Computational and Statistical Methods in Chemistry workshop

Molecular Dynamics Simulations of Biomolecular Systems

Hands-on part 1: Introduction to Molecular dynamics

19. 10. 2021

Neli Sedej

In this workshop you will learn to set up a molecular dynamics (MD) simulation using Gromacs, use coarse grained Martini model and visualize biomolecular structures with VMD. The workshop consists of two parts. In the first part we will set up an all atom MD simulation of a protein ubiquitin in water and analyse the dynamics of wild type and mutant protein. We will construct a coarse grained model of ubiquitin. In the second part we will run a MD simulation of a coarse grained model of polysorbate 80 in water.
If you do not have the access to the tutorial files, you can still do the ubiquitin part of the tutorial by downloading the crystal structure of ubiquitin from the RCSB database. The gromacs input files for this part are attached in the appendix.

1 All atom MD simulation of ubiquitin

1.1 Visualisation of a protein structure using VMD

First move to the directory, which contains the crystal structure of the wild type ubiquitin.

```
cd ~/L14/session2/ub/system_wt/
```

The same structure can be obtained from the PDB database (`https://www.rcsb.org/`) under the PDB ID 1UBQ.
To visualise the molecule, open the .pdb file in VMD.

```
vmd wt.pdb
```

Now you can use the mouse to rotate the molecule and scroll to zoom in. You can translate the molecule after pressing "T". "R" switches back to the rotation mode. "P" turns on the selection mode. After clicking on an atom in selection mode, the properties of the atom will be displayed in the terminal.
To change the representation of the molecule go to `Graphics>Representations` Here you can try different drawing methods. "NewCartoon" is often used to show the secondary structure of the protein.
VMD offers numerous visualization and analysis tools. For example, the distance between two atoms can be measured. To show a selection of residues create new representation and type `resid 4 35` (or any two residues) in Selected Atoms. Then choose a representation that shows individual atoms (e. g. "Licorice"). Measure the distance between two selected atoms by going to `Mouse>Label>Bonds`. Then click on the two atoms.
Now we will open the structure of the mutant. Go to `File>New Molecule..` and find the mutant structure in *session2/ub/system_l67d* directory. Show the mutated residue (resid 26) in both structures, to see the mutation.

1.2 Preparing the system for simulation

1.2.1 Solvating the system

We will perform the simulation of ubiquitin in water. Before running the simulation we have to add water and ions to the system. The crystal structure already contains some water molecules that could be resolved by X-ray. In some cases, retaining the crystal waters may be important for the protein structure, but this would require some manual editing of the files. Therefore we will just delete them in this case. Open wt.pdb with a text editor and delete all lines that include `HETATM ... HOH ...`.
The .pdb file has to be converted to .gro file. Both files contain the coordinates of the atoms.

```
gmx pdb2gmx -f wt.pdb -o wt.gro -p wt.top
```

The pdb2gmx command also generated the topology file (wt.top). The topology file contains the information about the force field. Each atom, bond, angle and dihedral in the structure is defined and the files from which the force field parameters are read is listed. The force field files are located in */usr/share/gromacs/top/oplsaa.ff/* directory. We will use this file when running minimization and simulations. Another file, *posre.itp* was generated. We will use this file for setting up the position restraints during the equilibration.
Before adding the solvent we have to define the box size. We will use a cubic shape of a box.

```
gmx editconf -f wt.gro -o wt_ed.gro -bt cubic -d 1.2
```

The -d option determines the box size: Edges of the box will be placed at least 1.2 nm from any atom of the protein.
Now we can add water molecules

```
gmx solvate -cp wt_ed.gro -cs spc216.gro -p wt.top -o wt_sol.gro
```

spc216.gro is a file with water coordinates from the gromacs library. We also provided topology file in the input, so that the water molecules are added the the topology. To obtain a more realistic environment, we will also add 0.15M NaCl. The genion command replaces an appropriate number of water molecules with ions.

```
gmx grompp -f em.mdp -c wt_sol.gro -o str.tpr -p wt.top

gmx genion -s str.tpr -o wt_ion.gro -conc 0.15 -p wt.top
```

1.2.2 Energy minimization

Starting the simulation from a strained initial structure can lead to very large forces. This may result in system collapsing.

The *em.mdp* file contains the input parameters for energy minimization. We first prepare the .tpr file, which is a binary file that contains the information about the coordinates, force field and the input parameters. Then we will run the minimization using the mdrun command.

```
gmx grompp -f em.mdp -c wt_ion.gro -p wt.top -o em_wt.tpr

gmx mdrun -deffnm em_wt -v
```

Energy can be extracted from the .edr file

```
gmx energy -f em_wt.edr -o ener_em_wt.xvg -xvg none
```

You can plot the energy in gnuplot

```
gnuplot

> plot "ener_em_wt.xvg" w l
```

1.3 Equilibration

Before we begin with the main MD run, we will equilibrate the solvent. We generated the system by inserting the protein into a box of pure solvent, the interface between the protein and the solvent is not yet optimized. Starting a normal simulation may again result in a collapse of the simulation. Therefore we will perform a short MD run with restrained positions of the protein atoms. The atoms are restrained to their initial positions, as defined in the *posre.itp* file, which was generated by the pdb2gmx command. The restraints impose an energetic cost for movement of the protein atoms from their initial position. This pervents large structural changes in the protein, while the solvent can equilibrate freely.

```
gmx grompp -f eq.mdp -c em_wt.gro -r em_wt.gro -p wt.top -o
                        eq_wt.tpr

gmx mdrun -deffnm eq_wt -v
```

1.4 Production run

Our system is now ready for the production run. Use the *run.mdp* file to run the simulation

```
gmx grompp -f run.mdp -c eq_wt.gro -p wt.top -o run_wt.tpr

gmx mdrun -deffnm eq_wt
```

The same procedure should be used to construct the system of the mutant.

1.5 Analysis

The trajectories are located in the *session2/ub/results* directory. If your run is finished, you can replace it with your own.
To visualise the molecule in vmd, both the trajectory (.xtc) and the coordinate (.gro) files have to be provided

```
vmd run_wt.xtc run_wt.gro
```

In case some atoms jump across the periodic boundaries to the other side of the box, you can edit the trajectory to keep the protein whole with the trjconv command.

```
gmx trjconv -s ../system_wt/run_wt.tpr -f run_wt.xtc -o
    pbc_run_wt.xtc -pbc mol
```

It is more convenient to remove the rotation and the translation while visually inspecting the conformational dynamics of the protein. Again we use the trjconv command.

```
gmx trjconv -s ../system_wt/run_wt.tpr -f pbc_wt.xtc -o
    reor_run_wt.xtc -fit rot+trans
```

Choose the protein backbone for fit and system for output.

To compare the dynamics of the two trajectories we will calculate the root mean square deviation (RMSD). RMSD is defined as

$$\mathrm{RMSD} = \sqrt{\sum_{i=1}^{N} \frac{(\mathbf{r}_{\mathrm{i}}(t) - \mathbf{r}_{\mathrm{i,ref}})^2}{N}}, \tag{1}$$

where i runs trough the selected atoms (we will calculate RMSD for backbone atoms), $\mathbf{r}_i = (x_i, y_i, z_i)$, $\mathbf{r}_{\mathrm{ref}}$ are the coordinates of the reference structure (the starting structure in our case) and N is the number of selected atoms. The coordinates in the trajectory have to be aligned to the reference structure. Gromacs will do that for us, therefore we can use the trajectory directly from the MD run.

```
gmx rms -f run_wt.xtc -s ../system_files/em_wt.tpr -o rms_wt.xvg
    -xvg none
```

The -xvg none option will make the output file readable for gnuplot. Choose "Backbone" both for fit and the RMSD calculation. Then do the same for the mutant.
Plot RMSDs with gnuplot. Did you expect the difference?

2 Coarse grained simulations with Martini

In the second part of the workshop we will construct coarse grained (CG) models of ubiquitin and polysorbate 80 (psb). We will perform simulations of psb molecules in water. We will use the Martini force field, a popular coarse-grained model, in which molecules consist of beads, that account for 3-5 atoms.

2.1 Coarse grained model of ubiquitin

Protein molecule can be easily converted into Martini CG model with *martinize.py* script (located in the *system_wt* directory). The martinize.py script can also be obtained from the Martini website (`http://cgmartini.nl/index.php/tools2/proteins-and-bilayers/204-martinize`)

```
python martinize.py -f wt.pdb -o cg_wt.top -x cg_wt.pdb
```

2.2 Coarse grained simulation of polysorbate 80

Polysorbate 80 (psb) is a nonionic surfactant. We will use a CG model of 8 psb molecules to investigate their interaction in water. The files are located in */L14/session2/psb/cg*

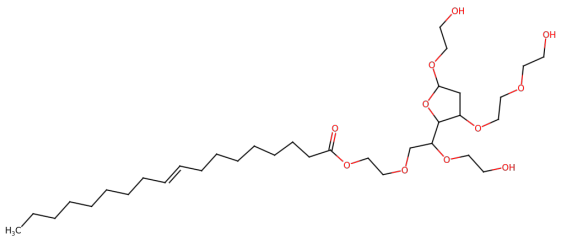

Figure 1: Structure of polysorbate 80

psb.pdb is the all atom structure of a single psb molecule. To convert it to a CG model use mappsb.py script[1]

```
python mappsb.py psb.itp psb.pdb psb.map cg_psb.gro
```

[1]Unfortunately there is an error in the mappsb.py file. Copy the mappsb.py file from system_files directory, this is the correct version.

We have provided the file with the force field parameters for psb (.itp), the .map file, which assigns the atoms in the all atom structure to CG beads and the all atom coordinate file (.pdb).
The mappsb.py script places beads at the center of mass of the atoms that are included in the bead. This may result in a non optimal structure. Therefore we will perform an energy minimization in vacuum before solvating the system.

```
gmx editconf -f cg_psb.gro -o cg_psb_ed.gro -bt cubic -d 1.2

gmx grompp -f em_vac.mdp -c cg_psb_ed.gro -p psb.top -o
                        em_psb.tpr

                gmx mdrun -deffnm em_psb -v
```

Now we will make 8 copies of psb.

```
python copymol.py em_psb.gro psb_8.gro
```

The number of psb molecules in the *pdb.top* file has to be changed by hand.

Adjust the box size and solvate the molecule. Use the *solvent.gro* as the coordinates for the solvent. Each water bead in the Martini model represents 4 molecules of water. Be careful to use the **-radius 0.21** option. This option tells Gromacs that our water beads are bigger than normal water molecules.

```
gmx editconf -f psb_8.gro -bt cubic -d 1.2 -o psb_8_ed.gro

gmx solvate -cp psb_8_ed.gro -cs solvent.gro -radius 0.21 -o
                psb_8_sol.gro -p psb.top
```

Ions are included in the box, therefore we do not have to add them separately. However their names were not assigned correctly to the .top file. We have of NA+ and CL- ions separately, while the solvate command grouped them together as ION. The ions are also not listed sequentially in the .gro file. The *rearange_gro.py* will arrange the ions in a sequential order and count them.

```
python rearange_gro.py psb_8_sol.gro
```

The script writes the number of ions. Add them to the topology. If the number of ions is highly unequal, change some of the ions names in .gro file to obtain an approximately neutral system.
The system is now ready for the minimization. As for ubiquitin, we will perform minimization, equilibration and the production simulation.

```
gmx grompp -f em.mdp -c psb_8_sol_ed.gro -p psb.top -o
                        em_psb_8.tpr
                gmx mdrun -deffnm em_psb_8 -v
 gmx grompp -f eq.mdp -c em_psb_8.gro -r em_psb_8.gro -p psb.top
                        -o eq_psb_8.tpr
                gmx mdrun -deffnm eq_psb_8 -v
gmx grompp -f run.mdp -c eq_psb_8.gro -p psb.top -o run_psb_8.tpr
                gmx mdrun -deffnm run_psb_8
```

How do the psb molecules interact? To color the hydrophobic and the hydrophilic parts of the molecules use the psbrep.tcl script in the vmd shell.

```
                        > source psbrep.tcl
```

3 Appendix: .mdp files for the ubiquitin tutorial.

em.mdp

```
integrator = steep nsteps = 2000
nstlist = 10
cutoff-scheme = verlet
vdw-type = cut-off
rvdw = 1.0
coulombtype = pme
rcoulomb = 1.0
```

eq.mdp

```
define = -DPOSRES
integrator = md
nsteps = 2500
dt = 0.002
nstlist = 10
rlist = 1.0
coulombtype = pme
rcoulomb = 1.0
cutoff-scheme = verlet
vdw-type = cut-off
rvdw = 1.0
tcoupl = v-rescale
tc-grps = protein non-protein
tau-t = 0.1 0.1
ref-t = 298 298
Pcoupl = Berendsen
tau-p = 1.0
compressibility = 1e-5 1e-5 1e-5 0 0 0
ref-p = 1.0
refcoord-scaling = all
nstenergy = 100
constraints = all-bonds
nstxtcout = 500
```

run.mdp

```
integrator = md
nsteps = 500000
dt = 0.002
nstlist = 10
rlist = 1.0
coulombtype = pme
rcoulomb = 1.0
cutoff-scheme = verlet
vdw-type = cut-off
rvdw = 1.0
tcoupl = v-rescale
tc-grps = protein non-protein
tau-t = 0.1 0.1
ref-t = 298 298
nstxtcout = 1000
nstenergy = 1000
constraints = all-bonds
nstxtcout = 1000
```