GROMACS TUTORIAL

Penetratin Peptide

0. Introduction

GROMACS is a major free, open-source, and fast code developed for Molecular Dynamics (MD) simulations. Its continuous updates (1 major release/year), speed, efficiency and flexibility, along with the inbuilt availability of force fields specific for proteins, make it one of the most popular choices for biomolecular simulations. (http://www.gromacs.org/).

To see what gromacs version is installed on your PC (without the $ sign!):

$ gmx –version

All gromacs commands can be used with the syntax:

$ gmx <command>

And all the commands have a help message that can be displayed using the option "-h", e.g.:

$ gmx pdb2gmx -h

For basic and advanced GROMACS tutorial, please visit: http://www.mdtutorials.com/gmx/

0.1 Files and suggested directory tree

You will need the following files to run a simulation:

1. *Atom coordinates:* ***.pdb** (or *.gro)
2. *Topology file:* ***.top**
3. *MD parameters:* ***.mdp**

The pdb file (penetratin.pdb) and *.mdp files are required input files, i.e. you have to make sure you have them and tell GROMACS where to find them! The topology (topol.top) file on the other hand will be generated by GROMACS once you select the forcefield. We suggest you create a new folder for each simulation you run, put all the necessary files into the simulation folder itself, and let GROMACS write into that same folder. For example:

00-simulation1/
penetratin.pdb
em.mdp
nvt.mdp
npt.mdp
md.mdp
topol.top

Keep in mind that GROMACS is a command-line program: if you don't specify the full paths to the required input files, but only the file name (example call: $ **gmx make_ndx -s struct.tpr -o index.ndx**), GROMACS will expect the input file (struct.tpr) to be in the folder you're calling it from, and will write the output file (index.ndx) into the same current folder. If struct.tpr is not in the current folder, GROMACS will throw an error and fail! GROMACS error messages are pretty explicit, so chances are **READING THE ERROR** is enough to troubleshoot most issues (e.g. if a required file is missing).

0.2 Molecule: Cell-Penetrating Peptide

The discovery of Cell-Penetrating Peptides (CPPs) represents an important breakthrough for the delivery of large cargo molecules or nanoparticles for several clinical applications. A main feature of CPPs is the ability to penetrate the cell membrane at low micromolar concentrations in vivo and in vitro, without binding any chiral

Prof. Marco A. Deriu
Biomeccanica Multiscala

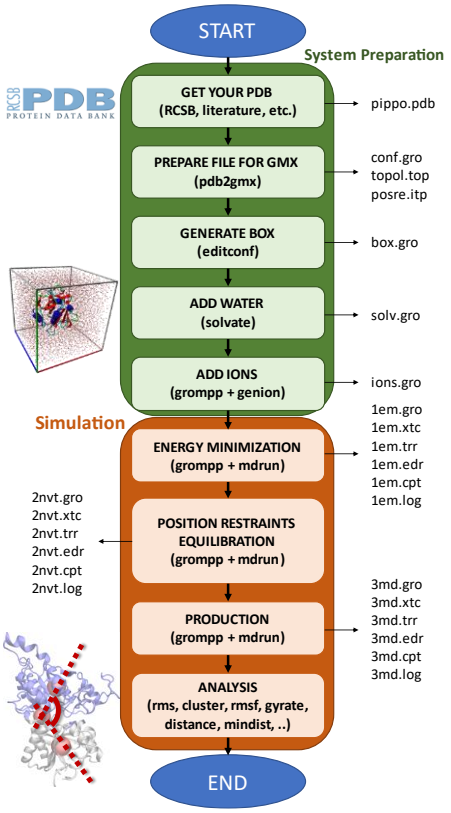
receptors and without causing significant membrane damage. This ability offers significant therapeutic potential, as targeting areas normally difficult to access for drugs.
TAT peptide and Drosophila Antennapedia homeodomain-derived penetratin peptide (pAntp) are the most extensively studied CPPs. In particular, the pAntp is a 16-residues long cationic peptide derived from the third helix of the homeodomain of the Drosohila transcription factor Antennapedia. This amphipatic CPP, largely unstructured in solution, is positively charged at neutral pH (since it contains four lysine and three arginine residues). Interestingly, the pAntp is able to cross biological membranes and enter a hydrophobic environment upon interaction with negatively-charged molecules, like phosphatic acid (PA) or phosphatidylserine (PS). However, mechanisms by which pAntp comes into the cells have not been completely understood. Proposed mechanisms of pAntp cellular uptake hypothesize direct crossing of the peptide through the membrane at low peptide concentrations (1µM) and an endocytotic pathway at high concentrations (10µM). Several studies have suggested that pAntp amphiphilicity may not be enough to drive the membrane penetration, indicating instead tryptophan as key player. Replacement of thryptophan by phenylanine resulted in a loss of penetration activity when interacting with membranes and bicelles. Moreover, a recent computational work has characterized the binding mode of pAntp-DPPC bilayers, proposing arginine, lysine and tryptophan as driving the penetration mechanism.
This extraordinary ability of CPPs to penetrate cell membranes has brought to designate them as perfect functionalization molecules for drug delivery systems.
For example, pAntp might be employed to decorate Magnetic Nanoparticles (MNPs), thus combining their fascinating physico-chemical properties with a cell-penetrating ability to design novel effective therapeutic strategies as well as innovative biotechnology methodologies. Size, biocompatibility and excellent magnetic properties, have made MAG and Silica-coated MAG the object of a remarkable amount of research in the last decade and numerous biomedical applications have been reported. Recently, MNPs combined with magnetic fields were used to enable cell positioning under non-permissive conditions, local gene therapy and/or optimization of MNP- assisted lentiviral gene transfer.
Functionalization strategies comprise grafting with organic molecules, including small organic biomolecules such as CPPs, and/or coating with an inorganic layer (e.g., silica).
Design and properties prediction of functionalization strategies may be addressed by computational molecular modelling. In this context, Kubiak-Ossowska and coworkers have recently employed computational modelling to investigate the adsorption of TAT peptides onto three silica surface models. This work has suggested that TAT-Silica adsorption mechanism is driven by electrostatic and hydrophobic interactions mainly involving arginine residues and the nanoparticle surface.

0.3 MD simulations

Briefly, the general workflow of an MD simulation with GROMACS is divided in the following steps (see also "*MD_FlowChart.pdf"*):

1) ***System Preparation***
   - Retrieve starting structure (e.g. RCSB)
   - GROMACS structure conversion (from pdb to gro*)
   - Topology (and position restraints) generation
   - System box generation and addition of water and ions
2) ***Energy Minimization***
3) ***Simulation***
   - (Equilibration: Molecular Dynamics Simulation normally with position restraints)
   - Production: Molecular Dynamics Simulation
4) ***Analysis of the Simulation***

*note that both **.gro** and **.pdb** files both contain atomic coordinates and only differ for minor aspects:
- .gro coordinates are in nm, whereas *.pdb coordinates in Å
- .gro file can contain also atoms' velocities

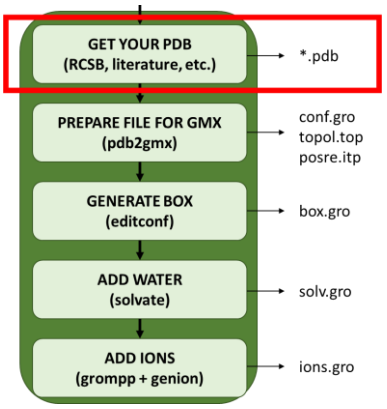


Prof. Marco A. Deriu
Biomeccanica Multiscala

GROMACS can work with both types of files and does not strictly need the .gro file format for most operations. It can be however handy to have a .gro file, for example, to quickly extract the box vector.

More on that later!

1. STEP 1 – PDB conversion and system preparation



The file of interest for the present tutorial is called **penetratin.pdb**. Using a visualization software (for example VMD, PyMol, UCSF Chimera, Schrödinger Maestro, MOE, …) you can view and rotate the structure. We'll use VMD for this tutorial.
Create a new folder (e.g. **00-simulation**) and put the penetratin.pdb file inside.

Open a terminal session, *cd* to the folder and type:

$ vmd penetratin.pdb

You can view the structure in 3D. Also, by heading to *Graphics -> Representations* you can change the *Drawing Method* to *NewCartoon* to view the secondary structure. Go ahead and play with the drawing methods, colours and options to tweak the visualization a bit! Once you're done, close vmd.

Let's now have a look at the actual pdb file. To do so, open penetratin.pdb using a text editor (nano, gedit, vi, kate, less,…), for example:

$ nano penetratin.pdb

You will see that the pdb is essentially a space-separated text file organized into columns. Briefly:

- The first column defines the row type (e.g. REMARK tells you that this row is a comment, ATOM tells you that this row contains actual atomic coordinates, etc..).
- Columns 7, 8 and 9 contain the x,y,z coordinates of the atoms of the system.

For more information: https://www.cgl.ucsf.edu/chimera/docs/UsersGuide/tutorials/pdbintro.html.

1.1 Make index files (gmx make_ndx)

We now introduce a very powerful tool in GROMACS: index files. Index files are created starting from an atomic system (e.g. a pdb file), and split atoms into specific groups. These can be useful to restrict operations and analyses only to specific parts of the system, for example, if you want to see fluctuations of the alpha-Carbons only, you can select that specific group in the analysis tool.

To have detailed information about the command to make index files, just type:

$ gmx make_ndx -h

Let's first create an index file (it will not be created by default when you launch the simulation!):

$ gmx make_ndx -f penetratin.pdb -o index.ndx

You will see that make_ndx will read the pdb file (option "-f"), create some default groups and then wait for your input. You can manually add groups by typing the atoms manually (we will see the selection syntax later on…), or simply accept the default groups and save them by typing *q* and hitting *Enter*. The final index file (index.ndx) will be saved with the name chosen in the option "-o".

1.2 Edit molecular configuration (gmx editconf)

As mentioned, the previous groups obtained with *gmx make_ndx* can be used to tell GROMACS to do operations on a specific subset of atoms only. Let's say for example I want a pdb file containing only the alpha Carbons of

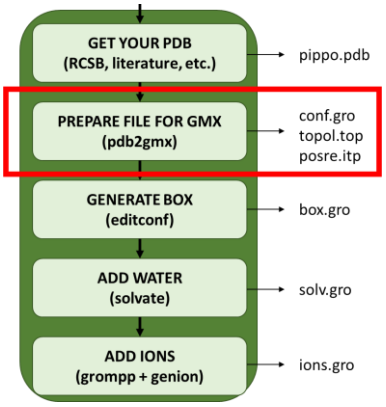
my starting penetratin.pdb structure. I can achieve this by calling *editconf* (which, as the name suggests, is an editor for molecular configurations) and also including the *index.ndx* file as input:

$ gmx editconf -f penetratin.pdb -o calpha.pdb -n index.ndx

In simpler words, this command is saying:
"Call the editconf utility, read the penetratin.pdb and the index.ndx files, and write the result to calpha.pdb"

Indeed, *editconf* will prompt for a group. Choose C-alpha by typing *3* and hitting *Enter*. *Editconf* will confirm the selection and write atoms belonging to the C-alpha group to the file *calpha.pdb* (check it by opening the file with a text editor!). Easy, right?

1.3 Prepare file for Gromacs (gmx pdb2gmx)



Now, let's get our hands dirty. As you might remember, we have to generate a topology for the system and convert it to the .gro file format. It is a good idea to backup our starting file first (it's easy to mess things up), so just make a copy in the current folder:

$ cp penetratin.pdb backup.pdb

The tool for conversion is pdb2gmx. As always, you can use the -h flag to print some help:

$ gmx pdb2gmx -h

The main options of the tool are:
-f : input file (coordinates, so .pdb or .gro)
-p: topology output file (a text file, ending in .top)
-o: structure output file (coordinates after processing, again .pdb or .gro)

Let's call the tool by issuing the following command:

$ gmx pdb2gmx -f penetratin.pdb -ignh -heavyh -i penetratin_posre.itp -p penetratin.top -o penetratin.gro

We added a couple of options here. Briefly, they are:
-ignh : it stands for "ignore hydrogens", so it will ignore all the hydrogens in the input coordinate file
-heavyh: makes hydrogen atoms heavy to reduce oscillation frequency
-i : write position restraint include file for topology

To build the topology, you will need to select a forcefield. This selection is interactive, as you can see from your terminal. Selecting a forcefiled can be thought of as "choosing which parameters to use for the classical mechanical model of the atomic system". Different forcefields contain parameters that work best for different types of molecules and simulation conditions. The main FFs are:

1: AMBER03 protein, nucleic AMBER94 (Duan et al., J. Comp. Chem. 24, 1999-2012, 2003)
2: AMBER94 force field (Cornell et al., JACS 117, 5179-5197, 1995)
3: AMBER96 protein, nucleic AMBER94 (Kollman et al., Acc. Chem. Res. 29, 461-469, 1996)
4: AMBER99 protein, nucleic AMBER94 (Wang et al., J. Comp. Chem. 21, 1049-1074, 2000)
5: AMBER99SB protein, nucleic AMBER94 (Hornak et al., Proteins 65, 712-725, 2006)
6: AMBER99SB-ILDN protein, nucleic AMBER94 (Lindorff-Larsen et al., Proteins 78, 1950-58, 2010)
7: AMBERGS force field (Garcia & Sanbonmatsu, PNAS 99, 2782-2787, 2002)
8: CHARMM27 all-atom force field (CHARM22 plus CMAP for proteins)
9: GROMOS96 43a1 force field
10: GROMOS96 43a2 force field (improved alkane dihedrals)
11: GROMOS96 45a3 force field (Schuler JCC 2001 22 1205)

12: GROMOS96 53a5 force field (JCC 2004 vol 25 pag 1656)
13: GROMOS96 53a6 force field (JCC 2004 vol 25 pag 1656)
14: GROMOS96 54a7 force field (Eur. Biophys. J. (2011), 40,, 843-856, DOI: 10.1007/s00249-011-0700-9)
15: OPLS-AA/L all-atom force field (2001 aminoacid dihedrals)

We will choose the *AMBER99SB-ILDN* forcefield, so select it by typing *6* and hitting *enter*. *pdb2gmx* will then ask you for the water model to use for simulations. We will select *TIP3P*, so enter *1*.

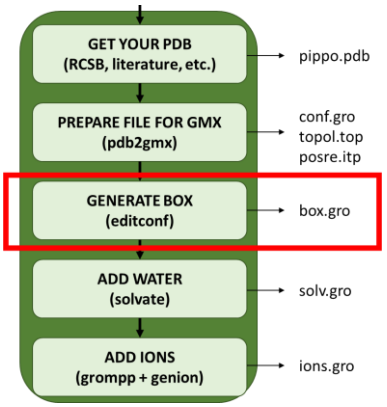
Pdb2gmx should have created all the necessary files we've been talking about in the current folder. After listing files with

$ ls

You should see:
calpha.pdb (from the previous exercise)
index.ndx
penetratin.gro
penetratin.pdb
penetratin.top
penetratin_posre.itp

Have a look at the *.gro file with a text editor to see the differences with the pdb file!

1.4 Generate Box (gmx editconf)



Now, we defined the parameters and water model for the simulation. Now we have to actually put the system inside a box and fill it with water. And guess what tool we are going to use to edit a molecular configuration? Of course, editconf:

$ gmx editconf -f penetratin.gro -bt cubic -d 0.8 -o box.gro

This command is saying "hey editconf, get the atoms from *penetratin.gro*, put them into a cubic box *(-bt cubic*) with a size so that the minimum distance of the atoms from the edge of the box is 0.8 nm *(-d 0.8*). Save the result into *box.gro*".

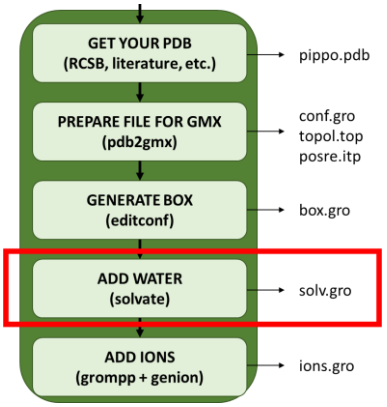
Now we have a small peptide inside a cubic box and with all the necessary simulation parameters. If you want to see the box around the peptide, open the box gro with VMD:

$ vmd box.gro

And type "pbc box" in the vmd terminal or in the Tk console.

Prof. Marco A. Deriu
Biomeccanica Multiscala

1.5 Add water (gmx solvate)



We now have to fill the box with actual water. This is done by *gmx solvate*:

$ gmx solvate -cp box.gro -cs spc216.gro -p penetratin.top -o solvated.gro

This means: "take my box.gro as the solute (*-cp*) and fill it with the SPC216 water model *(-cs spc216.gro*). Write the new solvated system to solvated.gro *(-o solvated.gro).* Also, update the topology file *(-p penetratin.top)*."
Wait, we specified a file (spc216.gro) that is not in the current folder. How did gromacs find it? Let's see if you can find out! In the meantime, let's check our topology after this step: using the option -p in the previous command, gromacs updated the molecules of your system in the topology file (penetratin.top):

$ tail -10 penetratin.top

This will print the last 10 lines of the file:

```
#include "amber99sb-ildn.ff/ions.itp"

[ system ]
; Name
Penetratin-Silica Umbrella simulation in water

[ molecules ]
; Compound        #mols
Protein_chain_B     1
SOL              3015
```

See how gromacs added an appropriate amount of solvent molecules (3015 molecules in this case)? Now let's have an actual look at the solvated system:

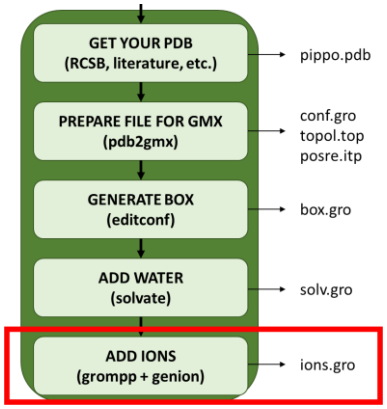
$ vmd solvated.gro

Again, play with the drawing method and colour to make things clearer. As you can see, there are quite a bit of water molecules all around, which means it will take more time to simulate. Ideally, we would like to just surround our peptide with an appropriate layer of water, but as you can see the corners of the cubic box are much farther away from the peptide than the sides and are filled with "useless" water. The ideal shape would be a box whose faces are equidistant to the surface of the protein everywhere, i.e. a sphere. In the finite world of gromacs, the discrete geometrical object closest to a sphere is a dodecahedron. So let's try again with a dodecahedron box. There is one small problem: in the previous steps, when we solvated the cubic box, gromacs has updated the topology with that specific amount of water molecules: indeed, the last line of penetratin.top should read something like:
SOL 3015
If we are going to create a new box and solvate it, we need to revert the topology first, so that it does not include the old amount of water anymore. Simply delete this last line with any text editor before proceeding. Now that we removed the old amount of water from the topology, let's recreate the box and re-solvate it (NOTE THAT WE ARE USING DIFFERENT OUTPUT FILE NAMES TO NOT OVERWRITE THE OLD ONES):

$ gmx editconf -f penetratin.gro -bt dodecahedron -d 0.8 -o box_dodecahedron.gro
$ gmx solvate -cp box_dodecahedron.gro -cs -p penetratin.top -o solvated_dodecahedron.gro

Now have a look at the penetratin.top file again: how is the number of water molecules now compared to the amount from the cubic box? Did we make things better? Note that the distance constraint of having a minimum distance between the edges of the box and the peptide of 0.8 nm is still maintained!

1.6 Add ions (gmx grompp, gmxgenion)



We are almost ready! As a last step, it makes sense to add a proper amount of Na+ and Cl- ions to the water we just added, for two main reasons:

1) Make the system neutral should it be charged (by adding the proper amount of counterions)
2) Simulating physiological salt concentrations (circa 0.15M)

GROMACS has a specific tool to do that, which is called *genion*. The only caveat is that genion needs a ***.tpr** input file to work. You can think of the tpr file as the union of a ***.gro** (coordinates), a ***.top** (topology) and an ***.mdp** (simulation parameters).

****

IMPORTANT

In other words, to run any sort of simulation, you obviously need:

1) *A coordinate file, like a *.gro, which contains the system to be simulated (WHAT SYSTEM)*
2) *A topology, like a *.top, which contains the parameters to model such a system (WHAT MODEL)*
3) *A run parameter file, like an *.mdp, which contains instructions for the simulation, like the timestep to use, the cutoffs, the total simulation time, etc. (HOW TO SIMULATE)*

*These three basic ingredients of any simulation (***.gro**, ***.top**, ***.mdp**) can then be combined into a single file (the *.tpr file), so that the engine carrying out the simulations (which is gmx mdrun) has all the necessary information in one single file.*

****

To see how this works, let's create a dummy *.mdp file, and leave it empty for now:

$ touch dummy.mdp

At this point we can try to create our *.tpr file, since we have all the necessary ingredients (the *.gro, *.top and *.mdp files). ***Gmx grompp*** (which stands for gromacs pre-processor) does that:

$ gmx grompp -f dummy.mdp -c solvated_dodecahedron.gro -p penetratin.top -o ions.tpr

As expected, *grompp* takes the three input files and generates a single output file. In this case, it will contain default simulation parameters since our dummy.mdp file was empty, but we can use this file to fill the box with genion!

*NOTE: grompp will check that the topology *.top and coordinates *.gro indeed contain the same number of atoms (i.e. the topology represents the system in the coordinate file). If this is not the case, it will throw an error saying that the number of atoms doesn't match, and it won't generate the *.tpr. If this happens to you, you likely messed something up during solvation and the number of water molecules in the topology don't correspond to the ones in the coordinate file. It's best, in this case, to delete the SOL line in the topology, and start over with the solvation.*

Now that we have the tpr, we can fill the solvated dodecahedron with ions at the proper concentration:

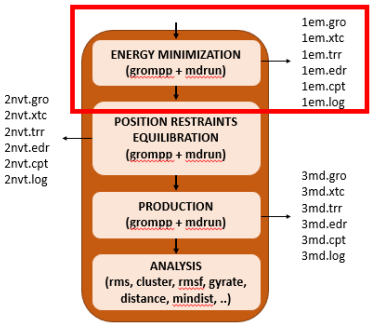
$ gmx genion -s ions.tpr -p penetratin.top -o box_ions.gro -conc 0.15 -neutral

Here we said: "*genion*, read the tpr file containing the solvated box *(-s ions.tpr*) and try to fill it with ions so that the system is electrostatically neutral *(-neutral*) and the total salt concentration in 0.15 mol/L *(-conc 0.15*). While doing so, also update the topology (*-p penetratin.top*), and save the new system with water and ions into a new file *(-o box_ions.gro*)."

Genion will ask you for the group of solvent molecules to insert the ions into, so we select group 13 (SOL) and hit enter. Note that genion also takes the topology file as input, and updates it with the added amount of NA and CL molecules (open the topology and check!).

At this point, we're ready for the simulation. Our system is parametrized with a forcefield, is placed into a proper box, solvated with water and contains the right amount of ions.

2. STEP 2 – Energy Minimization



The first step for the MD pipeline is Energy Minimisation. This is needed because there might be atoms (both of the protein and the solvent) placed in non-ideal positions, resulting in high forces distorting the molecule. So, i*f we started straight away with the simulation, it would likely crash!*

We will use the ***steepest descent algorithm*** to perform the energy minimization.

To keep things organized, we will do the EM in its own subfolder, so let's create it:

$ mkdir 00-em

Now, move the supplied em.mdp file into this new folder, and open it. This file contains the instructions to perform energy minimization. In particular, the most relevant lines are:

integrator *= steep*
…
nsteps *= 1000*
…
emtol *= 1000*

Which mean respectively:

- ***steep***: use the steepest descent algorithm (actually, it isn't strictly an integrator as for the md code)
- do the energy minimization for a maximum of 1000 steps (***nsteps***) OR until the maximum force drops below 1000 kJ*mol^{-1}*nm^{-1} (***emtol***).

Again, we are in a situation where we have everything we need for a simulation (coordinates, topology and parameters) and we need to merge them into a single file. What do we do? You guessed it, we generate a *.tpr with grompp (don't forget to cd to the new 00-em directory first!):

$ gmx grompp -f em.mdp -c ../box_ions.gro -p ../penetratin.top -o em.tpr

NOTE: the coordinate and topology files are not in the current folder anymore, since we cd'ed to 00-em, so we have to tell grompp that they are located one directory above the current one (../box_ions.gro and ../penetratin.top).

Know we can feed the unified input file (em.tpr) into the engine doing the actual calculations, which is ***gmx mdrun***. In genral, mdrun takes many parameters, such as:

-s input file (.tpr).
-o output file (trajectory, not important in minimization)
-e output file (energy)

If you noticed though, we called the tpr file with the name *em.tpr*. The mdrun tool has a very useful option called *-deffnm*, which stands for "*define file names*": with this option, you can tell *mdrun* that all the required input files (e.g. the ***.tpr**) and all the produced output files (e.g. ***.xtc, *.edr, *.gro**, …) will all have the same name (with the proper extension). This means that you don't have to tell mdrun explicitly all the input and output file names. For example, if you set:
… -deffnm em …
mdrun will:

a) Look for **em.tpr** as the input file
b) Name the output files **em.xtc**, **em.trr**, **em.gro**, **em.edr**, etc…

Prof. Marco A. Deriu
Biomeccanica Multiscala

This is way quicker than specifying all the file names explicitly, and this is what we will do. Run the minimization:

$ gmx mdrun -deffnm em -v

Which means: "run the calculations on the *em.tpr* input file and save all the outputs with the root name *em (-deffnm em)*. Also, tell me what you're doing while you're running (verbose mode, *-v*)".

GROMACS will start the energy minimization calculation and will stop once the force drops below the *emtol* setting or once the maximum number of steps is reached (*whatever happens first!*).

2.1 Analysing Energy Minimization Data (gmx energy, xmgrace)

Once the energy minimization has finished, we have to check that our system actually reached a low potential energy and plateaued at that level, i.e. that nothing went wrong during the minimization, that the number of steps was sufficient to reach a minimized potential energy, and that we can proceed further. To view the potential energy as a function of the number of minimization steps, we need to use ***gmx energy***, which is a tool that extracts energy data from the .edr file and prints it in ASCII format (for example as a tab-separated xy data file) which can then be plotted. Let's extract the energies:

$ gmx energy -f em.edr -o potentialE.xvg

Which again means: "Read the energy file produced during minimization *(-f em.edr*), and write output data into an xvg text file *(-o potentialE.xvg*)"
gmx energy will prompt you for what you wish to extract and save as a text file. We are interested in the potential energy, so in the interactive prompt we'll enter *10* and hit *enter* twice to confirm.

If you now *ls* the current folder, you'll find the new potentialE.xvg file.

Tip: use ls -l to list the contents in a column, and use ls -lh to make file sizes human-readable!

To plot the data we'll use Grace, which is a simple tool for data plotting:

$ *xmgrace potentialE.xvg*

You should see that the system reached an energy minimum and plateaued at that value (it should be in the order of 10^{-5} – 10^{-6} at least). If everything looks ok, close Grace.

****

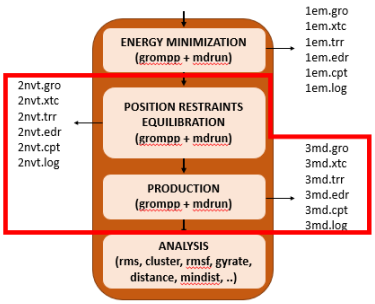
IMPORTANT

Do note that on this plot the default label for the x axis is "Time (ps)". Of course, when using a Gradient Descent algorithm as we did, as opposed to an integrator, the concept of timestep is pretty meaningless. The x axis, in this case, does in fact represent the number of steps, and not time!!

****

Now everything is energy-minimized, which means that the most obvious clashes between atoms and the poorest atomic positions (e.g. inconsistent bond stretches, etc.) have been reasonably solved. We are ready to run a Molecular Dynamics simulation!

3. STEP 3 – Molecular Dynamics



Input files required to run a Molecular Dynamics (MD) simulation are again a molecular structure, a topology and a files with simulation settings:

1. (minimized) structure file (minimized.gro)
2. topology file (penetratin.top)
3. molecular dynamics parameters file (posre.mdp/md.mdp)

3.1 Equilibration with position restraints

EM ensured that we have a reasonable starting structure, in terms of geometry and solvent orientation. To begin real dynamics, we must equilibrate the solvent and ions around the protein. If we were to attempt unrestrained dynamics at this point, the system may collapse. The reason is that the solvent is mostly optimized within itself, and not necessarily with the solute. It needs to be brought to the temperature we wish to simulate. Remember that *posre.itp* file that pdb2gmx generated a long time ago? We're going to use it now. The purpose of posre.itp is to apply a position restraining force on the heavy atoms of the protein (anything that is not hydrogen). Movement is permitted, but only after overcoming a substantial energy penalty. The utility of position restraints is that they allow us to equilibrate our solvent around our protein, without the added variable of structural changes in the protein. The origin of the position restraints (the coordinates at which the restraint potential is zero) is provided via a coordinate file passed to the -r option of grompp.

We'll use the file "posre.mdp" which contains the parameters for a molecular dynamics simulation with position restraints. Have a look to posre.mdp:

$ less posre.mdp

An extensive explanation of the simulation parameters will be given during the lectures of the BM course. Note that the comprehensive list of all simulation parameters is available at https://manual.gromacs.org/documentation/current/user-guide/mdp-options.html.

Then, create a folder to simulate with position restraints:

$ mkdir 2posre; cd 2posre

And create the tpr file and run the simulation (same steps done for the energy minimization):

$ gmx grompp -f ../mdp/posre.mdp -c ../1em/1em.gro -r ../1em/1em.gro -p ../penetratin.top -o 2posre.tpr
$ gmx mdrun -deffnm 2posre -v

Analyze system temperature with gmx energy and xmgrace:

$ gmx energy -f 2posre.edr -o temperature.xvg
$ xmgrace temperature.xvg

Is the system well equilibrated in terms of temperature? And in terms of pressure??

3.2 Production without position restraints

Upon completion of the equilibration phase, the system is now well-equilibrated at the desired temperature and pressure. We are now ready to release the position restraints and run production MD for data collection.

Return at the root of your working directory, create a folder for the molecular dynamics production and enter in it:

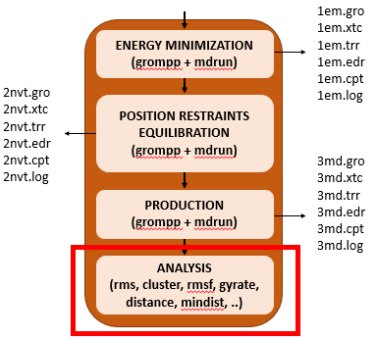
$ cd .. ; mkdir 3md; cd 3md

Create the tpr and run the simulation: we will use the file “md.mdp” for molecular dynamics parameters which do not include the position restraints of the previous step. Note that we will use the structure file coming from the previous equilibration with position restraints, i.e. 2posre.gro

$ gmx grompp -f ../mdp/md.mdp -c ../2posre/2posre.gro -p ../penetratin.top -o 3md.tpr

$ gmx mdrun -deffnm 3md -v

4. STEP 4 - Analysis of the Simulation



Again, the best option to keep everything clear and order is to go back to the root of the working directory and create a specific folder for the analysis:

$ cd .. ; mkdir 4-analysis; cd 4-analysis

4.1 Energy (gmx energy)

Let’s look at the total energy of the System: run the gmx energy command and select “Potential” as done after the energy minimization.

$ gmx energy -f ../4md/md.edr -o Etot.xvg

Plot the energy with xmgrace:

$ xmgrace Etot.xvg

Now, let’s check the temperature:

$ gmx energy -f md.edr -o temperature.xvg

Plot the temperature with xmgrace:

$ *xmgrace temperature.xvg*

Check if the distribution of the temperature is normal!!

4.2 Structure deviation (gmx rms)

gmx rms -h

gmx rms -s md.tpr -f md.xtc

xmgrace -nxy rmsd.xvg

Where does the RMSD reach a plateau? What’s happening?

Check the structure at this time

trjconv -h (per avere l’help)

trjconv -s md.tpr -f md_out.xtc -dt 20 -b tempoA -o last.pdb -fit rot+trans

4
1

../soft/./rasmol last.pdb

Restrict protein (write it on the shell) '**Colours**',→ '**Chain**'
'Display',→ 'Backbone'

Compactness of the structure

gmx gyrate -s topol.tpr -f traj.xtc 4
1

xmgrace -nxy gyrate.xvg

Create a trajectory movie
gmx trjconv -s topol.tpr -f traj.xtc -skip 10 -o movie.pdb 0

Ramachandran

gmx rama -s topol.tpr -f traj.xtc

or vmd is able too (procheck is better though!!!!!)

vmd movie.pdb or
vmd starting.pdb traj.xtc