LA-UR-14-26188

Title:	GROMACS Tutorial
Author(s):	Junghans, Christoph
Intended for:	Q-bio Summer School, 2014-07-27 (Albuquerque, New Mexico, United States)
Issued:	2014-08-05

Disclaimer:
Los Alamos National Laboratory, an affirmative action/equal opportunity employer,is operated by the Los Alamos National Security, LLC for the National NuclearSecurity Administration of the U.S. Department of Energy under contract DE-AC52-06NA25396. By approving this article, the publisher recognizes that the U.S. Government retains nonexclusive, royalty-free license to publish or reproduce the published form of this contribution, or to allow others to do so, for U.S. Government purposes. Los Alamos National Laboratory requests that the publisher identify this article as work performed under the auspices of the U.S. Departmentof Energy. Los Alamos National Laboratory strongly supports academic freedom and a researcher's right to publish; as an institution, however, the Laboratory does not endorse the viewpoint of a publication or guarantee its technical correctness.

GROMACS Tutorial

Christoph Junghans

Los Alamos National Laboratory
NM, USA

July 28, 2014

Plan for the Course

- Installing GROMACS
- From PDB to MD
- HPC Tips & Tricks
- Question & Answer Session
- Advanced Simulations
  - Spectral clustering (MDSCTK)
  - Coarse-graining (www.votca.org)
  - Martini
- What is next ?

Installing GROMACS

Dependencies

GROMACS is a GPL code and free of charge.

Hard Dependencies

- C++ compiler
- cmake
- GNU make (or MSVS or Google Ninja)

Soft Dependencies

- fftw (can be built-in, legal implications)
- Intel MKL
- Nvidia Cuda

Installing GROMACS

Considerations

Single vs. Double precision

- Why do you want double?
- Can you afford a factor two in speed?
- Do you do NVE or normal mode analysis?

MPI vs. Threads

- Are you going to use more than one node (shared memory)?
- Communication overhead vs. performance gain

Intel vs. GNU

- Are you an Intel fan boy?
- Are you using the latest gcc/icc/MKL/fftw ?

Installing GROMACS

Preparations

- Download GROMACS: wget↩

  ```
  ftp://ftp.gromacs.org/pub/gromacs/gromacs-5.0.tar.gz
  ```

- Unpack tarball:

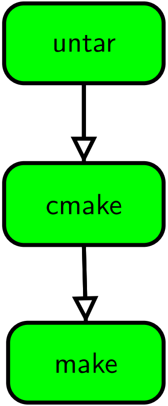
  ```
  tar -xzvf gromacs-5.0.tar.gz
  ```

- Make a build directory:

  ```
  cd gromacs-5.0
  mkdir build
  cd build
  ```

- Run cmake:

  `cmake ..` (or `ccmake` or `cmake-gui`)



Installing GROMACS

Cmake Options

Add options to `cmake -DSOME=THING -DSOMETHING=ELSE ..`

Useful options

- Enable/Disable MPI: `-DGMX_MPI=<ON|OFF>`
  if disabled make sure to use: `-DGMX_THREAD_MPI=ON`
- Enable/Disable GPU: `-DGMX_GPU=<ON|OFF>`
- Install prefix: `-DCMAKE_INSTALL_PREFIX=<PATH>`
- Single or Double: `-DGMX_DOUBLE=<ON|OFF>`
- Enable minimal viewer: `-DGMX_X11=<ON|OFF>`
- Enable built-in fftw: `-DGMX_BUILD_OWN_FFTW=ON`

In short: `cmake ..  -DGMX_X11=ON`↩
`-DCMAKE_INSTALL_PREFIX=$HOME/gromacs`

Installing GROMACS

Build & Install

- Get number of cores:
  - Linux: `grep -c processor /proc/cpuinfo`
  - MacOs: `sysctl -n hw.ncpu`
- Build: `make -j<number of cores>` (add VERBOSE=1 for verbose output)
- Install: `make -j<number of cores> install`
- Setup paths: `source↩ $HOME/gromacs/bin/GMXRC.bash`
- Run: `gmx`

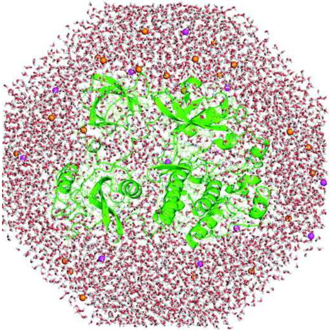
"Move about like a Scientist, lay down, get kissed"
(Red Hot Chili Peppers)

From PDB to MD

Introduction

PDB is a standard format for protein structures, which contains:

- Atom positions
- Atom types
- Residue name & number
- *No* topological information (bonds, angle)
- *No* force field information
- *No* solvent



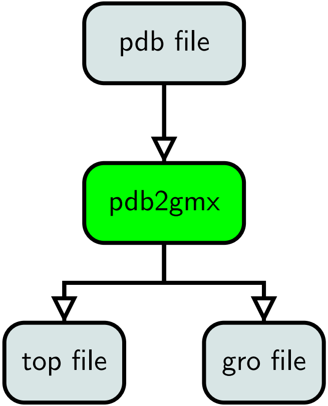
Src kinase protein, from: M. Karplus and J. Kuriyan, PNAS **102**),6679-6685 (2005).

From PDB to MD

Considerations

pdb2gmx Inputs

- Force field: interactive or `-ff` option
- GROMACS supports multiple force fields!
- Water model: interactive or `-water` option
- Interactive termini selection or `-ter`
- Hydrogen removal: `-ignh` option
- Hydrogen to virtual site: `-vsite hydrogens` option



From PDB to MD

Force Fields in a Nutshell

- Force fields describe the interactions between the atoms
- Incl. their functional form
- Bonded and non-bonded interactions

Non-bonded interactions

$$U(r) = 4\epsilon\left[(\sigma/r)^{12} - (\sigma/r)^{6}\right]$$

or

$$U(r) = C_{12}/r^{12} - C_6/r^6$$

or Buckingham or tabulated

Bonded interactions

- Bonds
- Angle
- Dihedrals

Supported Force Fields

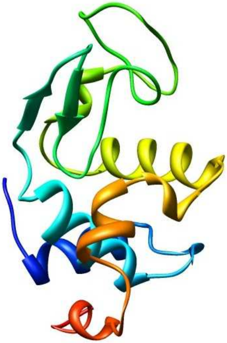
- CHARMM
- Amber
- GROMOS
- OPLS
- Martini

From PDB to MD

1AKI Conversion

This tutorial is modeled after Justin Lemkul's 1AKI tutorial[1]

- Goto `http://www.rcsb.org/`
- Get 1AKI as PDB (text)
- `gmx pdb2gmx -f 1AKI.pdb`
- Select OPLS-AA force field and SPC/E water
- Look at topol.top

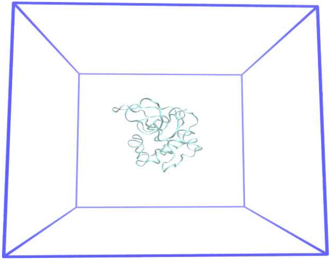


[1] Justin's tutorial(link)

From PDB to MD

1AKI Solvating

- Put 1AKI into a cubic box with distance 1 nm from the boundary
  ```
  gmx editconf -f conf.gro↩
  -o conf2.gro -c -d 1.0↩
  -bt cubic
  ```
- Solvate in spce water
  ```
  gmx solvate -cp conf2.gro↩
  -cs spc216.gro -o↩
  conf3.gro -p topol.top
  ```
- Compare `topol.top` with `#topol.top.1#` (backup file)
- Create runtime setting file



From PDB to MD

Runtime Settings

Create grompp.mdp:

```
integrator      = steep
emtol           = 1000.0
emstep          = 0.01
nsteps          = 50000
nstlist         = 1
rlist           = 1.0
coulombtype     = PME
rcoulomb        = 1.0
rvdw            = 1.0
pbc             = xyz
```

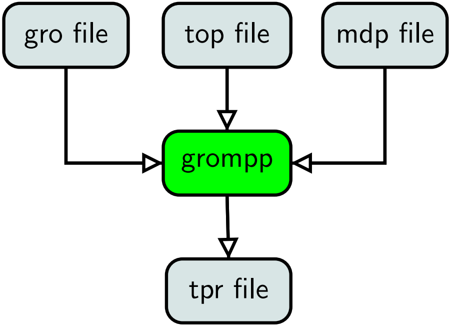
- Use steepest descent (not really an integrator)
- Initial step size 0.01nm
- Minimize until max force is 1000.0 kJ/mol/nm
- All cutoffs = 1nm
- Use smooth PME for electrostatic
- Use periodic boundary conditions in xyz
- Need more details? See the gromacs manual!

From PDB to MD

GROMACS Preprocessor

- Compile a tpr file: `gmx↩ grompp -c conf3.gro`
- Uh, a charged system
- To check defaults look at mdout.mdp
- Add counter ions: `gmx↩ genion -o conf4.gro↩ -p topol.top -pname↩ NA -nname CL -nn 8`
- Select SOL to replace Water molecules
- Compare `topol.top` with `#topol.top.2#` (backup file)

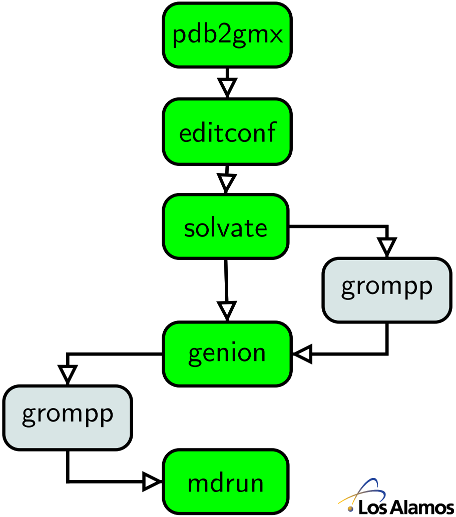
Note: tpr is actually the only file needed for the simulation (except tables)



From PDB to MD

1AKI Minimization

- Recompile tpr: `gmx↩ grompp -c conf4.gro`
- Minimize `gmx mdrun -v`
- Look at `md.log`
- Calculate energy (not actually calculated, but just converted) `gmx energy`
- Select potential energy
- Look at `energy.xvg` using `xmgrace`



MD in a Nutshell

GROMACS stands for "GROningen Mixture of Alchemy and Childrens' Stories"[2] and is a molecular dynamics code optimized for bio-simulations.

$$m_i \ddot{\vec{r}}_i = \sum_j F_{ij}(r_{ij}) = \dot{\vec{p}}_i$$

Molecular dynamics is the “art” of integrating Newton's equation of motion.

[2] Or: "GROningen MAchine for Chemical Simulation"

MD in a Nutshell

Integrators

Simple idea: Taylor expansion

Velocity Verlet Algorithm

$$\vec{r}_i(t+\Delta t) = \vec{r}_i(t) + \frac{\Delta t}{m_i}\vec{p}_i(t) + \frac{\Delta t^2}{2m_i}\vec{F}_i(t) + \mathcal{O}(\Delta t^3) \ ,$$

$$\vec{p}_i(t+\Delta t) = \vec{p}_i(t) + \frac{\Delta t}{2}\left(\vec{F}_i(t) + \vec{F}_i(t+\Delta t)\right) + \mathcal{O}(\Delta t^3)$$

- Only correct up to 2nd order
- One force call per time step
- Higher order will need more force calculations per time step

MD in a Nutshell

Integrators

Alternative Formulation:

Leap-Frog Algorithm

$$\vec{p}_i(t+\Delta t/2) = \vec{p}_i(t-\Delta t/2) + \Delta t \vec{F}_i(t) + \mathcal{O}(\Delta t^2)$$

$$\vec{r}_i(t+\Delta t) = \vec{r}_i(t) + \frac{\Delta t}{m_i}\vec{p}_i(t+\Delta t/2) + \mathcal{O}(\Delta t^3) \ ,$$

- Leap-Frog and Velocity Verlet differ when combined with Thermostats and Barostats
- What is a good integrator? A symplectic integrator![3]

[3] See the gromacs manual!

MD in a Nutshell

Ensembles

- Newton's equation conserved energy → NVE
  `integrator=md`
- Thermostat → NVT
  `tcoupl=...`
- Barostat → NPT
  `pcoupl=...`

Ensemble should correspond to experimental conditions!

MD in a Nutshell

Thermostats

Thermostats are additions to equations of motion to go from NVE to NVT.

4 Groups:

- Local thermostats couple to every particle
- Global thermostats couple to the total kinetic energy
- Stochastic thermostats have random variables
- Deterministic thermostats have no random variables

MD in a Nutshell

Rescaling Thermostats

- Simple rescaling thermostat (Global, deterministic)
  - Rescales total kinetic energy $K' = \lambda K$ with $\lambda = \sqrt{T/T_0}$
  - Iso-kinetic ensemble
- Berendsen thermostat (Global, deterministic)[4] `tcouple=Berendsen`
  - Rescales with a finite rate:

$$\frac{\mathrm{d}K}{\mathrm{d}t} = \frac{K(T) - K(T_0)}{\tau}$$

- "flying ice cube" problem
- No canonical ensemble!

[4] Berendsen et al., JCP **81**,3684 (1984)

MD in a Nutshell

Rescaling Thermostats

- Stochastic velocity rescaling thermostat (global)[5]
  `tcouple=v-rescale`
  - Chooses kinetic energy from the canonical distribution:

$$\frac{\mathrm{d}K}{\mathrm{d}t} = \frac{K(T) - K(T_0)}{\tau} + 2\sqrt{\frac{K(T)K(T_0)}{\tau}}\eta$$

  - Canonical ensemble!
  - equivalent to Langevin thermostat in K
- "flying ice cube" problem
- Localization: acts on kinetic energy of separate groups indepently

[5] Bussi, Donadio, Parrinello JCP **126**, 014101 (2007)

MD in a Nutshell

Stochastic Dynamics

- Add a stochastic and damping force to each particle:

$$m_i\ddot{\vec{r}}_i = \vec{F}_i - \xi_i\vec{v}_i + \sigma_i\vec{\eta}_i \ ,$$

with $\xi_i k_\mathrm{B} T = \sigma_i^2$ (Fluctuation-Dissipation theorem)

- Every particle is coupled to its own distribution
- Changes the dynamics
- Stabilizes dynamics → bigger time step
- Also known as Langevin thermostat
- No momentum conservation
- Can be used as implicit solvent
- `integrator=sd`

MD in a Nutshell

Andersen Thermostats

- Choose velocities randomly from a Maxwell distribution[6]
- Used to generate starting velocities
  `gen_vel=yes`

[6] Andersen, JCP **72**,2384 (1980)

MD in a Nutshell

Nosé-Hoover Thermostats

- Add an extra degree of freedom to the Hamiltonian

$$H = \sum_{i=1}^{N} \frac{\vec{p}_i}{2m_i} + U\left(\vec{r}_1, \vec{r}_2, \ldots, \vec{r}_N\right) + \frac{p_\xi^2}{2Q} + N_f kT\xi$$

- Deterministic
- Virtual mass Q needs to be chosen
- Extension: Nosé-Hoover chain - thermostat the thermostat[7]. `nh-chain-length=X`

$$H = \sum_{i=1}^{N} \frac{\vec{p}_i}{2m_i} + U\left(\vec{r}_1, \vec{r}_2, \ldots, \vec{r}_N\right) + \sum_{k=1}^{M} \frac{p_{\xi_k}^2}{2Q'_k} + N_f kT\xi_1 + kT\sum_{k=2}^{M} \xi_k$$

- `tcouple=nose-hoover`

[7] Martyna, Klein and Tuckerman, JCP**97**, 2635 (1992)

MD in a Nutshell

Barostats

Barostats are extensions to the equations of motion to go from NVT to NPT.

- Berendsen barostat
  `pcoupl=Berendsen`
  - Rescales simulation box to the right pressure with a constant rate

$$\frac{\partial \mathbf{P}}{\partial t} = \frac{\mathbf{P}_0 - \mathbf{P}}{\tau_p}.$$

- Parrinello-Rahman barostat[8]
  `pcoupl=Parrinello-Rahman`
  - Adds degree of freedom for the box vectors

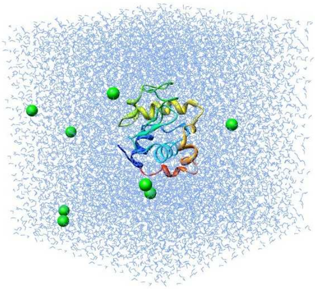
$$\frac{\partial \vec{b}^2}{\partial t^2} = V\mathbf{W}^{-1}\vec{b}'^{-1}\left(\mathbf{P} - \mathbf{P}_{ref}\right).$$

[8] Parrinello and Rahman, J Appl Phys **52**, 7182 (1981)

MD Simulations

1AKI constrained NVT equilibration

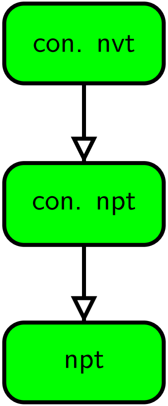
- Use new directory:
  `mkdir nvt; cd nvt`
- Copy over `confout.gro`
  `cp ../confout.gro conf.gro`
- Copy over `topol.top` and `posre.itp`
- Get Christoph's mdp file
- Let's discuss it.
- Compile a tpr file: `gmx grompp`
- Run `gmx mdrun -v`
- Use `gmx energy` to look at the pressure



MD Simulations

1AKI constrained NPT Equilibration

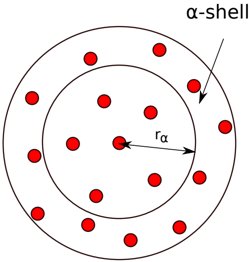
- Use new directory:
  `mkdir npt; cd npt`
- Copy over `confout.gro`
  `cp ../nvt/confout.gro conf.gro`
- Copy over `topol.top` and `posre.itp`
- Get Christoph's mdp file
- Let's discuss the input
- Compile a tpr file: `gmx grompp`
- Run `gmx mdrun -v`
- Use `gmx energy` to look at the box size



MD Simulations

1AKI MD Production

- Use new directory:
  `mkdir md; cd md`
- Copy over `confout.gro`
  `cp ../npt/confout.gro conf.gro`
- Copy over `topol.top`
- Get Christoph's mdp file
- Let's discuss the input
- Compile a tpr file: `gmx grompp`
- Run `gmx mdrun -v`
- Use `gmx energy` to look at the energies
- Use `gmx rdf` to calculate the radial dist. function

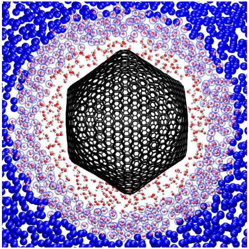


MD Simulations

Outlook

Things we have not discussed:[9]

- Free energy and PMF calculations
- Replica exchange (mpi version only)
- Coarse-grained simulations
- GROMACS supports multiple force fields
- Shell molecular dynamics simulations
- Adaptive resolution simulations
- Essential dynamics
- Implicit solvent simulations
- many, many more analysis features (try `gmx help commands`)
- . . .



[9] See the gromacs manual!

GROMACS on HPC

Introduction

"GROMACS Runs One Microsecond At Cannonball Speeds" using:

- Optimized neighbor list for water
- Vectorized force kernels
- Parallelization:
  - Domain Decomposition
  - Threaded particle blocks (Verlet only)
  - Threaded PME calculations
  - Offload to GPUs (Verlet only)
- Separated nodes for PME calculations (use `gmx tune_pme`)

Don't panic in case of a doubt let `gmx mdrun` decide!

GROMACS on HPC

Test System

- Create topol.top:

  ```
  #include "gromos43a1.ff/forcefield.itp"
  #include "gromos43a1.ff/spce.itp"
  [ system ]
  Water
  [ molecules ]
  ```

- Create a box of water

  ```
  gmx solvate -cs spc216.gro -p topol.top↩
  -box 5 5 5 -o conf.gro
  ```

- Get Christoph's mdp file
- Build different system sizes using:

  ```
  gmx genconf -nbox X X X
  ```

GROMACS on HPC

Considerations

Options of `gmx mdrun` to play

- `-nb`
- `-nt`
- `-ntomp`
- `-ntmpi`
- `-dd`
- `-dlb`
- `-npme`

Other tweaks

- MPI vs threads
- Group scheme (`cutoff-scheme=Group`)
- Verlet scheme (`cutoff-scheme=Verlet`)

GROMACS on HPC

GROMACS in the Queue

- to wall clock time use: `gmx mdrun -maxh xxx`
- `gmx mdrun` creates checkpoint every 15min (-cpt option)
- to start from a checkpoint use -cpi option (smart option)
- chain jobs[10]

```
...script header
gmx mdrun -maxh 36 -cpi state.cpt -append
[ $? -eq 0 ] && [ ! -f confout.gro ] && msub $0
```

[10] moab: `msub -l depend=jobid xxx.log`

Conclusion

- GROMACS is great!
- GROMACS is fast!
- GROMACS is free!
- There is a manual out there! (RTFM)
- Join gmx-users mailing list and ask questions!

The End

Thank you for your attention !

Los Alamos
NATIONAL LABORATORY
EST. 1943