GROMACS Introductory Tutorial

Gromacs ver 3.3.1

Author: John E. Kerrigan, Ph.D.
AST/IST, University of Medicine and Dentistry of NJ
675 Hoes Lane
Piscataway, NJ 08554

Phone: (732) 235-4473
Fax: (732) 235-5252
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] 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 gromacs (*.gro) 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/).

We will use MOE to preview the PDB file. It is advisable to use DeepView (Download DeepView from http://www.expasy.ch/spdbv/) 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.

Open 1OMB.PDB in Deep View (type **spdbv** 1OMB.pdb in your unix shell). Deep View fixes the structure and replaces a missing oxygen atom in a proline residue. Use File > Save > Layer and save the new file as fws.pdb.

Edit the file in any text editor and make sure that the HEADER and COMPND line have a name (actually, any name will do!). In addition, delete the SPDBV lines added to the end of the file by DeepView.

HEADER FWS TOXIN
COMPND FWS TOXIN

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 your fws.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.

pdb2gmx –ignh –ff G43a1 **–f** 1OMB.pdb **–o** fws.pdb **–p** fws.top

Setup the Box for your simulations

editconf -bt cubic **–f** fws.pdb **–o** fws.pdb **-c –d** 0.9

What you have done in this command is specify that you want to use a simple cubic periodic box. The **–d** 0.9 flag sets the dimensions of the box based upon setting the box edge approx 0.9 nm (i.e. 9.0 angstroms) from the molecule(s) periphery. We are using a small dimension for convenience and speed. The molecule is centered in the box by the **–c** flag. Ideally you should set **–d** at no less than 0.85 nm for most systems. [2]

[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.pdb **–cs** spc216.gro **–o** fws_b4em.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, you are using the "SPC" water model. [3] The **genbox** program will add the correct number of water molecules needed to solvate your box of given dimensions.

Setup the energy minimization.

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. (Note: the path to the c pre-processor may be different on your machine. Use the **which** command to locate [i.e. which cpp]!)

Content of em.mdp:

```
title               =  FWS
cpp                 =  /usr/bin/cpp ; the c pre-processor
define              =  -DFLEXIBLE
constraints         =  none
integrator          =  steep
dt                  =  0.002    ; ps !
nsteps              =  400
nstlist             =  5
ns_type             =  grid
rlist               =  0.9
coulombtype         =  PME
rcoulomb            =  0.9
rvdw                =  1.4
fourierspacing          =  0.12
fourier_nx        =  0
fourier_ny        =  0
fourier_nz        =  0
pme_order         =  4
ewald_rtol        =  1e-5
optimize_fft            =  yes
;
;       Energy minimizing stuff
```

```
;
emtol               =  1000.0
emstep              =  0.01
```

Important aspects of the em.mdp file:

title – The title can be any given text description (limit 64 characters; keep it short and simple!)
cpp – location of the pre-processor
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 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
rvdw – distance for the LJ or Buckingham potential cut-off

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_b4em.pdb **–p** fws.top **–o** fws_em.tpr

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. 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 fws_em.tpr **–o** fws_ion.pdb **–nname** CL- **–nn** 2 **–g** fws_ion.log

Where **–nnname** 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. The **–g** flag gives a name to the output log for genion.

When you process this command, you will be prompted to provide a continuous group of solvent molecules, which should be Group 12 (SOL). Type 12 then <enter>. You will notice that two solvent molecules have been replaced by Cl ions. Now you must make the following additions/corrections to your fws.top file:

Add

#include "ions.itp" (Note: This should be present by default already in version 3.2 & greater)

after the include statement for the force field. Subtract 2 from the total number of SOL molecules and add a line for 2 Cl molecules in the [molecules] section at the end of the file. You will also need to modify the temperature coupling scheme for your pr_md.mdp and md.mdp files.

Now, you will use fws_ion.pdb in place of fws_b4em.pdb for generating the actual input tpr using grompp for energy minimization.

Temperature coupling scheme for the pr_md.mdp and md.mdp files after adding chloride ions.

```
; Berendsen temperature coupling is on in three groups
Tcoupl              =  berendsen
tau_t               =  0.1          0.1   0.1
tc_grps             =  protein      sol   CL-
ref_t               =  300          300   300
```

Remember: 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 neutralized the overall net charge of the model.

grompp –f em.mdp **–c** fws_ion.pdb **–p** fws.top **–o** fws_em.tpr

Run the energy minimization in background (Use the ampersand & at the end of the command).

mdrun –s fws_em.tpr **–o** fws_em.trr **–c** fws_b4pr.pdb **–g** em.log **–e** em.edr **&**

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 -50** em.log

```
Steepest Descents converged to Fmax < 1000 in 207 steps
Potential Energy  = -1.8840495e+05
Maximum force     =  9.8538837e+02 on atom 109
Norm of force     =  5.3166636e+03
```

Setup the position-restrained 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 20.0 ps time scale (at least an order of magnitude greater). The settings below will work well for the Gromacs force field. Consult the Gromacs manual for optimal settings for other force fields (e.g. the GROMOS 96 force field uses nstlist = 5 and rvdw = 1.4 is required). Under **coulombtype**, PME stands for "Particle Mesh Ewald" electrostatics. [4, 5] 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. 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). [6] Study the mdp file below.

Content of pr.mdp.

```
title                =  FWS
warnings             =  10
cpp                  =  /usr/bin/cpp
define               =  -DPOSRES
constraints          =  all-bonds
integrator           =  md
dt                   =  0.002  ; ps !
nsteps               =  10000  ; total 20.0 ps.
nstcomm              =  1
nstxout              =  250    ; collect data every 0.5 ps
nstvout              =  1000
nstfout              =  0
nstlog               =  10
nstenergy            =  10
nstlist              =  5
ns_type              =  grid
rlist                =  0.9
coulombtype          =  PME
rcoulomb             =  0.9
rvdw                 =  1.4
fourierspacing           =  0.12
fourier_nx         =  0
fourier_ny         =  0
fourier_nz         =  0
pme_order          =  4
ewald_rtol         =  1e-5
optimize_fft             =  yes
; Berendsen temperature coupling is on in two groups
Tcoupl               =  berendsen
tau_t                =  0.1    0.1   0.1 ; use items in bold if you have Cl ions
tc-grps              =  protein sol  CL-
ref_t                =  300    300   300
; Pressure coupling is on
```

```
Pcoupl              =  berendsen
tau_p               =  0.5
compressibility     =  4.5e-5
ref_p               =  1.0
; Generate velocites is on at 300 K.
gen_vel             =  yes
gen_temp            =  300.0
gen_seed            =  173529
```

Important things to know about the mdp file.

The –DPOSRE 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. [6]
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; 0.9 nm = 9.0 angstroms) for the electrostatic and van der Waals terms.

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

Tcoupl = berendsen [7] (type of temperature coupling; another option is nose-hoover)

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 berendsen barostat for controlling the simulation pressure. [7]

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.
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).

Pre-process the pr.mdp file.

grompp –f pr.mdp **–c** fws_b4pr.pdb **–r** fws_b4pr.pdb **–p** fws.top **–o** fws_pr.tpr

mdrun –s fws_pr.tpr **–o** fws_pr.trr **–c** fws_b4md.pdb **–g** pr.log **–e** pr.edr **&**

Use the **tail** command to check the pr.log file.

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.

Content of md.mdp for explicit solvation (*Special Note*: For *in vacuo*, remove the entries for "sol" in the temperature coupling section. For the simulation that includes the counterions, add an entry for the ions in the temperature coupling section.)

```
title                =  FWS
cpp                  =  /usr/bin/cpp
constraints          =  all-bonds
integrator           =  md
dt                   =  0.002  ; ps !
nsteps               =  25000  ; total 50 ps.
nstcomm              =  1
nstxout              =  500    ; collect data every 1 ps
nstvout              =  0
nstfout              =  0
nstlist              =  5
ns_type              =  grid
rlist                =  0.9
coulombtype          =  PME
rcoulomb             =  0.9
rvdw                 =  1.4
fourierspacing          =  0.12
fourier_nx        =  0
fourier_ny        =  0
fourier_nz        =  0
pme_order         =  4
ewald_rtol        =  1e-5
optimize_fft            =  yes
; Berendsen temperature coupling is on in two groups
Tcoupl                =  berendsen
tau_t                 =  0.1    0.1       0.1
tc-grps               =  protein    sol   CL-
ref_t                 =  300        300   300
```

```
; Pressure coupling is on
Pcoupl              =  berendsen
tau_p               =  0.5
compressibility     =  4.5e-5
ref_p               =  1.0
; Generate velocites is on at 300 K.
gen_vel             =  yes
gen_temp            =  300.0
gen_seed            =  173529
```

grompp –f md.mdp **–c** fws_b4md.pdb **–r** fws_b4md.pdb **–p** fws.top **–o** fws_md.tpr

mdrun –s fws_md.tpr **–o** fws_md.trr **–c** fws_pmd.pdb **–g** md.log **–e** md.edr **&**

Use the **tail** command to check the md.log file.

You may compress the trajectory using **trjconv** to save on disk space.

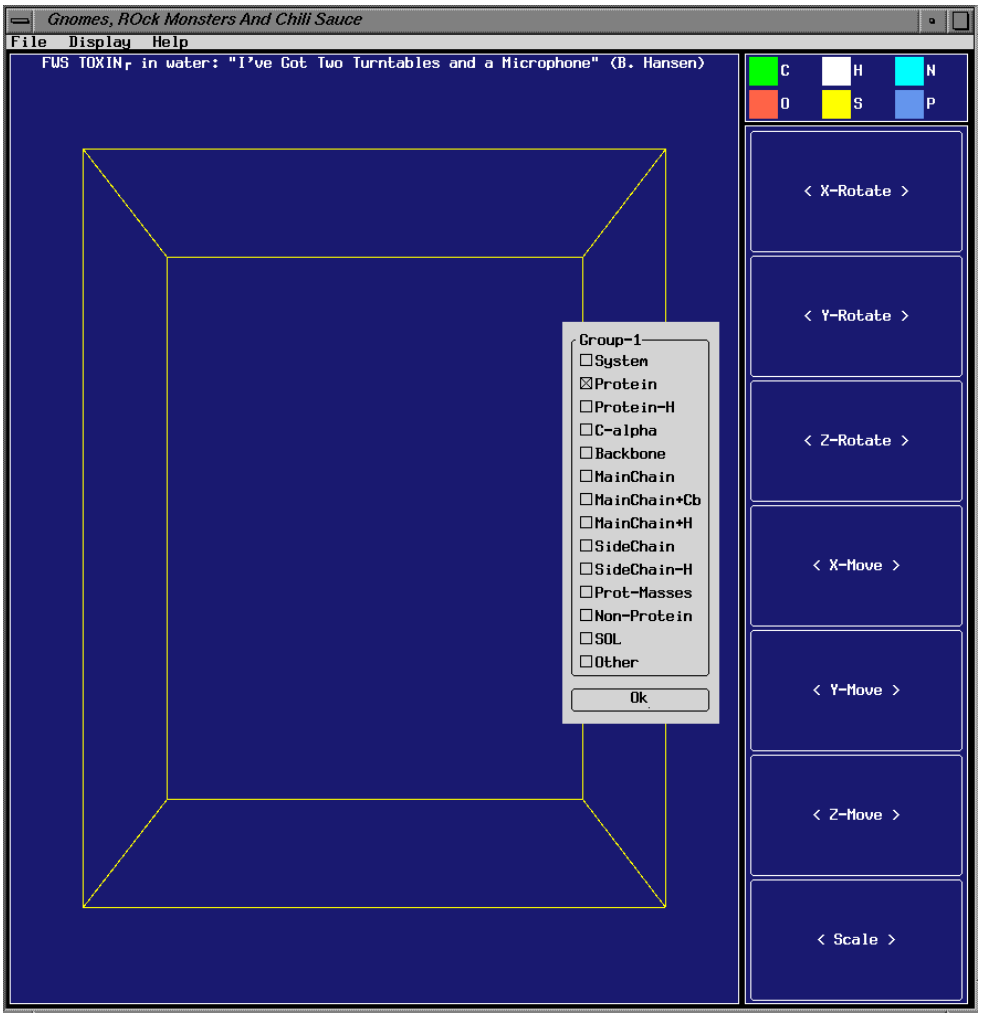
trjconv –f filename.trr **–o** filename.xtc

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

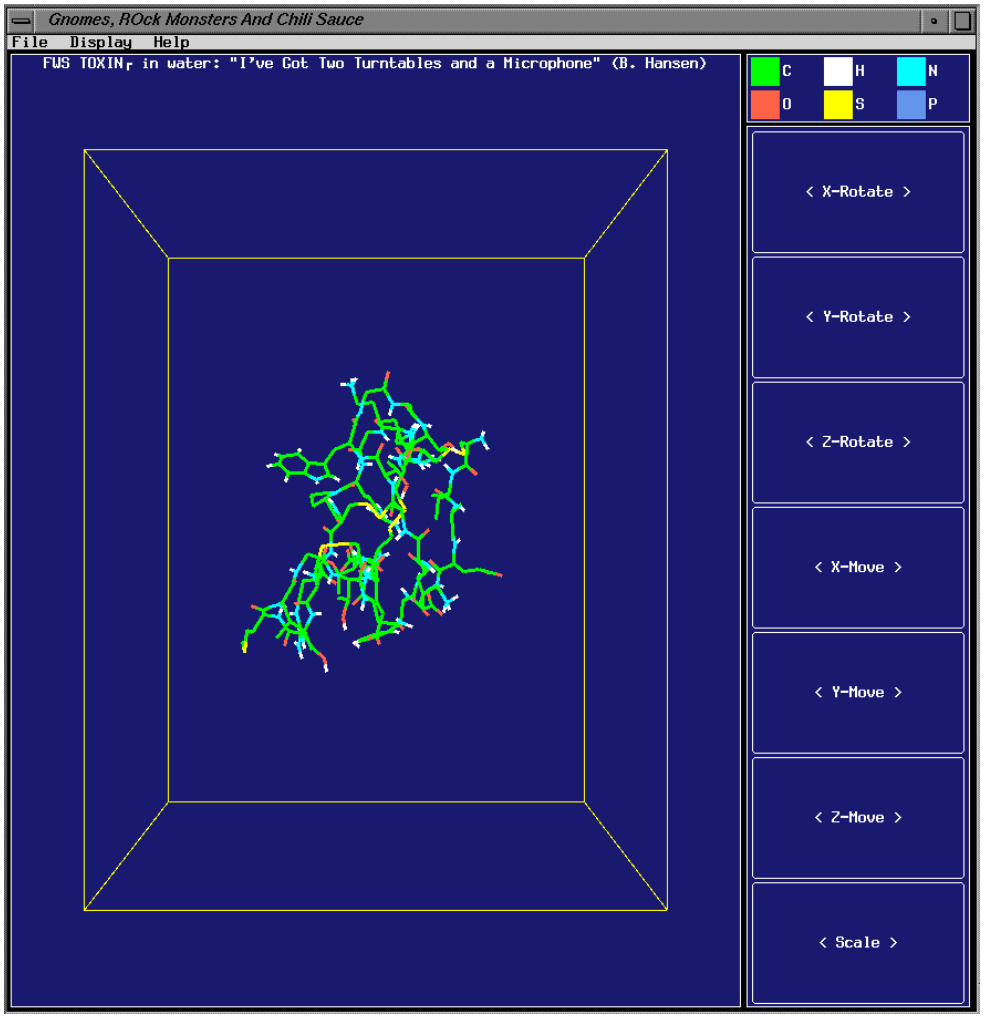
Use **ngmx** to view the trajectory (you may also download to your PC and use VMD to view the trajectory).

ngmx –f fws_md.trr (or fws_md.xtc) **–s** fws_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 Ok.



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 most 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

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
```

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

```
energygrps_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** clg_b4md.pdb **–p** clg.top **–n** clg_ter.ndx **–o** clg_md.tpr

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** fws_pmd.pdb **–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). 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** 801 **–e** 1000 **–av** traj_avg.pdb

Warning – Average structures tend to be crude. Minimization is recommended.

g_energy

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

g_energy –f md.edr **–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, we see the following …

```
  Select the terms you want from the following list
-------------------------------------------------
G96Angle       Proper-Dih.    Improper-Dih.  LJ-14          Coulomb-14
LJ-(SR)        LJ-(LR)        Coulomb-(SR)   Coul.-recip.   Potential
Kinetic-En.    Total-Energy   Temperature    Pressure-(bar) Box-X
Box-Y          Box-Z          Volume         Density-(SI)   pV
Vir-XX         Vir-XY         Vir-XZ         Vir-YX         Vir-YY
Vir-YZ         Vir-ZX         Vir-ZY         Vir-ZZ         Pres-XX-(bar)
Pres-XY-(bar)  Pres-XZ-(bar)  Pres-YX-(bar)  Pres-YY-(bar)  Pres-YZ-(bar)
Pres-ZX-(bar)  Pres-ZY-(bar)  Pres-ZZ-(bar)  #Surf*SurfTen  Pcoupl-Mu-XX
Pcoupl-Mu-YY   Pcoupl-Mu-ZZ   Mu-X           Mu-Y           Mu-Z
T-Protein      T-SOL          T-CL-          Lamb-Protein   Lamb-SOL
Lamb-CL-
```

For potential energy, type “Potential” <enter>
Hit the <enter> key again

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

```
Last frame read 500 time  100.000

Statistics over 50001 steps [ 0.0000 thru 100.0000 ps ], 1 data sets

Energy                      Average       RMSD     Fluct.      Drift  Tot-Drift
-------------------------------------------------------------------------------
Potential                   -172216    287.932    287.925  0.0672381    6.72394
```

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

xmgrace -nxy fws_pe.xvg

Grace is not available on UMDNJ computers at this time. 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.*

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 fws_md.trr **–s** fws_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

g_rms –s fws_md.tpr **–f** fws_md.trr **–o** fws_rmsd.xvg

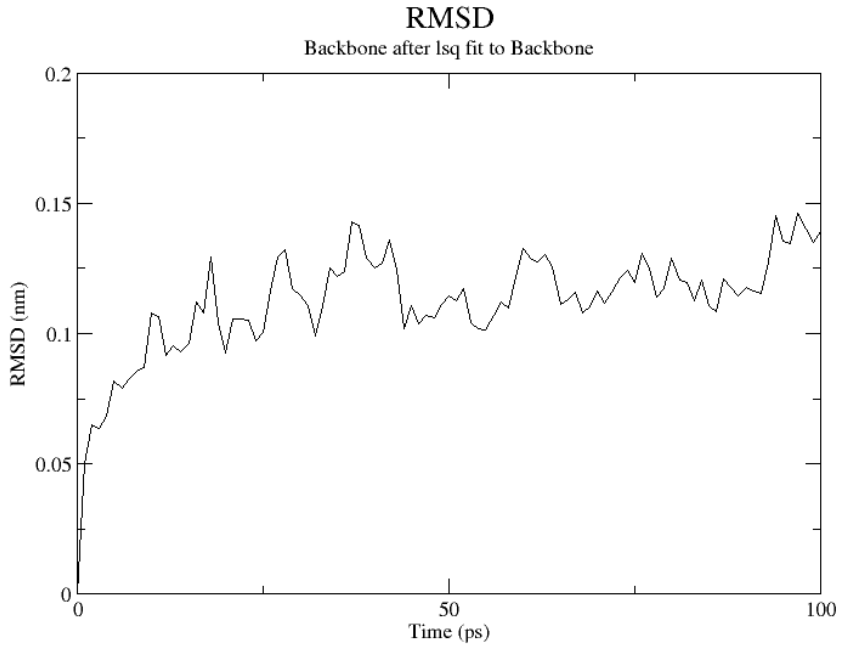
Select group 4 (backbone) for the least squares fit. The program will generate a plot of rmsd over time (rmsd.xvg) which you may import into Excel as a space delimited file.

g_rmsf

Computes root mean square fluctuation of atom positions. Like **g_covar** this command may be used to compute average structures and may be faster than g_covar. 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 **–ox** traj_avg.pdb

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



In the example above, we might use the 30 to 90 ps range for computing an average structure as we observe a leveling off in deviation from the reference (initial structure).

Warning – Average structures tend to be crude. Minimization is recommended.

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
;
cpp                =  /usr/bin/cpp
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
epsilon_r            =  6.0
Tcoupl               =  no
Pcoupl               =  no
gen_vel              =  no
```

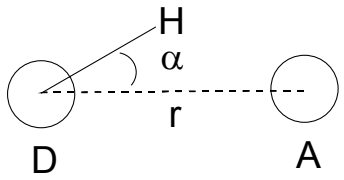
```
;  Conjugate gradient with morse potential
;   in vacuo
;     User kerrigan (236)
;
;     Input file
;
cpp                  =  /usr/bin/cpp
constraints          =  none
integrator           =  cg
nsteps               =  3000
;
;     Energy minimizing stuff
;
emtol                =  100
emstep               =  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
epsilon_r            =  6.0
Tcoupl               =  no
Pcoupl               =  no
gen_vel              =  no
```

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 fws_md.trr **–s** fws_md.tpr **–num** fws_hnum.xvg



Geometrical relations used by g_hbond.

The defaults in Gromacs 3.2 are:

$r \leq 0.35nm$

$\alpha \leq 30°$

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 fws_md.trr **–s** fws_md.tpr

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

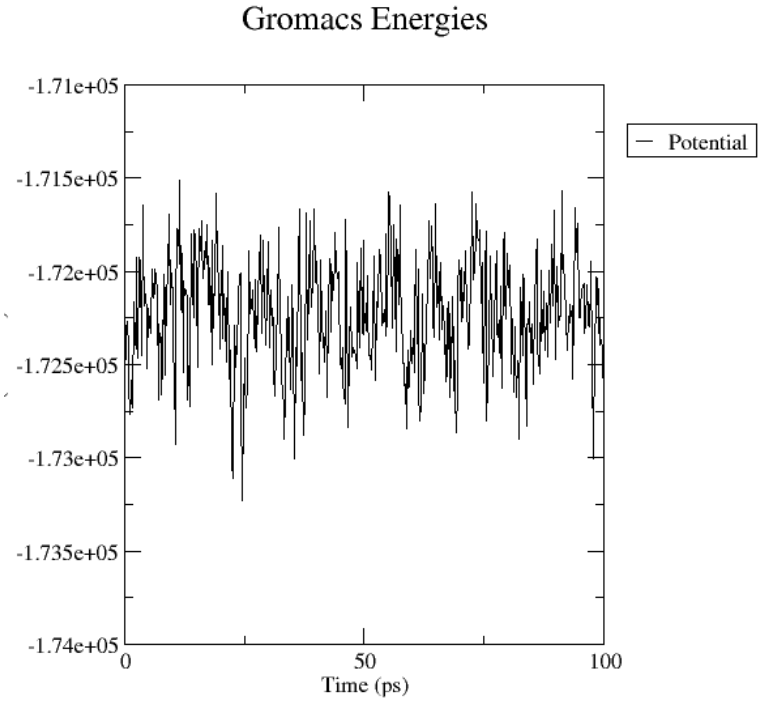
To get a specific frame (3000 ps in this example) instead of the whole trajectory as a pdb file use additionally the **-dump** option, e.g.

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

Results

All sample data was collected using Gomacs version 3.3.1 in single precision.

Potential Energy Plot



Computation of structure average:

First, compute the average structure from each dynamics trajectory using either **g_covar** or **g_rmsf**. In a real research run you would not necessarily compute the average of the entire trajectory. You would just compute the average for the time points where you have reached some equilibrium.

In our run, we computed the average for 60 ps of simulation using g_rmsf (it's faster).

g_rmsf –f fws_md.trr **–s** fws_md.tpr **–b** 30 **–e** 90 **–ox** fws_avg.pdb

At the prompt for group, select Group 1 (Protein). If you just want look at the Backbone atoms, then use Group 4.

Perform other analysis tasks as recommended by your instructor.

Appendix.

How to restart a crashed run.

tpbconv -s prev.tpr **-f** prev.trr **-e** prev.edr **-o** restart.tpr

mdrun -s restart.tpr **-deffnm** myrestart (the **-deffnm** option sets the default filename for all file options to mdrun)

How to extend a run.

tpbconv -f traj.trr **-s** topol.tpr **-e** ener.edr **-o** tpxout.tpr **–until** $VALUE

Where $VALUE = ps (e.g. if you wanted to extend a 2 ns run to 5 ns, then $VALUE = 5000)

How to setup a parallel run

grompp –np # **–f** md.mdp **–c** fws_b4md.pdb **–r** fws_b4md.pdb **–p** fws.top **–o** fws_md.tpr

Where the **-np** flag is used to designate the number of nodes for parallel execution.

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

mprun –np # /products/gromacs/sparc-sun-solaris2.8/bin/**mdrun_mpi -s** fws_md.tpr **-o** fws_md.trr **-c** fws_pmd.pdb **–g** md.log **-e** md.edr

Use the above command in your Sun Grid Engine batch submission script!

Bibliography

1. 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.
2. 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.
3. Berendsen, H.J., J.P. Postma, W.F. van Gunsteren, and J. Hermans, *Interaction models for water in relation to protein hydration.*, in *Intermolecular Forces*, B. Pullman, Editor. 1981, D. Reidel Publishing Co.: Dordrecht. p. 331-342.
4. Essmann, U., L. Perera, M.L. Berkowitz, T. Darden, H. Lee, and L. Pedersen, *A smooth particle mesh ewald potential.* J. Chem. Phys., 1995. **103**: p. 8577-8592.
5. 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.
6. Hess, B., H. Bekker, H. Berendsen, and J. Fraaije, *LINCS: A Linear Constraint Solver for molecular simulations.* J. Comp. Chem., 1997. **18**: p. 1463-1472.
7. Berendsen, H.J.C., J.P.M. Postma, W.F. vanGunsteren, A. DiNola, and J.R. Haak, *Molecular dynamics with coupling to an external bath.* J. Chem. Phys., 1984. **81**(8): p. 3584-3590.