then calculate $\sigma_{yy}$.

You can now estimate the line tension $\gamma$ of the bilayer by the simple formula $\gamma = -\sigma_{yy} L_x L_z/2$, where $L_x$ and $L_z$ are the box dimensions (look them up in the parameter script you used). You can also provide error bars to your calculation of $\gamma$ by using the error made on $\sigma_{yy}$. The units of $\gamma$ are $\epsilon/\sigma$, where $\epsilon \simeq k_{\mathrm{B}} T_{\mathrm{room}}/1.1$ and $\sigma \simeq 1\,\mathrm{nm}$.

# 6 Extracting the bending modulus from actively bent membranes

This simulation shows a very simple and efficient way of calculating the bending modulus $\kappa$ of a bilayer from an actively bent membrane system. It does not rely on analyzing the power spectrum of the fluctuations of a flat bilayer (as it is usually done) but rather the force exerted by a cylindrical membrane along its periodic axis. More details can be found elsewhere.[8] A representative conformation of the system is shown in Fig. 3.

To simulate such a system, adapt the script given in Sec. 4 in the following way:

- $1,000$ lipids
- set the box size to $\{40\ 40\ 20\}$
- the correct initial geometry can be set using the following command

  ```
  set geometry { geometry "cylinder -shuffle" }
  ```

- reduce the main time step to $\delta t = 0.002\,\tau$
- set the thermostat to Langevin
- analyze the system every $2\,\tau$, and take at least 100 measurements

[7] In order to calculate accurate errors, one should really consider correlation times. This is a measure of the characteristic time over which data points are correlated. One can, for instance, use a blocking error analysis routine.

[8] V. A. Harmandaris and M. Deserno, J. Chem. Phys. **125**, 204905 (2006)

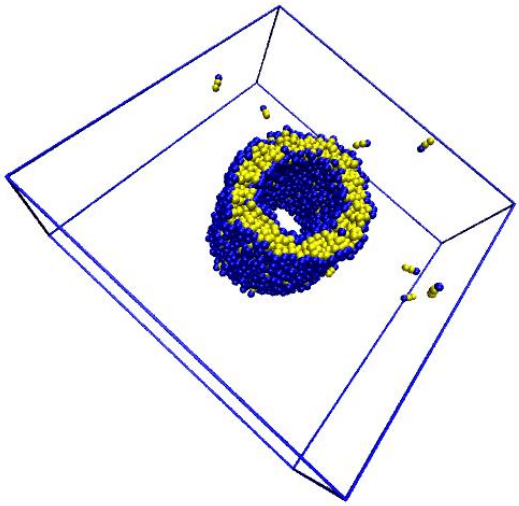


Figure 3: Equilibrated conformation of the cylinder system described in Sec. 6.

- add the following analysis scripts:
  - `stress_tensor`
  - `cylinder_radius`

The reference provided above showed that the bending modulus can be extracted by measuring the force along the axis of the cylinder $F_z$, as well as the cylinder's radius $R$: $\kappa = F_z R/(2\pi)$. Similarly to Sec. 5, the force along $z$ can be obtained from measuring the $\sigma_{zz}$ component of the stress tensor: $F_z = \sigma_{zz} L_x L_y$. As in the previous exercise, you will need to first estimate the time to reach equilibrium, and then calculate the average value and error of the mean of $\sigma_{zz}$ in order to extract the force $F_z$ (with error bars). The radius of the cylinder can be extracted in a similar fashion from `time_vs_cylinder_radius_tmp` (use the second column which displays the *average* radius between the inner and outer leaflets of the bilayer, the third and fourth column show the inner and outer contributions). Calculate the bending modulus $\kappa = F_z R/2\pi$ including error bars. $\kappa$ has units of $\epsilon$.

# 7 Spontaneous vesicle formation

## 7.1 Theory

In Helfrich theory, [9] the free energy of a patch of lipid bilayer with an open edge is

$$E[\mathcal{P}] = \int_{\mathcal{P}} \mathrm{d}A \, \left\{ \frac{1}{2}\kappa(K - K_0)^2 + \overline{\kappa} K_{\mathrm{G}} \right\} + \oint_{\partial\mathcal{P}} \mathrm{d}s \, \gamma \, . \tag{2}$$

Here, $K = c_1 + c_2$ is the total curvature (the sum of the two local principal curvatures $c_1$ and $c_2$) and $K_{\mathrm{G}} = c_1 c_2$ is the Gaussian curvature. The two previous exercices have allowed us to extract estimates for the line tension $\gamma$ (Sec. 5) and the bending modulus $\kappa$ (Sec. 6). The only material parameter left is the Gaussian bending modulus $\overline{\kappa}$, with the assumption that the bilayer is symmetric, and thus has no spontaneous curvature $K_0$.

Due to the Gauss-Bonnet Theorem in differential geometry, the surface integral over the Gaussian curvature remains constant if there is no change in boundary or topology. Thus it's difficult to measure $\overline{\kappa}$ in experiments and in simulations because of the lack of control over boundary and topology.

This section provides the essential part of a new method to measure this parameter $\overline{\kappa}$ by studying the spontaneous vesicle formation process. During such process, the line tension will try to shrink the boundary in order to reduce the energy, while on the other hand it will also curve the patch and induce higher bending energy. Thus, it's a competition between line tension and bending, which prefers a vesicle and a flat bilayer as the final state, respectively.

By the assupmtion that during the folding process, the shape of a circular bilayer stays as a part of a sphere (which means one curvature is enough to describe the evolution of the system), one can write down the analytical expression of free energy as a function of curvature $c$. Given the $\gamma$ and $\kappa$ measured in the previous sections, together with a reasonable estimation of $\overline{\kappa} \simeq -\kappa$, one can show that for a patch of around 1000 lipids, the vesicle state is globally stable, yet there is a barrier between the flat and the vesicle state at the critical curvature

$$c^* = \frac{1}{R} = \sqrt{\frac{4\pi}{A} - \frac{\gamma^2}{4(2\kappa + \overline{\kappa})^2}}. \tag{3}$$

Thus, if $c^*$ can be obtained, then $\overline{\kappa}$ becomes available.

## 7.2 Method

A good way to get $c^*$ is to simulate a piece of bilayer starting from different initial curvatures and measure the probability of folding into vesicles, $P_{\mathrm{fold}}$. Near $c^*$ is the top of the barrier where $P_{\mathrm{fold}}$ is around $1/2$ and has the steepest slope. [10] $P_{\mathrm{fold}}$ provides a much more clear signal of the trasition than some complicated free energy calculation when $c^*$ is the target.

[9] W. Helfrich, *Zeitschrift für Naturforschung C* **28**, 693 (1973).

[10] This folding probability follows the theory of *splitting probability*. A careful calculation shows at $c^*$, the $P_{\mathrm{fold}}$ is not exactly 0.5, but not far from it.

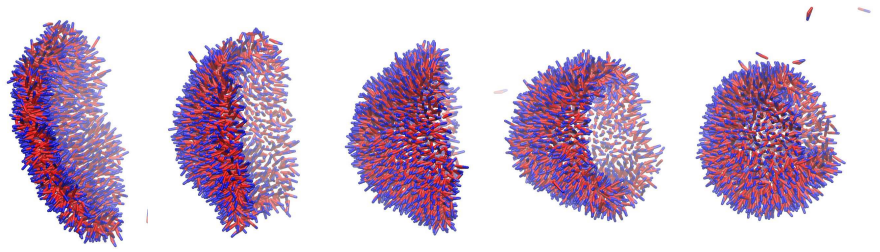


Figure 4: Time evolution snapshots of the curved pancake system simulation described in Sec. 7.

Because of the time limitation here, we cannot afford a seriously search for $c^*$. Instead, what we will do is a rough test of a broadly used estimation on $\overline{\kappa} \simeq -\kappa$. Let's focus on a spherical cap system of **1000** lipids.

First estimate the $c^*$ according to Eq. 3, with $\overline{\kappa} \simeq -\kappa$. Here the area of the patch can be approximated as $A = (1.20\sigma^2/\text{lipid}) \times (\#\text{ of lipids}/2)$. Second, set up a spherical cap of a curvature $c^*$ and check if it closes up into a vesicle or expands into a flat patch. In the end, results from all participants will be collected together and thus the folding probability $P_{\text{fold}}$ can be calculated. **NOTE:** for a more meaningful result on $P_{\text{fold}}$, please check the results in Section 8 *before* you start the folding simulation.

## 7.3 Simulation

Here are the changes to the script in Sec. 4 you need to apply:

- change the number of lipids to 1000
- set a large enough box size to avoid any periodicity in the membrane (e.g. {100 100 100})
- set the initial radius of curvature to be $r = 1/c^*$ by the following script:

  ```
  set geometry { geometry "sphere_cap -c {$center} -r $r" }
  ```

  `$center` is the position of the center of the sphere where the cap is sitting on.
- you may keep a time step of $\delta t = 0.01\,\tau$ for this simulation
- keep a DPD thermostat

Does the initial configuration relax to a vesicle, or go back to a flat bilayer?

# 8 Solutions

Note that the simulations carried out here in the tutorial have been set to extremely small simulation times. This strongly affects the accuracy of certain observables, especially the stress tensor statistics. The results are subject to non-negligible errors. The reader is advised to repeat the present simulations with longer simulation times, thus allowing better statistics.

Section 8.1 and 8.2 will give some sample simulations results as demonstration. More accurate measurements will be applied to Section 8.3 in order to get a better estimation on the critical curvature $c^*$.

## 8.1 Line tension measurement

Parameters used in a sample simulation:

- $2,000$ lipids
- total time $t = 2,100\,\tau$
- box size $L = 50\,\sigma$

Diagonal stress tensor components:

- $\sigma_{xx} = 0.0008 \pm 0.0021\,\epsilon/\sigma^3 \to 0$
- $\sigma_{yy} = -0.011 \pm 0.0018\,\epsilon/\sigma^3$
- $\sigma_{zz} = 0.0023 \pm 0.003\,\epsilon/\sigma^3 \to 0$

This allows us to obtain the line tension:

$$\gamma = \frac{1}{2}(\sigma_{xx} - \sigma_{yy})L_x L_z = 3.4 \pm 0.3\epsilon/\sigma \tag{4}$$

## 8.2 Bending modulus from cylindrical membrane

Parameters used in a sample simulation:

- $1,000$ lipids
- total time $t = 200\,\tau$
- box size $\{60\ 60\ 20\}\,\sigma$

Measurements:

- Stress tensor component: $\sigma_{zz} = -0.0042 \pm 0.0008\,\epsilon/\sigma^3$.
- radius of the cylinder: $R = 5.171 \pm 0.004\,\sigma$

We can now extract the bending modulus:

$$\kappa = \frac{F_z R}{2\pi} = 12 \pm 3\,\epsilon \tag{5}$$

### 8.3 Vesicle formation

Following the half-bilayer methods described in Section 5, a set of more serious simulations present a more accurate $\gamma = 3.35 \pm 0.07\,\epsilon/\sigma$. Also, a set of cylinder simulations around a meaningful radius of curvature give $\kappa = 13.7 \pm 0.3\,\epsilon$.

Assuming $\overline{\kappa} \simeq -\kappa$, the critical curvature is estimated as

$$c^* = \sqrt{\frac{4\pi}{A} - \frac{\gamma^2}{4(2\kappa + \overline{\kappa})^2}} = 0.077\,/\sigma \tag{6}$$

where the area $A = 0.5 \times 1.20 \times 1000\sigma^2$. Idealy, when setting up the initially curved bilayer, the radius should be $r^* = 1/c^* = 13.0\sigma$. However, as it turns out that $\overline{\kappa}$ is slightly above $-\kappa$ ($\sim -0.9\,\kappa$), $r^*$ is actually close to

$$r^* = 1/c^* = 11.5\sigma. \tag{7}$$

Thus, for this tutorial, two $r^*$ values are recommended: $10.5\,\sigma$ and $12.5\,\sigma$. This will give a bracket on the actual value of $r^*$, and thus a rough estimation of $\overline{\kappa}$.

# A Software Setup

## A.1 ESPResSo

This section will provide basic instructions for those who want to experience ESPResSo and the `mbtools` on other computers. You can find the latest version of the ESPResSo package at `http://espressomd.org/wordpress/download`. Detailed installation instructions is available in the package.

One of the useful ideas of ESPResSo is that one only needs to compile those *features* that are needed. For instance, if someone is trying to study a system without any explicit charges, all the features related to electrostatics can be left aside.

Thus, for the purpose of this tutorial, the first step is to make sure the following ESPResSo features are activated in the `myconfig.h` file and compiled:

```
#define EXTERNAL_FORCES
#define DPD
#define LENNARD_JONES
#define TABULATED
#define LJCOS
#define LJCOS2
#define LENNARD_JONES_GENERIC
```

## A.2 mbtools

The `mbtools` package is contained within ESPResSo in the subdirectory

`$ESPRESSO_DIR/packages/mbtools`

it includes all the necessary Tcl routines required to run the package (e.g., create initial configurations, setup the particles and interactions, run the analysis).

`mbtools` needs several specific Tcl libraries that are contained in the package `tcllib`. To check whether this library is installed and recognized by ESPResSo , start ESPResSo and type

```
package require cmdline
```

If a number (e.g., 1.3.x) appears then `tcllib` is properly installed and linked to the main Tcl library. Otherwise, one can download and install the library from

`http://www.tcl.tk/software/tcllib/`

Recompile ESPResSo after installing the package. In case ESPResSo does not automatically find the library, it is possible to manually source it when ESPResSo starts. Assuming `tcllib` was installed in directory `/usr/local/tcllib`, create (or add to) `~/.espressorc` file (in your home directory):

```
lappend auto_path "/usr/local/tcllib/lib"
```

## A.3 VMD

The final step is to link a VMD script available in ESPResSo that allows to load many trajectory files at once. In your home directory, edit (or create) the configuration file `~/.vmdrc` and add

```
source $ESPRESSO_DIR/tools/vmd_plg.tcl
```

assuming ESPResSo was installed in `$ESPRESSO_DIR/`.

## A.4 Gnuplot

Gnuplot[11] is a very powerful tool to make scientific plots and do some data analysis. Here we only give a couple of examples that are useful for the purpose of this tutorial.

- Plot column $i$ as a function of column $j$ over the range of $[x_a, x_b]$:

  ```
  plot [xa,xb] 'file_name' using j:i
  ```

- Define and fit function F(x) to data in column $i$ and $j$:

  ```
  #Just an example of a straight line
  F(x) = a * x + b
  fit [xa,xb] F(x) 'file_name' using j:i via a,b
  ```

  After this, you will get the values of the fitting parameters and their *approximated* errors. You can also check the fitting by

  ```
  plot [xa,xb] 'file_name' using j:i, F(x)
  ```

[11] `http://www.gnuplot.info/`