

GROMACS Coarse-Graining Workshop

CSC, Helsinki, Finland

January 28-30, 2009

General Information

The tutorial exercises described here aim to help you familiarize yourself with some of the basic and more advanced aspects of using GROMACS to perform coarse-grained simulations. The main focus is on the semi-empirical MARTINI model. Familiarity with GROMACS is assumed; for explanation of how to work with GROMACS and the specification of force field and run parameters we refer to the GROMACS user manual and web-pages.

These tutorials follow a historical track, starting with the MARTINI model for lipids. It then goes on to describe how to set up simulations with proteins (and peptides), and ends with a state-of-the-art demonstration of reverse mapping (i.e. fine graining from a coarse-grained structure). As an alternative to the MARTINI model, a hierarchical modeling strategy is offered, in which first an effective interaction potential based on atomistic-level simulations is derived and then used.

The material can be covered in as much time as you like, and the speed at which you go through this material depends on the computational facilities available, your GROMACS skills, and the time you take to consider what you're doing. However, the general idea of this workshop is, that the lipid modules are addressed on the first day, the protein modules on the second day, and the multigraining modules on the second and/or third day. The set-up is modular, however, and depending on your particular interest, you may change the order of the exercises.

Some parts of the tutorial may take too long to fit in the time allotted during the workshop. You may then either leave runs overnight to return to them the next day, or use "pre-prepared" intermediate and result files that are provided by the organizers.

Tutorial WED 28: MARTINI Lipids

Lars Schäfer

Today's aim is to create and study properties of coarse-grained models of lipid bilayer membranes. First, we will attempt to spontaneously self-assemble a DSPC bilayer and check various properties that characterize lipid bilayers, such as the area per lipid, bilayer thickness, P2 order parameters, and diffusion. Next, we will change the nature of the lipid head groups and tails, and study how that influences the properties. Finally, there is a slightly more advanced exercise on how to parameterize a MARTINI CG model of diC18:2-PC, a PC lipid with a doubly unsaturated tail, based on an all-atom simulation.

You can find more on the MARTINI force-field plus some additional topology files at the MARTINI homepage:

<http://md.chem.rug.nl/~marrink/coarsegrain.html>

1 Spontaneous assembly: topology files and system setup

Unpack the `martinlip.tar.gz`. (It expands locally, so to keep things clean put it in a separate directory, e.g. `LIPIDS`.) We will begin with self-assembling a DSPC (distearoyl-phosphatidylcholine) bilayer from a random configuration of lipids and water in the simulation box. First, create this random configuration of 128 DSPC starting from a single DSPC molecule:

- `genbox -ci dspc_single.gro -nmol 128 -box 7 7 7 -try 100 -o 128_noW.gro`

Next, minimize the system...

- `grompp -f em.mdp -c 128_noW.gro -p dspc.top -maxwarn 10`
- `mdrun -v -c 128_minimised.gro`

...add 6 CG waters (i.e., 24 all-atom waters) per lipid...

- `genbox -cp 128_minimised.gro -cs water.gro -o waterbox.gro -maxsol 768`

...and minimize again (adapt dspc.top to reflect the water beads added to the system):

- `grompp -f em.mdp -c waterbox.gro -p dspc.top -maxwarn 10`
- `mdrun -v -c minimised.gro`

2 Self-assembly simulation

Now, you are ready to run the self-assembly MD simulation. About 25 ns should suffice.

- `grompp -f md.mdp -c minimised.gro -p dspc.top -maxwarn 10`
- `mdrun`

This might take ca. 25 min on a dual core machine. You may want to check the progress of the simulation to see whether the bilayer has already formed before the end of the simulation. You may do this most easily by starting the inbuilt GROMACS viewer:

- `ngmx`

In the meantime, have a close look at the MARTINI parameter files; in particular, the interactions and bead types. What are the differences between PC and PE head groups as well as between saturated and unsaturated tails?

Check whether you got a bilayer. If yes, check if the formed membrane is normal to the z-axis (i.e., membrane in the xy-plane). If the latter is not the case, rotate the system accordingly (with `editconf`). In case you did not get a bilayer at all, continue with the pre-formed one from “`dspc.bilayer.gro`”.

Set up a simulation for another 15 ns at zero surface tension (switch to semiisotropic pressure coupling) at a higher temperature of 341 K. This is above the experimental gel to liquid-crystalline transition temperature of DSPC. You will find how to change pressure coupling and temperature in the GROMACS manual:

<http://www.gromacs.org/documentation/reference/online.html>

3 Analysis

From the latter simulation, calculate properties such as

- area per lipid
- bilayer thickness
- P2 order parameters of the bonds
- lateral diffusion of the lipids

In general, for the analysis, you might want to discard the first few ns of your simulation (equilibration time).

3.1 Area per lipid

To get the (projected) area per lipid, you can simply divide the area of your simulation box (Box-X times Box-Y from `g_energy`) by half the number of lipids in your system.¹

3.2 Bilayer thickness

To get the bilayer thickness, use `g_density`. You can get the density for a number of different functional groups in the lipid (e.g., phosphate and ammonium headgroup beads, carbon tail beads, etc) by feeding an appropriate index-file to `g_density` (make one with `make_ndx`; you can select, e.g., the phosphate beads by typing “a P*”). The bilayer thickness you can obtain from the distance between the headgroup peaks in the density profile.

A more appropriate way to compare to experiment is to calculate the electron density profile. The `g_density` tool also provides this option. However, you need to supply the program with a data-file containing the number of electrons associated with each bead (option `-ei electrons.dat`). The format is described in the GROMACS manual.

Compare your results to those from small-angle neutron scattering experiments (Balgavy et al., *Biochim. et Biophys. Acta* 2001, 1512, 40-52):

¹Note that this might not be strictly OK, because the self-assembled bilayer might be slightly asymmetric in terms of number of lipids per monolayer, i.e., the areas per lipid are different for the two monolayers. However, to a first approximation, we will ignore this here.

- thickness = 4.98 ± 0.15 nm
- area per lipid = 0.65 ± 0.05 nm²

3.3 Order parameters

Now, we will calculate the (second-rank) order parameter, which is defined as

$$P_2 = 1/2 (3 \langle \cos^2 \theta \rangle - 1),$$

where θ is the angle between the bond and the bilayer normal. $P_2 = 1$ means perfect alignment with the bilayer normal, $P_2 = -0.5$ anti-alignment, and $P_2 = 0$ random orientation. First, make a new directory “order” and unpack order.tar.gz into it. Copy the xtc and gro file there as well (should be named traj.xtc and conf_file.gro, respectively). You may need to compile “order_param.exe” from “order_param.f” using gfortran. The script “do-order.sh” will calculate P_2 for you. As it explains when you invoke it, it needs a number of arguments. The command

- `./do-order.sh 0 10000 20 0 0 1 64`

will for example read in a 10-ns trajectory of 64 lipids and average over 20 equally-spaced snapshots. P_2 is calculated relative to the z-axis.

3.4 Lateral diffusion

Before calculating the lateral diffusion, remove jumps over the box boundaries (trjconv -pbc nojump). Then, calculate the lateral diffusion using g_msd. Take care to remove the overall center of mass motion (-rmcomm),² and to fit the line only to the linear regime of the mean-square-displacement curve (-beginfit and -endfit options of g_msd). To get the lateral diffusion, choose the “-lateral z” option. To compare the diffusion coefficient obtained from a MARTINI simulation to a measured one, a conversion factor of about 4 has to be applied to account for the faster diffusion at the CG level due to the smoothed free energy landscape.

²In GROMACS 3, this option is not available. An additional “trjconv -center” will do the trick.

4 Different bilayers: unsaturated tails, headgroups

Next, investigate the effect of changes in the lipid tails and in the headgroups on the properties of the bilayer. We will i) introduce a double bond in the lipid tails, and ii) change the lipid head groups from PC to PE.

4.1 Unsaturated tails

To introduce a double bond in the tail, we will replace the DSPC lipids by DOPC (compare the MARTINI topologies of these two lipids). Replace DSPC by DOPC in your `.top` and `.mdp` files, and `grompp` will do the rest for you (you can ignore the "atom name does not match" warnings of `grompp`).

4.2 Changing the headgroups

Starting again from the final snapshot of the DSPC simulation, change the head groups from PC to PE.

For both new systems, run 15-ns MD simulations and compare the above properties (area per lipid, thickness, order parameter, diffusion) between the three bilayers (DSPC, DOPC, DSPE). Do the observed changes match your expectations? Why/why not? Discuss.

5 Refine CG parameters based on AA simulation

In this part, we will try to obtain improved MARTINI parameters for diC18:2-PC, a PC lipid with a doubly unsaturated tail. We will try to optimise the MARTINI parameters such that the dihedral and angle distributions within the lipid tail match those from a 100 ns all-atom simulation (all-atom data in "fg.xtc") as close as possible. To get started, unpack `lipid_refine.tar.gz` into a new directory.

The task can be divided into the following five steps:

1. Transform the all-atom (fine-grained, FG) trajectory into its coarse-

grained counterpart, based on a pre-defined mapping scheme³.

2. Calculate the angle and dihedral distributions. These will serve as the reference (“gold standard”) later.
3. Do a coarse-grained simulation of the system.
4. Calculate the angle and dihedral distributions, and compare to the reference.
5. Revise coarse-grained parameters, repeat steps 3 and 4 until you are satisfied with the result.

For the transformation from FG to CG, we will use the program `g_fg2cg`, which is part of a modified version of GROMACS that you either need to install yourself (see section Transformation run) or can use the “module load” option on the machines at CSC:

- `source /where-ever-your-installation-is/bin/GMXRC`
- `setenv GMXLIB /where-ever-your-installation-is/share/top`

Alternatively, you may skip this part and use the transformed trajectory provided in the tarball (`fg2cg.xtc`). First, transform from FG to CG:

- `g_fg2cg -pfg fg.top -pcg cg.top -c fg.gro -n 1 -o cg.gro`
- `g_fg2cg -pfg fg.top -pcg cg.top -c fg.xtc -n 1 -o fg2cg.xtc`

Then, make an index file needed for the calculation of the angle and dihedral distributions within the lipid tail:

- `make_ndx -f cg.gro -o angle.ndx`
- `aGL1 | aC1A | aD2A`
- `aC1A | aD2A | aD3A`
- `aD2A | aD3A | aC4A`
- `aC1A | aD2A | aD3A | aC4A`

Now, calculate the distributions with `g_angle`:

³The mapping is already defined, look at the [mapping] section in `dupc_fg.itp`

- `g_angle -f fg2cg.xtc -type angle -n angle.ndx -od fg2cg_ang{1,2,3}.xvg`
- `g_angle -f fg2cg.xtc -type dihedral -n angle.ndx -od fg2cg_dih1.xvg`

These are the target distributions that we want to obtain with the CG model. As a starting point, we will use the MARTINI parameters as defined in “dupc.cg.itp”, i.e., all angles at 180 degrees. Carry out a short CG simulation (starting from “cg_mdstart.gro”, you just have to add water to the cg.top). After comparing the angle and dihedral distributions to the fg2cg reference, change the angle parameters in “dupc.cg.itp” and repeat.

Tutorial THU 29: MARTINI Proteins

Martti Louhivuori

Today you will learn to set-up and run a MARTINI CG protein simulation starting from a PDB structure. Some variants adding extra harmonic bonds that serve to help stabilize the tertiary structure are also introduced. Finally, an advanced exercise will show you how to increase the resolution of a CG model to an atomistic one.

Intro: Keeping in line with the overall MARTINI philosophy, the coarse-grained protein model groups ~ 4 heavy atoms together in one CG bead. Each residue has one backbone bead and zero or more side-chain beads depending on the amino acid type. The secondary structure of the protein influences both the selected bead types and bond/angle/dihedral parameters of each residue as explained in Monticelli et al. (J Comput Chem Theory, 2008). It is noteworthy that, even though the protein is free to change its tertiary arrangement, local secondary structure is pre-defined and thus static throughout a simulation. Conformational changes that involve changes in the secondary structure are therefore beyond the scope of MARTINI CG proteins.

Setting up a CG protein simulation consists basically of two steps: 1) converting an atomistic protein structure into a CG model and 2) generating a suitable MARTINI topology. These steps are easy to do with publicly available tools distributed at the MARTINI web-site:

<http://md.chem.rug.nl/~marrink/coarsegrain.html>

6 Martini CG simulation of ubiquitin in water

The aim of this module is to set-up a CG simulation of ubiquitin in a water box. After getting the atomistic structure of ubiquitin (1UBQ), you'll need to convert it into a CG structure and to prepare a MARTINI topology for it. Once this is done the CG structure can be minimised, solvated and simulated. The steps you need to take are roughly the following:

1. Unpack the protein-tutorial.tar.gz archive.
2. Download (/copy) 1UBQ.pdb.

3. Remove HETATMs and other crud from the PDB-file.
4. Use the atom2cg-script (available from the MARTINI web-site) to convert the atomistic structure into a CG structure.
5. Figure out the secondary structure of the protein from the original PDB, e.g. by using dssp or do_dssp. When using do_dssp you'll need a DSSP binary, which can be downloaded from a CMBI website (under "Linux executable"):

<http://swift.cmbi.ru.nl/gv/dssp>

The binary may not work on all platforms. In case you have a working DSSP binary, but do not want to use GROMACS, the dssp2ssd.py script might be useful to convert DSSP output to a .ssd file required later. If all fails, appropriate .ssd files are provided for the proteins and peptides used in this tutorial and for future reference, the DSSP output is provided for all proteins in the PDB on the quoted website (under "FTP access").

6. Get the amino acid sequence of ubiquitin.
7. Check the man-page of the seq2itp-script (i.e. run "`seq2itp --help`") for the correct format of the secondary structure and amino acid sequence and prepare the files as instructed.
8. Use seq2itp-script (available from MARTINI web-site) to generate a MARTINI topology from the sequence and secondary structure files.
9. Prepare Gromacs input files: a system topology file, .mdp files etc.
10. Do a short minimization in vacuum.
11. Solvate the system with genbox (water-1bar-303K.gro is an equilibrated waterbox) and minimise. Remember to use a larger van der Waals distance when solvating to avoid clashes.
12. Do a short position restrained simulation.
13. Start production run.
14. ...
15. PROFIT!!! What sort of analysis can be done on this molecule? A mapped fine-grained simulation of ubiquitin is available in the archive UBQ_FG_mapped.tar.gz.

If you need help with the commands, think twice before raising your hand ;)

7 Elastic networks

The aim of this module is to see how application of elastic networks can be combined with the MARTINI model to conserve tertiary and quaternary structures more faithfully without sacrificing realistic dynamics of a protein. We offer two alternative routes. Please be advised that this is an active field of research and that there is as of yet no "gold standard".

First you should simulate a normal CG protein without an elastic network and then see what changes when you use a CG topology with an elastic network. The workflow to do so goes like this:

1. Copy HIV-1 Protease (1A8G.pdb) for yourself.
2. Remove HETATMs and the ligand (chain C) from the PDB-file.
3. Repeat steps 4–13 from the previous exercise. (You'll need to simulate the protein for hundreds of nanoseconds to see major changes in the structure).
4. Visualize the simulation. Look at the binding pocket of the protein. Does it stay closed, open up, or what? What happens to the overall protein structure after some time (200–300 ns)?

—

There are several ways in which the elastic network can be applied. In its simplest form you generate a number of extra bonds between (originally non-bonded) beads within a specified cutoff distance and with a specified (standard) force constant and add these to the original topology. Alternatively, you may use the structural information of the PDB file and specify CG bonds and angles that take the values as calculated directly from the PDB structure as their reference value.

7.1 Martini + elastic network

The first option to help preserve higher-order structure of proteins is to add to the standard MARTINI topology extra harmonic bonds between non-bonded beads based on a distance cutoff. Note that in standard MARTINI, long harmonic bonds are used to impose the secondary structure of extended elements (sheets) of the protein.

5. Generate a coarse-grained PDB structure you want to use as the reference structure.

6. Generate a list of distances between all beads within a certain distance. The GROMACS tool `genrestr` is well suited for this. First generate an index file containing only those beads you want to consider, the standard option would be the backbone beads.

```
> genrestr -f cg.pdb -n index -constr -o constraints
```

7. Select those restraints you want to keep, based on a lower and upper cutoff. In addition to MARTINI you would not want to duplicate already existing bonds, so the `select_restr.pl` script includes a lower cut-off value. The example below selects all instances from the full list in `constraints.itp` with a distance between 0.5 and 0.9 nm and generates a harmonic bond with a force constant of 500 kJ mol⁻¹ nm⁻².

```
> ./select_restr.pl constraints.itp 0.5 0.9 500 > AddElBonds.itp
```

8. Paste the resulting file into the standard MARTINI topology in the section [bonds].
9. Proceed as before and start a production run.

7.2 ElNeDyn

The second option to use elastic networks in combination with MARTINI puts more emphasis on remaining close to the overall structure as deposited in the PDB than standard MARTINI does. For this exercise you'll need two small FORTRAN programs available to the participants of this workshop. The main difference from the standard way (used in the previous exercise) is the use of a global elastic network between the backbone beads to conserve the conformation instead of relying on the angle/dihedral potentials and/or local elastic bonds to do the job. The position of the backbone beads is also slightly different. By default the center-of-mass of the peptide plane is used as the location of the backbone bead, but in the elastic network implementation the location of the C^α are used for this.

10. Unpack the `ElNeDyn.tar.gz` archive that contains all the necessary files for using the global elastic network.
11. Compile the FORTRAN programs (`pdb2CGpdb-2.1.f` and `topol-CG-2.1.f`) using `g77`.

12. Remove REMARKs / HEADERS etc. and hydrogens. You should have only ATOM records in the PDB for non-hydrogen atoms.
13. Add the number of residues & the number of atoms on the first line of the PDB-file, e.g. if you have 76 residues and 842 atoms, the first row should be "76 842".
14. Use pdb2CGpdb to transform the atomistic structure into a CG structure. (Note: the name of the output file is chosen deliberately because the next program expects a file with that name.)

```
> ./pdb2CGpdb < 1A8G.pdb > pdb-CG.pdb
```

15. Use topol-CG to generate a CG topology with an elastic network. (Note: the bead types are still wrong, but we'll fix it next.)

```
> ./topol-CG-2.1
```

```
Choose a force constant (kJ mol-1 nm-2) for ElNeDyn:
```

```
500
```

```
Choose a cutoff (nm) for ElNeDyn:
```

```
0.9
```

16. In this case, only the bead types need to be changed, because they are given only generic types by the program. Cut-n-paste the atom definitions from the normal MARTINI topology (prepared earlier using seq2itp) to the elastic network topology replacing the wrong atom definitions. (Hint: e.g. nedit can copy and paste columns. Just press Ctrl when selecting a region.)
17. Copy the same Gromacs input files used with the normal MARTINI topology.
18. Do a short minimization in vacuum.
19. Solvate the system with genbox and minimise.
20. Do two position restrained simulations:
 - (a) first with a slower time-step (1 fs) while restraining the positions of all the protein beads
 - (b) and then with a normal time-step (20 fs) while restraining the positions of the back-bone beads

21. Simulate the system for a long time without restraints.
22. Visualize the simulations. Analyze the differences between the simulations performed in this section.

8 Reverse transformation

Andrzej Rzepiela

Simulations at coarse-grained level allow the exploration of longer time-scales and larger length-scales than at the usual atomistic level. However, the loss of detail can seriously limit the questions that can be asked to the system. Methods that re-introduce atomic details in a CG structure are therefore of considerable interest. Such structures provide starting points in phase space for simulations at the more detailed level that may otherwise take too long to reach. As a demonstration of such a method, we will use restrained simulated annealing (SA) to increase the resolution of a system containing a WALP peptide spanning a DPPC bilayer. We will investigate how the number of steps allowed for the reconstruction influences the quality of the generated FG structure.

8.1 CG system preparation

1. Unpack the reverse.tar.gz archive that contains all necessary GRO-MACS files for this exercise.
2. Insert the WALP peptide (walp_cg.gro) into a DPPC bilayer (dppc_cg.gro), using genbox (remember about the -vdwd option). To achieve this, one should actually solvate the WALP peptide in the lipid bilayer + water system.
3. Construct a topology file for the peptide, selecting an α -helical secondary structure throughout the peptide. Refer to the protein tutorial to see what the required files are and what they should contain.
4. Modify the dppc_cg.top file to reflect the system you have generated (number of lipid and water molecules, add peptide).
5. Perform a 12 ns simulation to equilibrate the system.
6. Check the position of the peptide in the bilayer and its tilt.

8.2 Transformation run

In this part of exercise we will use a modified version of GROMACS that allows one to generate a fine grained (FG) structure from CG beads. The same version is also used to transform a FG structure to CG beads. Thus, the program allows one to switch between FG and CG representations. To achieve this, additional information is put in the topology file at the FG level in a section called [mapping]. The tool `pdb2gmx` can generate this mapping for proteins automatically. Two new functions in `mdrun` are responsible for restrained simulated annealing (SA) to converge from a “random” FG structure to a FG structure compatible with the CG structure, i.e. the FG positions are optimized in such a way that the CG structure is preserved in the FG-to-CG mapping (each CG bead is at the center of mass of the FG atoms that map to it). There is also small tool, called `g_fg2cg`, to generate CG structures from FG ones defined by the mapping and “random” FG starting structures based on CG input file, that are afterwards optimized during a SA run. The package is supplied in the archive “`gromacs_reverse.tar.gz`”.

1. Source the modified version of GROMACS

```
source /where-ever-your-installation-is/bin/GMXRC
or, for this tutorial on the available machines
module load gromacs/cg-2009-rev
```

2. Set path to GROMACS library

```
setenv GMXLIB /where-ever-your-installation-is/share/gromacs/top/
```

3. Modify the fine-grained `fg.top` file in such a way that the number of water and lipid molecules is the same as in the coarse-grained model. One `FG_W` corresponds to four normal water molecules.
4. Use `g_fg2cg` to construct input atomistic structure for a simulated annealing run. Take the last frame from the CG simulation (called later `cg.gro`) performed before. Check the output with `VMD` or `ngmx`.

```
g_fg2cg -pfg fg.top -pcg dppc_cg.top -n 0 -c cg.gro -o fg.gro
```

5. Modify the `fg.mdp` file. Set the last non-water CG bead (“water = nnnn”, see comments for additional `mdp` options near the bottom of the `fg.mdp` file) to the one from your `cg.gro` file. Leave the rest of the options default.

6. Use grompp to create a topol.tpr file for a single processor.

7. Perform a SA run by typing

```
mdrun -coarse cg.gro -v
```

8. Change the number of simulation steps to 1000 and SA time parameters to 0 1.5 0 1.5 0 1.5. Perform another SA run with the altered parameters.

9. Change the number of simulation steps to 5000 and SA time parameters to 0 7.5 0 7.5 0 7.5. Perform another SA run with the altered parameters.

10. Optionally (if you got enough time) change the number of simulation steps to 60000 and the SA time parameters to 0 100 0 100 0 100 . Perform this SA run.

11. Plot the potential energies for all runs and compare them.

12. Compare the FG WALP models from all runs. Check the secondary structure and ϕ and ψ angles (e.g. through using the Ramachandran plot in VMD) .

13. Compare the dihedral distribution for one of dihedral angles in a tail of the DPPC lipids.

Tutorial FRI 30: Structural modeling

Berk Hess

9 Coarse-graining methane in water

In this exercise we will integrate out the solvent (water) degrees of freedom for methane molecules solvated in water. A Potential of Mean Force (PMF) for the interaction between methane molecules in water is obtained and later used to perform an implicit solvent simulation which can be compared to simulations with full details. To speed up the calculations we use a united atom (GROMOS) force field for methane, but the same procedure will also work for multi-atomic solutes.

NOTE: for this exercise to work properly you need GROMACS version 4.0.3 or higher (GROMACS 4.0 and 4.0.2 do have a (small) error in the pressure when using the pull code).

9.1 Potential of mean force

First we will determine the mean force between two methanes in water by constraining the distance between the two methanes at several (constant) values.

1. Go to the directory pmf.

In this directory you will find a topology, configuration and parameter file for simulating 2 methanes in a rhombic dodecahedron box with 300 SPC/E water molecules. We will use the pull code for constraining the distance between the two methanes, the pull options are given on the mdp options web page.

2. Now add the pull options to the mdp file.

We want to do constraint pulling in geometry distance with one reference group and 1 pull group (note that with only 1 pull group the pull and reference group are not treated differently). You will need to make groups from residues (or atoms) 1 and 2 with the `make_ndx` program. Then you can select these groups with the pull options. Finally it is useful to reduce the amount of pull output. Set the `x` output to zero (the distance is constant so this is useless) and set the `f` output frequency to 5 steps. Before continuing it might be useful to check if

the mdp file by running:

```
grompp -c start.d0.28.pdb
```

3. The csh script DO_PULL will do constraint simulations at 14 constantly spaced distances between 0.28 and 0.80 nm. Each simulation will be put in a separate trajectory, will start from the last conformation of the previous distance and will run for 400 ps. Run the script:

```
./DO_PULL
```

This make take a while.

4. When all runs have finished (or also during running), you can get the mean force including error estimates by running a csh script:

```
./GET_MF d 50
```

Have a look at the mean force file all_d.xvg with xmgrace.

5. The mean force can be corrected for the entropic contribution and integrated using the integrate_mf.awk awk script:

```
./integrate_mf.awk all_d.xvg > pmf.xvg
```

Have a look at the PMF with xmgrace.

9.2 Atomistic reference simulations

Before we do the coarse-grained simulation, we will generate reference data to compare with. There are two directories, mix2 and mix10 with 2% and 10% methane/water mixtures of 1000 molecules, respectively. They are 400 ps, which gives somewhat noisy data. If your computer is fast, you can make them longer.

1. Go into these directories and run grompp and mdrun:

```
grompp mdrun -v
```

2. When the simulations are finished, generate methane-methane RDFs with the command:

```
g_rdf -bin 0.01 -s -b 50
```

Compare the RDFs and have a look at the two trajectories with VMD.

9.3 Implicit solvent simulations

To do implicit solvent simulations, we need to remove the water and use the PMF as a tabulated potential. In the directory `cg2` there is a topology file for coarse-grained methane. The CH4 atomtype has the `c6` parameter set to 1, such that the potential between CH4 and CH4 is equal to the dispersion column in the table file.

1. Generate a starting configuration in `cg2` by simply removing the water lines from the 2% atomistic configuration:

```
grep -v SOL ../mix2/conf.pdb > conf.pdb
```

Copy the `mdp` file from the `mix2` directory. For implicit solvent simulations we will use stochastic dynamics to mimick the friction of the water degrees of freedom and we will keep the box fixed to its initial value. Set the integrator to `sd`, set `vdwtype` to `user` (`mdrun` will then read the user tables), set `tau_t` to 1 ps and turn off the pressure coupling (`pcoupl = no`). Since the implicit solvent simulations run much faster than the explicit solvent simulations, we can increase `nsteps` by a factor of 10. Now generate the table file:

```
../make_table.awk ../pmf/pmf.xvg > table.xvg
```

2. Run the simulation:

```
grompp mdrun -v
```

Have a look at the trajectory with VMD.

3. Get the structure:

```
g_rdf -bin 0.01 -s -b 50
```

and compare it with the explicit solvent result.

4. Now we coarse-grain the 10% mixture. Create a directory `cg10`, copy `grompp.mdp` and `topol.top` from `cg2` and modify the number of molecules. Repeat the procedure above for the 10% mixture. You will see that the implicit solvent potentials do not work well for the 10% mixture.

9.4 Comparing dynamics

The dynamics of the implicit solvent simulations is mainly controlled by the friction coefficient $1/\tau.t$. We just set $\tau.t$ to 1 ps, which gives fast dynamics and therefore fast sampling.

1. To check how fast the dynamics is, we can look at the methane diffusion. Determine the mean square displacement (MSD) for mix2 and cg2 simulations:

```
g_msd -b 50
```

Compare the MSD curves for the explicit and implicit simulations. What time scaling factor is required to map the implicit solvent MSD onto the explicit solvent MSD? Scale $\tau.t$ with this factor and rerun the cg2 simulation and redetermine the MSD to see if it matches.