Mesoscopic membrane simulations with mbtools

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

September 30, 2015

1. Introduction

This tutorial introduces the `mbtools` package, which provides an ESPRESSO implementation of the mesoscopic membrane model of Cooke *et al.*[1] This coarse grained (CG) model has a fairly low resolution (three beads per lipid molecule) and, moreover, eliminates the embedding solvent entirely. While not being a good choice for questions that depend critically on chemically specific aspects of lipid molecules, it nevertheless self-assembles into fluid lipid bilayers that show the correct curvature elastic behavior on large[2] scales with physically meaningful and tunable elastic parameters. The present tutorial is not the right place to discuss much of the physical basis of both the model and its emerging physical behavior, and therefore the reader is encouraged to read up on this in a recent review article.[3] Instead, we will illustrate how the `mbtools` package can be used to efficiently work with this model in the context of ESPRESSO, and along the line learn a bit about the model's capabilities. Specific aspects of lipid bilayer physics we will look at are bilayer self-assembly, measuring the membrane edge tension, and determining both the mean and the Gaussian curvature modulus. References to pertinent literature will be given alongside.

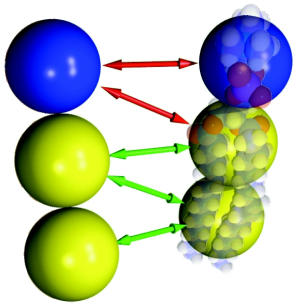

Figure 1: The lipid model.

General disclaimer: In what follows, you will determine a number of interesting physical observables of lipid bilayers from simulations. You will produce simulation trajectories and analyze their results in a variety of ways. While the theoretical considerations that will be employed to arrive at these observables are sound, the data analysis you will be asked to conduct is somewhat simplified. Specifically, we will not bother with a proper statistical error analysis—primarily because this would take too much time. You should know, though, that this is neither irrelevant nor indeed trivial. Specifically, one typically encounters two issues that complicate such an analysis and make it more technically sophisticated than a plain vanilla error propagation of independent samples. First, the relation between the measured observable and the underlying raw data might be so complicated that a simple error propagation will not work; either because a linear analysis is plainly insufficient, or because the analysis involves intricate computational manipulations which you cannot easily capture in a closed analytic expression. Second, the samples you collect and average over to reduce the error of the mean result from a simulation trajectory, and its individual frames are generally not statistically independent—correlation times depend on the observable you are looking at and may be negligibly small or distressingly large. Both these issues must be addressed if you wish to repeat any of this for work that you intend to actually publish, but in the following we will deliberately not bother with these issues, because statistical data analysis is not our primary topic.

[1] IR Cooke, K Kremer and M Deserno, *Tunable generic model for fluid bilayer membranes*, Phys. Rev. E. **72**, 011506 (2005).
[2] We will soon see that "large" really doesn't have to be much more than a few times the bilayer thickness.
[3] M Deserno, *Mesoscopic Membrane Physics: Concepts, Simulations, and Selected Applications*, Macromol. Rapid Commun. **30**, 752–771 (2009).

2. Background

Lipid membranes form the outer boundaries of all cells, and further compartmentalize many different organelles in all eukaryotic cells (such as the nucleus, mitochondria, or the endoplasmic reticulum). They spontaneously assemble from lipid molecules into two apposing monomolecular layers, thus forming a "lipid bilayer". This so-called self-assembly is driven by the lipids "desire" to shield their hydrophobic (water-insoluble) hydrocarbon tails from the surrounding aqueous solution. The types of aggregates that such amphiphilic molecules can form depends on a number of parameters, most notably the amphiphile's aspect ratio and shape,[4] but for the roughly cylindrically shaped lipids the two-dimensional bilayer morphology is the stable one, and in the following we will only be concerned with this case.

At the mesoscopic scale ($\sim 10\,\mathrm{nm}-10\,\mu\mathrm{m}$) the energetics of membranes can be modeled very well using a curvature-elastic continuum theory:[5]

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

where the integral extends over the entire membrane surface $\mathcal{S}$, $K = c_1 + c_2$ 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, and K_0 is the spontaneous bilayer curvature. 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 seen in the tutorial, can nevertheless self-assemble into fluid two-dimensional curvature-elastic bilayer membranes that exhibit physically meaningful (and within a certain range tunable) large-scale parameters.

A single lipid molecule is represented by only three beads, which gives us a realistic length-to-width aspect ratio of about 3:1 (see Fig. 1). To capture a lipid's amphiphilic nature, we need to define two different types of beads: hydrophobic and hydrophilic. The model does not contain any explicit solvent. While this increases the computational efficiency quite substantially, it requires some care when parametrizing the interactions between the lipid beads. Hydrophobicity in an implicit solvent model is usually represented by an effective attraction between nonpolar beads, but it turns out that the functional form of that attraction matters for once. While the use of a Lennard-Jones potential (between the hydrophobic lipid tail beads) allows the formation of lipid bilayer sheets under suitable conditions, these almost invariably seem to end up as gel-phase membranes, rather than the more interesting (and biologically relevant) fluid L_α phase. In the model considered here, this tendency to gel is avoided by giving the attractive interaction a longer range—an effect that does not appear to crucially depend on precisely how the range is extended.[6] Tuning (i) the range of this interaction and (ii) its depth (or, equivalently, the temperature) results in a two-dimensional phase diagram that contains a fairly broad phase in which fluid bilayers are thermodynamically stable. Their interesting large-scale physical parameters (such as for instance their bending rigidity) can now be determined by performing a suitable computational analysis. We will now look into this in some detail.

2.1. Outline

The following tutorial will first check the most basic property of the force-field, namely, that the lipids indeed self-assemble into a stable bilayer. Then the excess free energy of an open membrane edge (the so-called edge tension) will be determined by simulating a semi-periodic bilayer. Next, two methods will be discussed to extract the (mean) bending modulus of a membrane from an active shape deformation: either a cylindrical vesicle or a buckled sheet. Finally, we will look at the process of spontaneous vesicle formation from curved bilayers, which—in combination with the so far determined membrane parameters—is a convenient means to measure the Gaussian curvature modulus of a membrane.

[4] JN Israelachvili, DJ Mitchell and BW Ninham, *Theory of self-assembly of hydrocarbon amphiphiles into micelles and bilayers*, J. Chem. Soc., Faraday Trans. 2 **72**, 1525–1568 (1976).

[5] W Helfrich, *Elastic properties of lipid bilayers: theory and possible experiments*, Z. Naturforsch. C,**28**, 693 (1973).

[6] IR Cooke and M Deserno, *Solvent free model for self-assembling fluid bilayer membranes: Stabilization of the fluid phase based on broad attractive tail potentials*, J. Chem. Phys. **123**, 224710 (2005).

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[7,8] MD package for our simulations, where the membrane package `mbtools` has already been implemented. In addition, VMD[9] 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 the command line without any arguments:

```
Espresso
```

In 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 already been activated. If that is not the case, 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 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.

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:

```
1  ############################
2  set ident "self_assembly320"; # job name
3  set tabledir "./forcetables/"; # forcetable directory (unused)
4  set outputdir "./$ident/"; # folder name
5  set topofile "$ident.top"; # topology name
6
7  set use_vmd "offline"; # interactive VMD?
8
9  set moltypes [list { 0 lipid { 0 1 1 } { 0 1 } } ]; # molecule topology
10
11 set geometry { geometry random }; # define geometry
12
13 set n_molslist { n_molslist {  { 0 320 } } }; # number of molecules
14
```

[7] HJ Limbach, A Arnold, BA Mann and C Holm, *ESPResSo—An Extensible Simulation Package for Research on Soft Matter Systems*, Comput. Phys. Commun. **174**(9), 704–727 (2006).

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

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

```
15 # Add the bilayer to the total system
16 lappend system_specs [list $geometry $n_molslist]
17
18 set setbox_l  { 14. 14. 14. }; # system size
19
20 # ------ Warmup parameters ----------------------#
21 set warm_time_step 0.002
22 set free_warmsteps 0
23 set free_warmtimes 1
24
25 # ------ Integration parameters -----------------#
26 set main_time_step 0.01         ;# integration time step
27 set verlet_skin 0.4             ;# verlet skin
28 set systemtemp 1.1              ;# temperature
29 set thermo "DPD"                ;# DPD thermostat
30 set dpd_gamma 1.0               ;# DPD thermostat parameter "gamma"
31 set dpd_r_cut [expr 1.12 + 1.8] ;# D_Rc should be > Rc + wc
32
33 set int_steps   200             ;# number of integration steps per cycle
34 set int_n_times 1000            ;# number of integration cycles
35 set write_frequency 1           ;# frequency of config files
36 set analysis_write_frequency 1  ;# frequency of analysis output
37
38 # Bonded and bending Potentials
39 lappend bonded_parms [list 0 FENE 30 1.5 ]
40 lappend bonded_parms [list 1 harmonic 10.0 4.0 ]
41
42 # Non Bonded Potentials
43 set lj_eps 1.0; set lj_cutoff 2.5; set ljshift 0.0;
44 set ljoffset 0.0; set lj_sigmah 0.95; set lj_sigma 1.0
45 # Define the interaction matrix
46 lappend nb_interactions [list 0 0 lennard-jones $lj_eps $lj_sigmah \
47   [expr 1.1225*$lj_sigmah] [expr 0.25*$lj_eps] $ljoffset ]
48 lappend nb_interactions [list 0 1 lennard-jones $lj_eps $lj_sigmah \
49   [expr 1.1225*$lj_sigmah] [expr 0.25*$lj_eps] $ljoffset ]
50 lappend nb_interactions [list 1 1 lj-cos2 $lj_eps $lj_sigma $ljoffset 1.6 ]
51
52 # Analysis Parameters
53 lappend analysis_flags energy ;# calculate energy
54 ############################
```

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 named the same way.

While running the simulation, trajectory files are regularly being written as 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 are shown on Fig. 2. The box dimensions were set to $14\sigma \times 14\sigma \times 14\sigma$.

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.

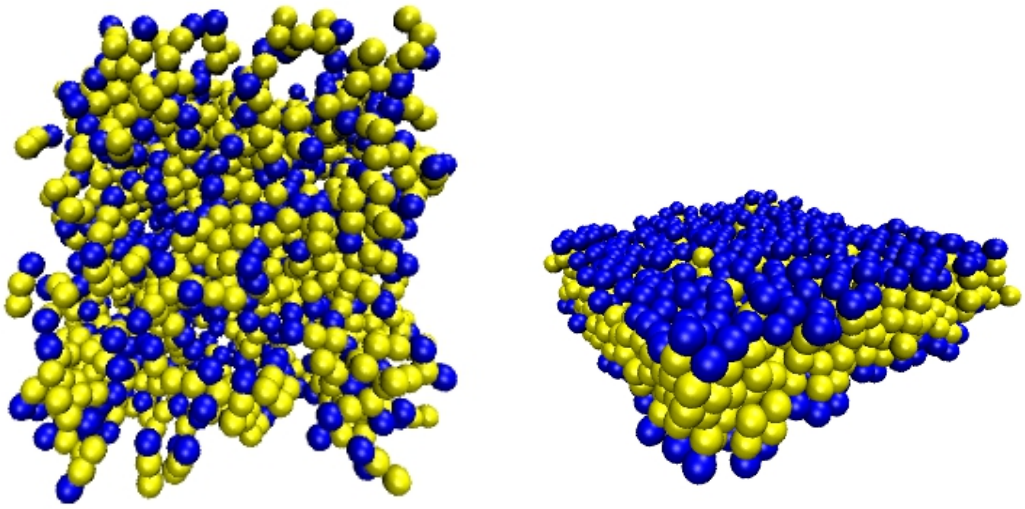

Figure 2: Initial (left) and final (right) conformations of a 320-randomly-placed-lipid simulation.

It might happen that at the end of the simulation you do not find as nice a bilayer as in Fig. 2. Rather, the lipids might be stuck in some intermediate global structure, or there seems to be a bilayer that has a hole which simply won't go away. If this happens (or even if it doesn't!), convince yourself that this has little to do with the actual lipid model and everything with the stochastic nature of the assembly process, combined with the fact that periodic boundary conditions permit more than the "trivial" types of periodically connected planes. If this outcome distresses you, you can increase the probability that a complete bilayer forms during the simulation by removing the possibility of "unusually connected" periodic boundary planes, and one simple way of doing this is to increase the length of one of the three box dimensions by, say, a factor of 2.

If you are interested, you may also check how the simulation time depends on the system size by changing the number of lipids. Note that changing the number of lipids will require you to also change the box size, such that the new bilayer is still neither too compressed nor too stretched across the periodic boundary conditions. Evidently, we want to keep the *area per lipid* the same (*i. e.*, the box area across which you wish to span the bilayer, divided by number of lipids *in one leaflet*), and this should tell you how to adjust the box dimensions.

Note that a DPD thermostat was used here to speed up the dynamics.

5. Line tension measurement

5.1. Theory

In this section, we will determine the edge tension of a semi-periodic flat bilayer at constant temperature and area. The simulation will consist of a bilayer spanning *half* of the xy plane, such that the bilayer is periodic only along the y-axis (see Fig. 3). The open edges of the bilayer will give rise to a pulling force that is precisely twice the edge tension of the system:

$$\gamma = -\frac{1}{2}\Pi_{yy} L_x L_z, \tag{2}$$

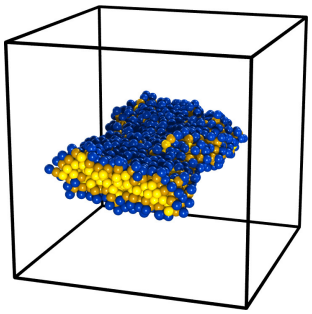

Figure 3: A half-bilayer for the determination of the edge tension.

where Π_{yy} is the diagonal components of the pressure tensor along y-direction; L_x and L_z are the box dimensions along the x- and z-axes, and the factor of $1/2$ accounts for the fact that *two* open edges pull across the box. The minus sign derives from the fact that the force will be a *pulling* force, for which the pressure tensor is negative, but the edge tension is usually given as a positive number.

You might wonder why there is no contribution to the overall pulling force from the membrane (surface)

tension. The reason is that this tension is automatically zero: if it were not, the two edges of the membrane would adjust their separation until it is.

5.2. Simulation

Adapt the script given in Section 4 to create the system represented in Fig. 3 using:

- 320 lipids, but a large box size of $20\sigma \times 20\sigma \times 20$ (Remember that we need open edges, and this can be achieved by simply using a box that is larger than what the lipids can 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 τ is the unit of time of the system (`main_time_step` is expressed in units of τ).
- add `stress_tensor` to the list of `analysis_flags`.

Before measuring the edge tension γ, make sure the system that you are simulating corresponds to what you expect by inspecting it with VMD. Specifically, if a half-bilayer formed, what is the direction across which it connects?

Equation (2) requires the measurement of the diagonal components of the stress tensor along the direction across which the bilayer is periodically connected. There should be no stress along the other two directions, because nothing periodically connects across those. Moreover, all off-diagonal components should vanish, too, essentially due to symmetry.

Convince yourself that if the simulation model had explicit solvent, Eqn (2) would need to be generalized. Along the y-direction there is not just a force coming from the two open edges, but also from the bulk contribution of the solvent. This pressure contribution must be subtracted away. This is not too hard, though, because the pressure can be simultaneously measured by looking at any of the other two diagonal components of the pressure tensor: Since the membrane doesn't periodically connect across those, the force across these two directions is solely due to the solvent pressure, and since the latter is isotropic, the solvent pressure along the x- or z-direction can be used as a proxy for the pressure along the y-direction, and we could write $\gamma = -\frac{1}{2}(\Pi_{yy} - \Pi_{xx})L_x L_z$.

You will need to take measurements over many configurations in order to gather statistics of the pressure tensor components. These measurements should be taken only after the system has had time to reach thermal equilibrium. Estimating the equilibration time is always difficult and a topic on which we do not wish to dwell here. One way to proceed, for now, is to plot the time dependence of the energy `time_vs_energy_tmp`[10]

[10] See Sec. 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.

and start measuring observables sufficiently long after the energy has relaxed (*i. e.*, no longer shows an "obvious" *drift*; it will always fluctuate, though). Bear in mind this does not unequivocally assure equilibrium (in fact, it can really only tell you the opposite: a signal that is still strongly drifting is clearly not yet in 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 Π_{ij} components as described at the top of the file. For example, the second and sixth columns are the diagonal terms Π_{xx} and Π_{yy} terms, respectively. Plot the time dependence of Π_{xx}, Π_{yy}, and Π_{zz}. Calculate the average and the error of the mean for the *equilibrated* part of the simulation only. If all samples had been statistically independent, the error of the mean would be $\sigma/\sqrt{n}$, where σ is the standard deviation of the distribution and n is the sample size.[11] First make sure that Π_{xx} and Π_{zz} average to 0 (within error bars), and then calculate Π_{yy}.

You can now estimate the edge tension γ of the bilayer by Eqn. (eq:edge-tension). You can also provide error bars to your calculation of γ by using the error made on Π_{yy} (but these are probably too small, because we have not accounted for the finite correlation time). The units of γ are ϵ/σ, 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

Probably the most common way to determine the bending rigidity of lipid membranes in simulations is to monitor the power spectrum of their thermal shape undulations. Continuum theory claims that for tensionless membranes the mean squared amplitude of a Fourier mode of wave vector $\vec{q}$ is proportional to q^{-4}. These amplitudes are very small, though, they do not probe strong curvatures, and non-curvature corrections tend to plague the analysis of this spectrum at shorter length scales. For these reasons, among others, it would be interesting to instead measure the bending rigidity by actively deforming a membrane and monitor the forces needed to do so.

In the following section, we will discuss two methods for doing this. The first such method creates a deformed membrane in the shape of a cylinder, and a suitable signal turns out to be the tensile force acting along that cylinder's symmetry axis. The second method actively buckles the membrane, and we then measure the force necessary to hold such a compressed buckle. The first method is conceptually easier to understand and to perform, but it does only work easily for highly coarse-grained solvent free membrane models (such as the Cooke model). The second method is more involved, but can be applied to a much wider range of models.

You can choose which method (or both) to attempt during this tutorial. If you want to finish Sec. 7 during this session, then we suggest that you only try the cylinder method.

6.1. Cylinder method

Background The cylinder method is a very simple and efficient way of calculating the bending modulus κ of a bilayer from an actively bent membrane system.[12] Imagine you have a cylindrical membrane of radius R and length L, and thus surface area $A = 2\pi RL$. Let's connect it over the periodic boundaries of our box, such that now every point on that cylinder is symmetrically equivalent to every other point (since we have translational symmetry along the cylinder and rotational symmetry around its axis). Specializing Eqn. (1) to the case of cylinders (and restricting to the case $K_0 = 0$), the total curvature energy is

$$E \;=\; \frac{1}{2}\kappa\left(\frac{1}{R}+0\right)^2 \times 2\pi RL \;=\; \frac{1}{2}\kappa\left(\frac{2\pi L}{A}\right)^2 \times A\,. \qquad (3)$$

This cylinder will be under an axial tension, since increasing its length (at constant area!) will reduce its radius and thus increase the curvature energy.

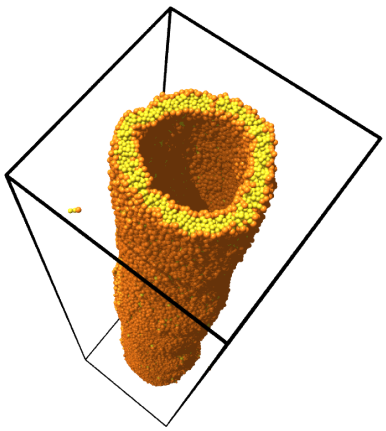

Figure 4: A cylindrical bilayer spanning across the periodic boundary conditions of a simulation box.

[11] 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.

[12] VA Harmandaris and M Deserno, *A novel method for measuring the bending rigidity of model lipid membranes by simulating tethers*, J. Chem. Phys. **125**, 204905 (2006).

This tensile force is given by

$$F \;=\; \left(\frac{\partial E}{\partial L}\right)_A \;=\; \kappa\left(\frac{2\pi L}{A}\right)\frac{2\pi}{A}\times A \;=\; \frac{2\pi\kappa}{R}\,. \tag{4}$$

Evidently, κ can be extracted by measuring the force along the axis of the cylinder, F, and the cylinder's radius R via the equation: $\kappa = FR/2\pi$. An example for what such a configuration would look like can be seen in Fig. 4.

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

Similarly to Sec. 5, the force along the z-direction can be obtained from measuring the Π_{zz} component of the pressure tensor: $F_z = -\Pi_{zz}L_xL_y$. Remember that you will need to first estimate the time to reach equilibrium, and then calculate the average value and error of the mean of Π_{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 $\kappa = F_zR/2\pi$ including error bars. Note that κ has units of ϵ.

Notice that the determination of the radius is not trivial: Even after eliminating troubles due to "stray lipids", dealing with periodic boundary conditions, and finding the cylinder axis, it will not suffice to simply determine the average distance of all lipids from that axis. (What even would that distance be, given that lipids are not point particles?) What we really want is the average radius of the bilayer midplane, and this does *not* coincide with the average distance of, say, the tail beads of all lipids from the axis. The reason is that the two bilayer leaflets are not identical: the outer leaflet contains more lipids than the inner one! What we therefore want is the (equally weighted arithmetic) average of the mean radius of (some specific bead of) the outer leaflet and (some specific bead of) the inner leaflet. To determine this, we need to know in which leaflet every lipid is. This, in turn, we can figure out by checking whether the lipid head points away from the cylinder axis (outer leaflet) or towards the axis (inner leaflet). If you're curious, you might want to check the source code which does all this for you.

6.2. Buckling method

Background. Consider a membrane placed inside a box, such that it exhibits a buckle along one of the directions. It's somewhat too big to fit in, as illustrated in Fig. 5, and hence there must exist a force—the buckling force—which should give access to the membrane bending rigidity. This idea was first put forward by Noguchi.[13]

Parameterizing the membrane shape by the angle $\psi(s)$, the total energy can be written as

$$E[\psi] \;=\; L_y\int_0^L \mathrm{d}s\,\left\{\frac{1}{2}\kappa\,\dot{\psi}^2 + f_x\,\left[\cos\psi - \frac{L_x}{L}\right]\right\}\,, \tag{5}$$

[13] H Noguchi, *Anisotropic surface tension of buckled fluid membranes*, Phys. Rev. E **83**, 061919 (2011).

where L is the length of the membrane contour along the buckle direction (which is of course larger than its horizontal projection L_x), κ is the membrane bending modulus, and f_x is a Lagrange multiplier needed to fix the length of the box in x-direction at a given value of the overall membrane area. Its value gives the lateral compressive stress along the x-direction, which is *not* known in advance but needs to be determined from the boundary conditions and constraints.

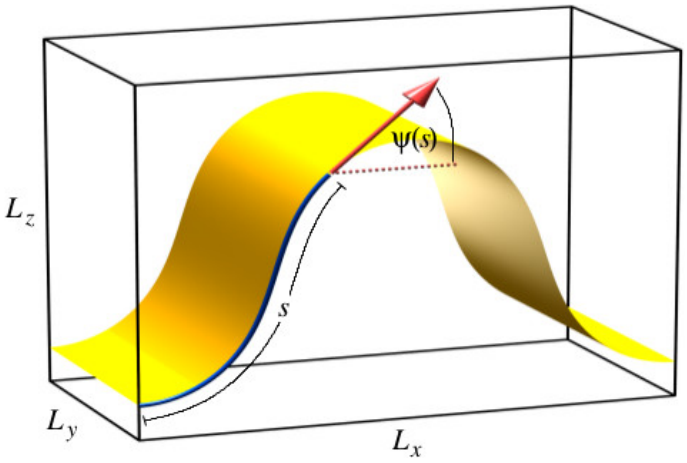

Figure 5: Illustration of the buckling geometry and membrane parametrization.

A functional variation gives the associated shape equation for the membrane. Solving it, we find the buckle shape, and from it the constraint force f_x, which we care about. As it turns out, the hardest part here is to satisfy the two global *constraints*, namely, that (i) the buckle has period L (if measured along the arc length variable s), and (ii) its horizontal projection has period L_z. While the general solution can be written down analytically in terms of elliptic integrals and elliptic functions, enforcing the constraints can unfortunately not be done in closed form. However, we can obtain all interesting quantities as accurate series expansions. For instance, the buckling stress f_x as a function of buckling strain $\gamma = (L - L_x)/L$ is found to be

$$f_x(\gamma) \;=\; \kappa \left(\frac{2\pi}{L}\right)^2 \left[1 + \frac{1}{2}\gamma + \frac{9}{32}\gamma^2 + \frac{21}{128}\gamma^3 + \frac{795}{8\,192}\gamma^4 + \frac{945}{16\,384}\gamma^5 + \frac{2\,247}{65\,536}\gamma^6 + \;\ldots\right] . \qquad (6)$$

The full derivation, plus a discussion of fluctuation corrections, can be found in Hu *et al.*[14] Thus, we can calculate κ from a simulation of a buckled membrane at a specific γ by measuring f_x. It turns out that $f_x(\gamma)$ is not actually such a strongly varying function of γ, hence any inaccuracies in determining γ (which require measuring L, the stress free length of the *unbuckled* membrane!) do not enter as a large source of error. Also, it turns out that this force is essentially unaffected by thermal fluctuations—something that is not really obvious, but all the more pleasing. Notice that fitting Eqn. (6) to some measured stress-strain relation really only has a single fitting parameter: κ. In fact, it would not even be necessary to measure $f_x(\gamma)$ for multiple values of the strain γ. One strain would be enough! However, it is always a good idea to check whether the functional form predicted by the theoretical model is indeed a good description of the data.

Simulation In order to calculate κ using the buckling method, we need to do three things:

1. find the length of a flat membrane under no stress,
2. create a buckled membrane at a particular strain γ, and
3. measure the force on the buckled membrane.

In order to find the proper length for the *flat* membrane, we must allow the length of the box in the x-direction to change until there is no residual stress on the bilayer. The file `simplebilayer.tcl` contains the parameters for this simulation. The main new elements of the file are:

- the initial geometry is set to flat with the command

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

- the number of lipids is set by the box size

[14] M Hu, P Diggins IV and M Deserno, *Determining the bending modulus of a lipid membrane by simulating buckling*, J. Chem. Phys. **138**, 214110 (2013).

```
set setbolx_l {45.0 8.0 15.0 }
set area [expr [lindex $setbox_l 0]*[lindex $setbox_l 1]]
set nlipid [expr $area*2/1.25]
set n_molslist [subst { n_molslist {  { 0 $nlipid } } }]
```

- the barostat is turned on using the following commands

```
set npt on
set p_ext 0.000
set piston_mass 0.5
set gamma_0 1.0
set gamma_v 0.0001
```

- add `boxl` to the list of `analysis_flags`.

While the zero stress simulation is running, you can start the simulation to create the buckled membrane. Once the zero stress simulation is finished, calculate the average length in the x-direction after the simulation has reached equilibrium. The values of dimensions of the simulation box are stored in the file `time_vs_boxl_tmp`. Specifically, the length in the x-direction is in the second column.

There are many ways to create a buckled membrane. Small buckles can be approximated by imposing a cosine-like undulate onto the z-coordinates,

$$h(z) \;\rightarrow\; h_0 + a\,\cos\frac{2\pi x}{L_z}\,, \tag{7}$$

but this approximation gets worse as γ increases. Strictly, the increased length of the membrane along the now undulating shape would have to be compensated for, so one might want to shrink the x-dimension of the box accordingly. For a cosine-wave of amplitude a, as used above, the length increase δL is easily seen to be

$$\frac{\delta L}{L_x} = \int_0^{L_x}\frac{\mathrm{d}x}{L_x}\left\{\sqrt{1+\big[h'(x)\big]^2}-1\right\} \approx \int_0^{L_x}\frac{\mathrm{d}x}{L_x}\,\frac{1}{2}\big[h'(x)\big]^2 = \int_0^{L_x}\frac{\mathrm{d}x}{L_x}\,\frac{1}{2}\left(\frac{2\pi a}{L_x}\right)^2\sin^2\frac{2\pi x}{L_x} = \left(\frac{\pi a}{L_x}\right)^2 . \tag{8}$$

However, even if we shrink the x-dimension of the box by that percentage amount, we are not creating a correct shape: This shrinkage should, strictly, depend on the *local* slope of the buckle, and these issues are not easy to deal with (especially if there is solvent). Moreover, neither is the buckle truly a cosine function, nor do lipids simply move up or down (without acquiring an average tilt).

Alternatively, a buckle can be created by actively compressing a flat bilayer—using a barostat to exert the compressive force, or in other words, increasing the external pressure in the x-direction. Of course, as the theory which we will use to analyze the buckling formalism teaches, the pressure needed to create a buckle in the first place depends on the strain: The bigger the strain, the bigger the pressure. Moreover, buckling is transition that requires a symmetry breaking. To assist the process, it is therefore helpful to impose a slight buckle—say, using the "cosine-method"—in order to "seed" the correct transformation, especially to avoid exciting higher order modes (more than one undulation per box). The latter is a bigger problem than one would maybe think: To avoid very lengthy compression protocols, one will probably want to compress the flat bilayer with a force that substantially exceeds the buckling threshold, but in this case the big energy one ends up pumping into the system might very well induce higher order buckles (or all kinds of other defects), unless we provide the seed of an easy mode to relax into.

In this tutorial we will actively buckle the membrane by stepwise shortening the box length in the x-direction. After each contraction, the bilayer is able to relax during a subsequent constant volume simulation. The commands necessary to buckle the membrane in this fashion are contained in the file `squeeze.tcl`, which replaces the file `main.tcl` as the parameter file. Take a few moments to compare `squeeze.tcl` to `main.tcl`, specifically the "Main Integration" section. The simulation is run by invoking

```
Espresso squeeze.tcl simplebilayer.tcl
```

The output pdb-files contain snapshots of the buckled membrane at different strains. Choose one of the snapshots to be the starting configuration for a simulation that calculates the force on a buckled membrane.

In order to measure the stress on the buckled membrane, we will use the run file `measure_stress.tcl`. Take a moment to look at it. This file has been modified from `main.tcl` to accept the checkpoint file from the buckling simulation. The main changes are in the Initialization section of the script. It is important to change the label to match the checkpoint file that you have chosen. If you choose `checkpoint.t2602.gz`, then the Initialization section should look like:

```
set lable "t2602"
set outputdir "./$lable"
set chkptfile "$outputdir/checkpoint.$lable.gz"
set in [open "|gzip -cd $chkptfile" "r"]
while { [blockfile $in read auto] != "eof" } {}
close $in
set outputdir "./$lable"
```

Before you can run the simulation, you need to set up the simulation by:

- create a folder with the same name as the "lable" variable
- copy the checkpoint file into the new folder
- copy the topology file from the squeezing simulation (`XXX.top`) into the new folder

You are now ready to run a new simulation file with the command:

```
Espresso measure_stress.tcl simplebilayer.tcl
```

Once the simulation is finished, you will need to first calculate the force in the x-direction. It is measured from the Π_{xx} component of the pressure tensor: $F_x = \Pi_{xx} L_y L_z$. The force is equal to the stress from Eqn. (6) times the width of the box, L_y. From this, calculate the bending modulus κ. Recall that the bending modulus has the units of energy, hence the results will come to in the natural energy unit of the simulation program, which is ϵ.

7. Spontaneous vesicle formation

7.1. Theory

We will now look at a patch of a lipid bilayer with an open edge. Its free energy can be expressed using Eqn. (1), but with two changes. First, since lipids can easily change between the two leaflets by moving around the edge, it is hard to argue for a nonzero spontaneous curvature, and so we will drop the term K_0. Second, because the bilayer now has an open edge, we need to add a term proportional to the length of that edge, multiplied by the edge tension γ:

$$E[\mathcal{S}] = \int_{\mathcal{S}} \mathrm{d}A \, \left\{ \frac{1}{2}\kappa K^2 + \overline{\kappa} K_{\mathrm{G}} \right\} + \oint_{\partial \mathcal{S}} \mathrm{d}s \, \gamma \, . \tag{9}$$

The previous exercises have allowed us to extract estimates for the line tension γ (Sec. 5) and the bending modulus κ (Sec. 6). The only material parameter left is the Gaussian bending modulus $\overline{\kappa}$, which we will now look into. This latter modulus is much trickier to determine, because of the subtle and beautiful *Gauss-Bonnet Theorem* from differential geometry: It states that the surface integral over the Gaussian curvature remains constant if there is neither a change in the topology nor in the boundary of the membrane patch in question. Hence, it is difficult to measure $\overline{\kappa}$ in experiments and in simulations, because one rarely has easy control over any of those two.

This section provides the essential part of a new method to measure this parameter $\overline{\kappa}$ by studying the process by which flat membrane patches close up to form closed vesicles.[15,16] This process is driven by the attempt of a

[15] M Hu, JJ Briguglio and M Deserno, *Determining the Gaussian Curvature Modulus of Lipid Membranes in Simulations*, Biophys. J. **102**(6) 1403–1410 (2012).

[16] M Hu, DH de Jong, SJ Marrink and M Deserno, *Gaussian curvature elasticity determined from global shape transformations and local stress distributions: a comparative study using the MARTINI model*, Faraday Discuss. **161**, 365–382 (2013).

flat membrane patch to lower its edge tension, but it is opposed by the curvature energy it gains in the process of doing so.

We can devise a simple model to quantitatively describe this: Assume the bilayer patch always takes the shape of an axisymmetric spherical cap of given area and some curvature c that will increase in the process of curving up. We can then write down the analytical expression of curvature energy as a function of that curvature c as follows.

Assume the patch has area A. The radius of the final vesicle is then $R = \sqrt{A/4\pi}$. Define the dimensionless reaction coordinate $x = (Rc)^2$ and the dimensionless control parameter $\xi = \gamma R/(2\kappa + \overline{\kappa})$. Simple geometry then shows that the excess energy of a curved patch, compared to a flat one, measured in units of the bending energy of the final vesicle, can then be written as

$$\frac{\Delta E(x,\xi)}{4\pi(2\kappa+\overline{\kappa})} \;=\; \Delta\tilde{E}(x,\xi) \;=\; x + \xi\Big[\sqrt{1-x}-1\Big]\,. \tag{10}$$

Studying this function, we readily find that for $\xi < 1$ the patch is stable and for $\xi > 1$ the vesicle is stable. However, for $1 < \xi < 2$ the stable vesicle state is separated from an initial flat patch state by a barrier located at $x^* = 1 - (\xi/2)^2$. In other words, its critical curvature is

$$c^* \;=\; \sqrt{\frac{4\pi}{A} - \frac{\gamma^2}{4(2\kappa+\overline{\kappa})^2}} \;=\; \sqrt{\frac{8\pi}{Na} - \frac{\gamma^2}{4(2\kappa+\overline{\kappa})^2}} \;\approx\; \sqrt{\frac{8\pi}{Na} - \Big(\frac{\gamma}{2\kappa}\Big)^2} \tag{11}$$

where N is the number of lipids, a is the area per lipid, and in the last step we used the rule of thumb that $\overline{\kappa} \approx -\kappa$. The latter, of course, could be way off, but for now we need a way to estimate c^* based on the knowledge we have acquired for now.

7.2. Method

A good way to get c^* is to simulate a piece of bilayer starting from different initial curvatures c and measure the probability of it folding into a vesicle, P_{fold}. For c near c^* we are on top of the barrier and would expect P_{fold} to be around $1/2$.[17]

Because of the time limitation here, we cannot afford a serious search for c^*. Instead, what we will do is to bracket the critical curvature from above and below, which will in turn also result in an upper and a lower value for $\overline{\kappa}$, respectively. In order to have some rough idea where the barrier lies to begin with, we will use the initial guess $\overline{\kappa} = -\kappa$ and see how well it serves us later.

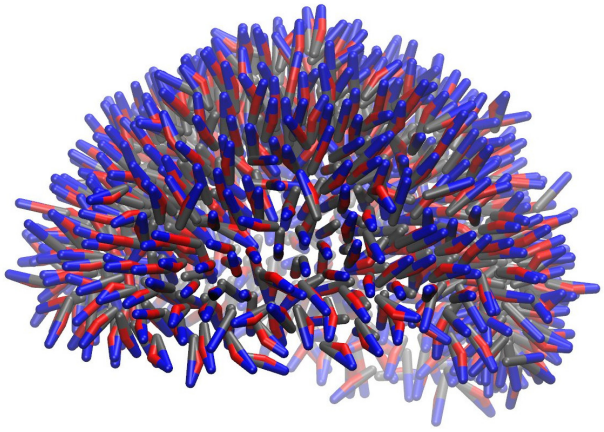

Figure 6: Illustration of a curved bilayer patch at a scaled curvature of $x \approx 0.55$.

Given what you know by now about γ and κ, and also using $a \approx 1.2\,\sigma^2$, you can show that a patch of **1000 lipids**—which we will now focus on—would have a critical radius of curvature of $1/c^* \approx 13\,\sigma$. Find the number that follows from your own measurements. Convince yourself that should you be able to actually pinpoint c^* in a careful set of simulations, you would in turn be able to calculate $\overline{\kappa}$.

Now set up a spherical cap of a curvature c^* and check if it closes up into a vesicle or expands into a flat patch. If you have time, do it repeatedly, so you vet statistics! In the end, results from all participants will be collected together and thus the folding probability P_{fold} can be estimated.

NOTE: for a more meaningful result on P_{fold}, please check the results in Section 8 *before* you start the folding simulation.

[17] This so-called *splitting probability* can be calculated exactly if we know the shape of the energy barrier and the diffusion constant that describes the stochastic barrier crossing process, but we will not go into this any further here—see the literature!

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. You are encouraged to repeat the present simulations with longer simulation times, thus allowing better statistics. And should you ever do anything like that in real research, then you need to think much harder about a proper handling of your correlated simulation trajectories.

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 $L = 50\,\sigma$

The diagonal pressure tensor components could for instance look like this:

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

Notice that only Π_{yy} is significantly different from zero. This now allows us to obtain the line tension:

$$\gamma = -\frac{1}{2}\Pi_{yy} L_x L_z = 3.4 \pm 0.3\,\epsilon/\sigma \tag{12}$$

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:

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

We can now extract the bending modulus:

$$\kappa = \frac{FR}{2\pi} = 12 \pm 3\,\epsilon \tag{13}$$

8.3. Vesicle formation

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

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

$$c^* \;\approx\; \sqrt{\frac{8\pi}{Na} - \left(\frac{\gamma}{2\kappa}\right)^2} \;\approx\; \sqrt{\frac{8\pi}{1000 \times 1.2\,\sigma^2} - \left(\frac{3.35\,\epsilon/\sigma}{2 \times 13.7\epsilon}\right)^2} \;\approx\; 0.077\,\sigma^{-1} \tag{14}$$

Ideally, when setting up the initially curved bilayer, the radius should be $r^* = 1/c^* = 12.9\,\sigma$. However, as it turns out that $\overline{\kappa}$ is slightly above $-\kappa$ ($\sim -0.9\,\kappa$), r^* is actually close to $r^* = 1/c^* = 11.5\,\sigma$.

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 also a bracket on $\overline{\kappa}$.

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/wiki/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:

```
1 #define EXTERNAL_FORCES
2 #define DPD
3 #define NPT
4 #define LENNARD_JONES
5 #define TABULATED
6 #define LJCOS
7 #define LJCOS2
8 #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). Notice that `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 ESPRESSOwas installed in `$ESPRESSO_DIR/`.

A.4. Gnuplot

Gnuplot[18] 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 should always visually check the fitting by

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

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