# GROMACS Introductory Tutorial

Gromacs ver 4.5

Author:      John E. Kerrigan, Ph.D.
                Associate Director, Bioinformatics
                The Cancer Institute of NJ
                195 Little Albany Street
                New Brunswick, NJ 08903

                Phone:         (732) 235-4473
                Fax:            (732) 235-6267
                Email:          kerrigje@umdnj.edu

**GROMACS Tutorial for Solvation Study of Spider Toxin Peptide.**

**Yu, H., Rosen, M. K., Saccomano, N. A., Phillips, D., Volkmann, R. A., Schreiber, S. L.:**
**Sequential assignment and structure determination of spider toxin omega-Aga-IVB.**
*Biochemistry* **32** *pp.* **13123 (1993)**

GROMACS is a high-end, high performance research tool designed for the study of protein dynamics using classical molecular dynamics theory.[1, 2] This versatile software package is Gnu public license free software. You may download it from http://www.gromacs.org . GROMACS runs on linux, unix, and on Windows (a recent development).

**Synopsis**. In this tutorial, you will study a toxin isolated from the venom of the funnel web spider. Venom toxins have been used in the past to identify cation channels. Calcium ion channels regulate the entry of this ion into cells. Nerve signals are highly governed by the balance of ions in neuronal cells. It is hypothesized that exposed positively charged residues in venoms like the spider toxin here bind favorably to the negatively charged entrance of the cell's ion channel. The spider toxin in this tutorial has its positively charged residues pointing predominantly to one side of the peptide. The ion channel is blocked, resulting in blocked nerve signal leading to paralysis and ultimately to death (presumably via respiratory failure).

We will study this peptide toxin using explicit solvation dynamics. First, we will compare an in vacuo model to a solvated model. We will solvate the peptide in a water box followed by equilibration using Newton's laws of motion. We will compare and contrast the impact of counterions in the explicit solvation simulation. We will seek answers to the following questions:

Is the secondary structure stable to the dynamics conditions?

Are the side chains of the positively charged residues predominantly displaced to one side of the peptide structure? Do the counterions hold these positively charged residues in place or do they move around?

What role does water play in maintaining the structure of proteins?

---

*Note*: You will generate structure files in this tutorial. To view these files, you must use VMD (Download from: http://www.ks.uiuc.edu/Research/vmd/ ). **In addition, you should obtain a copy of the GROMACS user manual at** http://www.gromacs.org .

Download 1OMB.PDB from the Protein Data Bank (http://www.rcsb.org/pdb/).

It is advisable to use DeepView (Download DeepView from http://www.expasy.ch/spdbv/ ) to preview the file if you know that your structure may be disordered (i.e. residues with missing side chains). DeepView will replace any missing side chains (However, beware as Deep View will mark those rebuilt side chains with a strange control character that can only be removed manually using a text editor!). There are no missing side chains in this pdb file, so we will not worry about that in this exercise.

Create a subdirectory in your unix account called "fwspider". Within this new directory, create the following directories: "invacuo", "wet", and "ionwet". Use sftp to copy the 1OMB.pdb file to the fwspider subdirectories (place a copy in each subdirectory within the fwspider directory). (IMPORTANT! Whenever you sftp a text file to a unix system from Windows, be sure to convert the file to a unix text file. Do this using the **to_unix** command (e.g. **to_unix** *filename filename* converts *filename* to a unix text file. In RedHat Linux, use the **dos2unix** command. Text editors in Windows like MS Word add control characters that may cause errors in unix programs.)

Process the pdb file with **pdb2gmx**. The **pdb2gmx** (to view the command line options for this command just type **pdb2gmx –h**; In fact, to get help for any command in Gromacs just use the **–h** flag) command converts your pdb file to a gromacs file and writes the topology for you. This file is derived from an NMR structure which contains hydrogen atoms. Therefore, we use the **–ignh** flag to ignore hydrogen atoms in the pdb file. The **–ff** flag is used to select the forcefield (G43a1 is the Gromos 96 FF, a united atom FF). The **–f** flag reads the pdb file. The **–o** flag outputs a new pdb file (various file formats supported) with the name you have given it in the command line. The **–p** flag is used for output and naming of the topology file. The topology file is very important as it contains all of the forcefield parameters (based upon the forcefield that you select in the initial prompt) for your model. Studies have revealed that the SPC/E water model [3] performs best in water box simulations. [4] Use the spce water model as it is better for use with the long-range electrostatics methods. So, we use the **–water** flag to specify this model.

**pdb2gmx –ignh –ff** gromos43a1 **–f** 1OMB.pdb **–o** fws.pdb **–p** fws.top **–water** spce

Setup the Box for your simulations

**editconf  -bt** octahedron **–f** fws.pdb **–o** fws-b4sol.pdb **–d** 1.0

What you have done in this command is specify that you want to use a truncated octahedron box. The **–d** 1.0 flag sets the dimensions of the box based upon setting the box edge approx 1.0 nm (i.e. 10.0 angstroms) from the molecule(s) periphery. Ideally you should set **–d** at no less than 0.9 nm for most systems. [5]

[Special Note:  **editconf** may also be used to convert gromacs files (*.gro) to pdb files (*.pdb) and vice versa. For example: **editconf –f** *file.gro* **–o** *file.pdb* converts *file.gro* to the pdb file *file.pdb*]

You may use the files generated from the this step to begin your in vacuo simulation. For the in vacuo work just move ahead to the energy minimization step followed by the molecular dynamics step (No position restrained dynamics necessary for in vacuo work. Why?).

Solvate the Box

**genbox –cp** fws-b4sol.pdb **–cs** spc216.gro **–o** fws-b4ion.pdb **–p** fws.top

The **genbox** command generates the water box based upon the dimensions/box type that you had specified using **editconf**. In the command above, we specify the spc water box. The **genbox**

program will add the correct number of water molecules needed to solvate your box of given dimensions.

**Preparation for running calculations**. The mdp file is used for controlling various settings in regard to force field, dynamics, periodic boundary conditions, etc., etc.

*Non-bonded interactions table* (From Berk Hess' presentation; Gromacs 2007 course).

| Force field | Neighb. List | Elec. Cut-off | PME Grid | VdW type | VdW cut-off | DispCorr |
|---|---|---|---|---|---|---|
| Gromos-UA | 1.0 | 1.0 | 0.135 | Shift | 0.9 | Yes |
| | | | | Cut-off | 1.4 | No |
| OPLS-AA | 0.9 | 0.9 | 0.125 | Shift | 0.8 | Yes |
| | | | | Cut-off | 1.4 | No |

<u>Notes</u>: The units for shift/cut-offs are nm. UA = united atom; AA = all-atom; DispCorr can only be used with periodic boundary conditions.

<u>Setup the energy minimization.</u>

Use the em.mdp file. Gromacs uses special *.mdp files to setup the parameters for every type of calculation that it performs. Look into the contents of this file. It specifies a steepest descents minimization to remove bad van der Waals contacts. Edit the file and change **nsteps** to 400. If the minimization fails to converge, re-submit with nsteps = 500. (The minimization should converge in less than 400 steps; however, different platforms respond differently.) To re-submit the job, you will need to re-run grompp. (<u>Note</u>: the path to the c pre-processor may be different on your machine. Use the **which** command to locate [i.e. which cpp]!)

<u>Content of em.mdp:</u>

```
define           =   -DFLEXIBLE
constraints      =   none
integrator       =   steep
dt               =   0.002     ; ps !
nsteps           =   400
nstlist          =   10
ns_type          =   grid
rlist            =   1.0
coulombtype      =   PME
rcoulomb         =   1.0
vdwtype          =   cut-off
rvdw             =   1.4
optimize_fft     =   yes
;
;      Energy minimizing stuff
;
emtol            =   1000.0
emstep           =   0.01
```

<u>Important aspects of the em.mdp file:</u>

**title** – The title can be any given text description (limit 64 characters; keep it short and simple!) Deprecated and no longer used in Gromacs 4.0.

**cpp** – location of the pre-processor. Deprecated: Gromacs 4.0 uses the default cpp for your system.

**define** – defines to pass to the pre-processor. –DFLEXIBLE will tell grompp to include the flexible SPC water model instead of the rigid SPC into your topology. This allows steepest descents to minimize further.

**constraints** – sets any constraints used in the model.

**integrator** – *steep* tells grompp that this run is a steepest descents minimization. Use *cg* for conjugate gradient.

**dt** – not necessary for minimization. Only needed for dynamics integrators (like **md**).

**nsteps** – In minimization runs, this is just the maximum number of iterations.

**nstlist** – frequency to update the neighbor list. nstlist = 10 (updates the list every 10 steps).

**rlist** – cut-off distance for short-range neighbor list.

**coulombtype** – tells gromacs how to model electrostatics. PME is particle mesh ewald method (please see the Gromacs user manual for more information).

**rcoulomb** – distance for the coulomb cut-off

**vdwtype** – tells Gromacs how to treat van der Waals interactions (cut-off, Shift, etc.)

**rvdw** – distance for the LJ or Buckingham potential cut-off

**fourierspacing** – Used to automate setup of the grid dimensions (`fourier_nx`, ...) for pme.

*EM Stuff*

**emtol** – the minimization converges when the max force is smaller than this value (in units of kJ $mol^{-1}$ $nm^{-1}$)

**emstep** – initial step size (in nm).

Now process the files with **grompp**. **grompp** is the pre-processor program (the gromacs pre-processor "grompp" Get it! Sigh!). **grompp** will setup your run for input into **mdrun**.

**grompp** –**f** em.mdp –**c** fws-b4ion.pdb –**p** fws.top –**o** ion.tpr –**maxwarn** 5

The –**f** flag in grompp is used to input the parameter file (*.mdp). The –**c** flag is used to input the coordinate file (the pdb file, *.pdb); -**p** inputs the topology and –**o** outputs the input file (*.tpr) needed for **mdrun**.

---

**Using genion and the tpr file to add ions**. You may use the tpr file generated here to add counterions to your model to neutralize any net charge. *In order for the Ewald equation we are using to describe long range electrostatics in our model to be valid, the net system charge must be neutral.* Our model has a net charge of +2.00. Therefore, we want to add two chloride ions. Copy the fws_em.tpr file that you used for your explicit solvated model to your "ionwet" subdirectory. In addition, copy your fws.top and posre.itp files from your explicit solvation model to your ionwet subdirectory. Use the genion command to add the chloride ions.

**genion** –**s** ion.tpr –**o** fws-b4em.pdb –**nname** CL –**nn** 2 –**p** fws.top –**g** ion.log

---

Where **–nname** is the negative ion name (CL for the Gromos G43a1 force field; see the ions.itp file for specifics wrt force field), **-nn** is the number of negative ions to add. Use **–pname** to add positively charged ions and **–np** to specify the number of positively charged ions to add. The **–g** flag gives a name to the output log for genion. An even easier method which creates a neutral system using a specified concentration of NaCl uses the **–neutral** and **–conc** flags as in the following …

**genion –s** ion.tpr **–o** fws-b4em.pdb **–neutral -conc** 0.15 **–p** fws.top **–g** ion.log

We will use the 0.15 M NaCl model via the command above. The **–norandom** flag places ions based on electrostatic potential as opposed to *random* placement (default). However, we will use the default (random placement). When you process this command, you will be prompted to provide a continuous group of solvent molecules, which should be Group 13 (SOL). Type 13 then <enter>. You will notice that a number of solvent molecules have been replaced by Na and Cl ions. You should also notice that the fws.top file has been updated to include the NA and CL ions in your topology accounting for the replaced water molecules.

*Caution*: The molecules listed here must appear in the exact same order as in your coordinate file!

---

Temperature coupling scheme for the pr_md.mdp and md.mdp files after adding chloride ions.
```
; Berendsen temperature coupling using velocity rescaling is on
Tcoupl              =  v-rescale
tau_t               =  0.1          0.1
tc_grps             =  protein      non-protein
ref_t               =  300          300
```

---

<u>Remember</u>: If you added the chloride ions, you will need to run the grompp step again. First remove the old fws_em.tpr file, then run the next grompp command below. We added chloride ions in our model to neutralize the overall net charge of the model.

**grompp –f** em.mdp **–c** fws-b4em.pdb **–p** fws.top **–o** em.tpr **–maxwarn** 5

Run the energy minimization and watch its progress.

**mdrun –v –deffnm** em

Use the **tail** command to check on the progress of the minimization.

**tail** –15 em.log

When the minimization is complete, you should see the following summary in your log file indicating that steepest descents converged. Use **tail -100** em.log

```
Steepest Descents converged to Fmax < 1000 in 240 steps
Potential Energy  = -1.9688684e+05
Maximum force     =  9.9480914e+02 on atom 106
Norm of force     =  5.4587994e+01
```

*Setup the position-restrained (PR) molecular dynamics*.  What is position-restrained MD?  You are restraining (or partially freezing, if you will) the atom positions of the macromolecule while letting the solvent move in the simulation.  This is done to "soak" the water molecules into the macromolecule.  The relaxation time of water is ~ 10 ps.  Therefore, we need to do dynamics in excess of 10 ps for position-restrained work.  This example uses a 50.0 ps time period (at least an order of magnitude greater is preferred).  Larger models (large proteins/lipids) may require longer pre-equilibration periods of more than 100 ps.  An even more rigorous approach uses one 100 ps equilibration to turn on the temperature coupling (NVT) followed by a separate 100 ps equilibration to additionally introduce the pressure coupling (NPT).  We will use a single 50.0 ps NPT equilibration to save time. The settings below are recommended for the Gromacs/Gromos forcefields.  Under **coulombtype**, PME stands for "Particle Mesh Ewald" electrostatics. [6, 7]  PME is the best method for computing long-range electrostatics (gives more reliable energy estimates especially for systems where counterions like $Na^+$, $Cl^-$, $Ca^{2+}$, etc. are used).  Due to the nature of this particular protein having exposed charged residues with a net +2 charge to the system, it is beneficial for us to use PME.  It is even more beneficial for us to use counterions to balance the charge and set the system to net neutral; otherwise, PME will not give reliable results.  The **all-bonds** option under **constraints** applies the Linear Constraint algorithm for fixing all bond lengths in the system (important to use this option when dt > 0.001 ps). [8]  Study the mdp file below.

Content of pr.mdp.

```
define              =   -DPOSRES
constraints         =   all-bonds
integrator          =   md
dt                  =   0.002  ; ps !
nsteps              =   25000  ; total 50.0 ps.
nstcomm             =   10
nstxout             =   500    ; collect data every 1.0 ps
nstxtcout           =   500
nstvout             =   5000
nstfout             =   0
nstlog              =   10
nstenergy           =   10
nstlist             =   10
ns_type             =   grid
rlist               =   1.0
coulombtype         =   PME
rcoulomb            =   1.0
vdwtype             =   cut-off
rvdw                =   1.4
pme_order           =   4   ; Use 6, 8 or 10 when running in parallel
ewald_rtol          =   1e-5
optimize_fft        =   yes
DispCorr            = no
; Berendsen temperature coupling is on
Tcoupl              =   v-rescale
tau_t               =   0.1    0.1
tc-grps             =   protein non-protein
ref_t               =   300    300
; Pressure coupling is on
Pcoupl              =   parrinello-rahman
```

```
Pcoupltype            =   isotropic
tau_p                 =   1.0
compressibility       =   4.5e-5
ref_p                 =   1.0
; Generate velocites is on at 300 K.
gen_vel               =   yes
gen_temp              =   300.0
gen_seed              =   -1
```

Important things to know about the mdp file.

The –DPOSRES in the **define** statement tells Gromacs to perform position restrained dynamics.

The **constraints** statement is as previously discussed. *all-bonds* sets the LINCS constraint for all bonds. [8]

The **integrator** tells gromacs what type of dynamics algorithm to use (another option is "sd" for stochastic dynamics).

**dt** is the time step (we have selected 2 fs; however, this must be entered in units of ps!).

**nsteps** is the number of steps to run (just multiply nsteps X dt to compute the length of the simulation).

**nstxout** tells gromacs how often to collect a "snapshot" for the trajectory. (e.g. nstxout = 250 with dt = 0.002 would collect a snapshot every 0.5 ps)

**coulombtype** selects the manner in which Gromacs computes electrostatic interactions between atoms. (PME is particle mesh ewald; there are other options like cut-off).

**rcoulomb** and **rvdw** are the cutoffs (in nm; 1.0 nm = 10.0 angstroms) for the electrostatic and van der Waals terms.

The temperature coupling section is very important and must be filled out correctly.

**Tcoupl** = v-rescale [9, 10] (type of temperature coupling where velocity is rescaled using a stochastic term.)

**tau_t** = Time constant for temperature coupling (units = ps). You must list one per tc_grp in the order in which tc_grps appear.

**tc_grps** = groups to couple to the thermostat (every atom or residue in your model must be represented here by appropriate index groups).

**ref_t** = reference temperature for coupling (i.e. the temperature of your MD simulation in degrees K). You must list one per tc_grp in the order in which tc_grps appear.

When you alter the temperature, don't forget to change the **gen_temp** variable for velocity generation.

**pcoupl** – is the Parrinello-Rahman barostat.
**pcoupltype** – isotropic means that the "box" will expand or contract evenly in all directions (x, y, and z) in order to maintain the proper pressure. <u>Note</u>: Use semiisotropic for membrane simulations.
**tau_p** – time constant for coupling (in ps).
**compressibility** – this is the compressibility of the solvent used in the simulation in $bar^{-1}$ (the setting above is for water at 300 K and 1 atm).
**ref_p** – the reference pressure for the coupling (in bar) (1 atm ~ 0.983 bar).
**gen_seed** = -1   Use random number seed based on process ID #.  More important for Langevin dynamics.[11]

Pre-process the pr.mdp file.

**grompp –f** pr.mdp **–c** em.gro **–p** fws.top **–o** pr.tpr **–maxwarn** 5

**nohup mdrun –deffnm** pr **&**

Use the **tail** command to check the pr.log file.  Our 50 ps run completed in 3 minutes, 35 seconds on a 2.66 GHz Core 2 Quad machine using all 4 cores.

The md.mdp parameter file looks very similar to the pr.mdp file.  There are several differences.  The define statement is not necessary as we are not running as position restrained dynamics.

<u>Content of md.mdp for explicit solvation</u>

```
constraints         =   all-bonds
integrator          =   md
dt                  =   0.002  ; ps !
nsteps              =   250000 ; total 500 ps.
nstcomm             =   10
nstxout             =   500     ; collect data every 1 ps
nstxtcout           =   500
nstvout             =   0
nstfout             =   0
nstlist             =   10
ns_type             =   grid
rlist               =   1.0
coulombtype         =   PME
rcoulomb            =   1.0
vdwtype             =   cut-off
rvdw                =   1.4
pme_order           =   4
ewald_rtol          =   1e-5
optimize_fft        =   yes
DispCorr            = no
; Berendsen temperature coupling is on
Tcoupl              =   v-rescale
tau_t               =   0.1    0.1
```

```
tc-grps              =  protein    non-protein
ref_t                =  300        300
; Pressure coupling is on
Pcoupl             =  parrinello-rahman
Pcoupltype         =  isotropic
tau_p              =  1.0
compressibility    =  4.5e-5
ref_p              =  1.0
; Generate velocites is on at 300 K.
gen_vel            =  yes
gen_temp           =  300.0
gen_seed           =  -1
```

**grompp –f** md.mdp **–c** pr.gro **–p** fws.top **–o** md.tpr **–maxwarn** 3

**nohup mdrun –deffnm** md **&**

Use the **tail** command to check the md.log file. (our 500 ps run finished in 50 minutes on a 2.66 GHz Core 2 Quad machine with all 4 cores.)

You may compress the trajectory using **trjconv** to save on disk space.  Note – we set nstxtcout = 500; therefore, you should already have a compressed copy of your trajectory.  However, for display purposes, you might need to use " **–pbc** nojump" to keep everything in the box.

**trjconv –f** md.trr **–s** md.tpr **–o** md.xtc **–pbc** nojump

Once you have made the *.xtc file, you may delete the *.trr file.

Use **ngmx** to view the trajectory (you may also download to your PC and use VMD to view the trajectory).
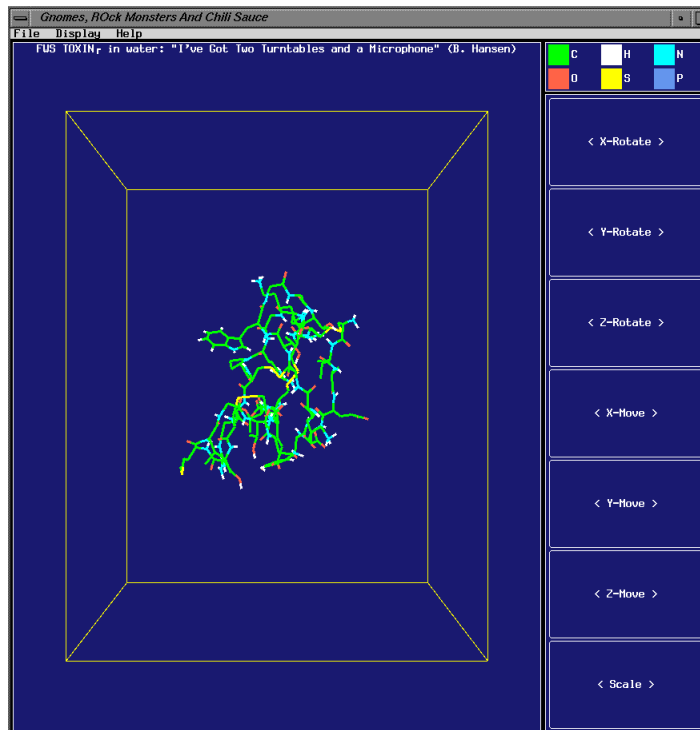
**ngmx –f** md.trr (or md.xtc) **–s** md.tpr

When the viewer comes up, you will see a dialog box with a number of selections.  Check the box labeled "Protein", then click <u>Ok</u>.
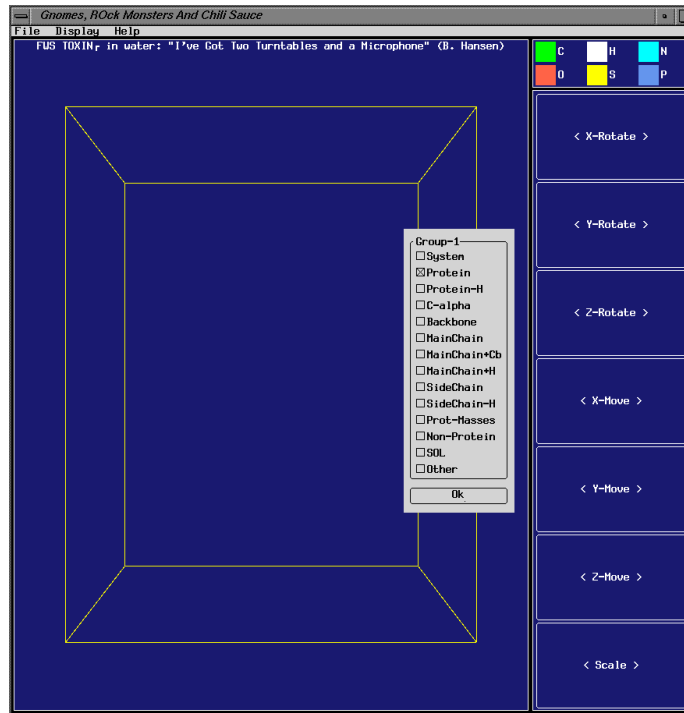
ngmx's initial startup dialog.



Selecting "Protein" permits us to view the protein only without interference from the ~3,000 water molecules that occupy the rest of the box.

Use **X-Rotate** to rotate the box up and down (left mouse click goes up; right mouse click goes down). Use **Y-Rotate** to rotate the box from left to right or vice versa (left mouse click rotates to the left and right mouse click rotates to the right). **Scale** at the bottom permits you to zoom in and zoom out (use left mouse button to zoom in and right mouse button to zoom out).

To view another group in the model go to **Display > Filter …** and the initial dialog box will come up again and you may opt to view some other index group (e.g. the backbone).

To view the animation of the MD trajectory go to **Display > Animate**. Controls used for the playback of the trajectory appear at the bottom of the window. Use the center arrow button to view one time step at time. Use the forward double arrow to play the whole trajectory, and use the "pause" button to the right to stop the trajectory playback. Use the double arrow to the left to reset the animation.
Unfortunately, the save as pdb option under the File menu in ngmx does not work. Therefore…
*The best way to view the trajectory and select snapshots to save as *.PDB files is to use visual molecular dynamics (VMD).* (Download from: http://www.ks.uiuc.edu/Research/vmd/ It's free to academics and it runs on unix and windows!).

**Analysis**

One of the major advantages of Gromacs (other than the fact that it is GNU public domain and free!) is the robust set of programs available for analyzing the trajectories. We will discuss a few of the more important analytic tools in the Gromacs arsenal here.

Groups

**make_ndx**

The make_ndx program is useful for generating groups (ID tags for specific atoms or residues that you may want to analyze). Gromacs generates default groups which may be adequate for your work as is. However, if you need to do more in depth analysis, use make_ndx to tag specific items in your model. See the manual for more information.

How to use **make_ndx** to setup index (ndx) files. Use make_ndx to identify particular groups that you might want to freeze during a simulation or gather special energy information. Let's look at an example where we want to freeze the N and C terminal amino acids of a protein. Always use make_ndx to create an index group for use with the **grompp** program.

In this sample case, we have a coordinate file of a collagen triple helix, post position restrained dynamics, where we want to freeze the N & C terminal for production run. We must first identify the residue #'s in the coordinate file (In our case clg_b4md.pdb) that identify the N & C termini. The command line is simple enough …

**make_ndx –f** clg_b4md.pdb **–o** clg_ter.ndx

GROMACS Tutorial

You will see the following output (we left out the descriptive info at the beginning) followed by a command prompt (>).

```
Reading structure file
Going to read 0 old index file(s)
Analysing residue names:
Opening library file /usr/share/gromacs/top/aminoacids.dat
There are:  2194     OTHER residues
There are:   108   PROTEIN residues
There are:     0       DNA residues
Analysing Protein...
Analysing Other...

  0 System              :  7365 atoms
  1 Protein             :   765 atoms
  2 Protein-H           :   687 atoms
  3 C-alpha             :   108 atoms
  4 Backbone            :   324 atoms
  5 MainChain           :   435 atoms
  6 MainChain+Cb        :   507 atoms
  7 MainChain+H         :   477 atoms
  8 SideChain           :   288 atoms
  9 SideChain-H         :   252 atoms
 10 Prot-Masses         :   765 atoms
 11 Non-Protein         :  6600 atoms
 12 DRG                 :    21 atoms
 13 SOL                 :  6579 atoms
 14 Other               :  6600 atoms

 nr : group       !   'name' nr name   'splitch' nr    Enter: list groups
 'a': atom        &   'del' nr         'splitres' nr   'l': list residues
 't': atom type   |   'keep' nr        'splitat' nr    'h': help
 'r': residue         'res' nr         'chain' char
 "name": group        'case': case sensitive          'q': save and quit

>
```

Use the 'r' command to enter the list of residue numbers that represent the N & C termini of the triple helix.

```
> r 1 36 37 72 73 108


 15 r_1_36_37_72_73_108 :    51 atoms
```

**Note:** You may also use a dash to specify a residue number range (e.g. to specify residues 1 to 36 use > r 1-36) … OR, better yet, lets specify a residue range only including backbone atoms.  Do this with the ampersand for example …

```
> r 1-36 & a C N CA
```

The default name (r_1_36_37_72_73_108 ) giving to the new index group that you have just created is cumbersome. Lets rename it using the name command. We will use the index group # (15) in the command.

```
> name 15 Terminal


> v

Turned verbose on

  0 System               :  7365 atoms
  1 Protein              :   765 atoms
  2 Protein-H            :   687 atoms
  3 C-alpha              :   108 atoms
  4 Backbone             :   324 atoms
  5 MainChain            :   435 atoms
  6 MainChain+Cb         :   507 atoms
  7 MainChain+H          :   477 atoms
  8 SideChain            :   288 atoms
  9 SideChain-H          :   252 atoms
 10 Prot-Masses          :   765 atoms
 11 Non-Protein          :  6600 atoms
 12 DRG                  :    21 atoms
 13 SOL                  :  6579 atoms
 14 Other                :  6600 atoms
 15 Terminal             :    51 atoms


 nr : group       !   'name' nr name    'splitch' nr    Enter: list groups
 'a': atom        &   'del' nr          'splitres' nr   'l': list residues
 't': atom type   |   'keep' nr         'splitat' nr    'h': help
 'r': residue         'res' nr          'chain' char
 "name": group        'case': case sensitive           'q': save and quit


>
```

We used the 'v' command to verify that our name change was successful. To finish up, use the 'q' command to save and quit and you're done.

Now how do we freeze the groups? Easy, add the following lines to your md.mdp file

```
energygrp_excl          =  Terminal Terminal Terminal SOL  ! To remove
computation of nonbonding interactions between the frozen groups with each other
and surroundings (i.e. the solvent, SOL)

freezegrps              =  Terminal  ! Index group to freeze
freezedim               =  Y Y Y     ! Freeze this group in all directions, x,
y, and z
```

Remember, you must include the new index file when you use this md.mdp file to create your run input file (tpr) using grompp. Use the –n flag to grompp. For example …

**grompp** –**f** md.mdp –**c** pr.gro –**p** clg.top –**n** clg_ter.ndx –**o** md.tpr

GROMACS Tutorial

Make a nice movie.  First remove high frequency noise with g_filter.  Perform in a separate directory named "movie"

**g_filter –s** ../md.tpr **–f** ../md.trr **–ol** filtered.pdb **–fit –nf** 5

Select 1 (Protein) when prompted for group. Load into pymol and use commands for display (set orientation, etc)

**pymol** filtered.pdb

dss (set secondary structure)
show cartoon
viewport 640,800
set ray_trace_frames,1
mpng frame_.png

Use an mpeg encoder to make the movie.  Alternatively, you can use the Linux ImageMagik "convert" command to build an animated gif file.

```
convert -delay 15 -loop 0 frame_*.png movie.gif
```

Properties

**g_confrms**

To compare the final structure to the original PDB file obtained from the PDB use g_confrms (for a description use **g_confrms –h**).  This program takes two structures and performs a least squares fit.

**g_confrms –f1** 1OMB.pdb **–f2** md.gro **–o** fit.pdb

You will be prompted to select a group (select the Backbone (group 4) both times).  The program reports an RMS deviation and produces an output file (fit.pdb).  The output file will show the two structures superimposed on each other.

**g_covar**

Use to compute the covariance matrix (see manual for more info).  The g_covar program may also be used to compute an average structure from a dynamics trajectory.  For example to compute the average of the last 200 ps of a 1 ns dynamics run use…

**g_covar –f** traj.xtc **–s** topol.tpr **–b** 800 **–e** 1000 **–av** traj_avg.pdb

Warning – Average structures tend to be crude and are not necessarily meaningful in a physical sense taken "as is".  Refinement using energy minimization is recommended.

**g_energy**

GROMACS Tutorial

Use this program to plot energy data, pressure, volume, density, etc.

**g_energy –f** md.edr **–s** md.tpr **–o** fws_pe.xvg

You will be prompted for the specific data that you want to include in the output file (*.xvg). The output file is a type of spreadsheet file that can be read using Xmgr or Grace. This file is a text file and can be read into Microsoft Excel; however, you will need to do some light editing of the file.

For example in our case above (your results may be different), we see the following …

```
Select the terms you want from the following list by
selecting either (part of) the name or the number or a combination.
End your selection with an empty line or a zero.
-------------------------------------------------------------------
  1  G96Angle        2  Proper-Dih.      3  Improper-Dih.    4  LJ-14
  5  Coulomb-14      6  LJ-(SR)          7  LJ-(LR)          8  Coulomb-(SR)
  9  Coul.-recip.   10  Potential       11  Kinetic-En.     12  Total-Energy
 13  Temperature    14  Pressure        15  Constr.-rmsd    16  Box-X
 17  Box-Y          18  Box-Z           19  Volume          20  Density
 21  pV             22  Enthalpy        23  Vir-XX          24  Vir-XY
 25  Vir-XZ         26  Vir-YX          27  Vir-YY          28  Vir-YZ
 29  Vir-ZX         30  Vir-ZY          31  Vir-ZZ          32  Pres-XX
 33  Pres-XY        34  Pres-XZ         35  Pres-YX         36  Pres-YY
 37  Pres-YZ        38  Pres-ZX         39  Pres-ZY         40  Pres-ZZ
 41  #Surf*SurfTen  42  Box-Vel-XX      43  Box-Vel-YY      44  Box-Vel-ZZ
 45  Mu-X           46  Mu-Y            47  Mu-Z            48  T-Protein
 49  T-non-Protein                      50  Lamb-Protein
 51  Lamb-non-Protein
```

For Potential energy, type  10 <enter>
Hit the <enter> key again

We get a summary with average PE and RMSD in kJ/mol

```
Statistics over 250001 steps [ 0.0000 through 500.0000 ps ], 1 data sets
All statistics are over 25001 points

Energy                      Average   Err.Est.      RMSD  Tot-Drift
-------------------------------------------------------------------------
Potential                    -174820        21   396.552   47.3972  (kJ/mol)
```

To read *.xvg files into Grace use the following command:

**xmgrace -nxy** fws_pe.xvg

You may obtain Grace from http://plasma-gate.weizmann.ac.il/Grace/ . Grace runs on Linux and in Unix only. *If you do not have Grace or Xmgr, just import or read the *.xvg file into MS Excel as a space delimited file.*
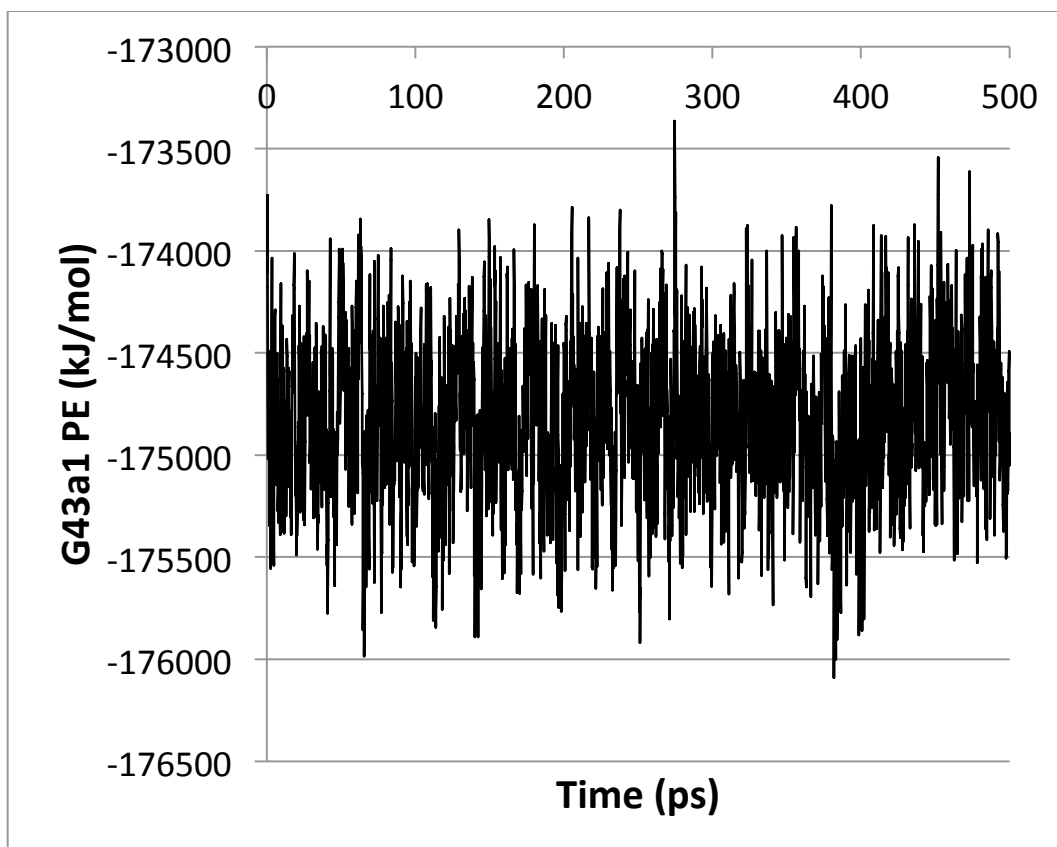
Potential energy plot of whole system.

**g_gyrate**

Use **g_gyrate** to measure the radius of gyration. This quantity gives a measure of the "compactness" of the structure. This gives a measure of the mass of the atom(s) relative the center of mass of the molecule.

**g_gyrate** –**f** md.trr –**s** md.tpr –**o** fws_gyrate.xvg

**g_rms and g_rmsdist**

These programs are useful for measuring root mean square deviations in the structure. Use **g_rms** to evaluate the deviation of the structure from the original starting structure over the course of the simulation. (The –**dt** 10 option tells the program to write every tenth frame)

**g_rms** –**s** *.tpr –**f** *.xtc –**dt** 10

To compare rmsd to the NMR structure use the following command …

**g_rms** –**s** em.tpr –**f** md.trr –**o** fws_rmsd.xvg

Select group 4 (Backbone) for the least squares fit. The program will generate a plot of rmsd over time (rmsd.xvg) which you may import into Excel as a space delimited file.

The **g_rmsdist** tool has the advantage that it computes the RMSD of interatomic distances. No LSQ fit is needed. The initial structure is taken as the reference. Compare the output of the g_rmsdist to g_rms (select group 4 again).

**g_rmsdist** –**s** em.tpr –**f** md.trr –**o** fws_rms-dist.xvg

**g_rmsf**

Computes root mean square fluctuation of atom positions. Like **g_covar** this command may be used to compute average structures. For example, to calculate the average of the last 500 ps of a 2 ns (2000 ps) dynamics simulation, use the following…

**g_rmsf** –**f** traj.xtc –**s** topol.tpr –**b** 1501 –**e** 2000 –**o** traj_rmsf.xvg –**ox** traj_avg.pdb

Select a range where you observe some equilibrium from your RMSD plot (computed using **g_rms**). For example …
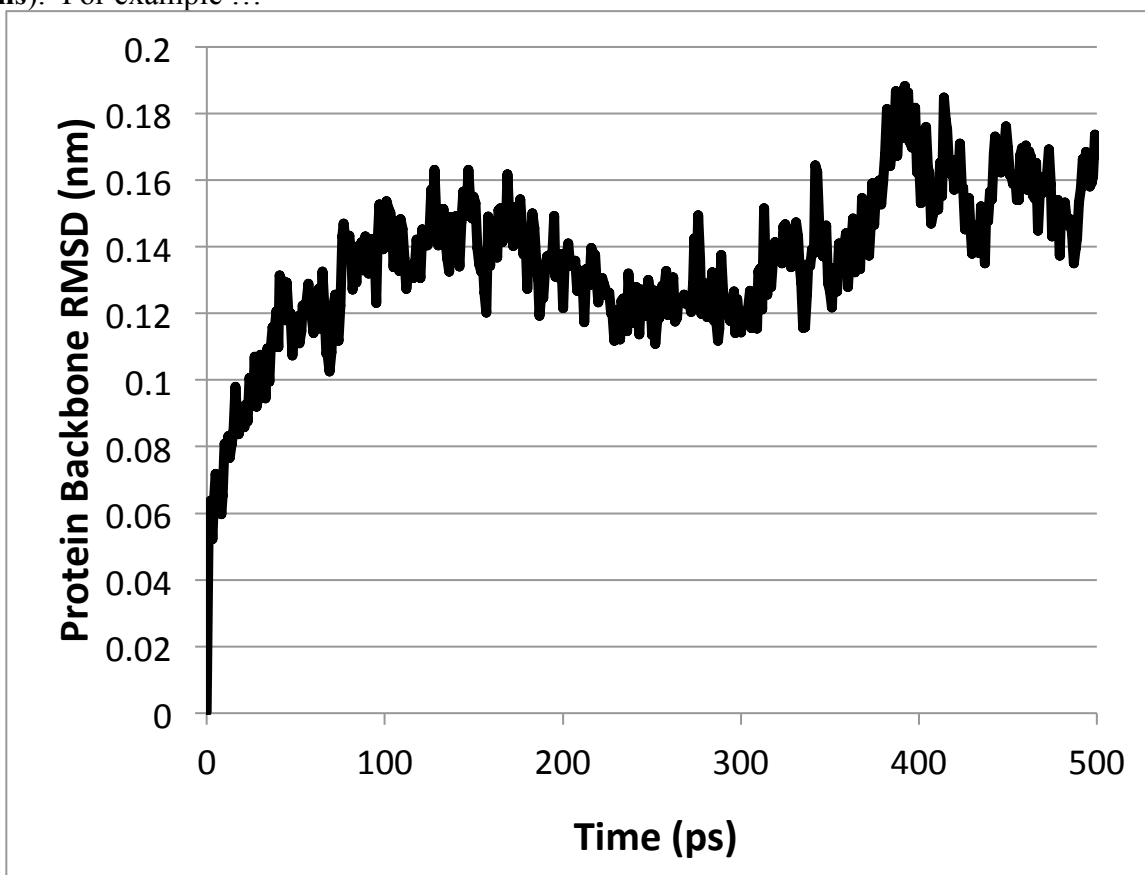


The example above is the result of our 500 ps simulation (your results might be different). In the example above, we might use the 400 to 500 ps range for computing an average structure as we observe a leveling off in deviation from the reference (initial structure). The command would be …

**g_rmsf –s** md.tpr **–f** md.trr **–b** 400 **–e** 500 **–ox** fws_avg.pdb

Select group 1 "Protein" when prompted.

Warning – Average structures tend to be very crude (especially hydrogen atom positions which are the most variable).  Minimization is recommended.

---

**An Aside: Has my Protein Equilibrated after my Production Run?**

This is an often-asked question on discussion lists.  The answer really depends upon the property (ies) you are studying.  Overall potential energy tends to be noisy and is a deceptive measure of system equilibration.  Protein backbone RMSD will give an indication of stability in the overall backbone, but what about a loop region?  You might need to zero-in on the region of interest in your analysis to be sure.  You will also need to know the approximate time scale of the property you are studying to really know whether or not you have run your production dynamics for a long enough period of time (for example, protein folding events can run the gamut of a few microseconds to minutes and even as long as hours in terms of timescales!).

---

Suggested em.mdp files setup for an *in vacuo* minimization.  Perform steepest descents first followed by conjugate gradient.  CAUTION! You may need to run **pdb2gmx** to setup a new set of topology files for your minimization especially if you have isolated a specific group in computing your average structure (as opposed to the entire system).

```
;   Steepest Descents in vacuo
;       User kerrigan (236)
;
;       Input file
;
constraints         =   none
integrator          =   steep
nsteps              =   400
;
;       Energy minimizing stuff
;
emtol               =   1000
emstep              =   0.01

nstcomm             =   1
ns_type             =   grid
morse               =   no
coulombtype         =   Shift
vdw_type            =   Shift
rlist               =   1.4
rcoulomb            =   1.2
rvdw                =   1.2
rcoulomb_switch     =   1.0
rvdw_switch         =   1.0
Tcoupl              =   no
Pcoupl              =   no
gen_vel             =   no
```

```
;  Conjugate gradient with morse potential
;   in vacuo
;     User kerrigan (236)
;
;     Input file
;
constraints         =   none
integrator          =   cg
nsteps              =   3000
;
;     Energy minimizing stuff
;
emtol               =   100
emstep              =   0.01
nstcgsteep          =   1000

nstcomm             =   1
morse               =   yes
coulombtype         =   Shift
vdw_type            =   Shift
ns_type             =   grid
rlist               =   1.4
rcoulomb            =   1.2
rvdw                =   1.2
rcoulomb_switch     =   1.0
rvdw_switch         =   1.0
Tcoupl              =   no
Pcoupl              =   no
gen_vel             =   no
```

The **g_rmsf** command may also be used to compute temperature factors. The computed *temperature factors* can be compared to x-ray crystal structure temperature factors. Use **–res** flag to get average per residue.

**g_rmsf –s** md.tpr **–f** md.xtc **–o** rmsf.xvg **–ox** average.pdb **-res –oq** bfactors.pdb

Use the "Protein" group. Using Pymol to view …

**pymol** bfactors.pdb

hide everything

show cartoon
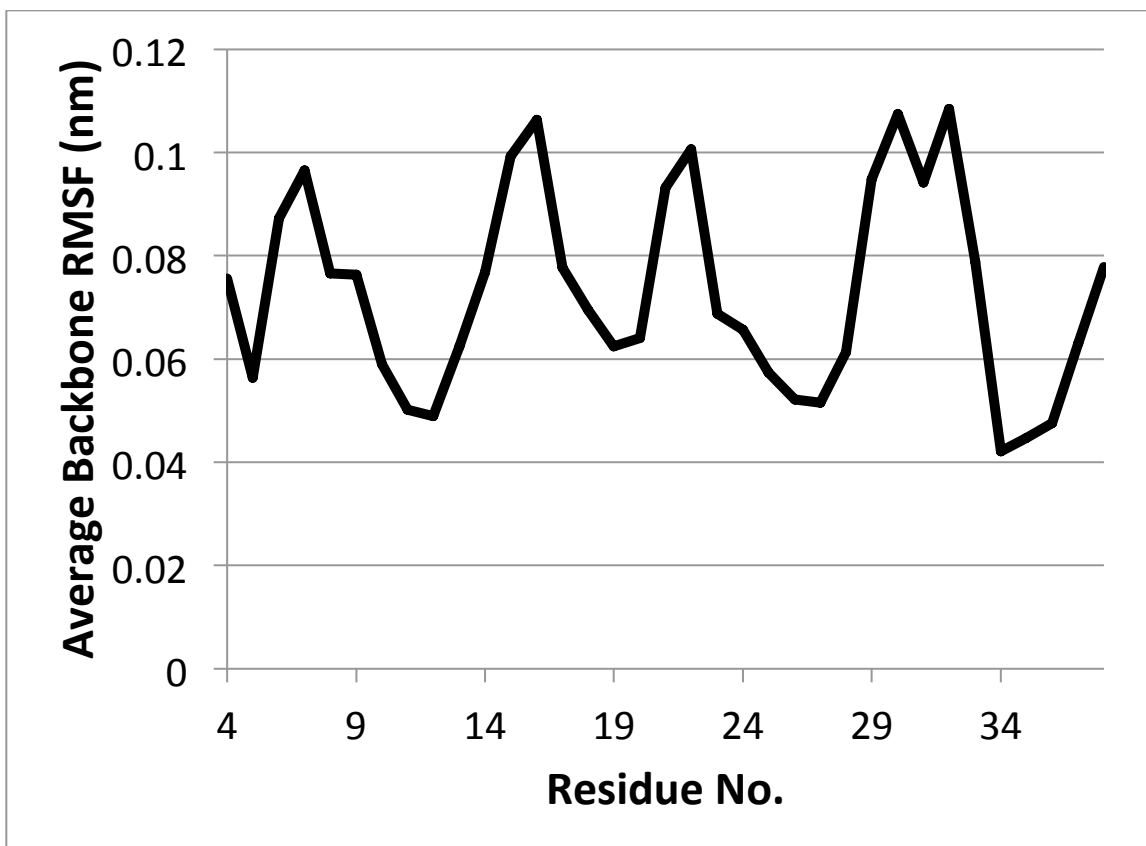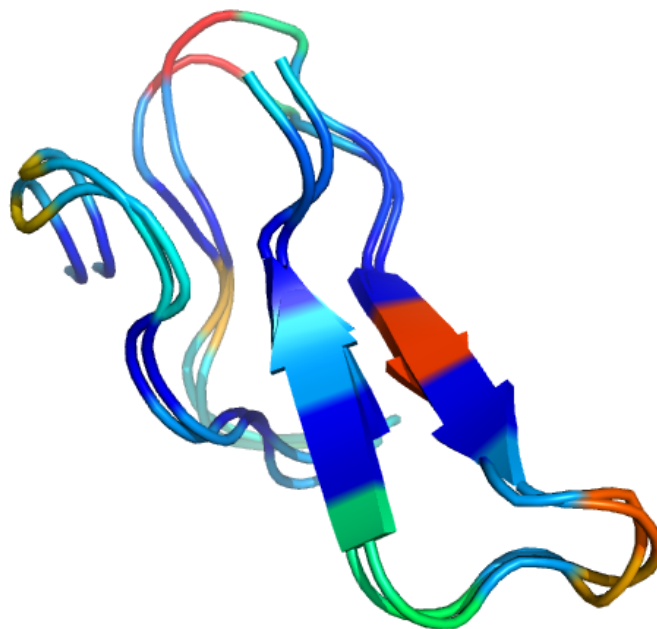
spectrum b, selection=bfactors

Q = XXX; cmd.alter("all", "q = b > Q and Q or b"); spectrum q

Where XXX is the highest bfactor value truncated.

Blue is cool regions; Green intermediate and Red is hot (most mobile)

**do_dssp**

Use the do_dssp command to analyze the secondary structure of your model.  You must have the dssp program (http://swift.cmbi.ru.nl/gv/dssp/ ) installed (in /usr/local/bin). [12]

**do_dssp –s** md.tpr **–f** md.trr **–b** 400 **–e** 500 **–o** fws_ss.xpm

Select Group 1 (Protein) for the calculation.  Use the xpm2ps program to convert the xpm file to eps format.  Then use the ImageMagick convert program to convert the eps file to png or whatever desired image format (For best results use Ghostview/Ghostscript).

**xpm2ps –f** fws_ss.xpm **–o** fws_ss.eps
**ghostview** fws_ss.eps
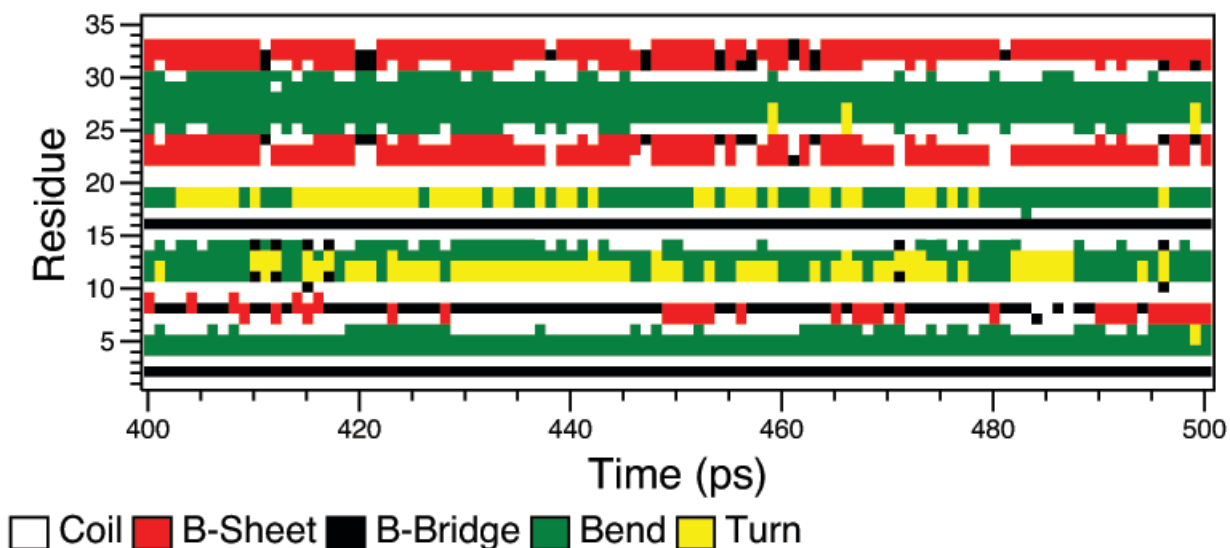


Where residue # is on the y axis and time (ps) is on the x-axis.  As seen from the original NMR structure below …



From the dssp plot above, we see 3 regions of red for the 3 sheets (displayed in yellow in the structure image to the left).   The one short region in the middle is the least stable.   The image to the left was prepared using the VMD program.

**g_hbond**

Use **g_hbond** to measure the number of hydrogen bonds, hbond distances & angles between molecules or groups through the course of the simulation.

**g_hbond** –**f** md.trr –**s** md.tpr –**num** fws_hnum.xvg



Geometrical relations used by g_hbond.
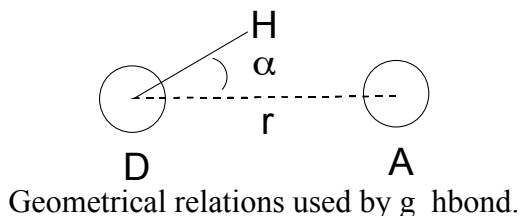
The defaults in Gromacs 4.5 are:

$r \leq 0.35 nm$

$\alpha \leq 30°$

Use the –**r** and –**a** flags to set other limits. By default, g_hbond analyzes the donor-acceptor ($r_{DA}$) distance). You may change this behaviour by setting the **-da** flag to "no". Setting **-da** to "no" instructs g_hbond to analyze the $r_{HA}$ distance.

See the Gromacs manual for more options.

**g_saltbr**

Use **g_saltbr** to analyze salt bridges between residues in your simulation. The program outputs a set of *.xvg files, which give distances between minus/minus, charged residues plus/minus (the most interesting) and plus/plus charged residues.

**g_saltbr** –**f** md.trr –**s** md.tpr

Cluster analysis

**g_rms** –**s** ../md.tpr –**f** ../md.trr –**f2** ../md.trr –**m** rmsd-matrix.xpm

**g_cluster** –**s** ../md.tpr –**f** ../md.trr –**dm** rmsd-matrix.xpm –**dist** rms-distribution.xvg –**o** clusters.xpm –**sz** cluster-sizes.xvg –**tr** cluster-transitions.xpm –**ntr** cluster-transitions.xvg –**clid** cluster-id-over-time.xvg –**cl** clusters.pdb –**cutoff** 0.25 –**method** gromos

**pymol** clusters.pdb
split_states clusters
delete clusters
dss
show cartoon

GROMACS Tutorial

How To Save – specific time points from a trajectory as *.PDB files:

To get a specific frame (3000 ps in this example) instead of the whole trajectory as a pdb file use additionally the **–dump** option, e.g.

**trjconv -f** traj.xtc **-s** file.tpr **-o** time_3000ps.pdb **-dump** 3000

To re-center your molecule back in the box use ...

**trjconv –f** traj.xtc **–o** traj_center.xtc **–s** str_b4md.gro[#] **–pbc** nojump **-center**

[#]Or use initial md.tpr file for a reference.
Perform other analysis tasks as recommended by your instructor.

**Principal Components Analysis (PCA)**

PCA methods help us determine what motions contribute most to the overall dynamics of the protein. In a system of N atoms, there exist 3N-6 modes of possible internal fluctuations (six degrees of freedom are required to describe the external rotation and translation of the system). For this analysis, we will focus on the protein backbone atoms. Perform in a separate directory.

**g_covar –s** ../md.tpr **–f** ../md.xtc **–o** eigenval.xvg **–v** eigenvect.trr **–xpma** covara.xpm

Select group "4" (Protein backbone) both for fit and analysis.

```
Trace of the covariance matrix: 0.910307 (nm^2)

Diagonalizing ...

Sum of the eigenvalues: 0.910307 (nm^2)

Writing eigenvalues to eigenval.xvg

Writing reference, average structure & eigenvectors 1--315 to eigenvec.trr
Wrote the log to covar.log
```
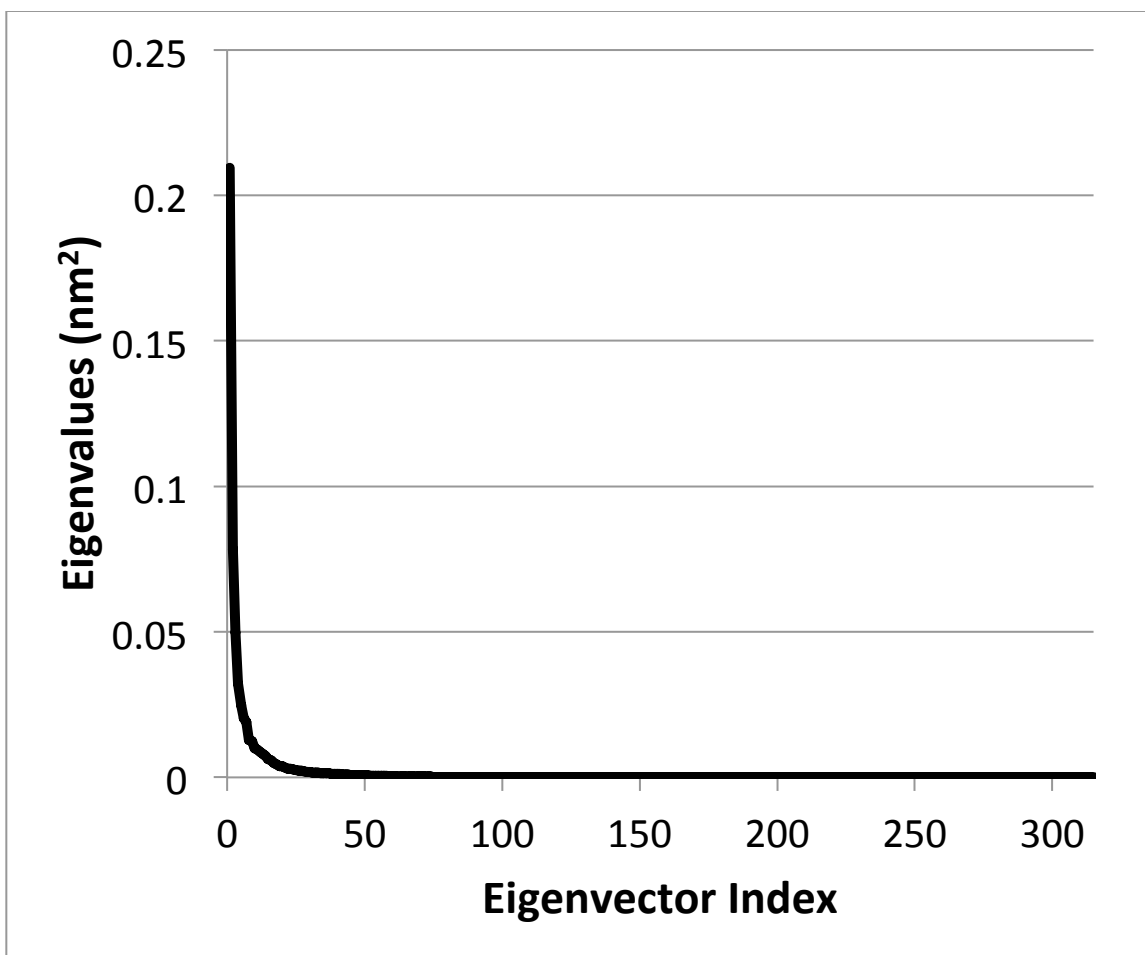
Use **xpm2ps** to make a pretty plot of the atomic covariance matrix. The **–do** flag outputs a config file which can be used to modify properties of the plot (axis titles, legend, etc.).

**xpm2ps –f** covara.xpm **–o** covara.eps **–do** covara.m2p

Use ghostview (or Photoshop) to view the plot (**gv** covara.eps). Use the **xpm2ps –rainbow** flag to look at other color schemes.

Only a very small number of eigenvectors (modes of fluctuation) contribute significantly to the overall motion of the protein.  This is typical.  To view the most dominant mode (1), use the following command …
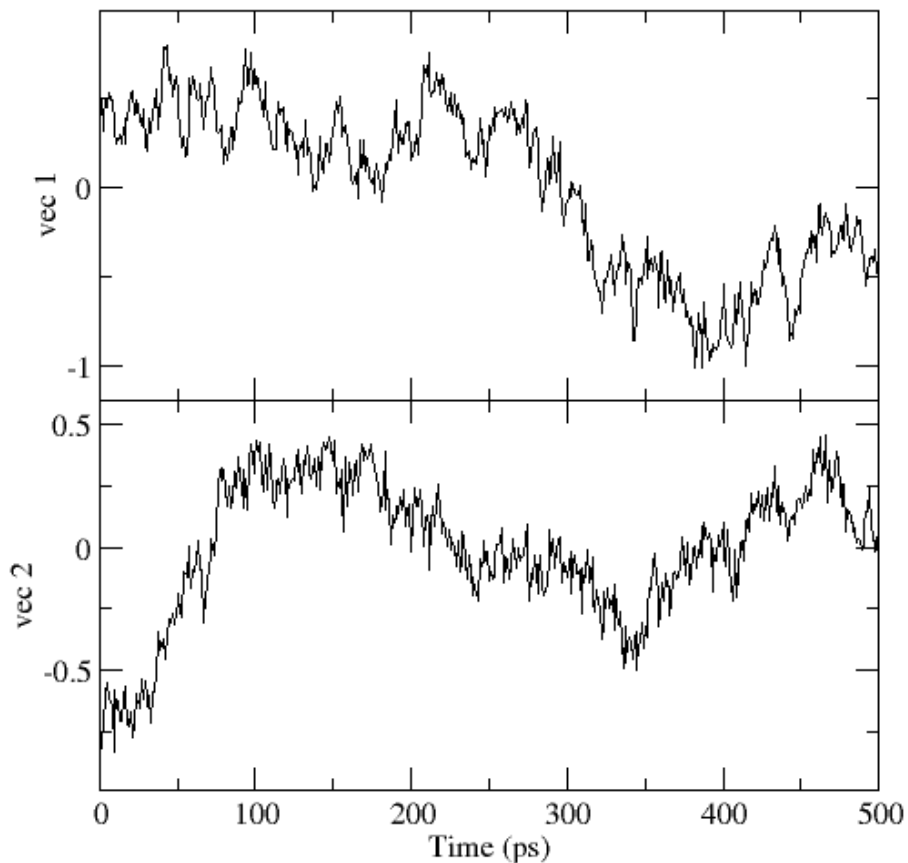
**g_anaeig** **–v** eigenvec.trr **–s** md.tpr **–f** md.xtc **–first** 1 **–last** 1 **–nframes** 100 **–extr** fws-ev1.pdb

For a 2D eigenvector projection use

**g_anaeig**  **–s** md.tpr **–f** md.xtc **–first** 1 **–last** 2 **-2d** proj-1-2.xvg

Or, for following the projection with time (as shown in the plot that follows) use **–proj** instead of **-2d** flag.
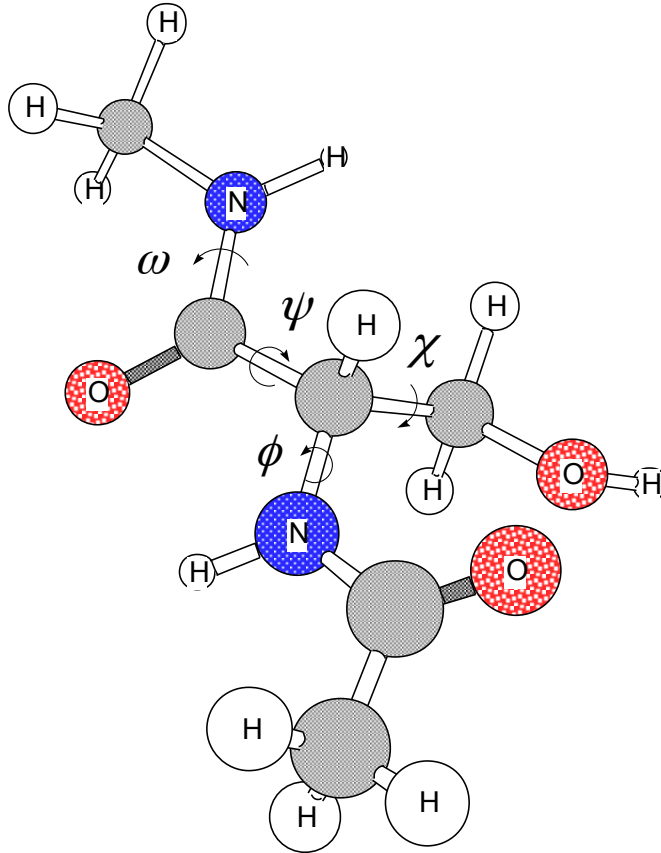
projection on eigenvectors (nm)



Projection for first 2 eigenvectors for FWS protein backbone atoms.

**Dihedral angle analysis**

Use (**mk_angndx –f** md.tpr **–n** mydih.ndx **–nohyd –type** dihedral) to make an index file containing groups of all dihedral angles in your system excluding hydrogen atoms.  Use **g_angle** to perform analysis.

*Dihedral angles in a polypeptide chain.* Adapted from Schulz and Schirmer. Schulz, G.E. and R.H. Schirmer, *Principles of Protein Structure*. Springer Advanced Texts in Chemistry, ed. C.R. Cantor. 1979, New York: Springer-Verlag. 314.

## Dihedral PCA

We will make an index file to study 4 dihedral angles (2 phi and 2 psi) spanning residues MET29, ILE30, and GLY31.  Use VMD (open pr.gro or md.gro) and pick the index numbers of the 4 atoms, which make up each dihedral angle.  Name the file "dangle.ndx".

**g_angle –f** ../md.trr  **–n** dangle.ndx **–or** dangle.trr **–type** dihedral

Write covar.ndx  by hand (2*N/3) atoms where N is the # of dihedral angles.  You will notice that **g_angle** will help you here in its output.  You will see a line in the **g_angle** output requesting 3 atom positions.

*There are 4 dihedrals. Will fill 3 atom positions with cos/sin*

[ test ]
1 2 3

Make the dummy **gro** file for the **g_covar** analysis.

**trjconv –s** ../md.tpr **–f** dangle.trr **–o** resiz.gro **–n** covar.ndx **–e** 0

Perform the **g_covar** analysis using

**g_covar –f** dangle.trr **–n** covar.ndx **–ascii –xpm –nofit –nomwa –noref –s** resiz.gro

To examine the projection of eigenvector 1 use …

**g_anaeig  -v** eigenvec.trr **-f** dangle.trr **-s** resiz.gro **-first** 1 **-last** 1 **-proj** proj-1
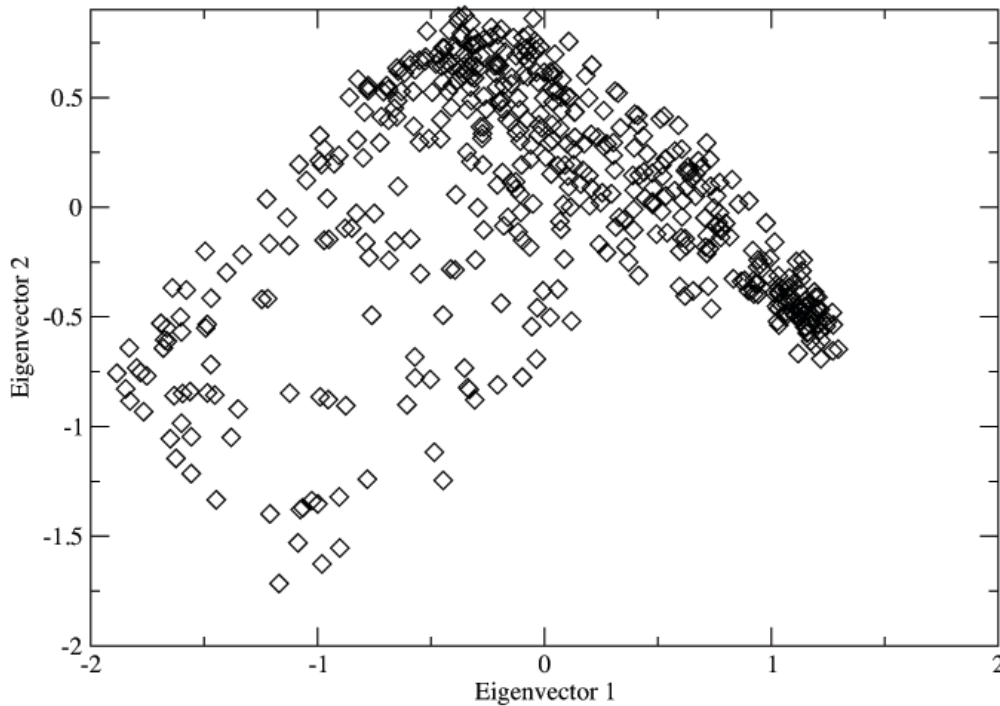
Select Group 0 (System) when prompted.

**xmgrace** proj-1.xvg

For a 2D projection of eigenvector 1 with eigenvector 2 use

**g_anaeig -v** eigenvec.trr **-f** dangle.trr  **-s** resiz.gro **-first** 1 **-last** 2 **-2d** 2dproj_1_2.xvg



2D projection of trajectory

The plot above shows we have roughly 2 clusters of conformations during the course of our 500 ps simulation.  Use **g_analyze** to check the *cosine content* of the projection of ev1 with ev1.  For example …

**g_analyze –f** proj1.xvg **–cc** proj1-cc.xvg

Large-scale motions in the system cannot be differentiated from random diffusion when cosine content is close to 1.[13]  Our cosine content for the dihedral angles in our study is ~ 0.11 for ev1, which is acceptable.

Use the **g_sham** program to view the free energy landscape.

**g_sham –f** 2dproj_1_2.xvg **–ls** gibbs.xpm **-notime**

**xpm2ps –f** gibbs.xpm **–o** gibbs.eps **–rainbow** red
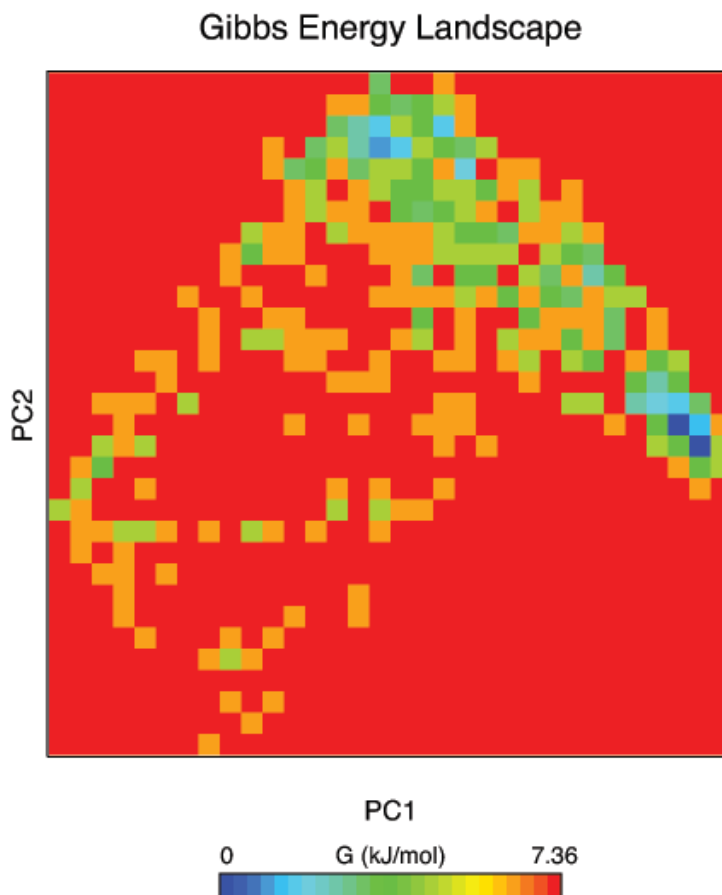


**Appendix.**

How to restart a crashed run. The mdrun program now uses a very handy checkpointing feature. Restarting crashed runs is easy with mdrun.

**mdrun -s** prev.tpr **-f** prev.trr **-e** prev.edr **-o** prev.trr **–g** prev.log **–cpi -append**

The **–cpi** flag tells mdrun to read the checkpoint file (state.cpt) as input.  The **–append** flag tells mdrun to append the data to the existing trajectory and log files.  See the mdrun manpage for more information (or use **mdrun –h** command for more info/help).

How to continue your runs. Use grompp or use tpbconv. For the grompp case, you will need to change the *gen_vel* value in the mdp file to "no"; set *unconstrained_start* to "yes" and read in the previous trajectory from previous trr or xtc file and the previous velocities from the edr file. For a more exact continuation using grompp, you will also need to set mdp variables like *tinit* and *init_step* to an appropriate value for continuation. Use **grompp –h** command for more information/help. This method is binary non-identical (introduces small negligible errors).

Method 1
**grompp -f** md_restart.mdp **-c** md_prev.gro **–t** md_prev.trr **-e** md_prev.edr **–p** topol.top **-o** md_restart.tpr **–maxwarn** 10

Method 2
**tpbconv -s** previous.tpr **-f** previous.trr **-e** previous.edr **-extend** timetoextendby **-o** next.tpr
**mdrun -s** next.tpr

Extending runs using tpbconv and mdrun checkpoint files in version 4.0 – Extending runs is even easier in version 4.0 and is binary identical because you are using the checkpoint file.

**tpbconv -s** previous.tpr **-extend** timetoextendby **-o** next.tpr
**mdrun -s** next.tpr **-cpi** previous.cpt

*Note*: you can use the **–append** flag for mdrun to add the new output to your existing files. Use the final *filename.cpt* file and NOT the *filename_prev.cpt* file!!!

How to concatenate trajectories from continued runs. Use **trjcat**.

**trjcat –f** md1.xtc md2.xtc md3.xtc … (etc) **–o** mdall.xtc **-settime**

Trajectories are read and concatenated in the order you provide. Use the **–settime** flag to interactively set the starting time for each trajectory (Note: the default time unit is ps).

How to concatenate energy files (edr files) from continued runs. Use **eneconv**.

**eneconv –f** md1.edr md2.edr md3.edr … (etc) **–o** mdall.edr **-settime**

Similar to trjcat in operation.

How to setup a parallel run  Simpler in version 4.0 now. No need to use the –np flag as mdrun will take the argument from the mpirun command.

**grompp  –f** md.mdp **–c** pr.gro **–p** fws.top **–o** md.tpr **–maxwarn** 10

Then use mdrun_mpi to run the parallel job. For example on the UMDNJ SunFire use …

**mpirun –np** # /products/gromacs/bin/**mdrun_mpi –deffnm** md

**Bibliography**

1.      Hess, B., et al., *GROMACS 4: Algorithms for highly efficient, load-balanced, and scalable molecular simulation.* J. Chem. Theor. Comp., 2008. **4**(3): p. 435-447.
2.      Lindahl, E., B. Hess, and D. van der Spoel, *GROMACS 3.0: a package for molecular simulation and trajectory analysis.* J. Mol. Model, 2001. **7**: p. 306-317.
3.      Berendsen, H.J., J. Grigera, and T. Straatsma, *The missing term in effective pair potentials.* J. Phys. Chem., 1987. **91**: p. 6269-6271.
4.      Hess, B. and N. van der Vegt, *Hydration thermodynamic properties of amino acid analogues: a systematic comparison of biomolecular force fields and water models.* J. Phys. Chem. B, 2006. **110**(35): p. 17616-26.
5.      Weber, W., P.H. Hünenberger, and J.A. McCammon, *Molecular Dynamics Simulations of a Polyalanine Octapeptide under Ewald Boundary Conditions: Influence of Artificial Periodicity on Peptide Conformation.* J. Phys. Chem. B, 2000. **104**(15): p. 3668-3575.
6.      Darden, T., D. York, and L. Pedersen, *Particle Mesh Ewald: An N-log(N) method for Ewald sums in large systems.* J. Chem. Phys., 1993. **98**: p. 10089-10092.
7.      Essmann, U., et al., *A smooth particle mesh ewald potential.* J. Chem. Phys., 1995. **103**: p. 8577-8592.
8.      Hess, B., et al., *LINCS: A Linear Constraint Solver for molecular simulations.* J. Comp. Chem., 1997. **18**: p. 1463-1472.
9.      Berendsen, H.J.C., et al., *Molecular dynamics with coupling to an external bath.* J. Chem. Phys., 1984. **81**(8): p. 3584-3590.
10.     Bussi, G., D. Donadio, and M. Parrinello, *Canonical sampling through velocity rescaling.* J. Chem. Phys., 2007. **126**(1): p. 14101.
11.     Cerutti, D.S., et al., *Vulnerability in Popular Molecular Dynamics Packages Concerning Langevin and Andersen Dynamics.* J Chem Theory Comput, 2008. **4**(10): p. 1669-1680.
12.     Kabsch, W. and C. Sander, *A dictionary of protein secondary structure.* Biopolymers, 1983. **22**: p. 2577-2637.
13.     Maisuradze, G.G. and D.M. Leitner, *Free energy landscape of a biomolecule in dihedral principal component space: sampling convergence and correspondence between structures and minima.* Proteins, 2007. **67**(3): p. 569-78.