GROMACS 2019 TUTORIAL

Before starting:

- you will need some basic Linux knowledge for this practical since all the tools that will be used are command lines tools.
- There is no breakline in commands line (commands are on 1 line)
- There is no space between the "-" and the argument letter ("- o" doesn't work, "-o" works!)
- Do not copy/paste commands from the PDF, it might not work.

External useful link :

- GROMACS tutorial: http://www.mdtutorials.com/gmx/lysozyme/index.html
- Second GROMACS tutorial (maybe more detailed): http://www.strodel.info/index_files/lecture/html/tutorial.html
- Linux basic commands: https://maker.pro/linux/tutorial/basic-linux-commands-for-beginners

Software installation

You will use GROMACS 2019 and VMD for this tutorial. You can install both softwares without root access thanks to anaconda (a python package manager that can also be used to install different softwares). Both VMD Gromacs are in anaconda Cloud, so it is perfect! Note that Gromacs is "precompiled" which means that it will not have the best performance for your computer. For better performance, you should compile it on your own (but it required skills and time…).

Ignore this part if you already have installed all the softwares.

```
wget https://repo.anaconda.com/miniconda/Miniconda3-latest-Linux-x86_64.sh
sh Miniconda3-latest-Linux-x86_64.sh -b
eval "$(~/miniconda3/bin/conda shell.bash hook)"
conda init
conda install -c bioconda -c conda-forge gromacs=2019 r-base vmd
mdanalysis python=3.6 pocl -y
```

Context

It is well admitted that prolines are "alpha-helices breakers", which means that they will alter the

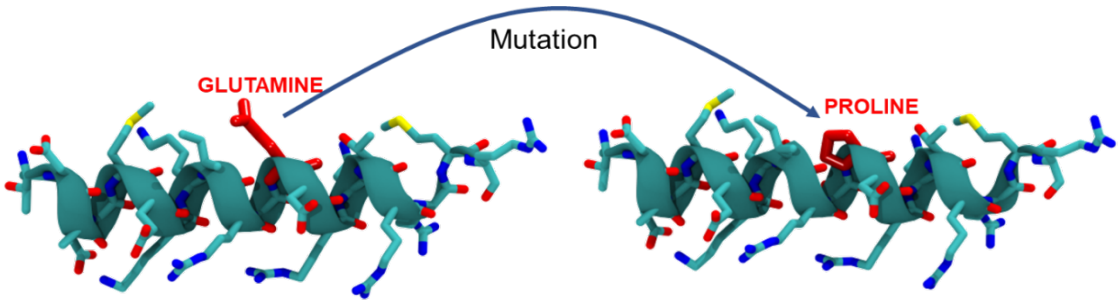


Fig 1 - Position of the proline in the nucleosome-interacting C-terminal alpha helix

conformation of an Alpha helix.

We will try to understand why prolines can "break" a helix with molecular dynamics simulation.

The system used for this PCLab is the nucleosome-interacting C-terminal alpha-helix (PDB ID 6UCH) which is an alpha helix that was obtained by NMR. The helix was extracted and renumbered from 1.

The GLN12 was replaced by a Proline (cf. Fig 1). We will not study the stability of the non-mutated helix with molecular dynamics simulation (both simulations would be too long, but the MD files will be provided in the next practical).

FILES for this practical :

- **helixPRO.pdb** → the mutated alpha Helix (Q12P, GLN 12 mutated into a PRO).
- **mdp/** → Folder that contains all parameters files for ionization, minimization, equilibration, and production.
- **Md_helixPRO_prod(dt_20).(xtc/gro/edr/tpr)**

1. Download all the file at this link : http://tubiana.me/teaching_files/biocat2020/materialbiocat.zip
2. You need to extract the archive (right click on the file → extract here).

the first part consists of preparing and running a MD, so you can work in the folder "**runningMD**".

The second part consists of analyzing the results, but it is optional since the first part can take some times. For the Analyzes part, use the files in "AnalyzingMD" folder and navigate into each sub-folder depending of the analyses you have to generate.

Table des matières

0 Before starting

>_ Command (or action) to execute

0.1 GROMACS pipeline

Open source and completely free for both academic and industrial, GROMACS is one of the most used molecular modelling software.
MD with GROMACS is iterative, which means that before running the calculation, you have to generate all parameters, topology, etc..

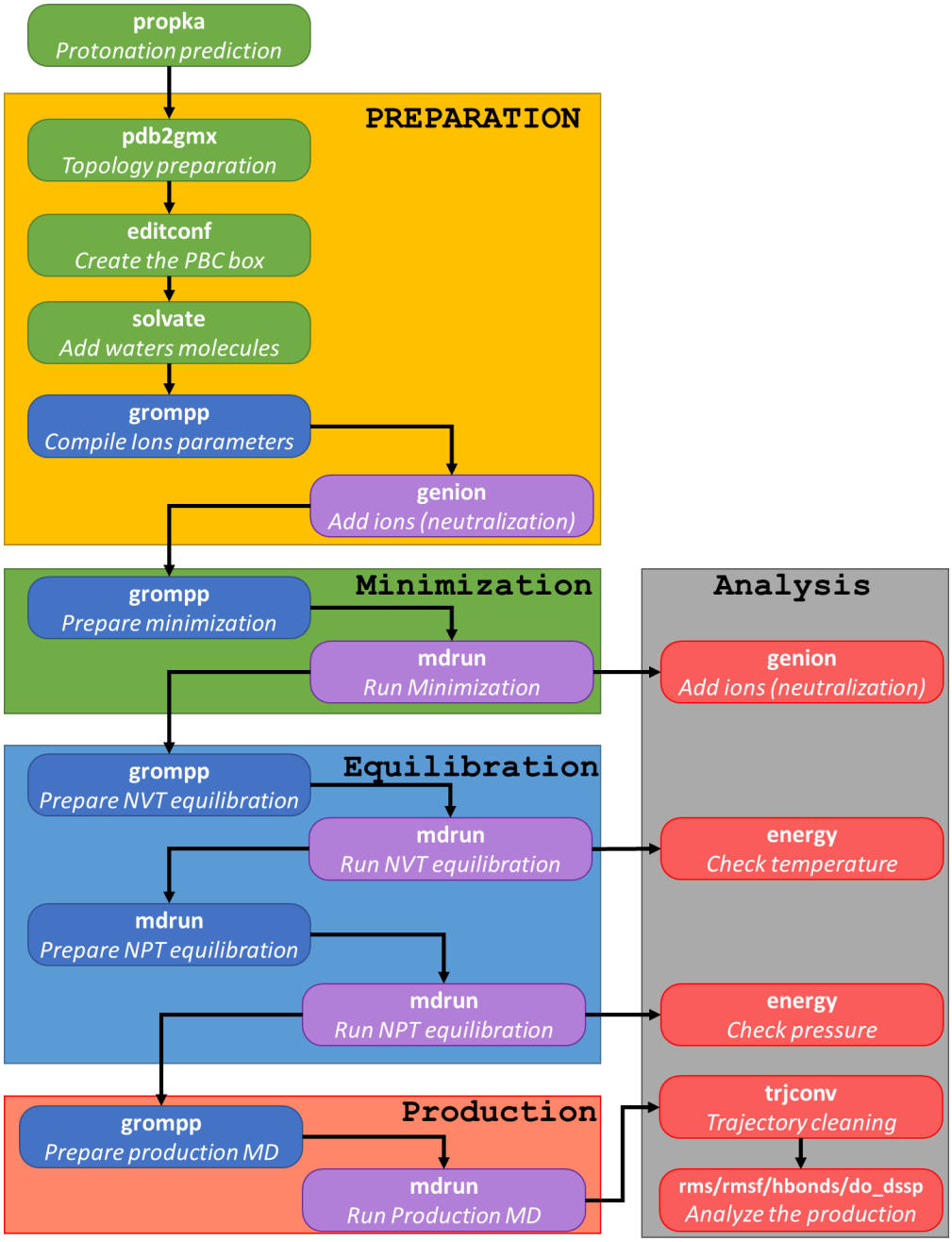


Fig 2 - GROMACS working pipeline. Reproduced and adapted from http://www.strodel.info/index_files/lecture/html/setup.html (originaly made by Oliver Schillinger). You can check his Gromacs tutorial if you need more explanation on some part, it's a very good one!

0.2 Genion and mdrun execution

In GROMACS, you need to "configure" a calculation before running it (*this phase is also called compiling)*. You need to gather all the parameters for the simulation, contained in a *mdp* file, the structure of your system of course and the topology that describe your system (how many bonds, which atoms are connected together….).

***Note about MDP files**: You don't have to put all parameters if you don't specify some parameters GROMACS will use the default value for each non-specified parameters*

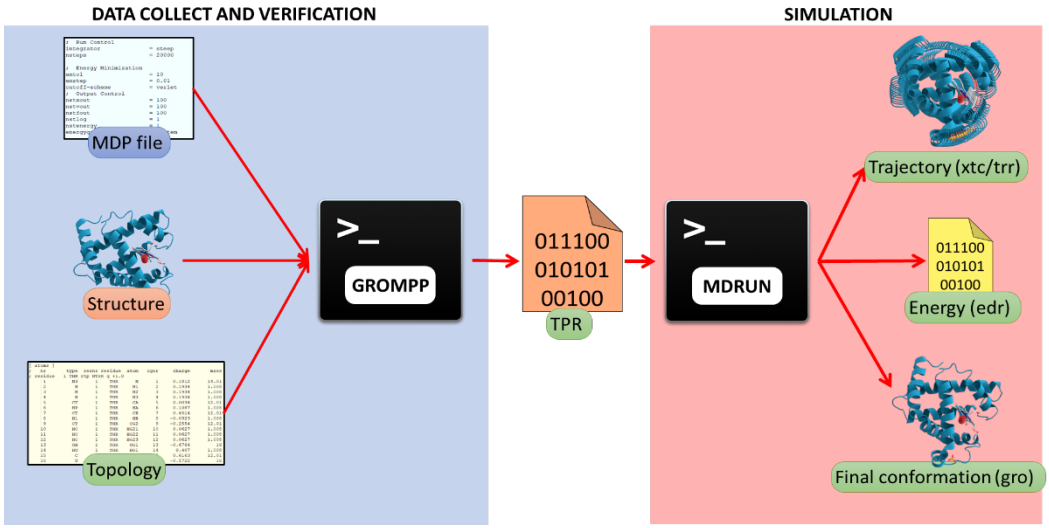


Fig 3 - Representative diagram of the preparation and execution of a simulation with GROMACS.

The software that gathers and checks the file before running the MD is called **grompp** and it will generate a **tpr** file (for **T**opology **P**ortable **R**un file). This is a binary file (*you can't read it with a regular text editor*) and it contains every parameter for the simulation, the initial coordinates, velocities, topology, restraints…..

This file is then given to MDRUN that will execute the MD based on the content of the tpr file. This workflow is described in Fig 3.

1 Preparation: protonation, solvatation, neutralization, and minimization.

1.1 Visualization

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?
- Are there close contacts (sterical clashes)?

You can, of course, check the PDB webpage if the structure is directly downloaded from the PDB webpage, and you can use a visualization software (VMD, pymol…) to look at the structure.

1.2 Protonation.

The first step (which is not mandatory, but better if you want to be more biological relevant), is the prediction of the protonation state. You can use the software "`propka`" to predict the

protonation state of your amino acids, but for today, we will use a webserver instead: `PDB2PQR` (since we will not need to install the software).

1. Go to http://nbcr-222.ucsd.edu/pdb2pqr_2.1.1/ and upload our working structure: helixPRO.pdb [tick "upload a PDB file" →choose a file → navigate to your working folder, click on helixPRO.pdb → open"].
2. Choose the force field
   For this practical, we will use the **Amber** force field. So better keep the same force field for everything!
3. Output naming shame
   Remember: in the force field, every amino acid is already parametrized, for each of their atoms. If you add or remove atoms, the software will realize that something is different and will not be able to process the file (or it will just use the "standard" protonation state and ignore the modification you've made.
   To bypass this issue, protonated and unprotonated amino acids were parametrized as well but recorded with a different amino acid name (eg: unprotonated TYR is TYM, protonated ASP is ASH….)
   >Since we will use **Amber** for the simulation, use this output naming scheme.
4. Available Options
   tick the "Add/keep chain IDs in the PQR file" option
5. pKa Options
   let the pH at 7 (neutral) and be sure that "use PROPKA" is ticked.
6. Click on Run!

The structure is small, you should have the results within 60 seconds, and you will have 3 **output files**.

- helixPRO.propka → pKa and protonation state for every amino acid than can be charged
- helixPRO.pqr → PDB output (PQR is a slightly modified version of the PDB, but the format stay the same)
- helixPRO.in → Input file for Electrostatic map calculation. Not needed for this course.

Download "helixPRO.pqr" and save it in your working directory (*right-click on the link → save as)*

Note: for this peptide (small protein), the protonation state is standard at pH 7. You can start the process again but changing the pH to 1 to see the differences (ASP will be changed to ASH with one extra hydrogen atom, and GLU in GLH with one extra hydrogen atom as well), but do not use the pH1 file for calculation.

1.3 Conversion in GROMACS Format

Now that you have prepared your structure for GROMACS, it is time to convert it to the GROMACS format.
GROMACS don't take ".pqr" format, but remember, pqr format is almost identical to PDB format. So the first thing to do:

- Rename **helixPRO.pqr** to **helixPRO_proto.pdb** (change the extension, say yes if you have a confirmation window)

Now convert **helixPRO_proto.pdb** to **helixPRO_processed.gro** using pdb2gmx *("gro" format is specific to GROMACS, but the structure generated can be opened in most of molecular viewer softwares).*

a. Open a terminal (CTRL+T on Linux)
b. Navigate into your working directory (command `cd` followed by the folder name)
c. Run the command
```
gmx pdb2gmx -f helixPRO_proto.pdb -o helixPRO_processed.gro -ignh
```
d. When prompted, choose the **AMBER99SB-ILDN protein** Force Field by pressing "**6**" then press the touch "Enter" to validate
e. Choose the "TIP3P" water model (press "**1**" then "Enter")

Arguments:

- `-f` → input file (pdb)
- `-ignh` → ignore hydrogens (gromacs will re-create the hydrogens with the right naming scheme).

From the output you can have the total charge of the system, how many dihedrals, impropers dihedral, angles, bonds, and pairs you have in your system.

You have 3 important files that are created:

- helixPRO_processed.gro: a .gro file containing the structure in GROMACS format.
- posre.itp: file containing all the restraints (on non-hydrogen atoms only). For more details see Restraints in GROMACS.
- topol.top: topology file (containing the number of atoms, which type of molecule you have in your system, where to find the parameters…)

1.4 Creation of the simulation box

Now that the protein was converted and prepared for GROMACS (with a specific force field), it is time to create a simulation box. Since we have a long and thin peptide, we will use a cubic box so if the helix rotates, it will still have the some space to move without too many border effects.

>_ Generate the box with this command:

```
gmx editconf -f  helixPRO_processed.gro -o helixPRO_newbox.gro -d 0.8 -bt cubic
```

Input files:
- GROMACS Converted structure file
- Boxsize
- Box type

Output:
- Structure with box encoded.

Arguments:
- `-f` → input file
- `-d` → minimum distance between the box and the and the protein.
- `-bt` → box type
- `-o` → output file (with the box)

Note:
The minimum distance between the box and the protein here is 0.8nm. If you want to simulate a bigger protein, you should use a bigger value to be sure to not have any border effect.

1.5 Solvatation

Some decades ago, the simulation was made in a vacuum, since the computation resources were not as powerful as nowadays, it was faster to simulate without waters.
Now that we have powerful computers (and supercomputers) the norm is to simulate the protein in a water box.
This is done by using a small pre-equilibrated system of water (a system of 216 water molecules) and repeat it over the simulation box. Then overlapping water molecules are removed.

>_ Use this command to solvate the simulation box:

```
gmx solvate -cp helixPRO_newbox.gro -cs spc216.gro -o helixPRO_solv.gro -p
topol.top
```

Input files
- Structure with box dimention encoded
- Water box model
- Topology

Outputs
- Structure box filled with solvent
- New topology with water information

Arguments:

- `-cp` → input file (protein with box)
- `-cs` → water box model
- `-p` → topology
- `-o` → output file (protein the box and the solvent)

In case you are wondering, Here's the concent of spc216.gro in Fig 4. This is the small equilibrated water system.

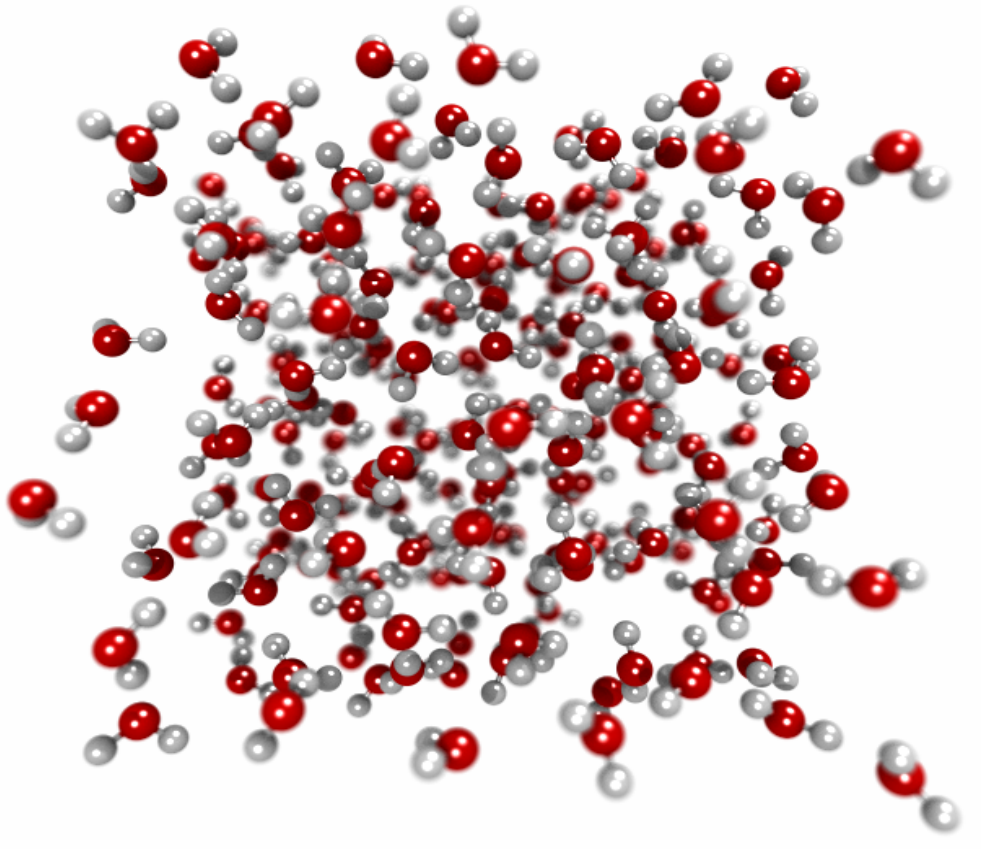


Fig 4 - spc216.gro, containing 216 water molecules

As an exercise *(optional)*, you can look how many solvent molecules were added in the first place, divide by 216 (the number of water molecules in `spc216.gro`) to have the number of duplicated "water box", and then look how many waters molecules were removed (because of overlaps)

1.6 Ionization

The calculation of the electrostatics has to be made in a neutral box. If your system is charged, you need to neutralize the charges with counter-ions.
The software that adds ion needs a "`tpr`" file. So we will use the software `grompp` to generate it (see Fig 3). To generate the tpr file write in your terminal:

```
gmx grompp -f mdp/ions.mdp -c helixPRO_solv.gro -p topol.top -o ions.tpr --maxwarn 1
```

Arguments:

- `-f` → mdp file containing all parameters
- `-c` → Input structure (c was used for "continuous")
- `-p` → Input topology
- `-o` → Output TPR file
- `--maxwarn` → Maximum warning authorized.

Inputs
- Genion parameters
- Structure with solvated box
- Topology

Outputs
- Compiled TPR file

Note:

- We have just generated the file "ions.tpr".
- `grompp` is mostly used to prepare the file for MD. It has a "checking" algorithm to evaluate a potential error that you can get. The argument `--maxwarn 1` is used to ignore the warning that you can have during this process. Here the warning is about electrostatic: the system net charge is not 0 so if you simulate a nonneutral charge system you can have severe artifacts. But during the ionization you will not simulate anything, just replace randomly some water molecules by ions. So you can ignore this warning

Now use genion to replace water molecules by ions.

```
gmx genion -s ions.tpr -o helixPRO_solv_ions.gro -p topol.top -pname NA -nname CL -neutral
```

When prompt, choose what you want to replace by ions : Solvent of course 😊! The group is 13 (or 12). Write "13" then press "Enter".

Arguments:

- `-s` → Topology Portable Run file
- `-o` → output file, it will be called
- `-p` → New topology that will be created (since you replace some water molecules with ions, you have to recreate the topology)
- `-pname` → Type of Positive ion (here $NA^{+)}$
- `-nname` → Type of Negative ion (here CL^{-})
- `-neutral` → We want to neutralize the system, this option calculates the number of necessary ions to neutralize the system (otherwise you have to specify the number of each ion by yourself).

Inputs
- Compiled TPR file
- Ions type
- Concentration (or neutral)

Outputs
- Solvated system with ions
- New topology with ions

As an exercise (*optional)*, you can look at how many ions were added (and which types).
An interesting thing to look at is which waters molecule were replaced and run the "genion" command again. You will see that the water molecules replaced are changed!
This depends on the SEED (random number) used! if you want to be totally reproducible, you can fix this SEED with the `-seed` argument of `genion`. You can have more details with the `-h` argument.

1.7 Minimization

Since we have now finished the preparation of the simulation box, we can minimize it to remove the potential clashes and optimize the geometry of all molecules.

All parameters for MD calculation are stored in an "em.mdp" file. You can open this file in a text editor or refer to Box 3 (page 27).

Compile and generate the TPR file with grompp

```
gmx grompp -f mdp/em.mdp -c helixPRO_solv_ions.gro -p topol.top -o em.tpr
```

Arguments:

- `-c`: input structure
- `-p`: topology
- `-o`: Topology Portable Run input file.

And run the minimization

```
gmx mdrun -v -deffnm em -nt 4
```

for details about input/output for Grompp and MDRUN, see Fig 3

Arguments:

- `-v`: "verbose", it will give you realtime feedback on the terminal
- `-deffnm`: basename for your input and output file (basename = file name without extension. Here the base name will be "em", the input is "em.tpr" and outputs: "em.trr"…
- `-nt`: number for cores for calculation. PCLab computers have 4 cores.

As an exercise *(optional)*, you can look at the potential energy at step=0 and at the end of the minimization and observe that indeed: the energy was well minimized!

2 Equilibration

When simulating, it is important to match the experimental conditions. Most of the time, experiments are done at constant temperature and pressure, so your simulations have to be done at constant temperature and pressure.

The equilibration process aims to bring "energy" to your system and avoid unnecessary distortion of the protein when the molecular dynamic is started. Right now, you don't have any velocity, temperature or pressure. It is like your system is "freeze". It is now time to "unfreeze" it!
You usually equilibrate your system in 2 steps :

- **NVT**: first you bring the temperature to your system (temperature is linked to velocity, so you will also add velocities on each atom)
- **NPT**: Once the temperature is equilibrated, you can equilibrate the pressure.

You can also equilibrate the temperature and the pressure in 1 step, but you increase the chance that your system crash…

In equilibration, all heavy proteins atoms are restrained to their starting positions (using the file posre.itp generated earlier, see Restraints in GROMACS. for more details), while the water is relaxing around the structure.

2.1 NVT equilibration

All the parameters for NVT equilibration are store in "nvt_300.mdp" (or refer to Box 4 - nvt_300.mdp page 28).
Equilibration and production use the same algorithm (per default leap-frog) and require a timestep (dt). Bonds will be constrained to enable a 2 fs time steps.

You can have the description of all those parameters with the online documentation (http://manual.gromacs.org/documentation/2019/user-guide/mdp-options.html) but I would like to emphasize on several parameters :

- the "integrator" is the algorithm that you use. Here it is MD (for Molecular Dynamic), which is a Leap-frog algorithm (`md-vv` for Velocity Verlet)
- For more details on `Nonbonded settings` (neighbor search), please refer to Two extra things about MD: (page 23).
- Same for `PME`, please read Two extra things about MD: (page 23).
- Look at "`Velocity generation`": it is set to "`yes`", and the temperature is 300K (around 27°C). This is also here where you set-up the random seed for MD (`gen_seed`). -1 is random.
- The timestep here is 2fs.
- The line "`define = -DPOSRES`" is the definition of restraint forces. You can check "Restraints in GROMACS. Page 23

Commands :
Compiling and preparing the input md file :

```
gmx grompp -f mdp/nvt_300.mdp -c em.gro -r em.gro -p topol.top -o nvt_300.tpr
```

New argument:

- -r: restrain file: this file contains the coordinate of the atoms that are restraint during the equilibration.

Runing the equilibration:

```
gmx mdrun -deffnm nvt_300 -v -nt 4
```

the -v argument is for "verbose" (or from the documentation: "be loud and noisy"). Actually, you will have real-time information about how many simulation steps were made, and the remaining ratings.
At the end of the mdrun execution, you will also have a summary of the performance in ns/day (how many nanoseconds of simulation can be produced in 24 hours).

Just for comparison, those calculations can be done on GPU (graphic card) if GROMACS was compiled for GPU usage, on 8 cores (AMD Ryzen 7 3700X) + Nvidia RTX 2070 SUPER, the performance was of 471ns/day for this system ….

2.2 NPT Equilibration

All the parameters for NVT equilibration are stored in "npt.mdp" (or refer to Box 5 - npt.mdp page 29).
On the MDP file, you can see that this time, we do not generate velocity (we keep the velocity that was generated during the NVT equilibration).

Same as for NVT Equilibration, you will use the output structure from the NVT this time (nvt_300.gro), to restart from the last step of the NVT equilibration but this time you will equilibrate the pressure
Note: If you compare "`em.gro`" and "`nvt_300.gro`" you will see 3 extra columns. Those columns are the velocities for each atom in each direction (X, Y, Z). This is all you need to restart or continue a molecular dynamics simulation.

Commands :

Compiling and preparing the input md file :

```
gmx grompp -f mdp/npt.mdp -c nvt_300.gro -r nvt_300.gro -p topol.top -o npt.tpr
```

Runing the equilibration:

```
gmx mdrun -deffnm npt -v -nt 4
```

NOTE:
Here are the systems that are equilibrated for 50ps, which is not a lot (because of the time constraints). If you simulate a bigger system, you should equilibrate a longuer (at least 100ps).

3 Production

Production is the final step. You can remove all restraints and let the system explore its conformational space. Some small changes have been made in the MDP files (see Box 6 – md_prod.mdp page 30).

Commands :

```
gmx grompp -f mdp/md_prod.mdp -c npt.gro -r npt.gro -p topol.top -o md_helixPRO.tpr
```

Running the simulation

```
gmx mdrun -v -deffnm md_helixPRO -nt 4
```

NOTE:
The calculation is taking too much time (you can check when the simulation will be over and calculate the performance by your own, but it should be similar to the production)
Instead of waiting for the results, you will use the file produced for you ☺.

4 Analysis – *(Optional)*

4.1 Analysis of the convergence before the production

It is important to check if the system converged during the minimization and equilibration. Normally those analysis is done just after the minimization and each equilibration.

4.1.1 Minimization

For minimization, you can check if the system has converged by looking at the potential energy contained in the .edr file.

1. Go into "mini" folder
2. Run this command:

```
gmx energy -s em.tpr -f em.edr -o potential.xvg
```

3. Choose the potential energy (write "`11 0`" then press '`enter`')
   At the end of the execution, you will be able to check the average potential energy (useful if you just want this data!)
4. Open the graphics with

```
xmgrace potential.xvg
```

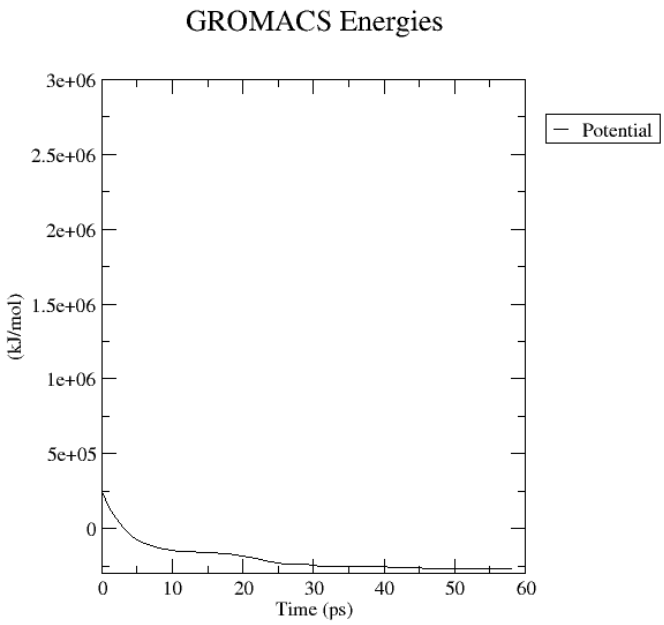


*Fig 5 - Potential energy during minimization. **Be carefull**, per default XMGrace think that the X-axis is in ps, but the minimization steps are not linked to the time. It is just steps.*

On Fig 5 you can observe the potential energy during the minimization steps. It starts from a very high value because the mutation introduced some cashed in the structure, but the energy decreases rapidly to converge after 45 steps. Note that the conjugate gradient + steepest descent is a fast algorithm on Gromacs. With Steepest descent only you can oscillate around your minima so you will maybe converge after a few hundred steps.

4.1.2 NVT Equilibration

Since in NVT equilibration we add velocities to the system by increasing the temperature, we want to be sure that we converge to our desired temperature (300K here)

1. Go into "nvt" folder
2. Run this command:

```
gmx energy -s nvt_300.tpr -f nvt_300.edr -o temperature.xvg
```

3. Select the temperature: "16 0" then press Enter.
   Here again, you will be able to check (on the terminal) the average temperature of the system.
4. Open the graphics with

```
xmgrace temperature.xvg
```

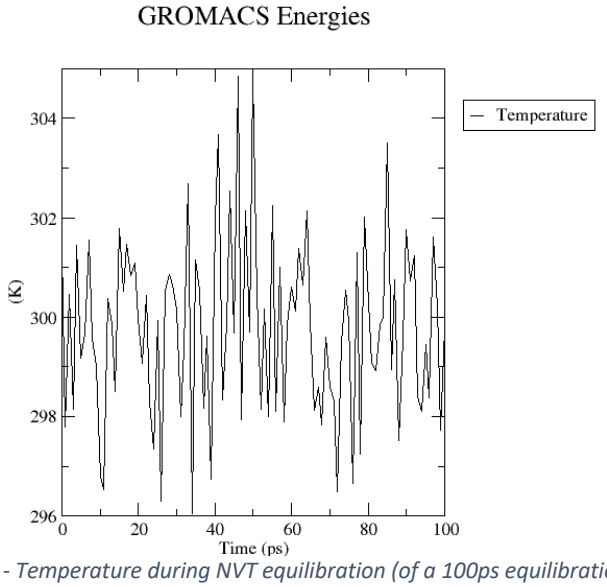


Fig 6 - Temperature during NVT equilibration (of a 100ps equilibration)

On Fig 6 you can see the temperature during the NVT equilibration. You can see that it is oscillating around our fixed temperature value (300K) and it is quite stable over time (the high oscillations are only due to the zooming level on the Y-axis). Also, the average value is around 300K, so it means that we are ready to continue to the next equilibration!

4.1.3 NPT Equilibration

Same as for NVT, we need to check the convergence of the pressure.

1. Go into "npt" folder
2. Run this command:

```
gmx energy -s npt_ab.tpr -f npt_ab.edr -o pressure.xvg
```

5. Select the pressure: "17 0" then press Enter.
   Here again, you will be able to check (on the terminal) the average pressure of the system.
3. Open the graphics with

```
xmgrace pressure.xvg
```

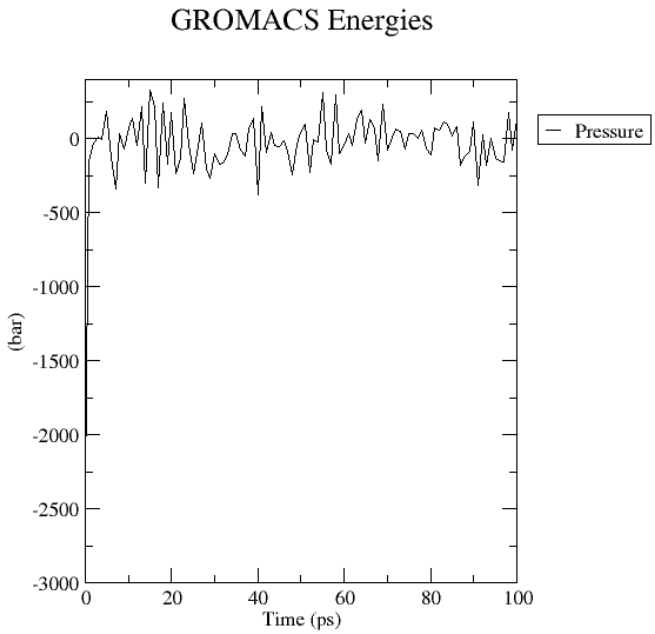


Fig 7 - Pressure during the equilibration (100ps)

On Fig 7 you can follow the pressure (in bar) over the 100ps NPT equilibration phase of the system. The average value of the pressure is -3.9 ± 180.4 bar (the reference pressure was set to 1 bar).
Pressure tends to fluctuate widely over an MD simulation, so statistically speaking, it is hard to distinguish a difference between our average (-3.9 ± 180.4 bar) and the reference value (1 bar). Moreover, pressure can be slow to converge, in real-case MD it would be interesting to extend the equilibration (up to 500ps) to see if the RMSD of the pressure decrease.

4.2 Cleaning trajectory

Files:

- md_helixPRO_prod.tpr
- md_helixPRO_prod_dt20.xtc
- md_helixPRO_prod.gro

The first thing you can do when the simulation is over, it's to visualize your MD.

1. Go into "prod" folder
2. Open the trajectory with VMD (md_helixPRO_prod.gro is the topology)

```
>- VMD md_helixPRO_prod.gro md_helixPRO_prod_dt20.xt
```

3. Write in the terminal "pbc box" to draw the simulation box
4. Show only the protein
   Graphics (menu) →Representation → erase 'all' in the selection and write "not waters"
5. Play the trajectory (▶ icon in main windows, bottom right)

You can see that the protein will be "broken" at some times. This is because during the MD, your protein can "move". It is important to "wrap" the atoms together, to make the molecule "full". You can do that with gromacs tools. trjconv is a tool used to align, wrap, cut the molecule.

Close your terminal and write the command:

```
gmx trjconv -s md_helixPRO_prod.tpr -f md_helixPRO_prod_dt20.xtc -o traj_wrapped.xtc -pbc mol
```

- -pbc: (mol) wrap each molecule in the box.

Gromacs will ask what do you want to save as output, press *0 then Enter* to save the full system (with water)

If you visualize your trajectory, you will see that the helix "jump" from one side to the other side of the box. We need to center the molecule inside the box and align the molecule to itself.

```
gmx trjconv -s md_helixPRO_prod.tpr -f traj_wrapped.xtc -o traj_wrapped_aligned.xtc -fit rot+trans -center
```

- -fit+trans: use rotation and translation to align the protein.
- -center: put the protein (if selected) in the center.

GROMACS will ask 3 things:

- on what you want to align the system, choose backbone (*4 then enter*)
- then on what you want to center the simulation box, choose a protein (*1 then enter*)
- and finally, the output group, choose all the system (*0 then enter*)

One last thing, we will not use the water here, we can create a smaller trajectory file by removing the water molecules.

1. `gmx trjconv -s md_helixPRO_prod.tpr -f traj_wrapped_aligned.xtc -o traj_wrapped_aligned_protein.xtc`
2. write `1` then `enter` to select only the protein when prompted.

It is not finished… yet… To have a topology file that corresponds to your trajectory without water and ions, you need to generate a new topology file (tpr for gromacs, pdb or gro for VMD)

PDB file for VMD (or other softwares). Note: -b 0 -e 0 means that you start the trajectory at 0 and stop at 0 (only the first frame will be saved)

1. `gmx trjconv -s md_helixPRO_prod.tpr -f traj_wrapped_aligned.xtc -o topology.pdb -b 0 -e 0`
   - -b: from time
   - -e: to time
2. choose the **protein**

TPR file for GROMACS (or other softwares)

1. `gmx convert-tpr -s md_helixPRO_prod.tpr -o protein.tpr`
2. choose the **protein**

4.3 Production analysis

4.3.1 RMSD

The RMSD (Root Mean Square Deviation) is one of the most important and fundamental properties to analyze if whether the protein is stable and close to the initial structure (it can be an experimental structure). The formula the of RMSD is

$$RMSD = \sqrt{\frac{\sum_{i=1}^{N} (r_i^t - r_i^0)^2}{N}}$$

Where r_i^t is the position of an atom at time *t* and r_i^0 at *t=0* (the initial position of the atom *i*, before the simulation). It is averaged by the number of atoms N. This formula means that you evaluate the deviation of a protein from its initial conformation over time.

1. Generate the RMSD of the helix simulation:

```
>_ gmx rms -s protein.tpr -f traj_wrapped_aligned_protein.xtc -o
rmsd_protein.xvg -tu ns
```

   - -tu: time unit = ns

1. Choose the **backbone** for alignment **AND** calculation
2. Open the output file with xmgrace (`xmgrace rmsd_protein.xvg`)

The alignment is very important. If you want to estimate the fluctuation of the backbone, you should align the protein on the backbone. If you align on a different reference, the results will be totally different.
Let's take the example of a protein with a ligand. If you calculate the RMSD of the ligand aligned on the protein, you will quantify the movement of the ligand regarding the protein (if the ligand leave the binding pocket and move a lot in the simulation box, you will observe a very high RMSD). But if you want to quantify only the intrinsic movements of the ligand, you should align the trajectory with the ligand as reference.

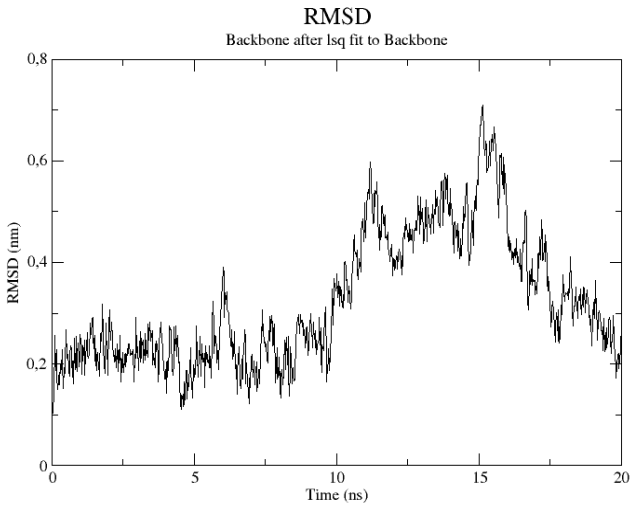


Fig 8 - RMSD of the helix's backbone (aligned on the backbone) on 20ns.

On the RMSD plot (Fig 8) you don't have a "plateau" (so no convergence in 20ns). But it seems that the structure is stabilizing around 0.2nm for 10ns. Then RMSD values increase to 0.7nm between 10ns and 15ns and then decrease to come back at 0.2nm at 20ns. It means that the helix deforms and then returns to a conformation close to its conformation at the beginning of dynamics.

4.3.2 RMSF

The RMSF (Root Mean Square Fluctuation) is a measure of the fluctuation of an atom (or an amino acid if the values are averaged by amino acids, which is common).

The formula is:

$$RMSF = \sqrt{\frac{\sum_{i=1}^{N} (r_i^t - \overline{r_i})^2}{N}}$$

where r_i^t is the position of an atom *i* at time *t* and $\overline{r_i}$ the average position of the atom *i* among all frames. N Is the number of frames. This formula means that you will observe the movement of an atom around its average position.

1. Generate the RMSF of the helix:

```
>_ gmx rmsf -s protein.tpr -f traj_wrapped_aligned_protein.xtc -o
rmsf_protein.xvg -res -oq bfactor.pdb
```

   - -res: average the RMSF values by amino acids.
   - -oq: put the RMSF values in the BFactor column in a PDB file, so you can see where are your most flexible region easily on Pymol and VMD.
2. Choose the **backbone**
3. Open the output file with xmgrace (`xmgrace rmsf_protein.xvg`)

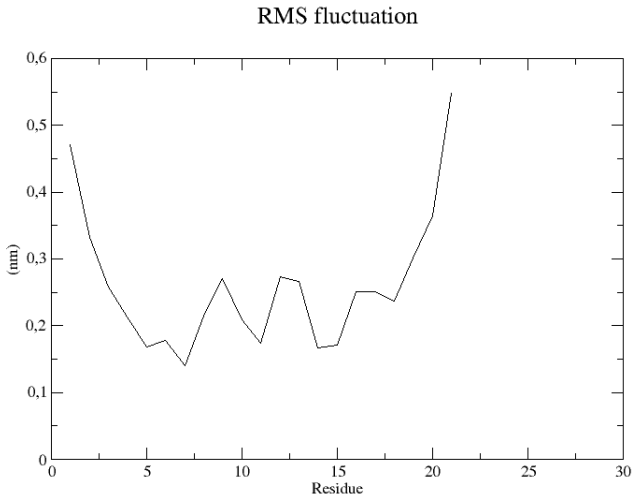


Fig 9 - RMSF of each helix's amino acids.

On the RMSF graph (Fig 9) you can observe that the firsts (N-ter) and the lasts (C-ter) amino acids are more flexible. It is because they are not constrained by other amino acids therefore they are freer to move. We can also see that amino acids 9, 12 a,d 13 is more flexible than other amino acids.

4.3.3 H-bonds

The structure of an alpha helix is maintained by a network of hydrogen bonds between the carboxyl oxygen of the backbone and amino group of 4th next amino acid **(n to n+4)**
We can follow the number of hydrogen bonds during the dynamic with the gromacs' H-bonds software.

1. `gmx hbond -s protein.tpr -f traj_wrapped_aligned_protein.xtc -tu ns -num hbonds_num.xvg -hx hbonds_helix.xvg`
   - -tu: time unit = ns
   - -num: total number of h-bonds
   - -hx: check only for n to n+1 to n+6
2. choose the "`mainchain+H`" `(7)` for the two groups
3. Open the output with

`xmgrace -nxy hbonds_helix.xvg`

nxy is an argument to read a file that contains multiple variables, here we have n to n+1, n+2, n+3, n+4, n+5, n+6

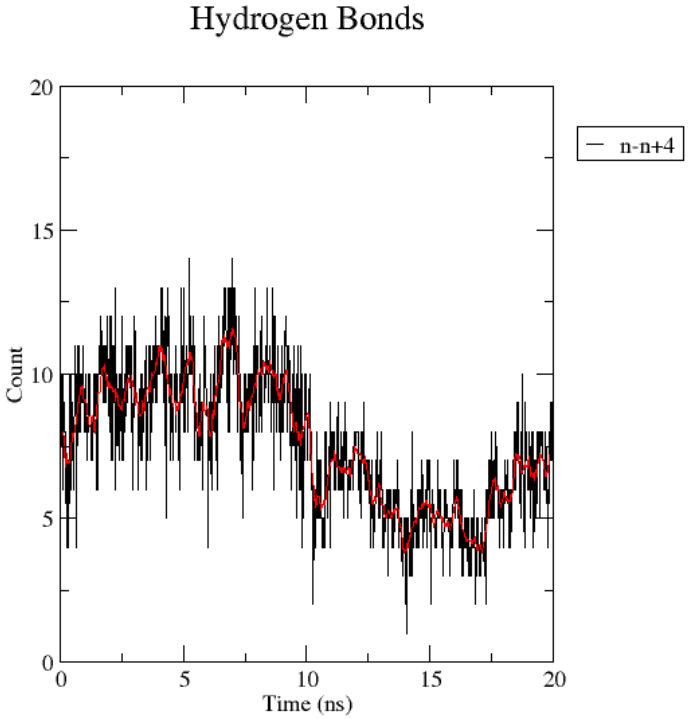


Fig 10 - Hydrogen bonding evolution of the helix main chain + hydrogens, with n,n+4 residue. Red curve = running average on 10 points.

For an alpha helix, is it useful to observe how many H-bonds you have over time. Since this helix is made of 21 amino acids, we can't have more than 18 H-bonds (21-3, after res 18, you can't form any H-bonds with amino acids 22…), and from the RMSF graph (Fig 9), N-Terminal and C-terminal are quite flexible (so cannot form a stable helix over time). So here we have at maximum 14 H-bonds and it gets down from 10 to 15ns. If we look at the RMSD graph (Fig 8), we see a correlation with the decrease of H-Bonds and the deviation from the initial helix structure. Then, from 15 to 20ns the number of H-bonds increase, indicating that the structure seems to form an α-helix again, which correlates with the RMSD graph again (the deviation decrease indicating that the structure comes back to something closer to its initial conformation). But as opposed to the RMSD evolution, we don't have the same amount of H-bonds at 20ns that between 0 and 10ns.

4.3.4 Secondary structure

Finally, we can check the secondary structure over time and see if the helix break and where. DSSP is a softwares (pre-installed on your computers) that calculate the secondary structure for proteins. Gromacs can be linked to DSSP and calculate the secondary structure on a trajectory and produce a secondary structure map for every amino acid over time. You need to execute those 3 commands :

1. `export DSSP=/usr/bin/dssp`
   this tell to gromacs where is the DSSP software
2. `gmx do_dssp -s protein.tpr -f traj_wrapped_aligned_protein.xtc -o helix_ss.xpm -ver 3 -tu ns -dt 0.05`
   - *ver is for the DSSP version, use 2 for the old version*
   - *tu: time unit in ns.*
   - *dt: timestep for SS calculation (every 0.05ns so every 50ps)*
3. Choose the protein
4. `gmx xpm2ps -f helix_ss.xpm -o helix_ss.eps -by 10 -bx 3`
   convert the output xpm map to a readable file with legends, axis, title….. by and bx are the scale of the axis to have a "readable" figure.

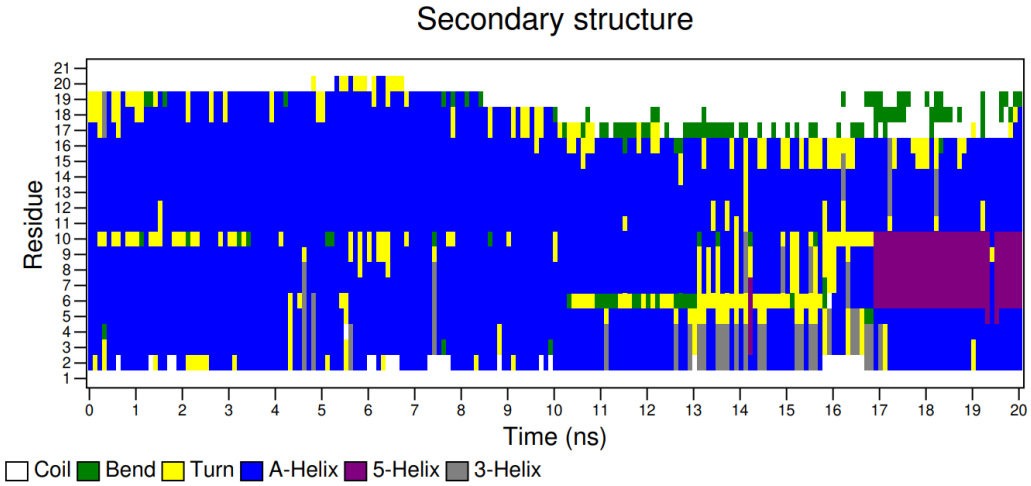


Fig 11 - Evolution of the helix Secondary structures during the 20ns MD.

On the secondary structure plot (Fig 11), each "column" is a secondary structure map of the structure (at time *t*) and each type of secondary structure is represented by a color. For example, the amino acid 19 at *t=0* is a *turn* (yellow).
We can see that in the first 10ns the helix is quite stable, but "broken" at position 10. This is due to the proline introduced in position 12: you can't form an H-bond in the helix backbone with a proline so this region is destabilized (*cf*. Fig 10).
From 10ns to 15ns, the "α-helix" become more unstable and we can see more *turn* and *bend* appearing. This corresponds to the decreasing amount of H-Bonds and the RMSD getting higher (*cf.* Fig 8*)*.
Finally, from 16ns the helix is formed again, but amino acids 10 to 6 are now in a 5-helix conformation (H-bonds formed between n,n+5), which is why in H-bonds graph we don't have the same amount of H-Bonds at 20ns than between 0 and 10ns.

If you wish, you can compare these results with the results of the wild-type helix (without mutation) on page 26.

5 Appendix

5.1 Two extra things about MD:

Neighbor search

To reduce the calculation of the non-bonded interactions you will use a cutoff distance, and you will ignore every atom far away from this cutoff distance.
Of course, you need to know which atoms are "neighbor" (which means which atoms an under this cutoff distance). So for every atom, you have to calculate the distance with every other atom to get this neighbor list. This is of course costly.
In MD, this neighbor list is calculated every *X* steps, *X* to be defined by the user (since two atoms will not move much under a certain number of steps).

PME (Particle Mesh Ewald)

Ignoring electrostatic potential beyond the cutoff value can lead to severe errors. That's why, after the cutoff values, another algorithm is used to calculate the electrostatic potential.
Based on Ewald's method, it consists of adding a virtual grid to the system, then for each point of the grid, interpolate their charge. The electrostatic potential is then calculated in the reciprocal space, after a Fourier transform, and the forces interpolated back to atoms in real space, whose positions are updated.

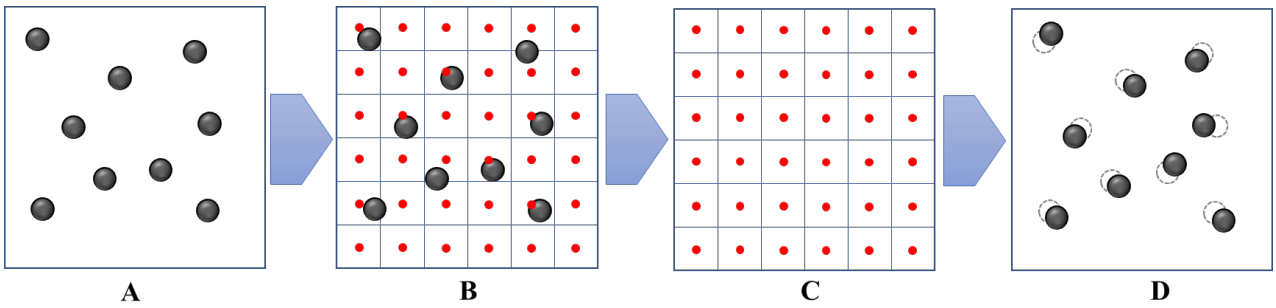


Fig 12 - PME approach on a two-dimensional network. (A) Charged particle system. (B) Charges are interpolated on a two-dimensional grid. (C) Potential and forces are evaluated at each point of the grid in reciprocal space with a Fourier transform. (D) The forces are interpolated back (inverse Fourier transform) to the particles, whose positions are then updated. Adapted from Christophe Chipot's lectures (French) (http://ecole.modelisation.free.fr/cours_chipot.pdf)

5.2 Restraints in GROMACS.

During the equilibration, you increase the temperature by scaling the velocities. If the velocities are too high, your system can crash (lead to an explosion). To avoid that kind of artifact, you have to apply restraints to all atom's positions.
The restraint is a Force applied on each restrained atoms, to remove the movement as much as possible. You will have a "shaking" effect on all atoms.

But how does it work with Gromacs?

After you used pdb2gmx, a file called "posre.itp" (for position restraint) is generated. If you open it with a text editor you will find this :

Box 1 – head of posre.itp

```
; In this topology include file, you will find position restraint
; entries for all the heavy atoms in your original pdb file.
; This means that all the protons which were added by pdb2gmx are
; not restrained.

[ position_restraints ]
; atom  type      fx      fy      fz
     1     1  1000  1000  1000
     5     1  1000  1000  1000
     7     1  1000  1000  1000
     9     1  1000  1000  1000
    13     1  1000  1000  1000
    15     1  1000  1000  1000
```

The header is quite informative: you will find restraints for all heavy atoms (*in MD, heavy atom = all non-hydrogen atoms).*
Then you have the field [position_restrains] with 5 columns.

- The atom number
- The type (*internal coding type for Gromacs*)
- The restraint force applied on X, Y, and Z (in $kJ.mol^{-1}.nm^{-2}$)

Now, look the end of the topology file (topol.top)

Box 2 - End of topol.top

```
; Include Position restraint file
#ifdef POSRES
#include "posre.itp"
#endif

; Include water topology
#include "amber99sb-ildn.ff/tip3p.itp"

#ifdef POSRES_WATER
; Position restraint for each water oxygen
[ position_restraints ]
;  i funct       fcx        fcy        fcz
   1    1       1000       1000       1000
#endif
```

The 4 first lines mean **"If you define a variable called POSRES, you include the restraint file".**

If you look at the position beginning of equilibration parameters files (`nvt_300.mdp` and `npt.mdp`) you will see the line "`define = -DPOSRES`". In those files, you say to the MD program "Define a Variable called `POSRES`" (`-D`, in the beginning, is an argument and therefore is not considered in the name of the variable).
It means that during equilibration, the file `posre.itp` will be integrated and you will have restraint forces applied on the atoms present in `posre.itp`.

Note :

The `posre.itp` file was created at the beginning of the process. You only have heavy atoms of the protein. Not the water. But in `topol.top` you have those lines :

```
#ifdef POSRES_WATER
; Position restraint for each water oxygen
[ position_restraints ]
;  i funct       fcx        fcy        fcz
   1    1       1000       1000       1000
#endif
```

It means that if you add "`define = -DPOSRES_WATER`" in the mdp files, the water will be restraint.

But maybe you have seen that we don't have this line on our MDP file… That's right! Water is free to move in our equilibration :-).

This is one way of doing the equilibration, some people like to restraint the water as well, but in that case, it requires more steps and we don't have the time during this practical.

5.3 Some advice on the MD pipeline

If you have a big or complex system (with a lot of atoms, or non-standard molecules like modified amino acids or other molecules), the equilibration may be a critical step and lead you to a crash. Do not hesitate to run multiple equilibration phase (like for the temperature: first at 150K, then 300K) with restraints on water molecules. You can also extend equilibration to 500ps for example.

Then before the production, you can have a "preproduction" phase when you remove restraints only on a water molecule, then on the full system for the production.

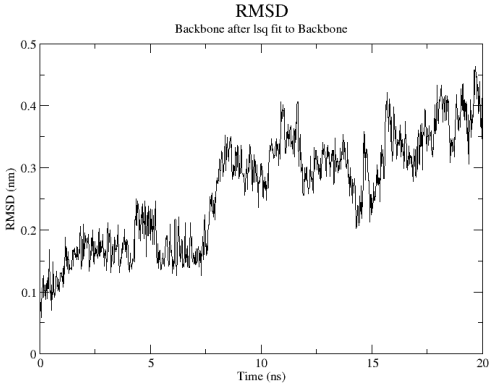
You will get something like this :

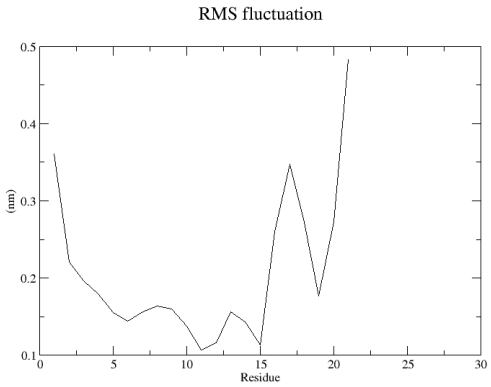
1. Preparation
2. Conversion
3. Box
4. Solvatation
5. Neutralization
6. Equilibration NVT 150K – 500ps
   Restraints on Protein + water
7. Equilibration NVT 300K – 500ps
   Restraints on Protein + water
8. Equilibration NPT – 500ps
   Restraints on Protein + water
9. Preproduction – 2ns
   Restraints on Protein only
10. Production - as long as you want.

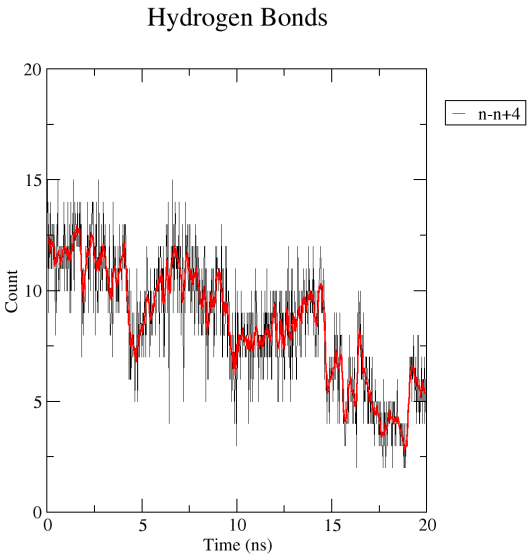
But also remember that there is no universal recipe for MD, it always depends on your input system.

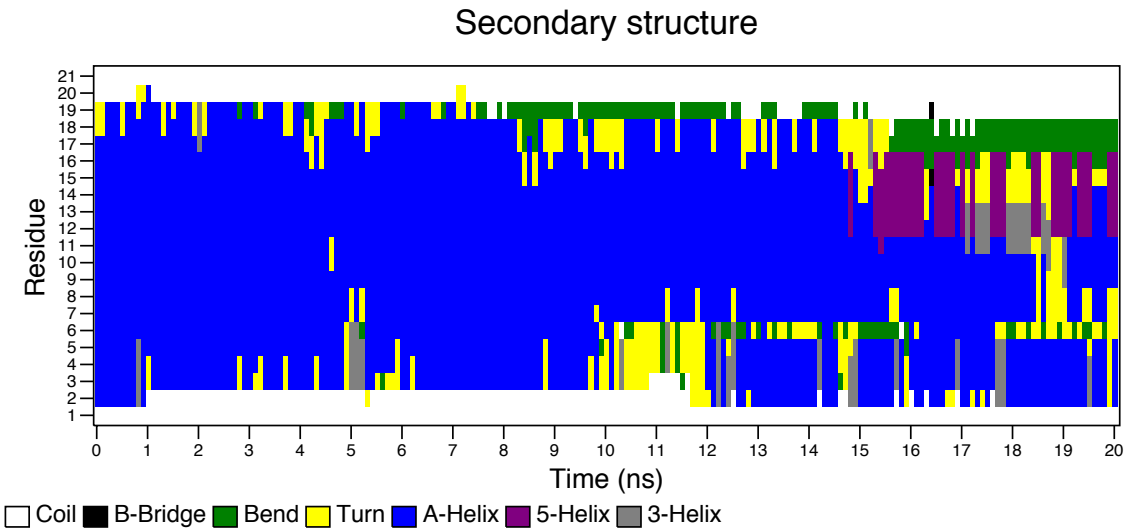
5.4 Result with the wild type Helix (without mutation)

Just for comparison, here are the same results with the helix without the mutation.









5.5 MDP files (parameter files)

5.5.1 Minimization parameters

Box 3 - em.mdp

```
;  Preprocessing
define                  = -DFLEXIBLE    ; defines to pass to the preprocessor

;  Run Control
integrator              = cg            ; Conjugate Gradient energy minimization
nstcgsteep              = 50            ; every 50 steps, do a steepest descent
nsteps                  = 1000          ; maximum number of steps to integrate

;  Energy Minimization
emtol                   = 1000          ; [kJ/mol/nm] minimization is converged when max force
is < emtol
emstep                  = 0.01          ; [nm] initial step-size
cutoff-scheme           = verlet
;  Output Control
nstxout                 = 100           ; [steps] freq to write coordinates to trajectory
nstvout                 = 100           ; [steps] freq to write velocities to trajectory
nstfout                 = 100           ; [steps] freq to write forces to trajectory
nstlog                  = 1             ; [steps] freq to write energies to log file
nstenergy               = 1             ; [steps] freq to write energies to energy file
energygrps              = System        ; group(s) to write to energy file

;  Neighbor Searching
nstlist                 = 100          ; [steps] freq to update neighbor list
ns_type                 = grid          ; method of updating neighbor list
pbc                     = xyz           ; periodic boundary conditions in all directions
rlist                   = 1.0           ; [nm] cut-off distance for the short-range neighbor
list

;  Electrostatics
coulombtype             = PME           ; Particle-Mesh Ewald electrostatics
rcoulomb                = 1.0           ; [nm] distance for Coulomb cut-off

;  VdW
vdwtype                 = cut-off       ; twin-range cut-off with rlist where rvdw >= rlist
rvdw                    = 1.0           ; [nm] distance for LJ cut-off
DispCorr                = Ener          ; apply long range dispersion corrections for energy

;  Ewald
fourierspacing          = 0.12          ; [nm] grid spacing for FFT grid when using PME
pme_order               = 4             ; interpolation order for PME, 4 = cubic
ewald_rtol              = 1e-5          ; relative strength of Ewald-shifted potential at
rcoulomb
```

5.5.2 NVT Equilibration

Box 4 - nvt_300.mdp

```
title                    = NVT Equilibration

define                   = -DPOSRES  ; position restrain the protein


; Run parameters
integrator               = md        ; leap-frog integrator
nsteps                   = 25000     ; 2 * 25000 = 50 ps
dt                       = 0.002     ; 2 fs

; Output control
nstxout                  = 500       ; save coordinates every 1.0 ps
nstvout                  = 500       ; save velocities every 1.0 ps
nstenergy                = 500       ; save energies every 1.0 ps
nstlog                   = 500       ; update log file every 1.0 ps

; Bond parameters
continuation             = no        ; first dynamics run
constraint_algorithm     = LINCS     ; holonomic constraints
constraints              = all-bonds   ; bonds involving H are constrained
lincs_iter               = 1         ; accuracy of LINCS
lincs_order              = 4         ; also related to accuracy


; Nonbonded settings
cutoff-scheme            = Verlet    ; Buffered neighbor searching
ns_type                  = grid      ; search neighboring grid cells
nstlist                  = 10        ; 100 fs, largely irrelevant with Verlet
rcoulomb                 = 1.0       ; short-range electrostatic cutoff (in nm)
rvdw                     = 1.0       ; short-range van der Waals cutoff (in nm)
DispCorr                 = EnerPres  ; account for cut-off vdW scheme


; Electrostatics
coulombtype              = PME       ; Particle Mesh Ewald for long-range electrostatics
pme_order                = 4         ; cubic interpolation
fourierspacing           = 0.16      ; grid spacing for FFT


; Temperature coupling is on
tcoupl                   = Berendsen             ; modified Berendsen thermostat
tc-grps                  = Protein Non-Protein   ; two coupling groups - more accurate
tau_t                    = 0.1     0.1           ; time constant, in ps
ref_t                    = 300     300           ; reference temperature, one for each group, in K

; Pressure coupling is off
pcoupl                   = no        ; no pressure coupling in NVT

; Periodic boundary conditions
pbc                      = xyz       ; 3-D PBC

; Velocity generation
gen_vel                  = yes       ; assign velocities from Maxwell distribution
gen_temp                 = 300       ; temperature for Maxwell distribution
gen_seed                 = -1        ; generate a random seed
```

5.5.3 NPT Equilibration

Box 5 - npt.mdp

```
; 7.3.2 Preprocessing
title                    = NPT Equilibration
define                   = -DPOSRES              ; defines to pass to the preprocessor

; 7.3.3 Run Control
integrator               = md                    ; md integrator
tinit                    = 0                     ; [ps] starting time for run
dt                       = 0.002                 ; [ps] time step for integration
nsteps                   = 25000                ; maximum number of steps to integrate, 0.002 * 25000 = 50 ps
comm_mode                = Linear                ; remove center of mass translation
nstcomm                  = 1                     ; [steps] frequency of mass motion removal
comm_grps                = Protein Non-Protein   ; group(s) for center of mass motion removal
cutoff-scheme            = verlet

; 7.3.8 Output Control
nstxout                  = 5000          ; [steps] freq to write coordinates to trajectory
nstvout                  = 5000          ; [steps] freq to write velocities to trajectory
nstfout                  = 5000          ; [steps] freq to write forces to trajectory
nstlog                   = 5000          ; [steps] freq to write energies to log file
nstenergy                = 5000           ; [steps] freq to write energies to energy file
nstxtcout                = 5000          ; [steps] freq to write coordinates to xtc trajectory
xtc_precision            = 1000          ; [real] precision to write xtc trajectory
xtc_grps                 = System        ; group(s) to write to xtc trajectory
energygrps               = System        ; group(s) to write to energy file

; 7.3.9 Neighbor Searching
nstlist                  = 10             ; [steps] freq to update neighbor list
ns_type                  = grid          ; method of updating neighbor list
pbc                      = xyz           ; periodic boundary conditions in all directions
rlist                    = 1.0           ; [nm] cut-off distance for the short-range neighbor list

; 7.3.10 Electrostatics
coulombtype              = PME           ; Particle-Mesh Ewald electrostatics
rcoulomb                 = 1.0           ; [nm] distance for Coulomb cut-off

; 7.3.11 VdW
vdwtype                  = cut-off       ; twin-range cut-off with rlist where rvdw >= rlist
rvdw                     = 1.0           ; [nm] distance for LJ cut-off
DispCorr                 = EnerPres      ; apply long range dispersion corrections for energy

; 7.3.13 Ewald
fourierspacing           = 0.16          ; [nm] grid spacing for FFT grid when using PME
pme_order                = 4             ; interpolation order for PME, 4 = cubic
ewald_rtol               = 1e-5          ; relative strength of Ewald-shifted potential at rcoulomb

; 7.3.14 Temperature Coupling
tcoupl                   = Berendsen             ; modified Berendsen thermostat
tc-grps                  = Protein Non-Protein   ; two coupling groups - more accurate
tau_t                    = 0.1     0.1           ; time constant, in ps
ref_t                    = 300     300           ; reference temperature, one for each group, in K

; 7.3.15 Pressure Coupling
pcoupl                   = Berendsen                    ; pressure coupling where box vectors are variable
pcoupltype               = isotropic             ; pressure coupling in x-y-z directions
tau_p                    = 1.0                   ; [ps] time constant for coupling
compressibility          = 4.5e-5                ; [bar^-1] compressibility
ref_p                    = 1.0                   ; [bar] reference pressure for coupling
refcoord_scaling         = com

; 7.3.17 Velocity Generation
gen_vel                  = no            ; velocity generation turned off

; 7.3.18 Bonds
constraints              = all-bonds  ; convert all bonds to constraints
constraint_algorithm     = LINCS      ; LINear Constraint Solver
continuation             = yes        ; apply constraints to the start configuration
lincs_order              = 4          ; highest order in the expansion of the contraint coupling matrix
lincs_iter               = 1          ; number of iterations to correct for rotational lengthening
lincs_warnangle          = 30         ; [degrees] maximum angle that a bond can rotate before LINCS will complain
```

5.5.4 Production

Box 6 – md_prod.mdp

```
title             = production MD for protein in explicit water

; Run parameters
integrator        = md              ; leap-frog algorithm
nsteps            = 2500000        ; 0.002 * 2500000 = 5000 ps or 5 ns
dt                = 0.002           ; 2 fs

; Output control
nstxout           = 5000               ; save coordinates every 10 ps
nstvout           = 5000               ; save velocities every 10 ps
nstxtcout         = 5000             ; xtc compressed trajectory output every 10 ps
nstenergy         = 5000             ; save energies every 10 ps
nstlog            = 5000             ; update log file every 10 ps
nstcomm           = 1000               ; center of mass motion removal

; Bond parameters
constraint_algorithm = LINCS         ; holonomic constraints
constraints          = all-bonds     ; all bonds (even heavy atom-H bonds) constrained
lincs_iter           = 1             ; accuracy of LINCS
lincs_order          = 4             ; also related to accuracy

; Neighborsearching
ns_type           = grid             ; search neighboring grid cells
nstlist           = 25               ; with Verlet lists the optimal nstlist is >= 10, with GPUs >= 20.
rlist             = 1.0              ; short-range neighborlist cutoff (in nm)
rcoulomb          = 1.0              ; short-range electrostatic cutoff (in nm)
rvdw              = 1.0              ; short-range van der Waals cutoff (in nm)
rlistlong         = 1.0              ; long-range neighborlist cutoff (in nm)
cutoff-scheme     = Verlet

; Electrostatics
coulombtype       = PME              ; Particle Mesh Ewald for long-range electrostatics
pme_order         = 4                ; cubic interpolation
fourierspacing    = 0.16             ; grid spacing for FFT

; Temperature coupling is on
tcoupl          = V-rescale               ; v-rescale is used now to have a canonical space
tc-grps         = Protein Non-Protein     ; two coupling groups - more accurate
tau_t           = 0.1    0.1              ; time constant, in ps
ref_t           = 300    300             ; reference temperature, one for each group,in K

; Pressure coupling is on
pcoupl          = Parrinello-Rahman     ; pressure coupling is on for NPT
pcoupltype      = isotropic             ; uniform scaling of box vectors
tau_p           = 2.0                   ; time constant, in ps
ref_p           = 1.0                   ; reference pressure, in bar
compressibility = 4.5e-5                ; isothermal compressibility of water, bar^-1

; Periodic boundary conditions
pbc             = xyz                   ; 3-D PBC

; Dispersion correction
DispCorr        = EnerPres              ; account for cut-off vdW scheme

; Velocity generation
gen_vel         = no                    ; Velocity generation is off
gen_temp        = 300                   ; reference temperature, for protein in K
```


UNIVERSITAS
BERGENSIS