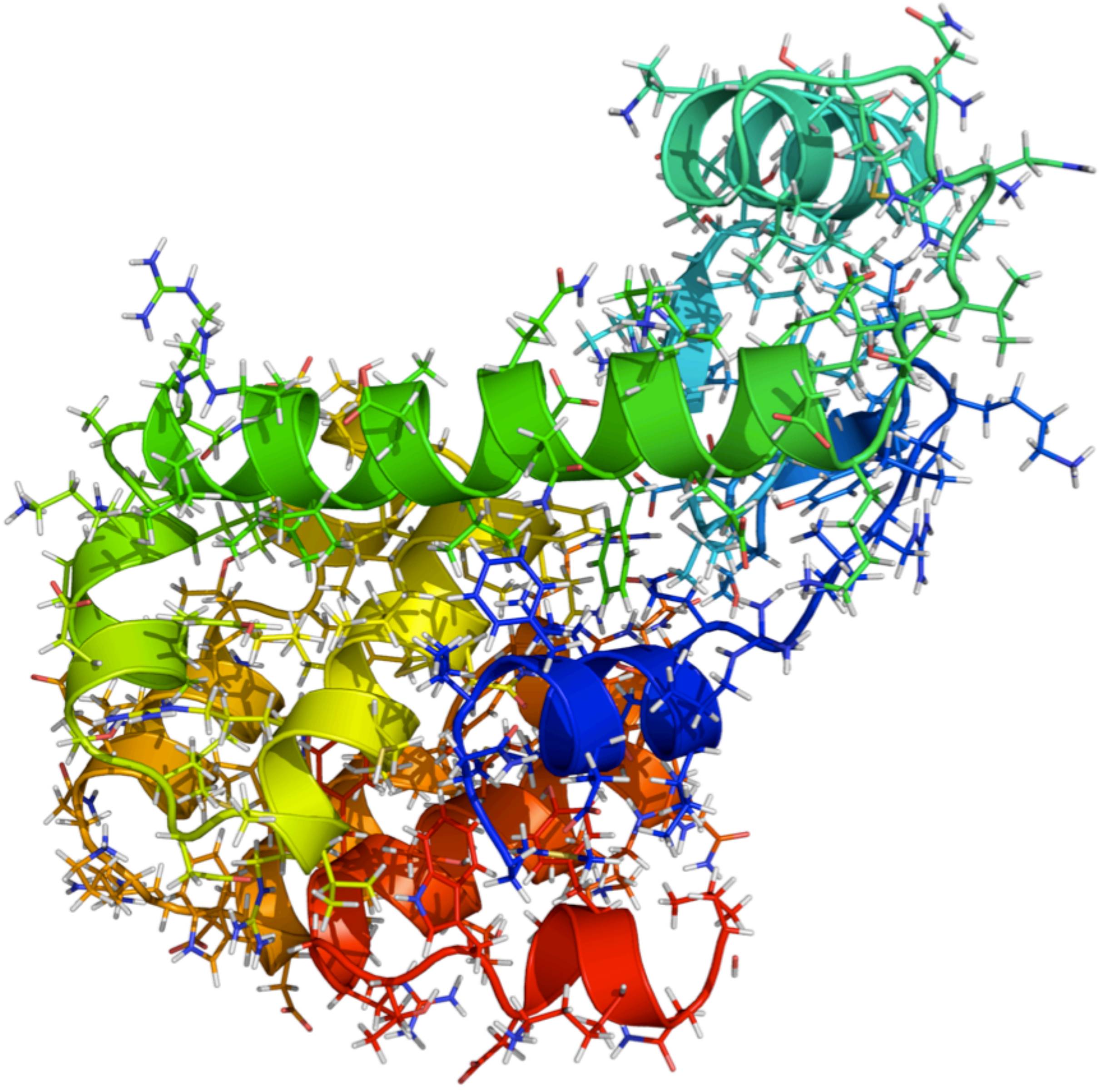
Molecular Simulation Methods with Gromacs



Hands-on tutorial

Introduction to Molecular Dynamics: Simulation of Lysozyme in Water

Background

The purpose of this tutorial is not to master all parts of Gromacs simulation and analysis tools in detail, but rather to give an overview and "feeling" for the typical steps used in practical simulations. Since the time available for this exercise is rather limited we can only perform a very short sample simulation, but you will also have access to a slightly longer pre-calculated trajectory for analysis.
In principle, the most basic system we could simulate would be water or even a gas like Argon, but to show some of the capabilities of the analysis tools we will work with a protein: Lysozyme. Lysozyme is a fascinating enzyme that has ability to kill bacteria (kind of the body's own antibiotic), and is present e.g. in tears, saliva, and egg white. It was discovered by Alexander Fleming in 1922, and one of the first protein X-ray structures to be determined (David Phillips, 1965). The two side-chains Glu35 and Asp52 are crucial for the enzyme functionality. It is fairly small by protein standards (although large by simulation standards :-) with 164 residues and a molecular weight of 14.4 kdalton - including hydrogens it consists of 2890 atoms (see illustration on front page), although these are not present in the PDB structure since they only scatter X-rays weakly (few electrons).
In the tutorial, we are first going to set up your Gromacs environments, have a look at the structure, prepare the input files necessary for simulation, solvate the structure in water, minimize & equilibrate it, and finally perform a short production simulation. After this we'll test some simple analysis programs that are part of Gromacs.

Setting up Gromacs & other programs

You will need a working Gromacs 4.6 installation on your system. If you're running at a supercomputing facility, chances are that it is already installed. The actual installation can depend on your system; you can find more detailed instructions at http://www.gromacs.org.
Gromacs works natively on a Unix-type system (such as Linux, or Mac OS X). You can use CygWin under windows, although this is a bit more cumbersome. Most of the packages needed to run Gromacs are available pre-packaged on most systems. Look at your rpm/dpkg package list if you're working working on a Linux system, MacPorts (http://macports.org) or Homebrew (http://mxcl.github.com/homebrew/) on a Mac, and the CygWin packaging system on Windows.
1. Gromacs uses FFTs (fast fourier transforms) for some interactions, in particular PME. While Gromacs comes with the slower FFTPACK libraries built-in, it is a very good idea to install the free and faster FFTW libraries from http://www.fftw.org. Make sure the package comes with floating point support enabled. If you need to install this from source, unpack the distribution tarball with **`tar -xzvf fftw-<version>.tar.gz`**,

go into the directory and issue **`./configure --enable-float`** (since Gromacs uses single precision by default) followed by `make` and finally become root and issue **`make install`**. If you do not have root access you can install it under your home directory, using the switch **`--prefix=/path/to/somewhere`** when you run the configure script.
2. Install VMD from http://www.ks.uiuc.edu/Research/vmd/ or alternatively PyMol from http://www.pymol.org.
3. Install the 2D plotting program Grace/xmgrace. Depending on your platform you might have to install the Lesstif libraries first (www.lesstif.org), and then get Grace from http://plasma-gate.weizmann.ac.il/Grace/.
4. To create beautiful secondary structure analysis plots, you can use the program DSSP from http://swift.cmbi.ru.nl/gv/dssp/. This is freely available for academic use but requires a license, so we will use the Gromacs built-in alternative **`my_dssp`** for this tutorial.

Download the tutorial material

Download the tutorial files from
http://www.gromacs.org/@api/deki/files/197/=gmx-tutorial-material.tar.gz to get started; you can use

```
wget http://www.gromacs.org/@api/deki/files/197/=gmx-tutorial-material.tar.gz
```

from a Unix command line. Then unpack with

```
tar xzvf gmx-tutorial-material.tar.gz
```

The PDB Structure

The first thing you need is a structure of the protein you want to simulate. Go to the Protein Data Bank at http://www.pdb.org, and search for “Lysozyme”. You should get lots of hits, many of which were determined with bound ligands, mutations, etc. We’ll use the structure **`1LYD.pdb`** here - download that pdb file.
It is always a good idea to look carefully at the structure before you start any simulations. Is the resolution high enough? Are there any missing residues? Are there any strange ligands? Check both the PDB web page and the header in the actual file. Finally, try to open it in a viewer like PyMOL/VMD and see what it looks like. In PyMOL you have some menus on the right where you can show (“S”) e.g. cartoon representations of the backbone!

Creating a Gromacs topology from the PDB file

When you believe you have a good starting structure the next step is to prepare it for simulation. As mentioned in the talks, we need to create a topology for this. Gromacs can do this automatically for you with the tool **pdb2gmx**. You can get some help on the available options with **pdb2gmx -h**. In this case, we will use a command like

```
pdb2gmx -f 1LYD.pdb -water tip3p
```

The program will ask you for a force field - for this tutorial I suggest that you use "OPLS-AA/L", and then write out lots of information. The important thing to look for is if there were any errors or warnings (it will say at the end). Otherwise you are probably fine. Read the help text of **pdb2gmx -h** and try to find out what the different files it created do, and what options are available for **pdb2gmx**! The important output files are

```
conf.gro  topol.top  posre.itp
```

If you issue the command several times, Gromacs will back up old files so their names start with a hash mark (#) - you can safely ignore (or erase) these for now.

Adding solvent water around the protein

If you started a simulation now it would be in vacuo. This was actually the norm 25-30 years ago, since it is much faster, but since we want this simulation to be accurate we want to add solvent.
However, before adding solvent we have to determine how big the simulation box should be, and what shape to use for it. There should be at least half a cutoff between the molecule and the box side, and preferably more. Due to the restricted time we'll be cheap here and only use 0.5nm. To further reduce the volume of the box we'll use a rhombic dodecahedron box:

```
editconf -f conf.gro -bt dodecahedron -d 0.5 -o box.gro
```

As an exercise, try a couple of different box sizes (the volume is written on the output!) and also other box shapes like "cubic" or "octahedron". What effect does it have on the volume?
The last step before the simulation is to add water in the box to solvate the protein. This is done by using a small pre-equilibrated system of water coordinates that is repeated over the box, and overlapping water molecules re- moved. The Lysozyme system will require roughly 6000 water molecules, which increases the number of atoms significantly (from 2900 to over 20,000). Gromacs does not use a special pre-equilibrated system for TIP3P water since water coordinates can be used with any model – the actual parameters are stored in the topology and force field. In Gromacs, a suitable command to solvate the new box would be

```
genbox -cp box.gro -cs spc216.gro -p topol.top -o solvated.gro
```

Solvent coordinates are taken from an SPC water system here, and the –p flag adds the new water to the topology. Before you continue it is a good idea to look at this system in PyMOL. However, PyMOL cannot read the Gromacs gro file directly. There are two simple ways to create a PDB file for PyMOL:
1. Use the ability of all Gromacs programs to write output in alternative formats, e.g.:

```
genbox -cp box.gro -cs spc216.gro -p topol.top -o solvated.pdb
```

2. Use the Gromacs `trjconv` program to convert it (use `-h` to get help on the options):

```
trjconv -s solvated.gro -f solvated.gro -o solvated.pdb
```

If you just use the commands like this, the resulting structure might look a bit strange, with water in a rectangular box even if the system is triclinic/octahedron/dodecahedron. This is actually fine, and depends on the algorithms used to add the water combined with periodic boundary conditions. However, to get a better-looking unit cell you can try the commands (backslash means the entire command should be on a single line):

```
trjconv -s solvated.gro -f solvated.gro \
     -o solvated_triclinic.pdb -pbc atom -ur tric

trjconv -s solvated.gro -f solvated.gro \
     -o solvated_compact.pdb -pbc atom -ur compact
```

We're not going to tell you what they do - play around and use the -h option to learn more :-)

Running energy minimization

The added hydrogens and broken hydrogen bond network in water would lead to quite large forces and structure distortion if molecular dynamics was started immediately. To remove these forces it is necessary to first run a short energy minimization. The aim is not to reach any local energy minimum, so e.g. 200 to 500 steps of steepest descent works very well as a stable rather than maximally efficient minimization. Nonbonded interactions and other settings are specified in a parameter file (**`em.mdp`**); it is only necessary to specify parameters where we deviate from the de- fault value, for example:

```
------em.mdp------
integrator      = steep
nsteps          = 200
nstlist         = 10
cutoff-scheme   = verlet
vdw-type        = cut-off
```

```
rvdw             = 1.0
coulombtype      = pme
rcoulomb         = 1.0
------------------
```

Comments on the parameters used:

We choose a standard cut-off of 1.0 nm, both for the **`neighborlist`** *generation and the coulomb and Lennard-Jones interactions.* **`nstlist=10`** *means it is updated at least every 10 steps, but for energy minimization it will usually be every step. Energies and other statistical data are stored every 10 steps (***`nstenergy`***), and we have chosen the more expensive Particle-Mesh-Ewald summation for electrostatic interactions. The treatment of non-bonded interactions is frequently bordering to religion. One camp advocates standard cutoffs are fine, another swears by switched-off interactions, while the third wouldn't even consider anything but PME. One argument in this context is that 'true' interactions should conserve energy, which is violated by sharp cut-offs since the force is no longer the exact derivative of the potential. On the other hand, just because an interaction conserves energy doesn't mean it describes nature accurately. In practice, the difference is most pronounced for systems that are very small or with large charges, but the key lesson is really that it is a trade-off. PME is great, but also clearly slower than cut-offs. Longer cut-offs are always better than short ones (but slower), and while switched interactions improve energy conservation they introduce artificially large forces. Using PME is the safe option, but if that's not fast enough it is worth investigating reaction-field or cut-off interactions. It is also a good idea to check and follow the recommended settings for the force field used.*

This mdp file uses the new interaction cut-off treatment introduced with Gromacs 4.6: the Verlet scheme. This replaces the old 'group' scheme and is not only more thermodynamically more correct, but it is also faster for proteins. Not all features (such as free energies, at the time of writing) are supported yet for this scheme, though.

Gromacs uses a separate preprocessing program **`grompp`** to collect parameters, topology, and coordinates into a single run input file (**`em.tpr`**) from which the simulation is then started (makes it easier to move it to a separate supercomputer). These two commands are

```
grompp -f em.mdp -p topol.top -c solvated.gro -o em.tpr
```

You should now see a note about the system having net charge. Normally this should be corrected by adding ions (salt) to the system, but we won't do that in this tutorial.
Run the resulting **`tpr`** file with (assuming that you run directly from the command line. On a cluster, you might need to submit this as a batch job):

```
mdrun -v -deffnm em
```

The **-deffnm** is a smart shortcut that uses "**em**" as the base filename for all options, but with different extensions. The minimization should finish in a couple of minutes. If you're running on the cluster, be sure to check the output of **em.log**.

Carefully equilibrating the water around the protein

To avoid unnecessary distortion of the protein when the molecular dynamics simulation is started, we first perform an equilibration run where all heavy protein atoms are restrained to their starting positions (using the file **posre.itp** generated earlier) while the water is relaxing around the structure. This should really be 50-100ps, but due to the time constraints we'll make do with 5ps (2500 steps) here. Bonds will be constrained to enable 2 fs time steps. Other settings are identical to energy minimization, but for molecular dynamics we also control the temperature and pressure with the v-rescale and Berendsen weak coupling algorithms. The settings used are:

```
------pr.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
------------------
```

For a small protein like Lysozyme it should be more than enough with 100 ps (50,000 steps) for the water to fully equilibrate around it, but in a large membrane system the slow lipid motions can require several nanoseconds of relaxation. The only way to know for certain is to watch the potential energy, and extend the equilibration until it has converged. To run this equilibration in Gromacs you execute

```
grompp -f pr.mdp -p topol.top -c em.gro -o pr.tpr
```

Run this with

```
mdrun -v -deffnm pr
```

Again, check the output **`pr.log`**.

Running the production simulation

The difference between equilibration and production run is minimal: the position restraints and pressure coupling are turned off, we decide how often to write output coordinates to analyze (say, every 100 steps), and start a significantly longer simulation. How long depends on what you are studying, and that should be decided before starting any simulations! For decent sampling the simulation should be at least 10 times longer than the phenomena you are studying, which unfortunately sometimes conflicts with reality and available computer resources. Start a simulation with a couple of thousand steps just to see what happens, but don't want for it to finish. Instead, copy the prepared **`run-long.xtc`** to **`run.xtc`**. This contains output data from a 10 ns simulation of this system (5 million steps, should take about 5-6 hours on an 8-core machine).

```
------run.mdp------
integrator      = md
nsteps          = 5000
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
------------------
```

The analysis normally only uses the compressed coordinates. Perform the production run as

```
grompp -f run.mdp -p topol.top -c pr.gro -o run.tpr
```

run this with

```
mdrun -v -deffnm run
```

Check the output again for any errors.

Analysis of the simulation

You will likely not have time to perform all these analyses, but pick one or a couple of them that sound interesting! At least for the longer ones you will get better results if you use the long trajectory from the web site, and there you can also find some plots that illustrate the results if you don't have time to perform the analysis yourself.

Deviation from X-ray structure

One of the most important fundamental properties to analyze is whether the protein is stable and close to the experimental structure. The standard way to measure this is the root-mean-square displacement (RMSD) of all heavy atoms with respect to the X-ray structure. GROMACS has a finished program to do this, as

```
g_rms -s em.tpr -f run.xtc
```

Note that the reference structure here is taken from the input before energy minimization. The program will prompt both for a fit group, and the group to calculate RMSD for – choose **`Protein-H`** (protein except hydrogens) for both. The output will be written to **`rmsd.xvg`**, and if you installed the Grace program you will directly get a finished graph with

```
xmgrace rmsd.xvg
```

The RMSD increases pretty rapidly in the first part of the simulation, but stabilizes around 0.19 nm, roughly the resolution of the X-ray structure. The difference is partly caused by limitations in the force field, but also because atoms in the simulation are moving and vibrating around an equilibrium structure. A better measure can be obtained by first creating a running average structure from the simulation and comparing the running average to the X-ray structure, which would give a more realistic RMSD around 0.16 nm.

Comparing fluctuations with temperature factors

Vibrations around the equilibrium are not random, but depend on local structure flexibility. The root-mean-square- fluctuation (RMSF) of each residue is straightforward to calculate over the trajectory, but more important they can be converted to temperature factors that are also present for each atom in a PDB file. Once again there is a program that will do the entire job:

```
g_rmsf -s run.tpr -f run.xtc -o rmsf.xvg -oq bfac.pdb
```

You can use the group **`C-alpha`** to get one value per residue. The overall agreement should be quite good (check the temperature factor field in this and the input PDB file, or cheat and open the file **`rmsf.pdf`**), which lends further credibility to the accuracy and stability of the simulation.

Secondary structure (Requires DSSP)

Another measure of stability is the protein secondary structure. This can be calculated for each frame with a pro- gram such as DSSP. If the DSSP program is installed and the environment variable DSSP points to the binary (`do_dssp -h` will give you help on that), the GROMACS program `do_dssp` can create time-resolved secondary structure plots. Since the program writes output in a special xpm (X pixmap) format you probably also need the GROMACS program xpm2ps to convert it to postscript:

```
do_dssp -s run.tpr -f run.xtc
```

```
xpm2ps -f ss.xpm -o ss.eps
```

Use the group "protein" for the calculation. The DSSP secondary structure definition is pretty tight, so it is quite normal for residues to fluctuate around the well-defined state, in particular at the ends of helices or sheets. For a (long) protein folding simulation, a DSSP plot would show how the secondary structures form during the simulation. If you do not have DSSP installed, you can compile the Gromacs built-in replacement **`my_dssp`** from the **`src/contrib`** directory.

Distance and hydrogen bonds

With basic properties accurately reproduced, we can use the simulation to analyze more specific details. As an example, Lysozyme appears to be stabilized by hydrogen bonds between the residues GLU22 and ARG137, so how much does this fluctuate in the simulation, and are the hydrogen bonds intact? To determine this, first create an index file with these groups as

```
make_ndx -f run.gro
```

At the prompt, create a group for GLU22 with “r 22”, ARG137 with “r 137”, and then “q” to save an index file as `index.ndx`. The distance and number of hydrogen bonds can now be calculated with the commands

```
g_dist -s run.tpr -f run.xtc -n index.ndx -o dist.xvg
```

```
g_hbond -s run.tpr -f run.xtc -n index.ndx -num hbnum.xvg
```

In both cases you should select the two groups you just created. Two hydrogen bonds should be present almost throughout the simulation.

Viewing the trajectory

A normal movie uses roughly 30 frames/second, so a 10-second movie requires 300 simulation trajectory frames. To make a smooth movie the frames should not be more 1-2 ps apart, or it will just appear to shake nervously. VMD can read **`.xtc`** trajectory files directly if you open them directly after you opened a corresponding **`.gro`** configuration file, such as the starting configuration. If you’re using PyMol, export a short trajectory from the first 2.5 ns in PDB format (readable by PyMOL) as

```
trjconv -s run.gro -f run.xtc -e 2500.0 -o movie.pdb
```

Choose the protein group for output rather than the entire system. If you open this trajectory in PyMOL you can immediately play it using the VCR-style controls on the bottom right, adjust visual settings in the menus, and even use photorealistic ray-tracing for all images in the movie. With MacPyMOL you can directly save the movie as a quicktime file, and on Linux you can save it as a sequence of PNG images for assembly in another program. Rendering a movie only takes a few minutes, and the final product lysozyme.mov is also available in the same location you found the other files.

Conclusion

Well, the main point of this exercise was to get your feet wet, and at least an idea of how to use Gromacs in practice. There are much more advanced analysis tools available, not to mention a wealth of simulation options. We’ll go through some of those during the rest of the workshop, but we’d also like to recommend the Gromacs manual which is available from our website: It goes quite a bit beyond just being a program manual, and explains many of the algorithms and equations in detail!