

Gromacs Introductory Tutorial

Gromacs ver 4.6

John E. Kerrigan, Ph.D.
Associate Director Bioinformatics
The Cancer Institute of New Jersey
Rutgers, The State University of NJ
120 Albany Street
New Brunswick, NJ 08903
USA

732 235-4473
732 235-6267

jkerriga@rutgers.edu
kerrigje@umdnj.edu

Gromacs 4.6 Tutorial

Funnel Web Spider toxin using Amber99SB-ILDN force field.

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. For this tutorial, we used Gromacs version 4.6 compiled with FFTW ver 3.3.3 libraries.

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. 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?

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 might 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. However, you will need to add OXT at the C-terminal end. This can be done using the “Build” menu in DeepView. Save the file as fws.pdb.

Create a project directory called “fwspider” and move 1OMB.pdb and fws.pdb to that directory. The **pdb2gmx** command creates input coordinates and topology for gromacs to use. The **-ignh** flag ignores any hydrogen atoms in the input pdb file. We use the **-ff** flag to designate the force field to be used and the **-water** flag to designate the water model. We are using the Amber99SB-ILDN force field[3] and the TIP3P water model [4].

```
pdb2gmx -ignh -ff amber99sb-ildn -f fws.pdb -o fws.gro -p fws.top -water tip3p
```

Next setup the periodic box. We will use a rhombic dodecahedron box as it is the most spherical box with the **-bt** flag. We use the **-d** flag to set the spacing distance in nm. You may also use editconf to convert “gro” files to “pdb” files.

```
editconf -f fws.gro -o fws-PBC.gro -bt dodecahedron -d 1.2
```

Now we begin setup of our system using *in vacuo* minimization. The following is the content of file em-vac-pme.mdp.

em-vac-pme.mdp

```
; Define can be used to control processes
define                = -DFLEXIBLE      ; Use flexible water model

; Parameters describing what to do, when to stop and what to save
integrator            = steep           ; Algorithm (steep = steepest descent
minimization)
emtol                 = 500.0           ; Stop minimization when the maximum force <
1.0 kJ/mol
nsteps               = 1000            ; Maximum number of (minimization) steps to
perform
nstenergy             = 1              ; Write energies to disk every nstenergy
steps
energygrps           = System          ; Which energy group(s) to write to disk

; Parameters describing how to find the neighbors of each atom and how to
calculate the interactions
nstlist              = 1               ; Frequency to update the neighbor list
ns_type              = grid            ; Method to determine neighbor list (simple,
grid)
coulombtype          = PME             ; Treatment of long range electrostatic
interactions
rlist                = 1.0             ; Cut-off for making neighbor list (short
range forces)
rcoulomb             = 1.0             ; long range electrostatic cut-off
rvdw                 = 1.0             ; long range Van der Waals cut-off
constraints          = none            ; Bond types to replace by constraints
pbc                  = xyz             ; Periodic Boundary Conditions (yes/no)
```

Use the gromacs pre-processor (grompp) to generate the run input file (tpr file).

```
grompp -f em-vac-pme.mdp -c fws-PBC.gro -p fws.top -o em-vac.tpr
```

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

Fill the periodic box with water using genbox.

```
genbox -cp em-vac.gro -cs spc216.gro -p fws.top -o fws-b4ion.gro
```

Solvated system energy minimization (em-sol-pme.mdp)

```
; Define can be used to control processes
define                = -DFLEXIBLE

; Parameters describing what to do, when to stop and what to save
integrator            = steep                ; Algorithm (steep = steepest descent
minimization)
emtol                 = 250.0                ; Stop minimization when the maximum force <
1.0 kJ/mol
nsteps                = 5000                ; Maximum number of (minimization) steps to
perform
nstenergy             = 1                   ; Write energies to disk every nstenergy
steps
energygrps            = System              ; Which energy group(s) to write to disk

; Parameters describing how to find the neighbors of each atom and how to
calculate the interactions
nstlist               = 1                   ; Frequency to update the neighbor list
ns_type                = grid               ; Method to determine neighbor list (simple,
grid)
coulombtype           = PME                 ; Treatment of long range electrostatic
interactions
rlist                 = 1.0                 ; short-range neighborlist cutoff (in nm)
rcoulomb               = 1.0                 ; long range electrostatic cut-off
rvdw                   = 1.0                 ; long range Van der Waals cut-off
constraints            = none               ; Bond types to replace by constraints
pbc                    = xyz                ; Periodic Boundary Conditions (yes/no)
```

```
grompp -f em-sol-pme.mdp -c fws-b4ion.gro -p fws.top -o ion.tpr
```

We will use this first tpr file (designated as “ion.tpr”) to add the ions to the system.

```
genion -s ion.tpr -o fws-b4em.gro -neutral -conc 0.15 -p fws.top -g ion.log
```

When prompted select - Solvent. The **genion** command will replace water molecules with ions. The **-neutral** flag insures the overall net charge of the system equals zero. The **-conc** flag sets the ion concentration to that designated (in the case above 0.15 M). The default salt used by genion is always NaCl. If you desire to use different cation and anion, use **-pname** (for positive ion) and **-nname** (for negative ion) flags.

```
grompp -f em-sol-pme.mdp -c fws-b4em.gro -p fws.top -o em-sol.tpr
```

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

Two-step equilibration procedure - Now we need to relax the solvent and ions while keeping the protein atom positions restrained. This will be accomplished in two phases: 100 ps NVT followed by 100 ps NPT ensembles. We will perform our dynamics using a temperature of 300 K which is close to room temperature for most laboratory environments. Some folks like to simulate at 310 K which is closer to body or physiological temperature. We use a dispersion correction for energy and pressure (see section 4.8 of Gromacs manual). For *coulombtype*, PME stands for “Particle Mesh Ewald” electrostatics.[5, 6] PME is the best method for computing long-range electrostatics in systems where we have charges (exposed charged amino acid residues, ions, polar lipids, etc.). The all-bonds option under *constraints* applies the Linear Constraint algorithm (lincs) for fixing all bond lengths in the system (important to use this option when the simulation time step $dt > 0.001$ ps).[7]

```
nvt-pr-md.mdp
```

```
; VARIOUS PREPROCESSING OPTIONS
define                = -DPOSRES

; RUN CONTROL PARAMETERS
integrator            = md
dt                    = 0.002 ; time step (in ps)
nsteps                = 50000 ; number of steps

; OUTPUT CONTROL OPTIONS
nstxout               = 500    ; save coordinates every ps
nstvout               = 500    ; save velocities every ps
nstenergy             = 500    ; save energies every ps
nstlog                = 500    ; update log file every ps
energygrps            = Protein Non-Protein

; NEIGHBORSEARCHING PARAMETERS
nstlist               = 5
ns_type               = grid
pbc                   = xyz
rlist                 = 1.0

; OPTIONS FOR ELECTROSTATICS AND VDW
coulombtype           = PME    ; Particle Mesh Ewald for long-range
electrostatics        =
pme_order              = 4     ; cubic interpolation
fourierspacing         = 0.16  ; grid spacing for FFT
rcoulomb               = 1.0
vdw-type               = Cut-off
rvdw                  = 1.0

; Temperature coupling
```

```

tcoupl          = v-rescale          ; Couple temperature to
external heat bath according to velocity re-scale method
tc-grps        = Protein Non-Protein ; Use separate heat baths for
Protein and Non-Protein groups
tau_t         = 0.1      0.1          ; Coupling time constant,
controlling strength of coupling
ref_t         = 300      300          ; Temperature of heat bath

; Dispersion correction
DispCorr      = EnerPres ; account for vdw cut-off

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

; GENERATE VELOCITIES FOR STARTUP RUN
gen_vel       = yes            ; Assign velocities to particles by taking
them randomly from a Maxwell distribution
gen_temp      = 300           ; Temperature to generate corresponding
Maxwell distribution
gen_seed      = -1            ; Seed for (semi) random number generation.
Different numbers give different sets of velocities

; OPTIONS FOR BONDS
constraints   = all-bonds ; All bonds will be treated as
constraints (fixed length)
continuation  = no              ; first dynamics run
constraint_algorithm = lincs    ; holonomic constraints
lincs_iter   = 1                ; accuracy of LINCS
lincs_order  = 4                ; also related to accuracy

```

grompp -f nvt-pr-md.mdp -c em-sol.gro -p fws.top -o nvt-pr.tpr

nohup mdrun -deffnm nvt-pr &

Use *tail -25 nvt-pr.log* to check run progress.

nvt-pr-md.mdp

```

; VARIOUS PREPROCESSING OPTIONS
define          = -DPOSRES

; RUN CONTROL PARAMETERS
integrator     = md
dt             = 0.002
nsteps        = 50000

; OUTPUT CONTROL OPTIONS
nstxout       = 500      ; save coordinates every ps
nstvout       = 500      ; save velocities every ps
nstfout       = 500      ; save forces every ps
nstenergy     = 500      ; save energies every ps
nstlog        = 500      ; update log file every ps
energygrps    = Protein Non-Protein

; NEIGHBORSEARCHING PARAMETERS

```

```

nstlist           = 5
ns-type          = Grid
pbc              = xyz
rlist            = 1.0

; OPTIONS FOR ELECTROSTATICS AND VDW
coulombtype      = PME
pme_order        = 4      ; cubic interpolation
fourierspacing   = 0.16  ; grid spacing for FFT
rcoulomb         = 1.0
vdw-type         = Cut-off
rvdw             = 1.0

; Temperature coupling
Tcoupl           = v-rescale
tc-grps         = Protein Non-Protein
tau_t            = 0.1    0.1
ref_t            = 300    300

; Dispersion correction
DispCorr         = EnerPres ; account for vdw cut-off

; Pressure coupling
Pcoupl           = Parrinello-Rahman
Pcoupltype       = Isotropic
tau_p            = 2.0
compressibility  = 4.5e-5
ref_p            = 1.0
refcoord_scaling = com

; GENERATE VELOCITIES FOR STARTUP RUN
gen_vel          = no

; OPTIONS FOR BONDS
constraints      = all-bonds
continuation     = yes      ; continuation from NVT
constraint_algorithm = lincs ; holonomic constraints
lincs_iter       = 1        ; accuracy of LINCS
lincs_order      = 4        ; also related to accuracy

```

grompp -f npt-pr-md.mdp -c em-sol.gro -p fws.top -o npt-pr.tpr

nohup mdrun -deffnm npt-pr &

npt-nopr-md.mdp

```

; RUN CONTROL PARAMETERS
integrator       = md
dt               = 0.002
nsteps           = 500000 ; 1 ns

; OUTPUT CONTROL OPTIONS
nstxout          = 500    ; save coordinates every ps
nstvout          = 500    ; save velocities every ps
nstfout          = 500    ; save forces every ps
nstenergy        = 500    ; save energies every ps
nstlog           = 500    ; update log file every ps

```

```

energygrps          = Protein Non-Protein

; NEIGHBORSEARCHING PARAMETERS
nstlist             = 5
ns-type             = Grid
pbc                 = xyz
rlist               = 1.0

; OPTIONS FOR ELECTROSTATICS AND VDW
coulombtype         = PME
pme_order           = 4      ; cubic interpolation
fourierspacing      = 0.16  ; grid spacing for FFT
rcoulomb            = 1.0
vdw-type            = Cut-off
rvdw                = 1.0

; Temperature coupling
Tcoupl              = v-rescale
tc-grps             = Protein Non-Protein
tau_t               = 0.1    0.1
ref_t               = 300    300

; Dispersion correction
DispCorr            = EnerPres ; account for vdw cut-off

; Pressure coupling
Pcoupl              = Parrinello-Rahman
Pcoupltype          = Isotropic
tau_p               = 2.0
compressibility      = 4.5e-5
ref_p               = 1.0

; GENERATE VELOCITIES FOR STARTUP RUN
gen_vel             = no

; OPTIONS FOR BONDS
constraints          = all-bonds
continuation         = yes      ; continuation from NPT PR
constraint_algorithm = lincs     ; holonomic constraints
lincs_iter          = 1         ; accuracy of LINCS
lincs_order         = 4         ; also related to accuracy

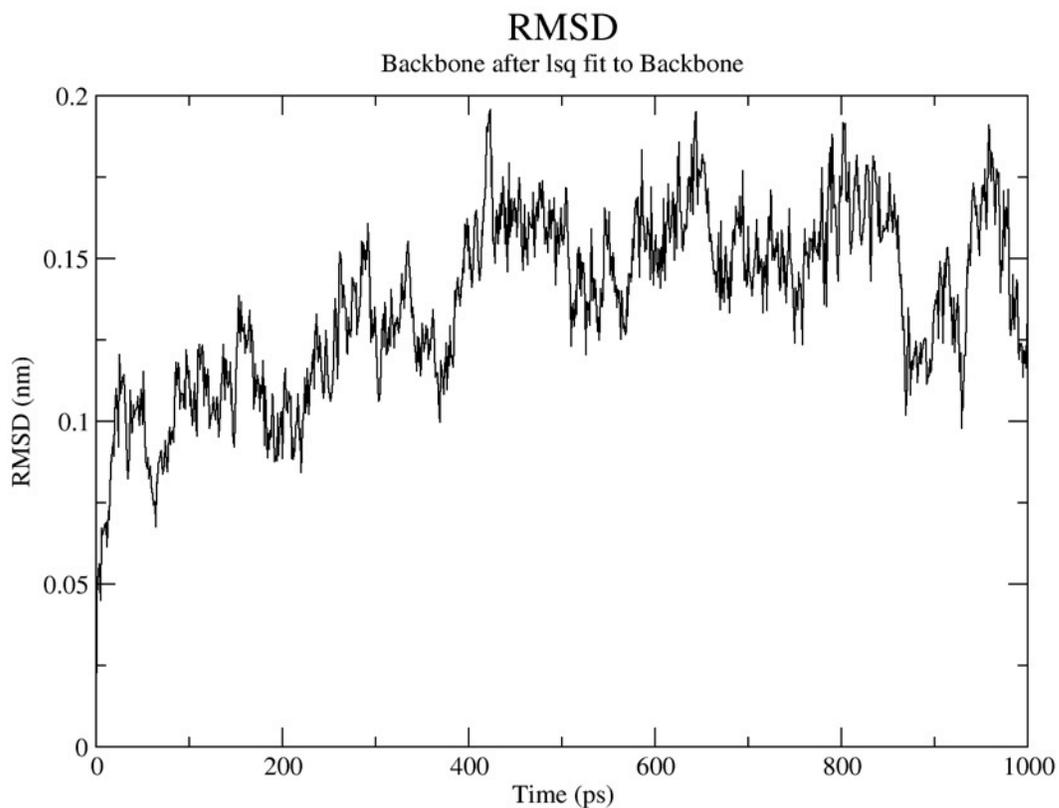
```

grompp -f npt-nopr-md.mdp -c em-sol.gro -p fws.top -o npt-nopr.tpr

nohup mdrun -deffnm npt-nopr &

g_rms -s npt-nopr.tpr -f npt-nopr.trr -o fws-bkbone-rmsd.svg

Select 4 (Backbone)



Average potential/kinetic/total energies and temperature.

g_energy -f npt-nopr.edr -o nrg-npt.xvg

Statistics over 500001 steps [0.0000 through 1000.0000 ps], 4 data sets
All statistics are over 5001 points

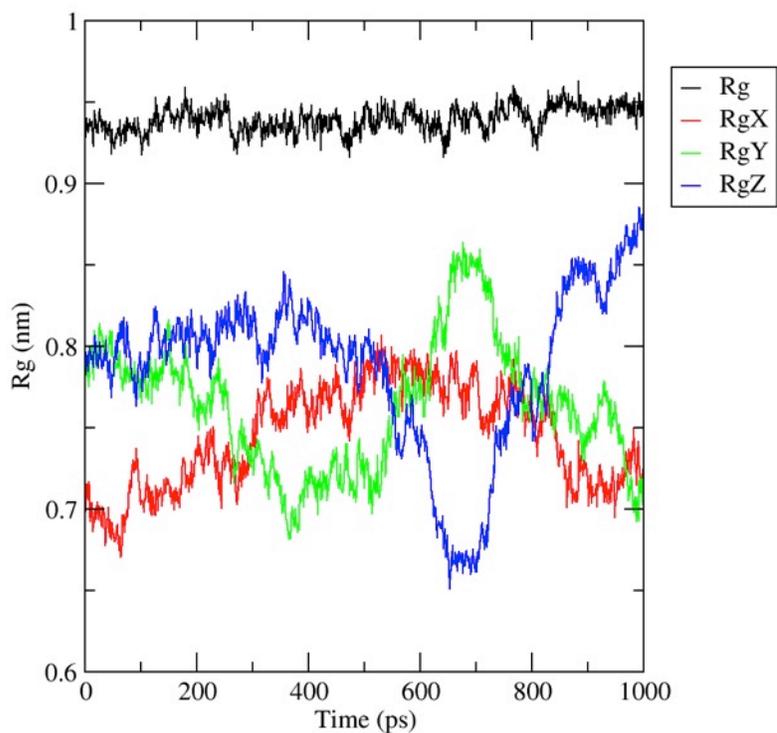
Energy	Average	Err.Est.	RMSD	Tot-Drift	
Potential	-176245	11	416.027	-50.7721	(kJ/mol)
Kinetic En.	31465.7	3	275.05	18.1918	(kJ/mol)
Total Energy	-144779	11	497.067	-32.5799	(kJ/mol)
Temperature	299.911	0.029	2.62161	0.173391	(K)

The radius of gyration provides a measure of the “compactness” of the protein. Use **g_gyrate** to evaluate.

g_gyrate -s npt-nopr.tpr -f npt-nopr.trr -o fws-gyrate.xvg

Select 1 Protein

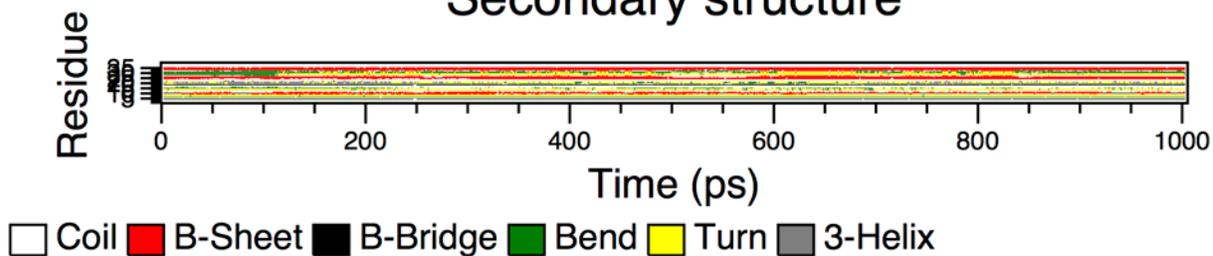
Radius of gyration



```
do_dssp -s npt-nopr.tpr -f npt-nopr.trr -o fws-ss.xpm
```

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

Secondary structure



Use the **g_rmsf** command to compute per residue temperature factors.

```
g_rmsf -s npt-nopr.tpr -f npt-nopr.trr -o fws-rmsf.xvg -ox fws-avg.pdb -res -oq fws-bfac.pdb
```

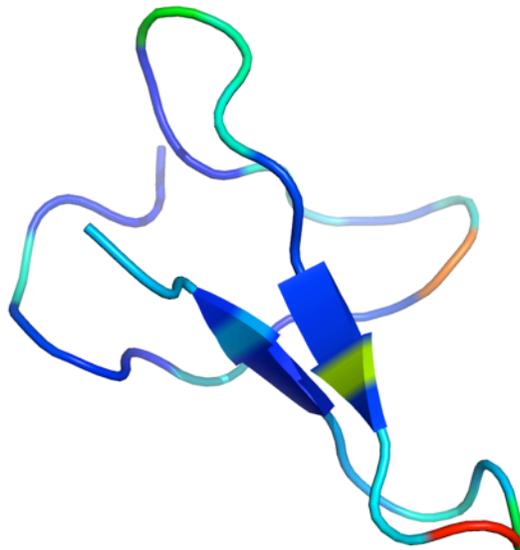
Select 1 Protein

```
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)

To prepare a 300 dpi PNG ray-traced image in pymol use the following commands:

```
ray 1200,1200
png bfac.png, dpi=300
```



Appendix:

npt-longrun.mdp

```
; RUN CONTROL PARAMETERS
integrator          = md
tinit              = 0           ; Starting time
dt                 = 0.002       ; 2 femtosecond time step for integration
nsteps             = YYY        ; Make it 50 ns
```

```

; OUTPUT CONTROL OPTIONS
nstxout          = 250000 ; Writing full precision coordinates every
0.5 ns
nstvout          = 250000 ; Writing velocities every 0.5 ns
nstlog           = 5000   ; Writing to the log file every 10ps
nstenergy       = 5000   ; Writing out energy information every 10ps
nstxtcout       = 5000   ; Writing coordinates every 10ps
energygrps      = Protein Non-Protein

; NEIGHBORSEARCHING PARAMETERS
nstlist          = 5
ns-type         = Grid
pbc             = xyz
rlist           = 1.0

; OPTIONS FOR ELECTROSTATICS AND VDW
coulombtype     = PME
pme_order       = 4           ; cubic interpolation
fourierspacing = 0.16       ; grid spacing for FFT
rcoulomb        = 1.0
vdw-type        = Cut-off
rvdw            = 1.0

; Dispersion correction
DispCorr        = EnerPres ; account for vdw cut-off

; Temperature coupling
Tcoupl         = v-rescale
tc-grps        = Protein Non-Protein
tau_t          = 0.1      0.1
ref_t          = 300      300

; Pressure coupling
Pcoupl         = Parrinello-Rahman
Pcoupltype     = Isotropic
tau_p          = 2.0
compressibility = 4.5e-5
ref_p          = 1.0

; GENERATE VELOCITIES FOR STARTUP RUN
gen_vel        = no

; OPTIONS FOR BONDS
constraints     = all-bonds
constraint-algorithm = lincs
continuation    = yes           ; Restarting after NPT without
position restraints
lincs-order     = 4
lincs-iter      = 1
lincs-warnangle = 30

```

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
```

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-overtime.xvg -cl clusters.pdb -cutoff 0.25 -method  
gromos
```

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

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
```

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 eigenvect.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 eigenvect.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
```

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 ($2 \cdot N/3$) 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
```

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.[8] 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
```

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
```

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
```

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 -deffnm next
```

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.

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. Lindorff-Larsen, K., et al., *Improved side-chain torsion potentials for the Amber ff99SB protein force field*. Proteins, 2010. **78**: p. 1950-1958.
4. Jorgensen, W., et al., *Comparison of Simple Potential Functions for Simulating Liquid Water*. J. Chem. Phys., 1983. **79**: p. 926-935.
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. Essmann, U., et al., *A smooth particle mesh ewald potential*. J. Chem. Phys., 1995. **103**: p. 8577-8592.
7. Hess, B., et al., *LINCS: A Linear Constraint Solver for molecular simulations*. J. Comp. Chem., 1997. **18**: p. 1463-1472.
8. 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.