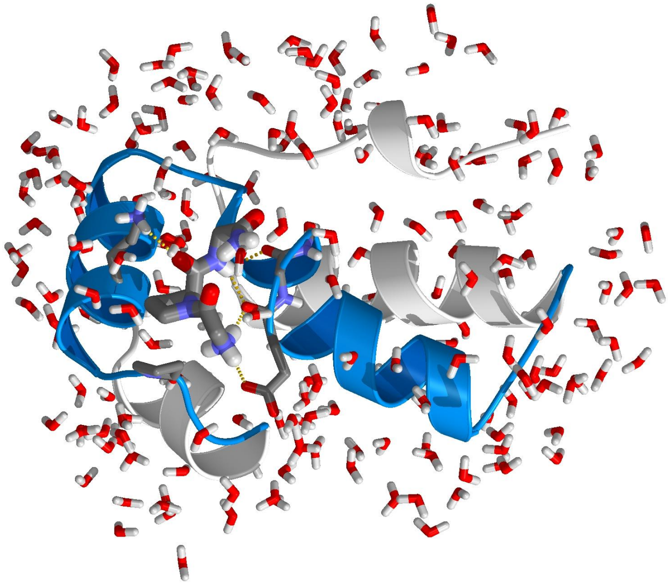
GROMACS
USER
MANUAL

Groningen Machine for Chemical Simulations



Version 3.0

Don’t forget to check out the online resources at www.gromacs.org.

GROMACS
USER MANUAL

Version 3.0

David van der Spoel
Aldert R. van Buuren
Emile Apol
Pieter J. Meulenhoff
D. Peter Tieleman
Alfons L.T.M. Sijbers
Berk Hess
K. Anton Feenstra
Erik Lindahl
Rudi van Drunen
Herman J.C. Berendsen

Department of Biophysical Chemistry, University of Groningen.
Nijenborgh 4, 9747 AG Groningen, The Netherlands.

Preface & Disclaimer

This manual is not complete and has no pretention to be so due to lack of time of the contributors – our first priority is to improve the software. It is meant as a source of information and references for the GROMACS user. It covers both the physical background of MD simulations in general and details of the GROMACS software in particular. The manual is continuously being worked on, which in some cases might mean the information is not entirely correct.

When citing this document in any scientific publication please refer to it as:

> van der Spoel, D., A. R. van Buuren, E. Apol, P. J. Meulenhoff, D. P. Tieleman, A. L. T. M. Sijbers, B. Hess, K. A. Feenstra, E. Lindahl, R. van Drunen and H. J. C. Berendsen, *Gromacs User Manual version 3.0*, Nijenborgh 4, 9747 AG Groningen, The Netherlands. Internet: www.gromacs.org (2001)

or, if you use BibTeX, you can directly copy the following:

```
@Manual{gmx30,
  title =  "Gromacs {U}ser {M}anual version 3.0",
  author = "David van der Spoel and Aldert R. van Buuren and
           Emile Apol and Pieter J. Meulen\-hoff and D. Peter
           Tieleman and Alfons L. T. M. Sij\-bers and Berk Hess
           and K. Anton Feenstra and Erik Lindahl and
           Rudi van Drunen and Herman J. C. Berendsen",
  address= "Nij\-enborgh 4, 9747 AG Groningen, The Netherlands.
           Internet: http://www.gromacs.org",
  year =   "2001"
}
```

We humbly ask that you cite the GROMACS papers [1, 2] when you publish your results. Any future developement depends on academic research grants, since the package is distributed as free software!

Any comments are welcome, please send them by e-mail to gromacs@gromacs.org.

Groningen, August 10, 2001.

Department of Biophysical Chemistry
University of Groningen
Nijenborgh 4
9747 AG Groningen
The Netherlands
Fax: +31-503634800

Online Resources

You can find more documentation and other material at our homepage www.gromacs.org. Among other things there is an online reference, several GROMACS mailing lists with archives and contributed topologies/force fields.

Manual and GROMACS versions

We try to release an updated version of the manual whenever we release a new version of the software, so in general it is a good idea to use a manual with the same major and minor release number as your GROMACS installation. Any revision numbers (like 3.0.1) are however independent, to make it possible to implement bugfixes and manual improvements if necessary.

GROMACS is *Free Software*

The entire GROMACS package is available under the GNU General Public License. This means it's free as in free speech, not just that you can use it without paying us money. For details, check the COPYING file in the source code or consult www.gnu.org/copyleft/gpl.html.

The GROMACS source code and Linux packages are available on our homepage!

Contents

Index

---

[62] de Loof, H., Nilsson, L., Rigler, R. Molecular dynamics simulations of galanin in aqueous and nonaqueous solution. J. Am. Chem. Soc. 114:4028–4035, 1992.

[63] Feenstra, K. A., Hess, B., Berendsen, H. J. C. Improving efficiency of large time-scale molecular dynamics simulations of hydrogen-rich systems. J. Comp. Chem. 20:786–798, 1999.

[64] Feenstra, K. A., Scheek, R. M., Berendsen, H. J. C., Mark, A. E. Analysis of the hierarchy of motion of globular proteins with implications for protein folding. (submitted to PROTEINS: Struct. Funct. Gen., feb. 2001).

[65] Allen, M. P., Tildesley, D. J. Computer Simulations of Liquids. Oxford: Oxford Science Publications. 1987.

[66] van der Spoel, D., Berendsen, H. J. C. Molecular dynamics simulations of Leu-enkephalin in water and DMSO. Biophys. J. 72:2032–2041, 1997.

[67] van der Spoel, D., van Maaren, P. J., Berendsen, H. J. C. A systematic study of water models for molecular simulation. J. Chem. Phys. 108:10220–10230, 1998.

[68] Smith, P. E., van Gunsteren, W. F. The viscosity of spc and spc/e water. Comp. Phys. Comm. 215:315–318, 1993.

[69] Balasubramanian, S., Mundy, C. J., Klein, M. L. Shear viscosity of polar fluids: Miolecular dynamics calculations of water. J. Chem. Phys. 105:11190–11195, 1996.

[70] Amadei, A., Linssen, A. B. M., Berendsen, H. J. C. Essential dynamics of proteins. PROTEINS: Struct. Funct. Gen. 17:412–425, 1993.

[71] Hess, B. Similarities between principal components of protein dynamics and random diffusion. Phys. Rev. **E** 62:8438–8448, 2000.

[72] Kabsch, W., Sander, C. Dictionary of protein secondary structure: Pattern recognition of hydrogen-bonded and geometrical features. Biopolymers 22:2577–2637, 1983.

[73] Williamson, M. P., Asakura, T. Empirical comparisons of models for chemical-shift calculation in proteins. J. Magn. Reson. Ser. B 101:63–71, 1993.

[74] Bekker, H., Berendsen, H. J. C., Dijkstra, E. J., Achterop, S., v. Drunen, R., v. d. Spoel, D., Sijbers, A., Keegstra, H., Reitsma, B., Renardus, M. K. R. Gromacs method of virial calculation using a single sum. In *Physics Computing 92* (Singapore, 1993). de Groot, R. A., Nadrchal, J., eds. . World Scientific.

[75] Berendsen, H. J. C., Grigera, J. R., Straatsma, T. P. The missing term in effective pair potentials. J. Phys. Chem. 91:6269–6271, 1987.

[76] Bekker, H. Ontwerp van een special-purpose computer voor moleculaire dynamica simulaties. Master's thesis. RuG. 1987.

[77] van Gunsteren, W. F., Berendsen, H. J. C. Molecular dynamics of simple systems. Practicum Handleiding voor MD Practicum Nijenborgh 4, 9747 AG, Groningen, The Netherlands 1994.

[46] van Gunsteren, W. F., Billeter, S. R., Eising, A. A., Hünenberger, P. H., Krüger, P., Mark, A. E., Scott, W. R. P., Tironi, I. G. Biomolecular Simulation: The GROMOS96 manual and user guide. Zürich, Switzerland: Hochschulverlag AG an der ETH Zürich. 1996.

[47] Morse, P. M. Diatomic molecules according to the wave mechanics. II. vibrational levels. Phys. Rev. 34:57–64, 1929.

[48] Berendsen, H. J. C., Postma, J. P. M., van Gunsteren, W. F., Hermans, J. Interaction models for water in relation to protein hydration. In: Intermolecular Forces. Pullman, B. ed. . D. Reidel Publishing Company Dordrecht 1981 331–342.

[49] Ferguson, D. M. Parametrization and evaluation of a flexible water model. J. Comp. Chem. 16:501–511, 1995.

[50] Jorgensen, W. L., Tirado-Rives, J. The OPLS potential functions for proteins. energy minimizations for crystals of cyclic peptides and crambin. J. Am. Chem. Soc. 110:1657–1666, 1988.

[51] Torda, A. E., Scheek, R. M., van Gunsteren, W. F. Time-dependent distance restraints in molecular dynamics simulations. Chem. Phys. Lett. 157:289–294, 1989.

[52] van Gunsteren, W. F., Mark, A. E. Validation of molecular dynamics simulations. J. Chem. Phys. 108:6109–6116, 1998.

[53] Berendsen, H. J. C., van Gunsteren, W. F. Molecular dynamics simulations: Techniques and approaches. In: Molecular Liquids-Dynamics and Interactions. et al., A. J. B. ed. NATO ASI C 135. Reidel Dordrecht, The Netherlands 1984 475–500.

[54] Ewald, P. P. Die Berechnung optischer und elektrostatischer Gitterpotentiale. Ann. Phys. 64:253–287, 1921.

[55] Darden, T., York, D., Pedersen, L. Particle mesh Ewald: An N-log(N) method for Ewald sums in large systems. J. Chem. Phys. 98:10089–10092, 1993.

[56] Essmann, U., Perera, L., Berkowitz, M. L., Darden, T., Lee, H., Pedersen, L. G. A smooth particle mesh ewald potential. J. Chem. Phys. 103:8577–8592, 1995.

[57] Hockney, R. W., Eastwood, J. W. Computer simulation using particles. New York: McGraw-Hill. 1981.

[58] Luty, B. A., Tironi, I. G., van Gunsteren, W. F. Lattice-sum methods for calculating electrostatic interactions in molecular simulations. J. Chem. Phys. 103:3014–3021, 1995.

[59] King, P. M., Mark, A. E., van Gunsteren, W. F. Re-parameterization of aromatic interactions in the GROMOS force-field. Private Communication 1993.

[60] Ryckaert, J. P., Bellemans, A. Far. Disc. Chem. Soc. 66:95, 1978.

[61] on Biochemical Nomenclature, I.-I. C. Abrreviations and symbols for the description of the conformation of polypeptide chains. tentative rules (1969). Biochemistry 9:3471–3478, 1970.

---

[30] Levitt, M., Sander, C., Stern, P. S. The normal modes of a protein: Native bovine pancreatic trypsin inhibitor. Proc. Natl. Acad. Sci. USA 10:181–199, 1983.

[31] Gō, N., Noguti, T., Nishikawa, T. Dynamics of a small globular protein in terms of low-frequency vibrational modes. Proc. Natl. Acad. Sci. USA 80:3696–3700, 1983.

[32] Brooks, B., Karplus, M. Harmonic dynamics of proteins: Normal modes and fluctuations in bovine pancreatic trypsin inhibitor. Proc. Natl. Acad. Sci. USA 80:6571–6575, 1983.

[33] Hayward, S., Gō, N. Collective variable description of native protein dynamics. Annu. Rev. Phys. Chem. 46:223–250, 1995.

[34] de Groot, B. L., Amadei, A., van Aalten, D. M. F., Berendsen, H. J. C. Towards an exhaustive sampling of the configurational spaces of the two forms of the peptide hormone guanylin. J. Biomol. Str. Dyn. 13(5):741–751, 1996.

[35] de Groot, B. L., Amadei, A., Scheek, R. M., van Nuland, N. A. J., Berendsen, H. J. C. An extended sampling of the configurational space of hpr from *e. coli*. PROTEINS: Struct. Funct. Gen. 26:314–322, 1996.

[36] Vriend, G. WHAT IF: a molecular modeling and drug design program. J. Mol. Graph. 8:52–56, 1990.

[37] Fincham, D. Parallel computers and molecular simulation. Mol. Sim. 1:1, 1987.

[38] Raine, A. R. C., Fincham, D., Smith, W. Systolic loop methods for molecular dynamics simulation. Comp. Phys. Comm. 55:13–30, 1989.

[39] van Gunsteren, W. F., Berendsen, H. J. C. Gromos-87 manual. Biomos BV Nijenborgh 4, 9747 AG Groningen, The Netherlands 1987.

[40] van Buuren, A. R., Marrink, S. J., Berendsen, H. J. C. A molecular dynamics study of the decane/water interface. J. Phys. Chem. 97:9206–9212, 1993.

[41] Mark, A. E., van Helden, S. P., Smith, P. E., Janssen, L. H. M., van Gunsteren, W. F. Convergence properties of free energy calculations: α-cyclodextrin complexes as a case study. J. Am. Chem. Soc. 116:6293–6302, 1994.

[42] Jorgensen, W. L., Chandrasekhar, J., Madura, J. D., Impey, R. W., Klein, M. L. Comparison of simple potential functions for simulating liquid water. J. Chem. Phys. 79:926–935, 1983.

[43] van Buuren, A. R., Berendsen, H. J. C. Molecular dynamics simulation of the stability of a 22 residue alpha-helix in water and 30 % trifluoroethanol. Biopolymers 33:1159–1166, 1993.

[44] Liu, H., Müller-Plathe, F., van Gunsteren, W. F. A force field for liquid dimethyl sulfoxide and liquid proporties of liquid dimethyl sulfoxide calculated using molecular dynamics simulation. J. Am. Chem. Soc. 117:4363–4366, 1995.

[45] Tironi, I. G., Sperb, R., Smith, P. E., van Gunsteren, W. F. A generalized reaction field method for molecular dynamics simulations. J. Chem. Phys. 102:5451–5459, 1995.

[13] Berendsen, H. J. C. Electrostatic interactions. In: Computer Simulation of Biomolecular Systems. van Gunsteren, W. F., Weiner, P. K., Wilkinson, A. J. eds. . ESCOM Leiden 1993 161–181.

[14] Hockney, R. W., Goel, S. P. J. Comp. Phys. 14:148, 1974.

[15] Verlet., L. Phys. Rev. 34:1311–1327, 1967.

[16] Berendsen, H. J. C., van Gunsteren, W. F. Practical algorithms for dynamics simulations.

[17] Berendsen, H. J. C., Postma, J. P. M., DiNola, A., Haak, J. R. Molecular dynamics with coupling to an external bath. J. Chem. Phys. 81:3684–3690, 1984.

[18] Nosé, S. A molecular dynamics method for simulations in the canonical ensemble. Mol. Phys. 52:255–268, 1984.

[19] Hoover, W. G. Canonical dynamics: equilibrium phase-space distributions. Phys. Rev. **A** 31:1695–1697, 1985.

[20] Berendsen, H. J. C. Transport properties computed by linear response through weak coupling to a bath. In: Computer Simulations in Material Science. Meyer, M., Pontikis, V. eds. . Kluwer 1991 139–155.

[21] Parrinello, M., Rahman, A. Polymorphic transitions in single crystals: A new molecular dynamics method. J. Appl. Phys. 52:7182–7190, 1981.

[22] Nosé, S., Klein, M. L. Constant pressure molecular dynamics for molecular systems. Mol. Phys. 50:1055–1076, 1983.

[23] Dick, B. G., Overhauser, A. W. Theory of the dielectric constants of alkali halide crystals. Phys. Rev. 112:90–103, 1958.

[24] Jordan, P. C., van Maaren, P. J., Mavri, J., van der Spoel, D., Berendsen, H. J. C. Towards phase transferable potential functions: Methodology and application to nitrogen. J. Chem. Phys. 103:2272–2285, 1995.

[25] van Maaren, P. J., van der Spoel, D. Molecular dynamics simulations of a water with a novel shell-model potential. J. Phys. Chem. B. 105:2618–2626, 2001.

[26] Ryckaert, J. P., Ciccotti, G., Berendsen, H. J. C. Numerical integration of the cartesian equations of motion of a system with constraints; molecular dynamics of n-alkanes. J. Comp. Phys. 23:327–341, 1977.

[27] Miyamoto, S., Kollman, P. A. SETTLE: An analytical version of the SHAKE and RATTLE algorithms for rigid water models. J. Comp. Chem. 13:952–962, 1992.

[28] Hess, B., Bekker, H., Berendsen, H. J. C., Fraaije, J. G. E. M. LINCS: A linear constraint solver for molecular simulations. J. Comp. Chem. 18:1463–1472, 1997.

[29] van Gunsteren, W. F., Berendsen, H. J. C. A leap-frog algorithm for stochastic dynamics. Mol. Sim. 1:173–185, 1988.

Bibliography

[1] Berendsen, H. J. C., van der Spoel, D., van Drunen, R. GROMACS: A message-passing parallel molecular dynamics implementation. Comp. Phys. Comm. 91:43–56, 1995.

[2] Lindahl, E., Hess, B., van der Spoel, D. Gromacs 3.0: A package for molecular simulation and trajectory analysis. In press 2001.

[3] van der Spoel, D., Vogel, H. J., Berendsen, H. J. C. Molecular dynamics simulations of N-terminal peptides from a nucleotide binding protein. PROTEINS: Struct. Funct. Gen. 24:450–466, 1996.

[4] van Gunsteren, W. F., Berendsen, H. J. C. Computer simulation of molecular dynamics: Methodology, applications, and perspectives in chemistry. Angew. Chem. Int. Ed. Engl. 29:992–1023, 1990.

[5] Fraaije, J. G. E. M. Dynamic density functional theory for microphase separation kinetics of block copolymer melts. J. Chem. Phys. 99:9202–9212, 1993.

[6] McQuarrie, D. A. Statistical Mechanics. New York: Harper & Row. 1976.

[7] van Gunsteren, W. F., Berendsen, H. J. C. Algorithms for macromolecular dynamics and constraint dynamics. Mol. Phys. 34:1311–1327, 1977.

[8] Nilges, M., Clore, G. M., Gronenborn, A. M. Determination of three-dimensional structures of proteins from interproton distance data by dynamical simulated annealing from a random array of atoms. FEBS Lett. 239:129–136, 1988.

[9] van Schaik, R. C., Berendsen, H. J. C., Torda, A. E., van Gunsteren, W. F. A structure refinement method based on molecular dynamics in 4 spatial dimensions. J. Mol. Biol. 234:751–762, 1993.

[10] Zimmerman, K. All purpose molecular mechanics simulator and energy minimizer. J. Comp. Chem. 12:310–319, 1991.

[11] Adams, D. J., Adams, E. M., Hills, G. J. The computer simulation of polar liquids. Mol. Phys. 38:387–400, 1979.

[12] Bekker, H., Dijkstra, E. J., Renardus, M. K. R., Berendsen, H. J. C. An efficient, box shape independent non-bonded force and virial algorithm for molecular dynamics. Mol. Sim. 14:137–152, 1995.

`-b`	time	`-1`	First frame (ps) to read from trajectory
`-e`	time	`-1`	Last frame (ps) to read from trajectory
`-dt`	time	`-1`	Only use frame when t MOD dt = first time (ps)

selected to combine the matrices. In this case, a new color map will be generated with a red gradient for negative numbers and a blue for positive.

If the color coding and legend labels of both matrices are identical, only one legend will be displayed, else two separate legends are displayed.

`-title` can be set to `none` to suppress the title, or to `ylabel` to show the title in the Y-label position (alongside the Y-axis).

With the `-rainbow` option dull grey-scale matrices can be turned into attractive color pictures.

Merged or rainbowed matrices can be written to an XPixelMap file with the `-xpm` option.

Files

`-f`	`root.xpm`	Input	X PixMap compatible matrix file
`-f2`	`root2.xpm`	Input, Opt.	X PixMap compatible matrix file
`-di`	`ps.m2p`	Input, Opt., Lib.	Input file for mat2ps
`-do`	`out.m2p`	Output, Opt.	Input file for mat2ps
`-o`	`plot.eps`	Output, Opt.	Encapsulated PostScript (tm) file
`-xpm`	`root.xpm`	Output, Opt.	X PixMap compatible matrix file

Other options

`-h`	bool	`no`	Print help info and quit
`-nice`	int	`0`	Set the nicelevel
`-w`	bool	`no`	View output xvg, xpm, eps and pdb files
`-frame`	bool	`yes`	Display frame, ticks, labels, title and legend
`-title`	enum	`top`	Show title at: `top`, `once`, `ylabel` or `none`
`-yonce`	bool	`no`	Show y-label only once
`-legend`	enum	`both`	Show legend: `both`, `first`, `second` or `none`
`-diag`	enum	`first`	Diagonal: `first`, `second` or `none`
`-combine`	enum	`halves`	Combine two matrices: `halves`, `add`, `sub`, `mult` or `div`
`-bx`	real	`0`	Box x-size (also y-size when -by is not set)
`-by`	real	`0`	Box y-size
`-rainbow`	enum	`no`	Rainbow colors, convert white to: `no`, `blue` or `red`
`-gradient`	vector	`0 0 0`	Re-scale colormap to a smooth gradient from white 1,1,1 to r,g,b
`-skip`	int	`1`	only write out every nr-th row and column
`-zeroline`	bool	`no`	insert line in xpm matrix where axis label is zero

E.73 xrama

xrama shows a Ramachandran movie, that is, it shows the Phi/Psi angles as a function of time in an X-Window.

Static Phi/Psi plots for printing can be made with g_rama.

Some of the more common X command line options can be used:
-bg, -fg change colors, -font fontname, changes the font.

Files

`-f`	`traj.xtc`	Input	Generic trajectory: xtc trr trj gro g96 pdb
`-s`	`topol.tpr`	Input	Generic run input: tpr tpb tpa

Other options

`-h`	bool	`no`	Print help info and quit
`-nice`	int	`0`	Set the nicelevel

List of Figures

---

the usual output will be written to file. When running with MPI, a signal to one of the mdrun processes is sufficient, this signal should not be sent to mpirun or the mdrun process that is the parent of the others.

Files

`-s`	`topol.tpr`	Input	Generic run input: tpr tpb tpa
`-o`	`traj.trr`	Output	Full precision trajectory: trr trj
`-x`	`traj.xtc`	Output, Opt.	Compressed trajectory (portable xdr format)
`-c`	`confout.gro`	Output	Generic structure: gro g96 pdb
`-e`	`ener.edr`	Output	Generic energy: edr ene
`-g`	`md.log`	Output	Log file
`-dgdl`	`dgdl.xvg`	Output, Opt.	xvgr/xmgr file
`-table`	`table.xvg`	Input, Opt.	xvgr/xmgr file
`-rerun`	`rerun.xtc`	Input, Opt.	Generic trajectory: xtc trr trj gro g96 pdb
`-ei`	`sam.edi`	Input, Opt.	ED sampling input
`-eo`	`sam.edo`	Output, Opt.	ED sampling output
`-j`	`wham.gct`	Input, Opt.	General coupling stuff
`-jo`	`bam.gct`	Input, Opt.	General coupling stuff
`-ffout`	`gct.xvg`	Output, Opt.	xvgr/xmgr file
`-devout`	`deviatie.xvg`	Output, Opt.	xvgr/xmgr file
`-runav`	`runaver.xvg`	Output, Opt.	xvgr/xmgr file
`-pi`	`pull.ppa`	Input, Opt.	Pull parameters
`-po`	`pullout.ppa`	Output, Opt.	Pull parameters
`-pd`	`pull.pdo`	Output, Opt.	Pull data output
`-pn`	`pull.ndx`	Input, Opt.	Index file

Other options

`-h`	bool	`no`	Print help info and quit
`-nice`	int	`19`	Set the nicelevel
`-deffnm`	string		Set the default filename for all file options
`-np`	int	`1`	Number of nodes, must be the same as used for grompp
`-v`	bool	`no`	Be loud and noisy
`-compact`	bool	`yes`	Write a compact log file
`-multi`	bool	`no`	Do multiple simulations in parallel (only with -np > 1)
`-glas`	bool	`no`	Do glass simulation with special long range corrections
`-ionize`	bool	`no`	Do a simulation including the effect of an X-Ray bombardment on your system

E.72 xpm2ps

xpm2ps makes a beautiful color plot of an XPixelMap file. Labels and axis can be displayed, when they are supplied in the correct matrix format. Matrix data may be generated by programs such as do_dssp, g_rms or g_mdmat.

Parameters are set in the `m2p` file optionally supplied with `-di`. Reasonable defaults are provided. Settings for the y-axis default to those for the x-axis. Font names have a defaulting hierarchy: titlefont -> legendfont; titlefont -> (xfont -> yfont -> ytickfont) -> xtickfont, e.g. setting titlefont sets all fonts, setting xfont sets yfont, ytickfont and xtickfont.

With `-f2` a 2nd matrix file can be supplied, both matrix files will be read simultaneously and the upper left half of the first one (`-f`) is plotted together with the lower right half of the second one (`-f2`). The diagonal will contain values from the matrix file selected with `-diag`. Plotting of the diagonal values can be suppressed altogether by setting `-diag` to `none`. With `-combine` an alternative operation can be

`-kt`	real	`400`	Angle force constant (kJ/mol/rad^2)
`-kp`	real	`5`	Dihedral angle force constant (kJ/mol/rad^2)
`-nexcl`	int	`3`	Number of exclusions
`-H14`	bool	`no`	Use 3rd neighbour interactions for hydrogen atoms
`-alldih`	bool	`no`	Generate all proper dihedrals
`-round`	bool	`yes`	Round off measured values
`-pairs`	bool	`yes`	Output 1-4 interactions (pairs) in topology file
`-name`	string	`ICE`	Name of your molecule

- The atom type selection is primitive. Virtually no chemical knowledge is used
- Periodic boundary conditions screw up the bonding
- No improper dihedrals are generated
- The atoms to atomtype translation table is incomplete (ffG43a1.n2t file in the $GMXLIB directory). Please extend it and send the results back to the GROMACS crew.

E.71 xmdrun

xmdrun is the experimental MD program. New features are tested in this program before being implemented in the default mdrun. Currently under investigation are: polarizibility, glass simulations, Free energy perturbation, X-Ray bombardments and parallel independent simulations.It reads the run input file (`-s`) and distributes the topology over nodes if needed. The coordinates are passed around, so that computations can begin. First a neighborlist is made, then the forces are computed. The forces are globally summed, and the velocities and positions are updated. If necessary shake is performed to constrain bond lengths and/or bond angles. Temperature and Pressure can be controlled using weak coupling to a bath.

mdrun produces at least three output file, plus one log file (`-g`) per node. The trajectory file (`-o`), contains coordinates, velocities and optionally forces. The structure file (`-c`) contains the coordinates and velocities of the last step. The energy file (`-e`) contains energies, the temperature, pressure, etc, a lot of these things are also printed in the log file of node 0. Optionally coordinates can be written to a compressed trajectory file (`-x`).

When running in parallel with PVM or an old version of MPI the `-np` option must be given to indicate the number of nodes.

The option `-dgdl` is only used when free energy perturbation is turned on.

With `-rerun` an input trajectory can be given for which forces and energies will be (re)calculated. Neighbor searching will be performed for every frame, unless `nstlist` is zero (see the `.mdp` file).

ED (essential dynamics) sampling is switched on by using the `-ei` flag followed by an `.edi` file. The `.edi` file can be produced using options in the essdyn menu of the WHAT IF program. mdrun produces a `.edo` file that contains projections of positions, velocities and forces onto selected eigenvectors.

The -table option can be used to pass mdrun a formatted table with user-defined potential functions. The file is read from either the current directory or from the GMXLIB directory. A number of preformatted tables are presented in the GMXLIB dir, for 6-8, 6-9, 6-10, 6-11, 6-12 Lennard Jones potentials with normal Coulomb.

The options `-pi`, `-po`, `-pd`, `-pn` are used for potential of mean force calculations and umbrella sampling. See manual.

When mdrun receives a TERM signal, it will set nsteps to the current step plus one. When mdrun receives a USR1 signal, it will set nsteps to the next multiple of nstxout after the current step. In both cases all

---

List of Tables

of the first n waters is made, the ordered trajectory can be used with any Gromacs program to analyze the n closest waters.

Files

`-f`	`traj.xtc`	Input	Generic trajectory: xtc trr trj gro g96 pdb
`-s`	`topol.tpr`	Input	Structure+mass(db): tpr tpb tpa gro g96 pdb
`-n`	`index.ndx`	Input, Opt.	Index file
`-o`	`ordered.xtc`	Output	Generic trajectory: xtc trr trj gro g96 pdb

Other options

`-h`	bool	`no`	Print help info and quit
`-nice`	int	`19`	Set the nicelevel
`-b`	time	`-1`	First frame (ps) to read from trajectory
`-e`	time	`-1`	Last frame (ps) to read from trajectory
`-dt`	time	`-1`	Only use frame when t MOD dt = first time (ps)
`-na`	int	`3`	Number of atoms in a molecule
`-da`	int	`1`	Atom used for the distance calculation

E.69 wheel

wheel plots a helical wheel representation of your sequence.The input sequence is in the .dat file where the first line contains the number of residues and each consecutive line contains a residuename.

Files

`-f`	`nnnice.dat`	Input	Generic data file
`-o`	`plot.eps`	Output	Encapsulated PostScript (tm) file

Other options

`-h`	bool	`no`	Print help info and quit
`-nice`	int	`19`	Set the nicelevel
`-r0`	int	`1`	The first residue number in the sequence
`-rot0`	real	`0`	Rotate around an angle initially (90 degrees makes sense)
`-T`	string		Plot a title in the center of the wheel (must be shorter than 10 characters, or it will overwrite the wheel)
`-nn`	bool	`yes`	Toggle numbers

E.70 x2top

x2top generates a primitive topology from a coordinate file. The program assumes all hydrogens are present when defining the hybridization from the atom name and the number of bonds. The program can also make an rtp entry, which you can then add to the rtp database.

Files

`-f`	`conf.gro`	Input	Generic structure: gro g96 pdb tpr tpb tpa
`-o`	`out.top`	Output, Opt.	Topology file
`-r`	`out.rtp`	Output, Opt.	Residue Type file used by pdb2gmx

Other options

`-h`	bool	`no`	Print help info and quit
`-nice`	int	`0`	Set the nicelevel
`-kb`	real	`400000`	Bonded force constant (kJ/mol/nm^2)

Using `-trunc` trjconv can truncate `.trj` in place, i.e. without copying the file. This is useful when a run has crashed during disk I/O (one more disk full), or when two contiguous trajectories must be concatenated without have double frames.

`trjcat` is more suitable for concatenating trajectory files.

Option `-dump` can be used to extract a frame at or near one specific time from your trajectory.

Files

`-f`	`traj.xtc`	Input	Generic trajectory: xtc trr trj gro g96 pdb
`-o`	`trajout.xtc`	Output	Generic trajectory: xtc trr trj gro g96 pdb
`-s`	`topol.tpr`	Input, Opt.	Structure+mass(db): tpr tpb tpa gro g96 pdb
`-n`	`index.ndx`	Input, Opt.	Index file
`-fr`	`frames.ndx`	Input, Opt.	Index file

Other options

`-h`	bool	`no`	Print help info and quit
`-nice`	int	`19`	Set the nicelevel
`-b`	time	`-1`	First frame (ps) to read from trajectory
`-e`	time	`-1`	Last frame (ps) to read from trajectory
`-tu`	enum	`ps`	Time unit: `ps`, `fs`, `ns`, `us`, `ms`, `s`, `m` or `h`
`-w`	bool	`no`	View output xvg, xpm, eps and pdb files
`-skip`	int	`1`	Only write every nr-th frame
`-dt`	time	`0`	Only write frame when t MOD dt = first time (ps)
`-dump`	time	`-1`	Dump frame nearest specified time (ps)
`-t0`	time	`0`	Starting time (ps) (default: don't change)
`-timestep`	time	`0`	Change time step between input frames (ps)
`-pbc`	enum	`none`	PBC treatment: `none`, `whole`, `inbox` or `nojump`
`-ur`	enum	`rect`	Unit-cell representation: `rect`, `tric` or `compact`
`-center`	bool	`no`	Center atoms in box
`-box`	vector	`0 0 0`	Size for new cubic box (default: read from input)
`-shift`	vector	`0 0 0`	All coordinates will be shifted by framenr*shift
`-fit`	bool	`no`	Fit molecule to ref structure in the structure file
`-pfit`	bool	`no`	Progressive fit, to the previous fitted structure
`-ndec`	int	`3`	Precision for .xtc and .gro writing in number of decimal places
`-vel`	bool	`yes`	Read and write velocities if possible
`-force`	bool	`no`	Read and write forces if possible
`-trunc`	time	`-1`	Truncate input trj file after this time (ps)
`-exec`	string		Execute command for every output frame with the frame number as argument
`-app`	bool	`no`	Append output
`-split`	time	`0`	Start writing new file when t MOD split = first time (ps)
`-sep`	bool	`no`	Write each frame to a separate .gro or .pdb file

E.68 trjorder

trjorder orders molecules according to the smallest distance to atoms in a reference group. It will ask for a group of reference atoms and a group of molecules. For each frame of the trajectory the selected molecules will be reordered according to the shortest distance between atom number `-da` in the molecule and all the atoms in the reference group. All atoms in the trajectory are written to the output trajectory.

trjorder can be useful for e.g. analyzing the n waters closest to a protein. In that case the reference group would be the protein and the group of molecules would consist of all the water atoms. When an index group

---

Chapter 1

Introduction

1.1 Computational Chemistry and Molecular Modeling

GROMACS is an engine to perform molecular dynamics simulations and energy minimization. These are two of the many techniques that belong to the realm of computational chemistry and molecular modeling. *Computational Chemistry* is just a name to indicate the use of computational techniques in chemistry, ranging from quantum mechanics of molecules to dynamics of large complex molecular aggregates. *Molecular modeling* indicates the general process of describing complex chemical systems in terms of a realistic atomic model, with the aim to understand and predict macroscopic properties based on detailed knowledge on an atomic scale. Often molecular modeling is used to design new materials, for which the accurate prediction of physical properties of realistic systems is required.

Macroscopic physical properties can be distinguished in (a) *static equilibrium properties*, such as the binding constant of an inhibitor to an enzyme, the average potential energy of a system, or the radial distribution function in a liquid, and (b) *dynamic or non-equilibrium properties*, such as the viscosity of a liquid, diffusion processes in membranes, the dynamics of phase changes, reaction kinetics, or the dynamics of defects in crystals. The choice of technique depends on the question asked and on the feasibility of the method to yield reliable results at the present state of the art. Ideally, the (relativistic) time-dependent Schrödinger equation describes the properties of molecular systems with high accuracy, but anything more complex than the equilibrium state of a few atoms cannot be handled at this *ab initio* level. Thus approximations are mandatory; the higher the complexity of a system and the longer the time span of the processes of interest is, the more severe approximations are required. At a certain point (reached very much earlier than one would wish) the *ab initio* approach must be augmented or replaced by *empirical* parameterization of the model used. Where simulations based on physical principles of atomic interactions still fail due to the complexity of the system (as is unfortunately still the case for the prediction of protein folding; but: there is hope!) molecular modeling is based entirely on a similarity analysis of known structural and chemical data. The QSAR methods (Quantitative Structure-Activity Relations) and many homology-based protein structure predictions belong to the latter category.

Macroscopic properties are always ensemble averages over a representative statistical ensemble (either equilibrium or non-equilibrium) of molecular systems. For molecular modeling this has two important consequences:

- The knowledge of a single structure, even if it is the structure of the global energy minimum, is not sufficient. It is necessary to generate a representative ensemble at a given temperature, in order to compute macroscopic properties. But this is not enough to compute thermodynamic equilibrium properties that are based on free energies, such as phase equilibria, binding constants, solubilities, relative stability of molecular conformations, etc. The computation of free energies and thermodynamic potentials requires special extensions of molecular simulation techniques.
- While molecular simulations in principle provide atomic details of the structures and motions, such details are often not relevant for the macroscopic properties of interest. This opens the way to simplify the description of interactions and average over irrelevant details. The science of statistical mechanics provides the theoretical framework for such simplifications. There is a hierarchy of methods ranging from considering groups of atoms as one unit, describing motion in a reduced number of collective coordinates, averaging over solvent molecules with potentials of mean force combined with stochastic dynamics [4], to *mesoscopic dynamics* describing densities rather than atoms and fluxes as response to thermodynamic gradients rather than velocities or accelerations as response to forces [5].

For the generation of a representative equilibrium ensemble two methods are available: (a) *Monte Carlo simulations* and (b) *Molecular Dynamics simulations*. For the generation of non-equilibrium ensembles and for the analysis of dynamic events, only the second method is appropriate. While Monte Carlo simulations are more simple than MD (they do not require the computation of forces), they do not yield significantly better statistics than MD in a given amount of computer time. Therefore MD is the more universal technique. If a starting configuration is very far from equilibrium, the forces may be excessively large and the MD simulation may fail. In those cases a robust *energy minimization* is required. Another reason to perform an energy minimization is the removal of all kinetic energy from the system: if several 'snapshots' from dynamic simulations must be compared, energy minimization reduces the thermal 'noise' in the structures and potential energies, so that they can be compared better.

1.2 Molecular Dynamics Simulations

MD simulations solve Newton's equations of motion for a system of N interacting atoms:

$$m_i \frac{\partial^2 \boldsymbol{r}_i}{\partial t^2} = \boldsymbol{F}_i, \; i = 1 \ldots N. \tag{1.1}$$

The forces are the negative derivatives of a potential function $V(\boldsymbol{r}_1, \boldsymbol{r}_2, \ldots, \boldsymbol{r}_N)$:

$$\boldsymbol{F}_i = -\frac{\partial V}{\partial \boldsymbol{r}_i} \tag{1.2}$$

The equations are solved simultaneously in small time steps. The system is followed for some time, taking care that the temperature and pressure remain at the required values, and the coordinates are written to an output file at regular intervals. The coordinates as a function of time

---

E.67 trjconv

trjconv can convert trajectory files in many ways:
1. from one format to another
2. select a subset of atoms
3. remove periodicity from molecules
4. keep multimeric molecules together
5. center atoms in the box
6. fit atoms to reference structure
7. reduce the number of frames
8. change the timestamps of the frames (`-t0` and `-timestep`)

The program `trjcat` can concatenate multiple trajectory files.

Currently seven formats are supported for input and output: `.xtc`, `.trr`, `.trj`, `.gro`, `.g96`, `.pdb` and `.g87`. The file formats are detected from the file extension. The precision of `.xtc` and `.gro` output is taken from the input file for `.xtc`, `.gro` and `.pdb`, and from the `-ndec` option for other input formats. The precision is always taken from `-ndec`, when this option is set. All other formats have fixed precision. `.trr` and `.trj` output can be single or double precision, depending on the precision of the trjconv binary. Note that velocities are only supported in `.trr`, `.trj`, `.gro` and `.g96` files.

Option `-app` can be used to append output to an existing trajectory file. No checks are performed to ensure integrity of the resulting combined trajectory file. `.pdb` files with all frames concatenated can be viewed with `rasmol -nmrpdb`.

It is possible to select part of your trajectory and write it out to a new trajectory file in order to save disk space, e.g. for leaving out the water from a trajectory of a protein in water. **ALWAYS** put the original trajectory on tape! We recommend to use the portable `.xtc` format for your analysis to save disk space and to have portable files.

There are two options for fitting the trajectory to a reference either for essential dynamics analysis or for whatever. The first option is just plain fitting to a reference structure in the structure file, the second option is a progressive fit in which the first timeframe is fitted to the reference structure in the structure file to obtain and each subsequent timeframe is fitted to the previously fitted structure. This way a continuous trajectory is generated, which might not be the case when using the regular fit method, e.g. when your protein undergoes large conformational transitions.

Option `-pbc` sets the type of periodic boundary condition treatment. `whole` puts the atoms in the box and then makes broken molecules whole (a run input file is required). `inbox` puts all the atoms in the box. `nojump` checks if atoms jump across the box and then puts them back. This has the effect that all molecules will remain whole (provided they were whole in the initial conformation), note that this ensures a continuous trajectory but molecules may diffuse out of the box. The starting configuration for this procedure is taken from the structure file, if one is supplied, otherwise it is the first frame. `-pbc` is ignored when `-fit` of `-pfit` is set, in that case molecules will be made whole.

Option `-ur` sets the unit cell representation for options `whole` and `inbox` of `-pbc`. All three options give different results for triclinc boxes and identical results for rectangular boxes. `rect` is the ordinary brick shape. `tric` is the triclinic unit cell. `compact` puts all atoms at the closest distance from the center of the box. This can be useful for visualizing e.g. truncated octahedrons.

Option `-center` centers the system in the box. The user can select the group which is used to determine the geometrical center. Use option `-pbc whole` in addition to `-center` when you want all molecules in the box after the centering.

With `-dt` it is possible to reduce the number of frames in the output. This option relies on the accuracy of the times in your input trajectory, so if these are inaccurate use the `-timestep` option to modify the time (this can be done simultaneously).

E.65 tpbconv

tpbconv can edit run input files in two ways.

1st. by creating a run input file for a continuation run when your simulation has crashed due to e.g. a full disk, or by making a continuation run input file. Note that a frame with coordinates and velocities is needed, which means that when you never write velocities, you can not use tpbconv and you have to start the run again from the beginning.

2nd. by creating a tpx file for a subset of your original tpx file, which is useful when you want to remove the solvent from your tpx file, or when you want to make e.g. a pure Ca tpx file. **WARNING: this tpx file is not fully functional**.

Files

`-s`	`topol.tpr`	Input	Generic run input: tpr tpb tpa
`-f`	`traj.trr`	Input, Opt.	Full precision trajectory: trr trj
`-n`	`index.ndx`	Input, Opt.	Index file
`-o`	`tpxout.tpr`	Output	Generic run input: tpr tpb tpa

Other options

`-h`	bool	`no`	Print help info and quit
`-nice`	int	`0`	Set the nicelevel
`-time`	real	`-1`	Continue from frame at this time (ps) instead of the last frame
`-extend`	real	`0`	Extend runtime by this amount (ps)
`-until`	real	`0`	Extend runtime until this ending time (ps)
`-unconstrained`	bool	`yes`	For a continuous trajectory, the constraints should not be solved before the first step (default)

E.66 trjcat

trjcat concatenates several input trajectory files in sorted order. In case of double time frames the one in the later file is used. By specifying `-settime` you will be asked for the start time of each file. The input files are taken from the command line, such that a command like `trjcat -o fixed.trr *.trr` should do the trick.

Files

`-o`	`trajout.xtc`	Output	Generic trajectory: xtc trr trj gro g96 pdb
`-n`	`index.ndx`	Input, Opt.	Index file

Other options

`-h`	bool	`no`	Print help info and quit
`-nice`	int	`19`	Set the nicelevel
`-b`	real	`-1`	First time to use
`-e`	real	`-1`	Last time to use
`-dt`	real	`0`	Only write frame when t MOD dt = first time
`-prec`	int	`3`	Precision for .xtc and .gro writing in number of decimal places
`-vel`	bool	`yes`	Read and write velocities if possible
`-settime`	bool	`no`	Change starting time interactively
`-sort`	bool	`yes`	Sort trajectory files (not frames)

---

type of bond	type of vibration	wavenumber (cm^{-1})
C-H, O-H, N-H	stretch	3000–3500
C=C, C=O,	stretch	1700–2000
HOH	bending	1600
C-C	stretch	1400–1600
H_2CX	sciss, rock	1000–1500
CCC	bending	800–1000
O-H···O	libration	400– 700
O-H···O	stretch	50– 200

Table 1.1: Typical vibrational frequencies (wavenumbers) in molecules and hydrogen-bonded liquids. Compare $kT/h = 200$ cm^{-1} at 300 K.

represent a *trajectory* of the system. After initial changes, the system will usually reach an *equilibrium state*. By averaging over an equilibrium trajectory many macroscopic properties can be extracted from the output file.

It is useful at this point to consider the limitations of MD simulations. The user should be aware of those limitations and always perform checks on known experimental properties to assess the accuracy of the simulation. We list the approximations below.

The simulations are classical

Using Newton's equation of motion automatically implies the use of *classical mechanics* to describe the motion of atoms. This is all right for most atoms at normal temperatures, but there are exceptions. Hydrogen atoms are quite light and the motion of protons is sometimes of essential quantum mechanical character. For example, a proton may *tunnel* through a potential barrier in the course of a transfer over a hydrogen bond. Such processes cannot be properly treated by classical dynamics! Helium liquid at low temperature is another example where classical mechanics breaks down. While helium may not deeply concern us, the high frequency vibrations of covalent bonds should make us worry! The statistical mechanics of a classical harmonic oscillator differs appreciably from that of a real quantum oscillator, when the resonance frequency ν approximates or exceeds k_BT/h. Now at room temperature the wavenumber $\sigma = 1/\lambda = \nu/c$ at which $h\nu = k_BT$ is approximately 200 cm^{-1}. Thus all frequencies higher than, say, 100 cm^{-1} are suspect of misbehavior in classical simulations. This means that practically all bond and bond-angle vibrations are suspect, and even hydrogen-bonded motions as translational or librational H-bond vibrations are beyond the classical limit (see Table 1.1). What can we do?

Well, apart from real quantum-dynamical simulations, we can do either of two things:
(a) If we perform MD simulations using harmonic oscillators for bonds, we should make corrections to the total internal energy $U = E_{kin} + E_{pot}$ and specific heat C_V (and to entropy S and free energy A or G if those are calculated). The corrections to the energy and specific heat of a one-dimensional oscillator with frequency ν are: [6]

$$U^{QM} = U^{cl} + kT\left(\frac{1}{2}x - 1 + \frac{x}{e^x - 1}\right) \tag{1.3}$$

$$C_V^{QM} = C_V^{cl} + k\left(\frac{x^2e^x}{(e^x-1)^2} - 1\right), \tag{1.4}$$

where $x = h\nu/kT$. The classical oscillator absorbs too much energy (kT), while the high-frequency quantum oscillator is in its ground state at the zero-point energy level of $\frac{1}{2}h\nu$.
(b) We can treat the bonds (and bond angles) as *constraints* in the equation of motion. The rational behind this is that a quantum oscillator in its ground state resembles a constrained bond more closely than a classical oscillator. A good practical reason for this choice is that the algorithm can use larger time steps when the highest frequencies are removed. In practice the time step can be made four times as large when bonds are constrained than when they are oscillators [7]. GROMACS has this option for the bonds, and for the bond angles. The flexibility of the latter is rather essential to allow for the realistic motion and coverage of configurational space [7].

Electrons are in the ground state

In MD we use a *conservative* force field that is a function of the positions of atoms only. This means that the electronic motions are not considered: the electrons are supposed to adjust their dynamics infinitely fast when the atomic positions change (the *Born-Oppenheimer* approximation), and remain in their ground state. This is really all right, almost always. But of course, electron transfer processes and electronically excited states can not be treated. Neither can chemical reactions be treated properly, but there are other reasons to shy away from reactions for the time being.

Force fields are approximate

Force fields provide the forces. They are not really a part of the simulation method and their parameters can be user-modified as the need arises or knowledge improves. But the form of the forces that can be used in a particular program is subject to limitations. The force field that is incorporated in GROMACS is described in Chapter 4. In the present version the force field is pair-additive (apart from long-range coulomb forces), it cannot incorporate polarizabilities, and it does not contain fine-tuning of bonded interactions. This urges the inclusion of some limitations in this list below. For the rest it is quite useful and fairly reliable for bio macro-molecules in aqueous solution!

The force field is pair-additive

This means that all *non-bonded* forces result from the sum of non-bonded pair interactions. Non pair-additive interactions, the most important example of which is interaction through atomic polarizability, are represented by *effective pair potentials*. Only average non pair-additive contributions are incorporated. This also means that the pair interactions are not pure, i.e., they are not valid for isolated pairs or for situations that differ appreciably from the test systems on which the models were parameterized. In fact, the effective pair potentials are not that bad in practice. But the omission of polarizability also means that electrons in atoms do not provide a dielectric constant as they should. For example, real liquid alkanes have a dielectric constant of slightly more than 2, which reduce the long-range electrostatic interaction between (partial) charges. Thus the simulations will exaggerate the long-range Coulomb terms. Luckily, the next item compensates this effect a bit.

Long-range interactions are cut-off

In this version GROMACS always uses a cut-off radius for the Lennard-Jones interactions

---

`-n`	`clean.ndx`	Output, Opt.	Index file
`-q`	`clean.pdb`	Output, Opt.	Generic structure: gro g96 pdb

Other options

`-h`	bool	`no`	Print help info and quit
`-nice`	int	`0`	Set the nicelevel
`-merge`	bool	`no`	Merge multiple chains into one molecule
`-inter`	bool	`no`	Set the next 6 options to interactive
`-ss`	bool	`no`	Interactive SS bridge selection
`-ter`	bool	`no`	Interactive termini selection, iso charged
`-lys`	bool	`no`	Interactive Lysine selection, iso charged
`-asp`	bool	`no`	Interactive Aspartic Acid selection, iso charged
`-glu`	bool	`no`	Interactive Glutamic Acid selection, iso charged
`-his`	bool	`no`	Interactive Histidine selection, iso checking H-bonds
`-angle`	real	`135`	Minimum hydrogen-donor-acceptor angle for a H-bond (degrees)
`-dist`	real	`0.3`	Maximum donor-acceptor distance for a H-bond (nm)
`-una`	bool	`no`	Select aromatic rings with united CH atoms on Phenylalanine, Tryptophane and Tyrosine
`-sort`	bool	`yes`	Sort the residues according to database
`-H14`	bool	`no`	Use 1-4 interactions between hydrogen atoms
`-ignh`	bool	`no`	Ignore hydrogen atoms that are in the pdb file
`-alldih`	bool	`no`	Generate all proper dihedrals
`-dummy`	enum	`none`	Convert atoms to dummy atoms: `none`, `hydrogens` or `aromatics`
`-heavyh`	bool	`no`	Make hydrogen atoms heavy
`-deuterate`	bool	`no`	Change the mass of hydrogens to 2 amu

E.64 protonate

`protonate` reads (a) conformation(s) and adds all missing hydrogens as defined in `ffgmx2.hdb`. If only `-s` is specified, this conformation will be protonated, if also `-f` is specified, the conformation(s) will be read from this file which can be either a single conformation or a trajectory.

If a pdb file is supplied, residue names might not correspond to to the GROMACS naming conventions, in which case these residues will probably not be properly protonated.

If an index file is specified, please note that the atom numbers should correspond to the **protonated** state.

Files

`-s`	`topol.tpr`	Input	Structure+mass(db): tpr tpb tpa gro g96 pdb
`-f`	`traj.xtc`	Input, Opt.	Generic trajectory: xtc trr trj gro g96 pdb
`-n`	`index.ndx`	Input, Opt.	Index file
`-o`	`protonated.xtc`	Output	Generic trajectory: xtc trr trj gro g96 pdb

Other options

`-h`	bool	`no`	Print help info and quit
`-nice`	int	`0`	Set the nicelevel
`-b`	time	`-1`	First frame (ps) to read from trajectory
`-e`	time	`-1`	Last frame (ps) to read from trajectory
`-dt`	time	`-1`	Only use frame when t MOD dt = first time (ps)

E.63 pdb2gmx

This program reads a pdb file, lets you choose a forcefield, reads some database files, adds hydrogens to the molecules and generates coordinates in Gromacs (Gromos) format and a topology in Gromacs format. These files can subsequently be processed to generate a run input file.

Note that a pdb file is nothing more than a file format, and it need not necessarily contain a protein structure. Every kind of molecule for which there is support in the database can be converted. If there is no support in the database, you can add it yourself.

The program has limited intelligence, it reads a number of database files, that allow it to make special bonds (Cys-Cys, Heme-His, etc.), if necessary this can be done manually. The program can prompt the user to select which kind of LYS, ASP, GLU, CYS or HIS residue she wants. For LYS the choice is between LYS (two protons on NZ) or LYSH (three protons, default), for ASP and GLU unprotonated (default) or protonated, for HIS the proton can be either on ND1 (HISA), on NE2 (HISB) or on both (HISH). By default these selections are done automatically. For His, this is based on an optimal hydrogen bonding conformation. Hydrogen bonds are defined based on a simple geometric criterium, specified by the maximum hydrogen-donor-acceptor angle and donor-acceptor distance, which are set by `-angle` and `-dist` respectively.

Option `-merge` will ask if you want to merge consecutive chains into one molecule, this can be useful for connecting chains with a disulfide brigde.

pdb2gmx will also check the occupancy field of the pdb file. If any of the occupanccies are not one, indicating that the atom is not resolved well in the structure, a warning message is issued. When a pdb file does not originate from an X-Ray structure determination all occupancy fields may be zero. Either way, it is up to the user to verify the correctness of the input data (read the article!).

During processing the atoms will be reordered according to Gromacs conventions. With `-n` an index file can be generated that contains one group reordered in the same way. This allows you to convert a Gromos trajectory and coordinate file to Gromos. There is one limitation: reordering is done after the hydrogens are stripped from the input and before new hydrogens are added. This means that you should not use `-ignh`.

The `.gro` and `.g96` file formats do not support chain identifiers. Therefore it is useful to enter a pdb file name at the `-o` option when you want to convert a multichain pdb file.

`-sort` will sort all residues according to the order in the database, sometimes this is necessary to get charge groups together.

`-alldih` will generate all proper dihedrals instead of only those with as few hydrogens as possible, this is useful for use with the Charmm forcefield.

The option `-dummy` removes hydrogen and fast improper dihedral motions. Angular and out-of-plane motions can be removed by changing hydrogens into dummy atoms and fixing angles, which fixes their position relative to neighboring atoms. Additionally, all atoms in the aromatic rings of the standard amino acids (i.e. PHE, TRP, TYR and HIS) can be converted into dummy atoms, elminating the fast improper dihedral fluctuations in these rings. Note that in this case all other hydrogen atoms are also converted to dummy atoms. The mass of all atoms that are converted into dummy atoms, is added to the heavy atoms.

Also slowing down of dihedral motion can be done with `-heavyh` done by increasing the hydrogen-mass by a factor of 4. This is also done for water hydrogens to slow down the rotational motion of water. The increase in mass of the hydrogens is subtracted from the bonded (heavy) atom so that the total mass of the system remains the same. Reference Feenstra et al., J. Comput. Chem. 20, 786 (1999).

Files

`-f`	`eiwit.pdb`	Input	Generic structure: gro g96 pdb tpr tpb tpa
`-o`	`conf.gro`	Output	Generic structure: gro g96 pdb
`-p`	`topol.top`	Output	Topology file
`-i`	`posre.itp`	Output	Include file for topology

---

and sometimes also for Coulomb. Due to the minimum-image convention (only one image of each particle in the periodic boundary conditions is considered for a pair interaction), the cut-off range can not exceed half the box size. That is still pretty big for large systems, and trouble is only expected for systems containing charged particles. But then real bad things may happen, like accumulation of charges at the cut-off boundary or very wrong energies! For such systems you should consider using one of the implemented long-range electrostatic algorithms.

Boundary conditions are unnatural

Since system size is small (even 10,000 particles is small), a cluster of particles will have a lot of unwanted boundary with its environment (vacuum). This we must avoid if we wish to simulate a bulk system. So we use periodic boundary conditions, to avoid real phase boundaries. But liquids are not crystals, so something unnatural remains. This item is mentioned in the last place because it is the least evil of all. For large systems the errors are small, but for small systems with a lot of internal spatial correlation, the periodic boundaries may enhance internal correlation. In that case, beware and test the influence of system size. This is especially important when using lattice sums for long-range electrostatics, since these are known to sometimes introduce extra ordering.

1.3 Energy Minimization and Search Methods

As mentioned in sec. 1.1, in many cases energy minimization is required. GROMACS provides a simple form of local energy minimization, the *steepest descent* method.

The potential energy function of a (macro)molecular system is a very complex landscape (or *hyper surface*) in a large number of dimensions. It has one deepest point, the *global minimum* and a very large number of *local minima*, where all derivatives of the potential energy function with respect to the coordinates are zero and all second derivatives are nonnegative. The matrix of second derivatives, which is called the *Hessian matrix*, has nonnegative eigenvalues; only the collective coordinates that correspond to translation and rotation (for an isolated molecule) have zero eigenvalues. In between the local minima there are *saddle points*, where the Hessian matrix has only one negative eigenvalue. These points are the mountain passes through which the system can migrate from one local minimum to another.

Knowledge of all local minima, including the global one, and of all saddle points would enable us to describe the relevant structures and conformations and their free energies, as well as the dynamics of structural transitions. Unfortunately, the dimensionality of the configurational space and the number of local minima is so high that it is impossible to sample the space at a sufficient number of points to obtain a complete survey. In particular, no minimization method exists that guarantees the determination of the global minimum. However, given a starting configuration, it is possible to find the *nearest local minimum*. Nearest in this context does not always imply nearest in a geometrical sense (i.e., the least sum of square coordinate differences), but means the minimum that can be reached by systematically moving down the steepest local gradient. Finding this nearest local minimum is all that GROMACS can do for you, sorry! If you want to find other minima and hope to discover the global minimum in the process, the best advice is to experiment with temperature-coupled MD: run your system at a high temperature for a while and then quench it slowly down to the required temperature; do this repeatedly! If something as a melting or glass transition temperature exists, it is wise to stay for some time slightly below that temperature and cool down slowly according to some clever scheme, a process called *simulated annealing*. Since no physical truth is required, you can use your phantasy to speed up this process. One trick that often works is to make hydrogen atoms heavier (mass 10 or so): although that will slow down the otherwise very rapid motions of hydrogen atoms, it will hardly influence the slower motions in the system while enabling you to increase the time step by a factor of 3 or 4. You can also modify the potential energy function during the search procedure, e.g. by removing barriers (remove dihedral angle functions or replace repulsive potentials by *soft core* potentials [8]), but always take care to restore the correct functions slowly. The best search method that allows rather drastic structural changes is to allow excursions into four-dimensional space [9], but this requires some extra programming beyond the standard capabilities of GROMACS.

Three possible energy minimization methods are:

- Those that require only function evaluations. Examples are the simplex method and its variants. A step is made on the basis of the results of previous evaluations. If derivative information is available, such methods are inferior to those that use this information.
- Those that use derivative information. Since the partial derivatives of the potential energy with respect to all coordinates are known in MD programs (these are equal to minus the forces) this class of methods is very suitable as modification of MD programs.
- Those that use second derivative information as well. These methods are superior in their convergence properties near the minimum: a quadratic potential function is minimized in one step! The problem is that for N particles a $3N \times 3N$ matrix must be computed, stored and inverted. Apart from the extra programming to obtain second derivatives, for most systems of interest this is beyond the available capacity. There are intermediate methods building up the Hessian matrix on the fly, but they also suffer from excessive storage requirements. So GROMACS will shy away from this class of methods.

The *steepest descent* method, available in GROMACS, is of the second class. It simply takes a step in the direction of the negative gradient (hence in the direction of the force), without any consideration of the history built up in previous steps. The step size is adjusted such that the search is fast but the motion is always downhill. This is a simple and sturdy, but somewhat stupid, method: its convergence can be quite slow, especially in the vicinity of the local minimum! The faster converging *conjugate gradient method* (see e.g. [10]) uses gradient information from previous steps. In general, steepest descents will bring you close to the nearest local minimum very quickly, while conjugate gradients brings you *very* close to the local minimum, but performs worse far away from the minimum.

E.61 nmrun

nmrun builds a Hessian matrix from single conformation. For usual Normal Modes-like calculations, make sure that the structure provided is properly energy-minimised. The generated matrix can be diagonalized by g_nmeig.

Files

`-s`	`topol.tpr`	Input	Generic run input: tpr tpb tpa
`-m`	`hessian.mtx`	Output	Hessian matrix
`-g`	`nm.log`	Output	Log file

Other options

`-h`	bool	`no`	Print help info and quit
`-nice`	int	`19`	Set the nicelevel
`-np`	int	`1`	Number of nodes, must be the same as used for grompp
`-v`	bool	`no`	Verbose mode
`-compact`	bool	`yes`	Write a compact log file

E.62 options

All GROMACS programs have 6 standard options, of which some are hidden by default:

Other options

`-h`	bool	`no`	Print help info and quit
`-nice`	int	`0`	Set the nicelevel

- Optional files are not used unless the option is set, in contrast to non optional files, where the default file name is used when the option is not set.
- All GROMACS programs will accept file options without a file extension or filename being specified. In such cases the default filenames will be used. With multiple input file types, such as generic structure format, the directory will be searched for files of each type with the supplied or default name. When no such file is found, or with output files the first file type will be used.
- All GROMACS programs with the exception of `mdrun`, `nmrun` and `eneconv` check if the command line options are valid. If this is not the case, the program will be halted.
- Enumerated options (enum) should be used with one of the arguments listed in the option description, the argument may be abbreviated. The first match to the shortest argument in the list will be selected.
- Vector options can be used with 1 or 3 parameters. When only one parameter is supplied the two others are also set to this value.
- For many GROMACS programs, the time options can be supplied in different time units, depending on the setting of the `-tu` option.
- All GROMACS programs can read compressed or g-zipped files. There might be a problem with reading compressed `.xtc`, `.trr` and `.trj` files, but these will not compress very well anyway.
- Most GROMACS programs can process a trajectory with less atoms than the run input or structure file, but only if the trajectory consists of the first n atoms of the run input or structure file.
- Many GROMACS programs will accept the `-tu` option to set the time units to use in output files (e.g. for `xmgr` graphs or `xpm` matrices) and in all time options.

`-nice`	int	`19`	Set the nicelevel
`-deffnm`	string		Set the default filename for all file options
`-np`	int	`1`	Number of nodes, must be the same as used for grompp
`-v`	bool	`no`	Be loud and noisy
`-compact`	bool	`yes`	Write a compact log file

E.59 mk_angndx

mk_angndx makes an index file for calculation of angle distributions etc. It uses a run input file (`.tpx`) for the definitions of the angles, dihedrals etc.

Files

`-s`	`topol.tpr`	Input	Generic run input: tpr tpb tpa
`-n`	`angle.ndx`	Output	Index file

Other options

`-h`	bool	`no`	Print help info and quit
`-nice`	int	`0`	Set the nicelevel
`-type`	enum	`angle`	Type of angle: `angle`, `g96-angle`, `dihedral`, `improper`, `ryckaert-bellemans` or `phi-psi`

E.60 ngmx

ngmx is the Gromacs trajectory viewer. This program reads a trajectory file, a run input file and an index file and plots a 3D structure of your molecule on your standard X Window screen. No need for a high end graphics workstation, it even works on Monochrome screens.

The following features have been implemented: 3D view, rotation, translation and scaling of your molecule(s), labels on atoms, animation of trajectories, hardcopy in PostScript format, user defined atom-filters runs on MIT-X (real X), open windows and motif, user friendly menus, option to remove periodicity, option to show computational box.

Some of the more common X command line options can be used:
-bg, -fg change colors, -font fontname, changes the font.

Files

`-f`	`traj.xtc`	Input	Generic trajectory: xtc trr trj gro g96 pdb
`-s`	`topol.tpr`	Input	Generic run input: tpr tpb tpa
`-n`	`index.ndx`	Input, Opt.	Index file

Other options

`-h`	bool	`no`	Print help info and quit
`-nice`	int	`0`	Set the nicelevel
`-b`	time	`-1`	First frame (ps) to read from trajectory
`-e`	time	`-1`	Last frame (ps) to read from trajectory
`-dt`	time	`-1`	Only use frame when t MOD dt = first time (ps)

- Balls option does not work
- Some times dumps core without a good reason

Chapter 2

Definitions and Units

2.1 Notation

The following conventions for mathematical typesetting are used throughout this document:

Item	Notation	Example
Vector	Bold italic	$\boldsymbol{r}_i$
Vector Length	Italic	r_i

We define the *lowercase* subscripts i, j, k and l to denote particles: $\boldsymbol{r}_i$ is the *position vector* of particle i, and using this notation:

$$\boldsymbol{r}_{ij} = \boldsymbol{r}_j - \boldsymbol{r}_i \tag{2.1}$$

$$r_{ij} = |\boldsymbol{r}_{ij}| \tag{2.2}$$

The force on particle i is denoted by $\boldsymbol{F}_i$ and

$$\boldsymbol{F}_{ij} = \text{force on } i \text{ exerted by } j \tag{2.3}$$

Please note that we changed notation as of ver. 2.0 to $\boldsymbol{r}_{ij} = \boldsymbol{r}_j - \boldsymbol{r}_i$ since this is the notation commonly used. If you encounter an error, let us know.

2.2 MD units

GROMACS uses a consistent set of units that produce values in the vicinity of unity for most relevant molecular quantities. Let us call them *MD units*. The basic units in this system are nm, ps, K, electron charge (e) and atomic mass unit (u), see Table 2.1.

Consistent with these units are a set of derived units, given in Table 2.2.

The **electric conversion factor** $f = \frac{1}{4\pi\varepsilon_o} = 138.935\,485(9)$ kJ mol^{-1} nm e^{-2}. It relates the mechanical quantities to the electrical quantities as in

$$V = f\frac{q^2}{r} \text{ or } F = f\frac{q^2}{r^2} \tag{2.4}$$

Quantity	Symbol	Unit
length	r	nm = 10^{-9} m
mass	m	u (atomic mass unit) = 1.6605402(10)×10^{-27} kg (1/12 of the mass of a C atom) $1.6605402(10) \times 10^{-27}$ kg
time	t	ps = 10^{-12} s
charge	q	e = electronic charge = $1.60217733(49) \times 10^{-19}$ C
temperature	T	K

Table 2.1: Basic units used in GROMACS. Numbers in parentheses give accuracy.

Quantity	Symbol	Unit
energy	E, V	kJ mol^{-1}
Force	$\boldsymbol{F}$	kJ mol^{-1} nm^{-1}
pressure	p	kJ mol^{-1} nm^{-3} = $10^{30}/N_{AV}$ Pa $1.660\,54 \times 10^6$ Pa = 16.6054 Bar
velocity	v	nm ps^{-1} = 1000 m/s
dipole moment	μ	e nm
electric potential	Φ	kJ mol^{-1} e^{-1} = $0.010\,364\,272(3)$ Volt
electric field	E	kJ mol^{-1} nm^{-1} e^{-1} = $1.036\,427\,2(3) \times 10^7$ V/m

Table 2.2: Derived units

Electric potentials Φ and electric fields $\boldsymbol{E}$ are intermediate quantities in the calculation of energies and forces. They do not occur inside GROMACS. If they are used in evaluations, there is a choice of equations and related units. We recommend strongly to follow the usual practice to include the factor f in expressions that evaluate Φ and $\boldsymbol{E}$:

$$\Phi(\boldsymbol{r}) = f \sum_j \frac{q_j}{|\boldsymbol{r} - \boldsymbol{r}_j|} \tag{2.5}$$

$$\boldsymbol{E}(\boldsymbol{r}) = f \sum_j q_j \frac{(\boldsymbol{r} - \boldsymbol{r}_j)}{|\boldsymbol{r} - \boldsymbol{r}_j|^3} \tag{2.6}$$

With these definitions $q\Phi$ is an energy and $q\boldsymbol{E}$ is a force. The units are those given in Table 2.2: about 10 mV for potential. Thus the potential of an electronic charge at a distance of 1 nm equals $f \approx 140$ units ≈ 1.4 V. (exact value: 1.439965 V)

Note that these units are mutually consistent; changing any of the units is likely to produce inconsistencies and is therefore *strongly discouraged*! In particular: if Å are used instead of nm, the unit of time changes to 0.1 ps. If the kcal/mol (= 4.184 kJ/mol) is used instead of kJ/mol for energy, the unit of time becomes 0.488882 ps and the unit of temperature changes to 4.184 K. But in both cases all electrical energies go wrong, because they will still be computed in kJ/mol, expecting nm as the unit of length. Although careful rescaling of charges may still yield consistency, it is clear that such confusions must be rigidly avoided.

In terms of the MD units the usual physical constants take on different values, see Table 2.3. All quantities are per mol rather than per molecule. There is no distinction between Boltzmann's constant k and the gas constant R: their value is $0.008\,314\,51$ kJ mol^{-1} K^{-1}.

E.58 mdrun

The mdrun program performs Molecular Dynamics simulations. It reads the run input file (`-s`) and distributes the topology over nodes if needed. The coordinates are passed around, so that computations can begin. First a neighborlist is made, then the forces are computed. The forces are globally summed, and the velocities and positions are updated. If necessary shake is performed to constrain bond lengths and/or bond angles. Temperature and Pressure can be controlled using weak coupling to a bath.

mdrun produces at least three output file, plus one log file (`-g`) per node. The trajectory file (`-o`), contains coordinates, velocities and optionally forces. The structure file (`-c`) contains the coordinates and velocities of the last step. The energy file (`-e`) contains energies, the temperature, pressure, etc, a lot of these things are also printed in the log file of node 0. Optionally coordinates can be written to a compressed trajectory file (`-x`).

When running in parallel with PVM or an old version of MPI the `-np` option must be given to indicate the number of nodes.

The option `-dgdl` is only used when free energy perturbation is turned on.

With `-rerun` an input trajectory can be given for which forces and energies will be (re)calculated. Neighbor searching will be performed for every frame, unless `nstlist` is zero (see the `.mdp` file).

ED (essential dynamics) sampling is switched on by using the `-ei` flag followed by an `.edi` file. The `.edi` file can be produced using options in the essdyn menu of the WHAT IF program. mdrun produces a `.edo` file that contains projections of positions, velocities and forces onto selected eigenvectors.

The -table option can be used to pass mdrun a formatted table with user-defined potential functions. The file is read from either the current directory or from the GMXLIB directory. A number of preformatted tables are presented in the GMXLIB dir, for 6-8, 6-9, 6-10, 6-11, 6-12 Lennard Jones potentials with normal Coulomb.

The options `-pi`, `-po`, `-pd`, `-pn` are used for potential of mean force calculations and umbrella sampling. See manual.

When mdrun receives a TERM signal, it will set nsteps to the current step plus one. When mdrun receives a USR1 signal, it will set nsteps to the next multiple of nstxout after the current step. In both cases all the usual output will be written to file. When running with MPI, a signal to one of the mdrun processes is sufficient, this signal should not be sent to mpirun or the mdrun process that is the parent of the others.

Files

`-s`	`topol.tpr`	Input	Generic run input: tpr tpb tpa
`-o`	`traj.trr`	Output	Full precision trajectory: trr trj
`-x`	`traj.xtc`	Output, Opt.	Compressed trajectory (portable xdr format)
`-c`	`confout.gro`	Output	Generic structure: gro g96 pdb
`-e`	`ener.edr`	Output	Generic energy: edr ene
`-g`	`md.log`	Output	Log file
`-dgdl`	`dgdl.xvg`	Output, Opt.	xvgr/xmgr file
`-table`	`table.xvg`	Input, Opt.	xvgr/xmgr file
`-rerun`	`rerun.xtc`	Input, Opt.	Generic trajectory: xtc trr trj gro g96 pdb
`-ei`	`sam.edi`	Input, Opt.	ED sampling input
`-eo`	`sam.edo`	Output, Opt.	ED sampling output
`-pi`	`pull.ppa`	Input, Opt.	Pull parameters
`-po`	`pullout.ppa`	Output, Opt.	Pull parameters
`-pd`	`pull.pdo`	Output, Opt.	Pull data output
`-pn`	`pull.ndx`	Input, Opt.	Index file

Other options

`-h`	bool	`no`	Print help info and quit

`-time`	real	`-1`	Take frame at or first after this time.
`-np`	int	`1`	Generate statusfile for # nodes
`-shuffle`	bool	`no`	Shuffle molecules over nodes
`-sort`	bool	`no`	Sort molecules according to X coordinate
`-rmdumbds`	bool	`yes`	Remove constant bonded interactions with dummies
`-load`	string		Releative load capacity of each node on a parallel machine. Be sure to use quotes around the string, which should contain a number for each node
`-maxwarn`	int	`10`	Number of warnings after which input processing stops
`-check14`	bool	`no`	Remove 1-4 interactions without Van der Waals

E.56 highway

highway is the gromacs highway simulator. It is an X-windows gadget that shows a (periodic) autobahn with a user defined number of cars. Fog can be turned on or off to increase the number of crashes. Nice for a background CPU-eater

Files

`-f`	`highway.dat`	Input	Generic data file
`-a`	`auto.dat`	Input	Generic data file

Other options

`-h`	bool	`no`	Print help info and quit
`-nice`	int	`0`	Set the nicelevel
`-b`	time	`-1`	First frame (ps) to read from trajectory
`-e`	time	`-1`	Last frame (ps) to read from trajectory
`-dt`	time	`-1`	Only use frame when t MOD dt = first time (ps)

E.57 make_ndx

Index groups are necessary for almost every gromacs program. All these programs can generate default index groups. You ONLY have to use make_ndx when you need SPECIAL index groups. There is a default index group for the whole system, 9 default index groups are generated for proteins, a default index group is generated for every other residue name.

When no index file is supplied, also make_ndx will generate the default groups. With the index editor you can select on atom, residue and chain names and numbers, you can use NOT, AND and OR, you can split groups into chains, residues or atoms. You can delete and rename groups.

The atom numbering in the editor and the index file starts at 1.

Files

`-f`	`conf.gro`	Input	Generic structure: gro g96 pdb tpr tpb tpa
`-n`	`index.ndx`	Input, Opt.	Index file
`-o`	`index.ndx`	Output	Index file

Other options

`-h`	bool	`no`	Print help info and quit
`-nice`	int	`0`	Set the nicelevel

Symbol	Name	Value
N_{AV}	Avogadro's number	$6.022\,136\,7(36) \times 10^{23}$ mol^{-1}
R	gas constant	$8.314\,510(70) \times 10^{-3}$ kJ mol^{-1} K^{-1}
k_B	Boltzmann's constant	idem
h	Planck's constant	$0.399\,031\,32(24)$ kJ mol^{-1} ps
$\hbar$	Dirac's constant	$0.063\,507\,807(38)$ kJ mol^{-1} ps
c	velocity of light	$299\,792.458$ nm/ps

Table 2.3: Some Physical Constants

Quantity	Symbol	Relation to SI
Length	r*	r σ^{-1}
Mass	m*	m M^{-1}
Time	t*	t $\sigma^{-1}\sqrt{\epsilon/M}$
Temperature	T*	k$_B$T ϵ^{-1}
Energy	E*	E ϵ^{-1}
Force	F*	F σ ϵ^{-1}
Pressure	P*	P $\sigma^3\epsilon^{-1}$
Velocity	v*	v $\sqrt{M/\epsilon}$
Density	ρ*	N σ^3 V^{-1}

Table 2.4: Reduced Lennard-Jones quantities

2.3 Reduced units

When simulating Lennard-Jones (LJ) systems it might be advantageous to use reduced units (*i.e.*, setting $\epsilon_{ii} = \sigma_{ii} = m_i = k_B = 1$ for one type of atoms). This is possible. When specifying the input in reduced units, the output will also be in reduced units. There is one exception: the *temperature*, which is expressed in $0.008\,314\,51$ reduced units. This is a consequence of the use of Boltzmann's constant in the evaluation of temperature in the code. Thus not T, but k_BT is the reduced temperature. A GROMACS temperature $T = 1$ means a reduced temperature of 0.008…units; if a reduced temperature of 1 is required, the GROMACS temperature should be 120.2717.

In Table 2.4 quantities are given for LJ potentials:

$$V_{LJ} = 4\epsilon\left[\left(\frac{\sigma}{r}\right)^{12} - \left(\frac{\sigma}{r}\right)^{6}\right] \tag{2.7}$$

grompp calls the c-preprocessor to resolve includes, macros etcetera. To specify a macro-preprocessor other than /lib/cpp (such as m4) you can put a line in your parameter file specifying the path to that cpp. Specifying `-pp` will get the pre-processed topology file written out.

If your system does not have a c-preprocessor, you can still use grompp, but you do not have access to the features from the cpp. Command line options to the c-preprocessor can be given in the `.mdp` file. See your local manual (man cpp).

When using position restraints a file with restraint coordinates can be supplied with `-r`, otherwise constraining will be done relative to the conformation from the `-c` option.

Starting coordinates can be read from trajectory with `-t`. The last frame with coordinates and velocities will be read, unless the `-time` option is used. Note that these velocities will not be used when `gen_vel = yes` in your `.mdp` file. If you want to continue a crashed run, it is easier to use `tpbconv`.

When preparing an input file for parallel `mdrun` it may be advantageous to partition the simulation system over the nodes in a way in which each node has a similar amount of work. The -shuffle option does just that. For a single protein in water this does not make a difference, however for a system where you have many copies of different molecules (e.g. liquid mixture or membrane/water system) the option is definitely a must.

A further optimization for parallel systems is the `-sort` option which sorts molecules according to coordinates. This must always be used in conjunction with `-shuffle`, however sorting also works when you have only one molecule type.

Using the `-morse` option grompp can convert the harmonic bonds in your topology to morse potentials. This makes it possible to break bonds. For this option to work you need an extra file in your $GMXLIB with dissociation energy. Use the -debug option to get more information on the workings of this option (look for MORSE in the grompp.log file using less or something like that).

By default all bonded interactions which have constant energy due to dummy atom constructions will be removed. If this constant energy is not zero, this will result in a shift in the total energy. All bonded interactions can be kept by turning off `-rmdumbds`. Additionally, all constraints for distances which will be constant anyway because of dummy atom constructions will be removed. If any constraints remain which involve dummy atoms, a fatal error will result.

To verify your run input file, please make notice of all warnings on the screen, and correct where necessary. Do also look at the contents of the `mdout.mdp` file, this contains comment lines, as well as the input that `grompp` has read. If in doubt you can start grompp with the `-debug` option which will give you more information in a file called grompp.log (along with real debug info). Finally, you can see the contents of the run input file with the `gmxdump` program.

Files

`-f`	`grompp.mdp`	Input	grompp input file with MD parameters
`-po`	`mdout.mdp`	Output	grompp input file with MD parameters
`-c`	`conf.gro`	Input	Generic structure: gro g96 pdb tpr tpb tpa
`-r`	`conf.gro`	Input, Opt.	Generic structure: gro g96 pdb tpr tpb tpa
`-n`	`index.ndx`	Input, Opt.	Index file
`-deshuf`	`deshuf.ndx`	Output, Opt.	Index file
`-p`	`topol.top`	Input	Topology file
`-pp`	`processed.top`	Output, Opt.	Topology file
`-o`	`topol.tpr`	Output	Generic run input: tpr tpb tpa
`-t`	`traj.trr`	Input, Opt.	Full precision trajectory: trr trj

Other options

`-h`	bool	`no`	Print help info and quit
`-nice`	int	`0`	Set the nicelevel
`-v`	bool	`yes`	Be loud and noisy

Files

`-f`	`traj.xtc`	Input, Opt.	Generic trajectory: xtc trr trj gro g96 pdb
`-f2`	`traj.xtc`	Input, Opt.	Generic trajectory: xtc trr trj gro g96 pdb
`-s1`	`top1.tpr`	Input, Opt.	Generic run input: tpr tpb tpa
`-s2`	`top2.tpr`	Input, Opt.	Generic run input: tpr tpb tpa
`-c`	`topol.tpr`	Input, Opt.	Structure+mass(db): tpr tpb tpa gro g96 pdb
`-e`	`ener.edr`	Input, Opt.	Generic energy: edr ene
`-e2`	`ener2.edr`	Input, Opt.	Generic energy: edr ene

Other options

`-h`	bool	`no`	Print help info and quit
`-nice`	int	`0`	Set the nicelevel
`-vdwfac`	real	`0.8`	Fraction of sum of VdW radii used as warning cutoff
`-bonlo`	real	`0.4`	Min. fract. of sum of VdW radii for bonded atoms
`-bonhi`	real	`0.7`	Max. fract. of sum of VdW radii for bonded atoms
`-tol`	real	`0`	Tolerance for comparing real values

E.54 gmxdump

gmxdump reads a run input file (`.tpa`/`.tpr`/`.tpb`), a trajectory (`.trj`/`.trr`/`.xtc`) or an energy file (`.ene`/`.edr`) and prints that to standard output in a readable format. This program is essential for checking your run input file in case of problems.

Files

`-s`	`topol.tpr`	Input, Opt.	Generic run input: tpr tpb tpa
`-f`	`traj.xtc`	Input, Opt.	Generic trajectory: xtc trr trj gro g96 pdb
`-e`	`ener.edr`	Input, Opt.	Generic energy: edr ene

Other options

`-h`	bool	`no`	Print help info and quit
`-nice`	int	`0`	Set the nicelevel
`-nr`	bool	`yes`	Show index numbers in output (leaving them out makes comparison easier, but creates a useless topology)

E.55 grompp

The gromacs preprocessor reads a molecular topology file, checks the validity of the file, expands the topology from a molecular description to an atomic description. The topology file contains information about molecule types and the number of molecules, the preprocessor copies each molecule as needed. There is no limitation on the number of molecule types. Bonds and bond-angles can be converted into constraints, separately for hydrogens and heavy atoms. Then a coordinate file is read and velocities can be generated from a Maxwellian distribution if requested. grompp also reads parameters for the mdrun (eg. number of MD steps, time step, cut-off), and others such as NEMD parameters, which are corrected so that the net acceleration is zero. Eventually a binary file is produced that can serve as the sole input file for the MD program.

grompp uses the atom names from the topology file. The atom names in the coordinate file (option `-c`) are only read to generate warnings when they do not match the atom names in the topology. Note that the atom names are irrelevant for the simulation as only the atom types are used for generating interaction parameters.

Chapter 3

Algorithms

3.1 Introduction

In this chapter we first give describe two general concepts used in GROMACS: *periodic boundary conditions* (sec. 3.2) and the *group concept* (sec. 3.3). The MD algorithm is described in sec. 3.4: first a global form of the algorithm is given, which is refined in subsequent subsections. The (simple) EM (Energy Minimization) algorithm is described in sec. 3.10. Some other algorithms for special purpose dynamics are described after this. In the final sec. 3.14 of this chapter a few principles are given on which parallelization of GROMACS is based. The parallelization is hardly visible for the user and is therefore not treated in detail.

A few issues are of general interest. In all cases the *system* must be defined, consisting of molecules. Molecules again consist of particles with defined interaction functions. The detailed description of the *topology* of the molecules and of the *force field* and the calculation of forces is given in chapter 4. In the present chapter we describe other aspects of the algorithm, such as pair list generation, update of velocities and positions, coupling to external temperature and pressure, conservation of constraints. The *analysis* of the data generated by an MD simulation is treated in chapter 8.

3.2 Periodic boundary conditions

The classical way to minimize edge effects in a finite system is to apply *periodic boundary conditions*. The atoms of the system to be simulated are put into a space-filling box, which is surrounded by translated copies of itself (Fig. 3.1). Thus there are no boundaries of the system; the artifact caused by unwanted boundaries in an isolated cluster is now replaced by the artifact of periodic conditions. If a crystal is simulated, such boundary conditions are desired (although motions are naturally restricted to periodic motions with wavelengths fitting into the box). If one wishes to simulate non-periodic systems, as liquids or solutions, the periodicity by itself causes errors. The errors can be evaluated by comparing various system sizes; they are expected to be less severe than the errors resulting from an unnatural boundary with vacuum.

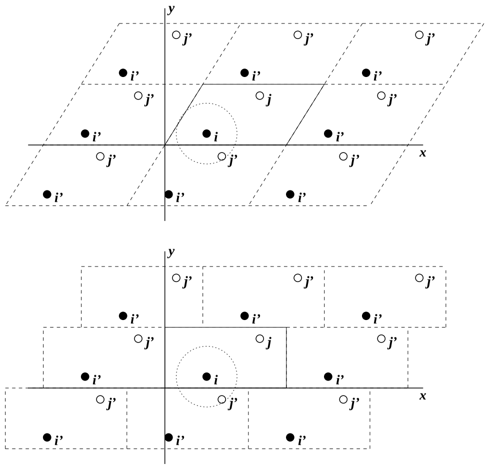


Figure 3.1: Periodic boundary conditions in two dimensions.

There are several possible shapes for space-filling unit cells. Some, as the *rhombic dodecahedron* and the *truncated octahedron* [11] approach a spherical shape better than a cubic box and are therefore more economical for studying an (approximately spherical) macromolecule in solution, since less solvent molecules are required to fill the box given a minimum distance between macromolecular images. However, a periodic system based on the rhombic dodecahedron or truncated octahedron is equivalent to a periodic system based on a *triclinic* unit cell. The latter shape is the most general space-filling unit cell; it comprises all possible space-filling shapes [12]. Therefore GROMACS is based on the triclinic unit cell.

GROMACS uses periodic boundary conditions, combined with the *minimum image convention:* only one - the nearest - image of each particle is considered for short-range non-bonded interaction terms. For long-range electrostatic interactions this is not always accurate enough, and GROMACS therefore also incorporates lattice sum methods like Ewald Sum, PME and PPPM.

Gromacs supports triclinic boxes of any shape. The box is defined by the 3 box vectors $\mathbf{a}$,$\mathbf{b}$ and $\mathbf{c}$. The box vectors must satisfy the following conditions:

$$a_y = a_z = b_z = 0 \tag{3.1}$$

$$a_x > 0, \quad b_y > 0, \quad c_z > 0 \tag{3.2}$$

$$|b_x| \leq \frac{1}{2}\,a_x, \quad |c_x| \leq \frac{1}{2}\,a_x, \quad |c_y| \leq \frac{1}{2}\,b_y \tag{3.3}$$

Equations (3.1) can always be statisfied by rotating the box. Equations (3.2) and (3.3) can always be statisfied by adding and subtracting box vectors.

Even when simulating using a triclinic box, GROMACS always puts the particles in a brick shaped volume, for efficiency reasons. This is illustrated in Fig. 3.1 for a 2-dimensional system. So from

---

Files

`-s`	`topol.tpr`	Input	Generic run input: tpr tpb tpa
`-n`	`index.ndx`	Input, Opt.	Index file
`-o`	`out.gro`	Output	Generic structure: gro g96 pdb
`-g`	`genion.log`	Output	Log file
`-pot`	`pot.pdb`	Output, Opt.	Protein data bank file

Other options

`-h`	bool	`no`	Print help info and quit
`-nice`	int	`19`	Set the nicelevel
`-np`	int	`0`	Number of positive ions
`-pname`	string	`Na`	Name of the positive ion
`-pq`	real	`1`	Charge of the positive ion
`-nn`	int	`0`	Number of negative ions
`-nname`	string	`Cl`	Name of the negative ion
`-nq`	real	`-1`	Charge of the negative ion
`-rmin`	real	`0.6`	Minimum distance between ions
`-random`	bool	`no`	Use random placement of ions instead of based on potential. The rmin option should still work
`-seed`	int	`1993`	Seed for random number generator

E.52 genpr

genpr produces an include file for a topology containing a list of atom numbers and three force constants for the X, Y and Z direction. A single isotropic force constant may be given on the command line instead of three components.

WARNING: genpr only works for the first molecule. Position restraints are interactions within molecules, therefore they should be included within the correct `[ moleculetype ]` block in the topology. Since the atom numbers in every moleculetype in the topology start at 1 and the numbers in the input file for genpr number consecutively from 1, genpr will only produce a useful file for the first molecule.

Files

`-f`	`conf.gro`	Input	Generic structure: gro g96 pdb tpr tpb tpa
`-n`	`index.ndx`	Input, Opt.	Index file
`-o`	`posre.itp`	Output	Include file for topology

Other options

`-h`	bool	`no`	Print help info and quit
`-nice`	int	`0`	Set the nicelevel
`-fc`	vector		
		`1000 1000 1000`	force constants (kJ mol-1 nm-2)

E.53 gmxcheck

gmxcheck reads a trajectory (`.trj`, `.trr` or `.xtc`) or an energy file (`.ene` or `.edr`) and prints out useful information about them.

Option `-c` checks for presence of coordinates, velocities and box in the file, for close contacts (smaller than `-vdwfac` and not bonded, i.e. not between `-bonlo` and `-bonhi`, all relative to the sum of both Van der Waals radii) and atoms outside the box (these may occur often and are no problem). If velocities are present, an estimated temperature will be calculated from them.

The program will compare run input (`.tpr`, `.tpb` or `.tpa`) files when both `-s1` and `-s2` are supplied.

- Molecules must be whole in the initial configurations.
- At the moment -ci only works when inserting one molecule.

E.50 genconf

genconf multiplies a given coordinate file by simply stacking them on top of each other, like a small child playing with wooden blocks. The program makes a grid of *user defined* proportions (`-nbox`), and interspaces the grid point with an extra space `-dist`.

When option `-rot` is used the program does not check for overlap between molecules on grid points. It is recommended to make the box in the input file at least as big as the coordinates + Van der Waals radius.

If the optional trajectory file is given, conformations are not generated, but read from this file and translated appropriately to build the grid.

Files

`-f`	`conf.gro`	Input	Generic structure: gro g96 pdb tpr tpb tpa
`-o`	`out.gro`	Output	Generic structure: gro g96 pdb
`-trj`	`traj.xtc`	Input, Opt.	Generic trajectory: xtc trr trj gro g96 pdb

Other options

`-h`	bool	`no`	Print help info and quit
`-nice`	int	`0`	Set the nicelevel
`-nbox`	vector	`1 1 1`	Number of boxes
`-dist`	vector	`0 0 0`	Distance between boxes
`-seed`	int	`0`	Random generator seed, if 0 generated from the time
`-rot`	bool	`no`	Randomly rotate conformations
`-shuffle`	bool	`no`	Random shuffling of molecules
`-sort`	bool	`no`	Sort molecules on X coord
`-block`	int	`1`	Divide the box in blocks on this number of cpus
`-nmolat`	int	`3`	Number of atoms per molecule, assumed to start from 0. If you set this wrong, it will screw up your system!
`-maxrot`	vector	`90 90 90`	Maximum random rotation
`-renumber`	bool	`no`	Renumber residues

- The program should allow for random displacement off lattice points.

E.51 genion

genion replaces solvent molecules by monoatomic ions at the position of the first atoms with the most favorable electrostatic potential or at random. The potential is calculated on all atoms, using normal GROMACS particle based methods (in contrast to other methods based on solving the Poisson-Boltzmann equation). The potential is recalculated after every ion insertion. If specified in the run input file, a reaction field or shift function can be used. The group of solvent molecules should be continuous and all molecules should have the same number of atoms. The user should add the ion molecules to the topology file and include the file `ions.itp`. Ion names for Gromos96 should include the charge.

The potential can be written as B-factors in a pdb file (for visualisation using e.g. rasmol). The unit of the potential is 0.001 kJ/(mol e).

For larger ions, e.g. sulfate we recommended to use genbox.

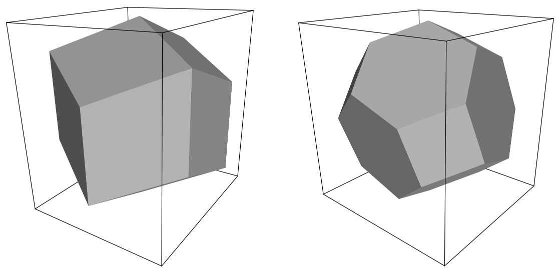


Figure 3.2: A rhombic dodecahedron and truncated octahedron (arbitrary orientations).

box type	image distance	box volume	box vectors **a**	box vectors **b**	box vectors **c**	box vector angles $\angle\mathbf{bc}$	$\angle\mathbf{ac}$	$\angle\mathbf{ab}$
cubic	d	d^3	d 0 0	0 d 0	0 0 d	90°	90°	90°
rhombic dodecahedron (xy-square)	d	$\frac{1}{2}\sqrt{2}d^3$ $\approx 0.71d^3$	d 0 0	0 d 0	$\frac{1}{2}d$ $\frac{1}{2}d$ $\frac{1}{2}\sqrt{2}d$	60°	60°	90°
rhombic dodecahedron (xy-hexagon)	d	$\frac{1}{2}\sqrt{2}d^3$ $\approx 0.71d^3$	d 0 0	$\frac{1}{2}d$ $\frac{1}{2}\sqrt{3}d$ 0	$\frac{1}{2}d$ $\frac{1}{6}\sqrt{3}d$ $\frac{1}{3}\sqrt{6}d$	60°	60°	60°
truncated octahedron	d	$\frac{4}{9}\sqrt{3}d^3$ $\approx 0.77d^3$	d 0 0	$\frac{1}{3}d$ $\frac{2}{3}\sqrt{2}d$ 0	$-\frac{1}{3}d$ $\frac{1}{3}\sqrt{2}d$ $\frac{1}{3}\sqrt{6}d$	$\approx 71°$	$\approx 71°$	$\approx 71°$

Table 3.1: The cubic box, the rhombic dodecahedron and the truncated octahedron.

the output trajectory it might seem like the simulation was done in a rectangular box. The program `trjconv` can be used to convert the trajectory to a different unit-cell representation.

It is also possible to simulate without periodic boundary conditions, but it is more efficient to simulate an isolated cluster of molecules in a large periodic box, since fast grid searching can only be used in a periodic system.

3.2.1 Some useful box types

The three most useful box types for simulations of solvated systems are described in Table 3.1. The rhombic dodecahedron (Fig. 3.2) is the smallest and most regular space-filling unit cell. Each of the 12 image cells is at the same distance. The volume is 71% of the volume of a cubic box with the same image distance. This saves about 29% of CPU-time when simulating a spherical or flexible molecule in solvent. A rhombic dodecahedron can have two different orientations, while fulfilling equations (3.1). The program `editconf` produces the orientation which has a square cross-section with the xy-plane. This orientation was chosen because the first two box vectors coincide with the x and y-axis, which is easier to comprehend. The other orientation can be useful for simulations of membrane proteins. In this case the cross-section with the xy-plane is a hexagon, which has an area which is 14% smaller than the area of a square with the same image distance. The height of the box (c_z) should be changed to obtain an optimal spacing. This box shape does not only save CPU-time, it also results in a more uniform arrangement of the proteins.

3.2.2 Cut-off restrictions

The minimum image convention implies that the cut-off radius used to truncate non-bonded interactions must not exceed half the shortest box vector for grid search:

$$R_c < \frac{1}{2}\min(\|\mathbf{a}\|, \|\mathbf{b}\|, \|\mathbf{c}\|), \tag{3.4}$$

otherwise more than one image would be within the cut-off distance of the force. When a macromolecule, such as a protein, is studied in solution, this restriction does not suffice. In principle a single solvent molecule should not be able to ‘see’ both sides of the macromolecule. This means that the length of each box vector must exceed the length of the macromolecule in the direction of that edge *plus* two times the cut-off radius R_c. It is common to compromise in this respect, and make the solvent layer somewhat smaller in order to reduce the computational cost. For efficiency reasons the cut-off with simple search in triclinic boxes (grid search always uses eq. (3.4)) is more restriced:

$$R_c < \frac{1}{2}\min(a_x, b_y, c_z) \tag{3.5}$$

Each unit cell (cubic, rectangular or triclinic) is surrounded by 26 translated images. Thus a particular image can always be identified by an index pointing to one of 27 *translation vectors* and constructed by applying a translation with the indexed vector (see 3.4.3).

3.3 The group concept

In the GROMACS MD and analysis programs one uses *groups* of atoms to perform certain actions on. The maximum number of groups is 256, but every atom can only belong to four different groups, one of each of the following kinds:

T-coupling group The temperature coupling parameters (reference temperature, time constant, number of degrees of freedom, see 3.4.4) can be defined for each T-coupling group separately. For example, in a solvated macromolecule the solvent (that tends to produce more heating by force and integration errors) can be coupled with a shorter time constant to a bath than a macromolecule, or a surface can be kept cooler than an adsorbing molecule. Many different T-coupling groups may be defined.

Freeze group Atoms that belong to a freeze group are kept stationary in the dynamics. This is useful during equilibration, e.g. to avoid that badly placed solvent molecules will give unreasonable kicks to protein atoms, although the same effect can also be obtained by putting a restraining potential on the atoms that must be protected. The freeze option can be used

---

E.49 genbox

Genbox can do one of 3 things:

1) Generate a box of solvent. Specify -cs and -box. Or specify -cs and -cp with a structure file with a box, but without atoms.

2) Solvate a solute configuration, eg. a protein, in a bath of solvent molecules. Specify `-cp` (solute) and `-cs` (solvent). The box specified in the solute coordinate file (`-cp`) is used, unless `-box` is set, which also centers the solute. The program `editconf` has more sophisticated options to change the box and center the solute. Solvent molecules are removed from the box where the distance between any atom of the solute molecule(s) and any atom of the solvent molecule is less than the sum of the VanderWaals radii of both atoms. A database (`vdwradii.dat`) of VanderWaals radii is read by the program, atoms not in the database are assigned a default distance `-vdw`.

3) Insert a number (`-nmol`) of extra molecules (`-ci`) at random positions. The program iterates until `nmol` molecules have been inserted in the box. To test whether an insertion is successful the same VanderWaals criterium is used as for removal of solvent molecules. When no appropriately sized holes (holes that can hold an extra molecule) are available the program tries for `-nmol` * `-try` times before giving up. Increase -try if you have several small holes to fill.

The default solvent is Simple Point Charge water (SPC), with coordinates from `$GMXLIB/spc216.gro`. Other solvents are also supported, as well as mixed solvents. The only restriction to solvent types is that a solvent molecule consists of exactly one residue. The residue information in the coordinate files is used, and should therefore be more or less consistent. In practice this means that two subsequent solvent molecules in the solvent coordinate file should have different residue number. The box of solute is built by stacking the coordinates read from the coordinate file. This means that these coordinates should be equlibrated in periodic boundary conditions to ensure a good alignment of molecules on the stacking interfaces.

The program can optionally rotate the solute molecule to align the longest molecule axis along a box edge. This way the amount of solvent molecules necessary is reduced. It should be kept in mind that this only works for short simulations, as eg. an alpha-helical peptide in solution can rotate over 90 degrees, within 500 ps. In general it is therefore better to make a more or less cubic box.

Setting -shell larger than zero will place a layer of water of the specified thickness (nm) around the solute. Hint: it is a good idea to put the protein in the center of a box first (using editconf).

Finally, genbox will optionally remove lines from your topology file in which a number of solvent molecules is already added, and adds a line with the total number of solvent molecules in your coordinate file.

Files

`-cp`	`protein.gro`	Input, Opt.	Generic structure: gro g96 pdb tpr tpb tpa
`-cs`	`spc216.gro`	Input, Opt., Lib.	Generic structure: gro g96 pdb tpr tpb tpa
`-ci`	`insert.gro`	Input, Opt.	Generic structure: gro g96 pdb tpr tpb tpa
`-o`	`out.gro`	Output	Generic structure: gro g96 pdb
`-p`	`topol.top`	In/Out, Opt.	Topology file

Other options

`-h`	bool	`no`	Print help info and quit
`-nice`	int	`19`	Set the nicelevel
`-box`	vector	`0 0 0`	box size
`-nmol`	int	`0`	no of extra molecules to insert
`-try`	int	`10`	try inserting -nmol*-try times
`-seed`	int	`1997`	random generator seed
`-vdwd`	real	`0.105`	default vdwaals distance
`-shell`	real	`0`	thickness of optional water layer around solute

`-ot`	`temp.xvg`	Output, Opt.	xvgr/xmgr file
`-ekr`	`ekrot.xvg`	Output, Opt.	xvgr/xmgr file

Other options

`-h`	bool	`no`	Print help info and quit
`-nice`	int	`19`	Set the nicelevel
`-b`	time	`-1`	First frame (ps) to read from trajectory
`-e`	time	`-1`	Last frame (ps) to read from trajectory
`-dt`	time	`-1`	Only use frame when t MOD dt = first time (ps)
`-tu`	enum	`ps`	Time unit: `ps`, `fs`, `ns`, `us`, `ms`, `s`, `m` or `h`
`-w`	bool	`no`	View output xvg, xpm, eps and pdb files
`-com`	bool	`no`	Plot data for the com of each group
`-mol`	bool	`no`	Index contains molecule numbers iso atom numbers
`-nojump`	bool	`no`	Remove jumps of atoms across the box
`-x`	bool	`yes`	Plot X-component
`-y`	bool	`yes`	Plot Y-component
`-z`	bool	`yes`	Plot Z-component
`-len`	bool	`no`	Plot vector length

E.48 g_velacc

g_velacc computes the velocity autocorrelation function. When the `-s` option is used, the momentum autocorrelation function is calculated.

With option `-mol` the momentum autocorrelation function of molecules is calculated. In this case the index group should consist of molecule numbers instead of atom numbers.

Files

`-f`	`traj.trr`	Input	Full precision trajectory: trr trj
`-s`	`topol.tpr`	Input, Opt.	Structure+mass(db): tpr tpb tpa gro g96 pdb
`-n`	`index.ndx`	Input, Opt.	Index file
`-o`	`vac.xvg`	Output	xvgr/xmgr file

Other options

`-h`	bool	`no`	Print help info and quit
`-nice`	int	`19`	Set the nicelevel
`-b`	time	`-1`	First frame (ps) to read from trajectory
`-e`	time	`-1`	Last frame (ps) to read from trajectory
`-dt`	time	`-1`	Only use frame when t MOD dt = first time (ps)
`-w`	bool	`no`	View output xvg, xpm, eps and pdb files
`-mol`	bool	`no`	Calculate vac of molecules
`-acflen`	int	`-1`	Length of the ACF, default is half the number of frames
`-normalize`	bool	`yes`	Normalize ACF
`-P`	enum	`0`	Order of Legendre polynomial for ACF (0 indicates none): `0`, `1`, `2` or `3`
`-fitfn`	enum	`none`	Fit function: `none`, `exp`, `aexp`, `exp_exp` or `vac`
`-ncskip`	int	`0`	Skip N points in the output file of correlation functions
`-beginfit`	real	`0`	Time where to begin the exponential fit of the correlation function
`-endfit`	real	`-1`	Time where to end the exponential fit of the correlation function, -1 is till the end

---

on one or two coordinates of an atom, thereby freezing the atoms in a plane or on a line. Many freeze groups can be defined.

Accelerate group On each atom in an 'accelerate group' an acceleration $\boldsymbol{a}^g$ will be imposed. This is equivalent to an external force. This feature makes it possible to drive the system into a non-equilibrium state and enables to perform non-equilibrium MD to obtain transport properties.

Energy monitor group Mutual interactions between all energy monitor groups are compiled during the simulation. This is done separately for Lennard-Jones and Coulomb terms. In principle up to 256 groups could be defined, but that would lead to 256×256 items! Better use this concept sparingly.

All non-bonded interactions between pairs of energy monitor groups can be excluded (see sec. 7.3.1). Pairs of particles from excluded pairs of energy monitor groups are not put into the pair list. This can result in a significant speedup for simulations where interactions within or between parts of the system are not required.

The use of groups in analysis programs is described in chapter 8.

3.4 Molecular Dynamics

A global flow scheme for MD is given in Fig. 3.3. Each MD or EM run requires as input a set of initial coordinates and - optionally - initial velocities of all particles involved. This chapter does not describe how these are obtained; for the setup of an actual MD run check the online manual at www.gromacs.org.

3.4.1 Initial conditions

Topology and force field

The system topology, including a description of the force field, must be loaded. These items are described in chapter 4. All this information is static; it is never modified during the run.

Coordinates and velocities

Then, before a run starts, the box size and the coordinates and velocities of all particles are required. The box size is determined by three vectors (nine numbers) $\boldsymbol{b}_1, \boldsymbol{b}_2, \boldsymbol{b}_3$, which represent the three basis vectors of the periodic box. While in the present version of GROMACS only rectangular boxes are allowed, three numbers suffice, but the use of three vectors already prepares for arbitrary triclinic boxes to be implemented in a later version.

If the run starts at $t = t_0$, the coordinates at $t = t_0$ must be known. The *leap-frog algorithm*, used to update the time step with Δt (see 3.4.4), requires that the velocities must be known at $t = t_0 - \frac{\Delta t}{2}$. If velocities are not available, the program can generate initial atomic velocities

THE GLOBAL MD ALGORITHM

---

1. Input initial conditions

Potential interaction V as a function of atom positions
Positions $\boldsymbol{r}$ of all atoms in the system
Velocities $\boldsymbol{v}$ of all atoms in the system
$\Downarrow$

---

repeat 2,3,4 required number of steps:

---

2. Compute forces

The force on any atom

$$\boldsymbol{F}_i = -\frac{\partial V}{\partial \boldsymbol{r}_i}$$

is computed by calculating the force between non-bonded atom pairs:

$$\boldsymbol{F}_i = \sum_j \boldsymbol{F}_{ij}$$

plus the forces due to bonded interactions (which may depend on 1, 2, 3, or 4 atoms), plus restraining and/or external forces.
The potential and kinetic energies and the pressure tensor are computed.
$\Downarrow$

3. Update configuration

The movement of the atoms is simulated by numerically solving Newton's equations of motion

$$\frac{\mathrm{d}^2\boldsymbol{r}_i}{\mathrm{d}t^2} = \frac{\boldsymbol{F}_i}{m_i}$$

or

$$\frac{\mathrm{d}\boldsymbol{r}_i}{\mathrm{d}t} = \boldsymbol{v}_i ; \quad \frac{\mathrm{d}\boldsymbol{v}_i}{\mathrm{d}t} = \frac{\boldsymbol{F}_i}{m_i}$$

$\Downarrow$

4. if required: **Output step**
write positions, velocities, energies, temperature, pressure, etc.

Figure 3.3: The global MD algorithm

When the box is cubic, one can use the option `-oc`, which averages the tcaf's over all k-vectors with the same length. This results in more accurate tcaf's. Both the cubic tcaf's and fits are written to `-oc` The cubic eta estimates are also written to `-ov`.

With option `-mol` the transverse current is determined of molecules instead of atoms. In this case the index group should consist of molecule numbers instead of atom numbers.

The k-dependent viscosities in the `-ov` file should be fitted to eta(k) = eta0 (1 - a k^2) to obtain the viscosity at infinite wavelength.

NOTE: make sure you write coordinates and velocities often enough. The initial, non-exponential, part of the autocorrelation function is very important for obtaining a good fit.

Files

`-f`	`traj.trr`	Input	Full precision trajectory: trr trj
`-s`	`topol.tpr`	Input, Opt.	Structure+mass(db): tpr tpb tpa gro g96 pdb
`-n`	`index.ndx`	Input, Opt.	Index file
`-ot`	`transcur.xvg`	Output, Opt.	xvgr/xmgr file
`-oa`	`tcaf_all.xvg`	Output	xvgr/xmgr file
`-o`	`tcaf.xvg`	Output	xvgr/xmgr file
`-of`	`tcaf_fit.xvg`	Output	xvgr/xmgr file
`-oc`	`tcaf_cub.xvg`	Output, Opt.	xvgr/xmgr file
`-ov`	`visc_k.xvg`	Output	xvgr/xmgr file

Other options

`-h`	bool	`no`	Print help info and quit
`-nice`	int	`19`	Set the nicelevel
`-b`	time	`-1`	First frame (ps) to read from trajectory
`-e`	time	`-1`	Last frame (ps) to read from trajectory
`-dt`	time	`-1`	Only use frame when t MOD dt = first time (ps)
`-w`	bool	`no`	View output xvg, xpm, eps and pdb files
`-mol`	bool	`no`	Calculate tcaf of molecules
`-k34`	bool	`no`	Also use k=(3,0,0) and k=(4,0,0)
`-wt`	real	`5`	Exponential decay time for the TCAF fit weights

E.47 g_traj

g_traj plots coordinates, velocities, forces and/or the box. With `-com` the coordinates, velocities and forces are calculated for the center of mass of each group. When `-mol` is set, the numbers in the index file are interpreted as molecule numbers and the same procedure as with `-com` is used for each molecule.

Option `-ot` plots the temperature of each group, provided velocities are present in the trajectory file.This implies `-com`.

Option `-ekr` plots the rotational kinetic energy of each group, provided velocities are present in the trajectory file. This implies `-com`.

Files

`-f`	`traj.xtc`	Input	Generic trajectory: xtc trr trj gro g96 pdb
`-s`	`topol.tpr`	Input	Structure+mass(db): tpr tpb tpa gro g96 pdb
`-n`	`index.ndx`	Input, Opt.	Index file
`-ox`	`coord.xvg`	Output, Opt.	xvgr/xmgr file
`-ov`	`veloc.xvg`	Output, Opt.	xvgr/xmgr file
`-of`	`force.xvg`	Output, Opt.	xvgr/xmgr file
`-ob`	`box.xvg`	Output, Opt.	xvgr/xmgr file

E.45 g_sorient

g_sorient analyzes solvent orientation around solutes. It calculates two angles between the vector from one or more reference positions to the first atom of each solvent molecule:
theta1: the angle with the vector from the first atom of the solvent molecule to the midpoint between atoms 2 and 3.
theta2: the angle with the normal of the solvent plane, defined by the same three atoms.
The reference can be a set of atoms or the center of mass of a set of atoms. The group of solvent atoms should consist of 3 atoms per solvent molecule. Only solvent molecules between `-rmin` and `-rmax` are considered each frame.

`-o`: angle distribution of theta1.

`-no`: angle distribution of theta2.

`-ro`: <cos(theta1)> and <3cos^2(theta2)-1> as a function of the distance.

`-ro`: the sum over all solvent molecules within distance r of cos(theta1) and 3cos^2(theta2)-1 as a function of r.

Files

`-f`	`traj.xtc`	Input	Generic trajectory: xtc trr trj gro g96 pdb
`-s`	`topol.tpr`	Input	Structure+mass(db): tpr tpb tpa gro g96 pdb
`-n`	`index.ndx`	Input, Opt.	Index file
`-o`	`sori.xvg`	Output	xvgr/xmgr file
`-no`	`snor.xvg`	Output	xvgr/xmgr file
`-ro`	`sord.xvg`	Output	xvgr/xmgr file
`-co`	`scum.xvg`	Output	xvgr/xmgr file

Other options

`-h`	bool	`no`	Print help info and quit
`-nice`	int	`19`	Set the nicelevel
`-b`	time	`-1`	First frame (ps) to read from trajectory
`-e`	time	`-1`	Last frame (ps) to read from trajectory
`-dt`	time	`-1`	Only use frame when t MOD dt = first time (ps)
`-w`	bool	`no`	View output xvg, xpm, eps and pdb files
`-com`	bool	`no`	Use the center of mass as the reference postion
`-rmin`	real	`0`	Minimum distance
`-rmax`	real	`0.5`	Maximum distance
`-nbin`	int	`20`	Number of bins

E.46 g_tcaf

g_tcaf computes tranverse current autocorrelations. These are used to estimate the shear viscosity eta. For details see: Palmer, JCP 49 (1994) pp 359-366.

Transverse currents are calculated using the k-vectors (1,0,0) and (2,0,0) each also in the y- and z-direction, (1,1,0) and (1,-1,0) each also in the 2 other plains (these vectors are not independent) and (1,1,1) and the 3 other box diagonals (also not independent). For each k-vector the sine and cosine are used, in combination with the velocity in 2 perpendicular directions. This gives a total of 16*2*2=64 transverse currents. One autocorrelation is calculated fitted for each k-vector, which gives 16 tcaf's. Each of these tcaf's is fitted to f(t) = exp(-v)(cosh(Wv) + 1/W sinh(Wv)), v = -t/(2 tau), W = sqrt(1 - 4 tau eta/rho k^2), which gives 16 tau's and eta's. The fit weights decay with time as exp(-t/wt), the tcaf and fit are calculated up to time 5*wt. The eta's should be fitted to 1 - a eta(k) k^2, from which one can estimate the shear viscosity at k=0.

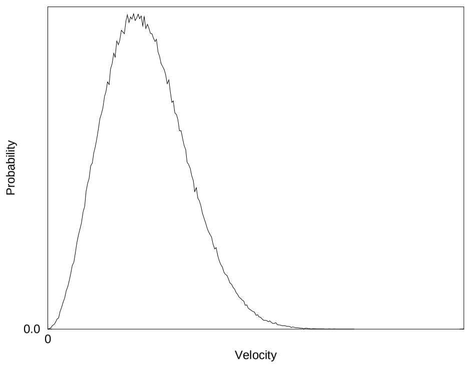


Figure 3.4: A Maxwellian distribution, generated from random numbers.

$v_i, i = 1 \dots 3N$ from a Maxwellian distribution (Fig. 3.4) at a given absolute temperature T:

$$p(v_i) = \sqrt{\frac{m_i}{2\pi kT}}\exp(-\frac{m_i v_i^2}{2kT}) \tag{3.6}$$

where k is Boltzmann's constant (see chapter 2). To accomplish this, normally distributed random numbers are generated by adding twelve random numbers R_k in the range $0 \le R_k < 1$ and subtracting 6.0 from their sum. The result is then multiplied by the standard deviation of the velocity distribution $\sqrt{kT/m_i}$. Since the resulting total energy will not correspond exactly to the required temperature T, a correction is made: first the center-of-mass motion is removed and then all velocities are scaled such that the total energy corresponds exactly to T (see eqn. 3.12).

Center-of-mass motion

The center-of-mass velocity is normally set to zero at every step. Normally there is no net external force acting on the system and the center-of-mass velocity should remain constant. In practice, however, the update algorithm develops a very slow change in the center-of-mass velocity, and thus in the total kinetic energy of the system, specially when temperature coupling is used. If such changes are not quenched, an appreciable center-of-mass motion develops eventually in long runs, and the temperature will be significantly misinterpreted. The same may happen due to overall rotational motion, but only when an isolated cluster is simulated. In periodic systems with filled boxes, the overall rotational motion is coupled to other degrees of freedom and does not give any problems.

3.4.2 Neighbor searching

As mentioned in chapter 4, internal forces are either generated from fixed (static) lists, or from dynamics lists. The latter concern non-bonded interactions between any pair of particles. When calculating the non-bonded forces, it is convenient to have all particles in a rectangular box. As shown in Fig. 3.1, it is possible to transform a triclinic box into a rectangular box. The output coordinates are always in a rectangular box, even when a dodecahedron or triclinic box was used for the simulation. Equations (3.1) ensure that we can reset particles in a rectangular box by first shifting them with box vector $\mathbf{c}$, then with $\mathbf{b}$ and finally with $\mathbf{a}$. Equations (3.3) ensure that we can find the 14 nearest triclinic images within a linear combination which does not involve multiples of box vectors.

Pair lists generation

The non-bonded pair forces need to be calculated only for those pairs i, j for which the distance r_{ij} between i and the nearest image of j is less than a given cut-off radius R_c. Some of the particle pairs that fulfill this criterion are excluded, when their interaction is already fully accounted for by bonded interactions. GROMACS employs a *pair list* that contains those particle pairs for which non-bonded forces must be calculated. The pair list contains the particle numbers and an index for the image displacement vectors that must be applied to obtain the nearest image, for all particle pairs that have a nearest-image distance less than `rshort`. The list is updated every `nstlist` steps, where `nstlist` is typically 10 for the GROMACS forcefield and 5 for the GROMOS-96 forcefield. There is an option to calculate the total non-bonded force on each particle due to all particle in a shell around the list-cutoff, *i.e*, at a distance between `rshort` and `rlong`. This force is calculated during the pair list update and retained during `nstlist` steps.

To make the neighbor list all particles that are close (i.e. within the cut-off) to a given particle must be found. This searching, usually called neighbor searching (NS), involves periodic boundary conditions and determining the *image* (see sec. 3.2). Without periodic boundary conditions a simple $O(N^2)$ algorithm must be used. With periodic boundary conditions a grid search can be used, which is $O(N)$.

Simple search

Due to equations (3.1) and (3.5), the vector $\boldsymbol{r}_{ij}$ connecting images within the cut-off R_c can be found by constructing:

$$\boldsymbol{r}''' = \boldsymbol{r}_j - \boldsymbol{r}_i \tag{3.7}$$

$$\boldsymbol{r}'' = \boldsymbol{r}''' - \mathbf{a} * \mathtt{round}(r_z'''/c_z) \tag{3.8}$$

$$\boldsymbol{r}' = \boldsymbol{r}'' - \mathbf{b} * \mathtt{round}(r_y''/b_y) \tag{3.9}$$

$$\boldsymbol{r}_{ij} = \boldsymbol{r}' - \mathbf{c} * \mathtt{round}(r_x'/a_x) \tag{3.10}$$

When distances between any two particles in a triclinic box are needed, many shifts of combinations of box vectors need to be considered to find the nearest image.

Other options

`-h`	bool	`no`	Print help info and quit
`-nice`	int	`19`	Set the nicelevel
`-b`	time	`-1`	First frame (ps) to read from trajectory
`-e`	time	`-1`	Last frame (ps) to read from trajectory
`-dt`	time	`-1`	Only use frame when t MOD dt = first time (ps)
`-w`	bool	`no`	View output xvg, xpm, eps and pdb files
`-solsize`	real	`0.14`	Radius of the solvent probe (nm)
`-ndots`	int	`24`	Number of dots per sphere, more dots means more accuracy
`-qmax`	real	`0.2`	The maximum charge (e, absolute value) of a hydrophobic atom
`-minarea`	real	`0.5`	The maximum charge (e, absolute value) of a hydrophobic atom
`-skip`	int	`1`	Do only every nth frame
`-prot`	bool	`yes`	Output the protein to the connelly pdb file too

E.44 g_sgangle

Compute the angle and distance between two groups. The groups are defined by a number of atoms given in an index file and may be two or three atoms in size. The angles calculated depend on the order in which the atoms are given. Giving for instance 5 6 will rotate the vector 5-6 with 180 degrees compared to giving 6 5.

If three atoms are given, the normal on the plane spanned by those three atoms will be calculated, using the formula P1P2 x P1P3. The cos of the angle is calculated, using the inproduct of the two normalized vectors.

Here is what some of the file options do:
-oa: Angle between the two groups specified in the index file. If a group contains three atoms the normal to the plane defined by those three atoms will be used. If a group contains two atoms, the vector defined by those two atoms will be used.
-od: Distance between two groups. Distance is taken from the center of one group to the center of the other group.
-od1: If one plane and one vector is given, the distances for each of the atoms from the center of the plane is given seperately.
-od2: For two planes this option has no meaning.

Files

`-f`	`traj.xtc`	Input	Generic trajectory: xtc trr trj gro g96 pdb
`-n`	`index.ndx`	Input	Index file
`-s`	`topol.tpr`	Input	Generic run input: tpr tpb tpa
`-oa`	`sg_angle.xvg`	Output	xvgr/xmgr file
`-od`	`sg_dist.xvg`	Output	xvgr/xmgr file
`-od1`	`sg_dist1.xvg`	Output	xvgr/xmgr file
`-od2`	`sg_dist2.xvg`	Output	xvgr/xmgr file

Other options

`-h`	bool	`no`	Print help info and quit
`-nice`	int	`19`	Set the nicelevel
`-b`	time	`-1`	First frame (ps) to read from trajectory
`-e`	time	`-1`	Last frame (ps) to read from trajectory
`-dt`	time	`-1`	Only use frame when t MOD dt = first time (ps)
`-w`	bool	`no`	View output xvg, xpm, eps and pdb files

`-acflen`	int	`-1`	Length of the ACF, default is half the number of frames
`-normalize`	bool	`yes`	Normalize ACF
`-P`	enum	`0`	Order of Legendre polynomial for ACF (0 indicates none): `0`, `1`, `2` or `3`
`-fitfn`	enum	`none`	Fit function: `none`, `exp`, `aexp`, `exp_exp` or `vac`
`-ncskip`	int	`0`	Skip N points in the output file of correlation functions
`-beginfit`	real	`0`	Time where to begin the exponential fit of the correlation function
`-endfit`	real	`-1`	Time where to end the exponential fit of the correlation function, -1 is till the end

E.42 g_saltbr

g_saltbr plots the difference between all combination of charged groups as a function of time. The groups are combined in different ways.A minimum distance can be given, (eg. the cut-off), then groups that are never closer than that distance will not be plotted.
Output will be in a number of fixed filenames, min-min.xvg,min-plus.xvg and plus-plus.xvg, or files for every individual ion-pair if selected

Files

`-f`	`traj.xtc`	Input	Generic trajectory: xtc trr trj gro g96 pdb
`-s`	`topol.tpr`	Input	Generic run input: tpr tpb tpa

Other options

`-h`	bool	`no`	Print help info and quit
`-nice`	int	`19`	Set the nicelevel
`-b`	time	`-1`	First frame (ps) to read from trajectory
`-e`	time	`-1`	Last frame (ps) to read from trajectory
`-dt`	time	`-1`	Only use frame when t MOD dt = first time (ps)
`-t`	real	`1000`	trunc distance
`-sep`	bool	`no`	Use separate files for each interaction (may be MANY)

E.43 g_sas

g_sas computes hydrophobic and total solvent accessible surface area. As a side effect the Connolly surface can be generated as well in a pdb file where the nodes are represented as atoms and the vertices connecting the nearest nodes as CONECT records. The area can be plotted per atom and per residue as well (option -ao). In combination with the latter option an `itp` file can be generated (option -i) which can be used to restrain surface atoms.

Files

`-f`	`traj.xtc`	Input	Generic trajectory: xtc trr trj gro g96 pdb
`-s`	`topol.tpr`	Input	Generic run input: tpr tpb tpa
`-o`	`area.xvg`	Output	xvgr/xmgr file
`-r`	`resarea.xvg`	Output	xvgr/xmgr file
`-q`	`connelly.pdb`	Output, Opt.	Protein data bank file
`-ao`	`atomarea.xvg`	Output, Opt.	xvgr/xmgr file
`-n`	`index.ndx`	Input, Opt.	Index file
`-i`	`surfat.itp`	Output, Opt.	Include file for topology

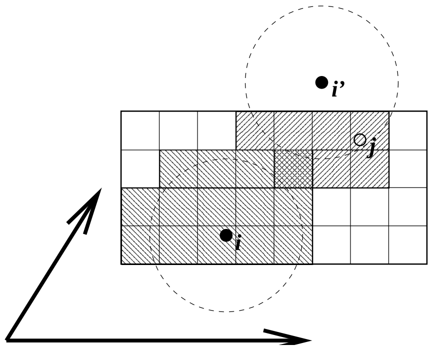


Figure 3.5: Grid search in two dimensions. The arrows are the box vectors.

Grid search

The grid search is schematically depicted in Fig. 3.5. All particles are put on the NS grid, with the smallest spacing $\geq R_c/2$ in each of the directions. In the direction of each box vector, a particle i has three images. For each direction the image may be -1,0 or 1, corresponding to a translation over -1, 0 or +1 box vector. We do not search the surrounding NS grid cells for neighbors of i and then calculate the image, but rather construct the images first and then search neighbors corresponding to that image of i. As can been seen in Fig. 3.5, for some images of i the same grid cell might be searched. This is not a problem, since at most one image will "see" the j-particle, due to the minimum image convention. For every particle, less than 125 (5^3) neighboring cells are searched. Therefore, the algorithm scales linear with the number of particles. Although the prefactor is large the scaling behavior makes the algorithm far superior over the standard $O(N^2)$ algorithm when the number of particles exceeds a few hundred. The grid search is equally fast for rectangular and triclinic boxes. Thus for most protein and peptide simulations the rhombic dodecahedron will be the preferable box shape.

Charge groups

Where applicable, neighbor searching is carried out on the basis of *charge groups*. A charge group is a small set of nearby atoms that have net charge zero. Charge groups are defined in the molecular topology. If the nearest image distance between the *geometrical centers* of the atoms of two charge groups is less than the cutoff radius, all atom pairs between the charge groups are included in the pair list. This procedure avoids the creation of charges due to the use of a cut-off (when one charge of a dipole is within range and the other not), which can have disastrous consequences for the behavior of the Coulomb interaction function at distances near the cut-off radius. If molecular groups have full charges (ions), charge groups do not avoid adverse cut-off effects, and you should consider using one of the lattice sum methods supplied by GROMACS [13].

If appropriately constructed shift functions are used for the electrostatic forces, no charge groups are needed. Such shift functions are implemented in GROMACS (see chapter 4) but must be used with care: in principle they should be combined with a lattice sum for long-range electrostatics.

3.4.3 Compute forces

Potential energy

When forces are computed, the potential energy of each interaction term is computed as well. The total potential energy is summed for various contributions, such as Lennard-Jones, Coulomb, and bonded terms. It is also possible to compute these contributions for *groups* of atoms that are separately defined (see sec. 3.3).

Kinetic energy and temperature

The temperature is given by the total kinetic energy of the N-particle system:

$$E_{kin} = \frac{1}{2}\sum_{i=1}^{N} m_i v_i^2 \tag{3.11}$$

From this the absolute temperature T can be computed using:

$$\frac{1}{2}N_{df}kT = E_{kin} \tag{3.12}$$

where k is Boltzmann's constant and N_{df} is the number of degrees of freedom which can be computed from:

$$N_{df} \;=\; 3N - N_c - N_{com} \tag{3.13}$$

Here N_c is the number of *constraints* imposed on the system. When performing molecular dynamics $N_{com} = 3$ additional degrees of freedom must be removed, because the three center-of-mass velocities are constants of the motion, which are usually set to zero. When simulating in vacuo, the rotation around the center of mass can also be removed, in this case $N_{com} = 6$. When more than one temperature coupling group is used, the number of degrees of freedom for group i is:

$$N^i_{df} \;=\; (3N^i - N^i_c)\frac{3N - N_c - N_{com}}{3N - N_c} \tag{3.14}$$

The kinetic energy can also be written as a tensor, which is necessary for pressure calculation in a triclinic system, or systems where shear forces are imposed:

$$\mathbf{E}_{kin} = \frac{1}{2}\sum_{i}^{N} m_i \boldsymbol{v}_i \otimes \boldsymbol{v}_i \tag{3.15}$$

Pressure and virial

The pressure tensor **P** is calculated from the difference between kinetic energy E_{kin} and the virial $\mathbf{\Xi}$

$$\mathbf{P} = \frac{2}{V}(\mathbf{E}_{kin} - \mathbf{\Xi}) \tag{3.16}$$

`-f`	`traj.xtc`	Input	Generic trajectory: xtc trr trj gro g96 pdb
`-s`	`topol.tpr`	Input	Structure+mass(db): tpr tpb tpa gro g96 pdb
`-n`	`index.ndx`	Input, Opt.	Index file
`-q`	`eiwit.pdb`	Input, Opt.	Protein data bank file
`-oq`	`bfac.pdb`	Output, Opt.	Protein data bank file
`-ox`	`xaver.pdb`	Output, Opt.	Protein data bank file
`-o`	`rmsf.xvg`	Output	xvgr/xmgr file
`-od`	`rmsdev.xvg`	Output, Opt.	xvgr/xmgr file
`-oc`	`correl.xvg`	Output, Opt.	xvgr/xmgr file
`-dir`	`rmsf.log`	Output, Opt.	Log file

Other options

`-h`	bool	`no`	Print help info and quit
`-nice`	int	`19`	Set the nicelevel
`-b`	time	`-1`	First frame (ps) to read from trajectory
`-e`	time	`-1`	Last frame (ps) to read from trajectory
`-dt`	time	`-1`	Only use frame when t MOD dt = first time (ps)
`-w`	bool	`no`	View output xvg, xpm, eps and pdb files
`-res`	bool	`no`	Calculate averages for each residue
`-aniso`	bool	`no`	Compute anisotropic termperature factors

E.41 g_rotacf

g_rotacf calculates the rotational correlation function for molecules. Three atoms (i,j,k) must be given in the index file, defining two vectors ij and jk. The rotational acf is calculated as the autocorrelation function of the vector n = ij x jk, i.e. the cross product of the two vectors. Since three atoms span a plane, the order of the three atoms does not matter. Optionally, controlled by the -d switch, you can calculate the rotational correlation function for linear molecules by specifying two atoms (i,j) in the index file.

EXAMPLES

g_rotacf -P 1 -nparm 2 -fft -n index -o rotacf-x-P1 -fa expfit-x-P1 -beginfit 2.5 -endfit 20.0

This will calculate the rotational correlation function using a first order Legendre polynomial of the angle of a vector defined by the index file. The correlation function will be fitted from 2.5 ps till 20.0 ps to a two parameter exponential

Files

`-f`	`traj.xtc`	Input	Generic trajectory: xtc trr trj gro g96 pdb
`-s`	`topol.tpr`	Input	Generic run input: tpr tpb tpa
`-n`	`index.ndx`	Input	Index file
`-o`	`rotacf.xvg`	Output	xvgr/xmgr file

Other options

`-h`	bool	`no`	Print help info and quit
`-nice`	int	`19`	Set the nicelevel
`-b`	time	`-1`	First frame (ps) to read from trajectory
`-e`	time	`-1`	Last frame (ps) to read from trajectory
`-dt`	time	`-1`	Only use frame when t MOD dt = first time (ps)
`-w`	bool	`no`	View output xvg, xpm, eps and pdb files
`-d`	bool	`no`	Use index doublets (vectors) for correlation function instead of triplets (planes)
`-aver`	bool	`yes`	Average over molecules

0.6 in this case) can be generated, by default averaging over equivalent hydrogens (all triplets of hydrogens named *[123]). Additionally a list of equivalent atoms can be supplied (`-equiv`), each line containing a set of equivalent atoms specified as residue number and name and atom name; e.g.:

```
3 SER HB1 3 SER HB2
```

Residue and atom names must exactly match those in the structure file, including case. Specifying non-sequential atoms is undefined.

Files

`-f`	`traj.xtc`	Input	Generic trajectory: xtc trr trj gro g96 pdb
`-s`	`topol.tpr`	Input	Structure+mass(db): tpr tpb tpa gro g96 pdb
`-n`	`index.ndx`	Input, Opt.	Index file
`-equiv`	`equiv.dat`	Input, Opt.	Generic data file
`-o`	`distrmsd.xvg`	Output	xvgr/xmgr file
`-rms`	`rmsdist.xpm`	Output, Opt.	X PixMap compatible matrix file
`-scl`	`rmsscale.xpm`	Output, Opt.	X PixMap compatible matrix file
`-mean`	`rmsmean.xpm`	Output, Opt.	X PixMap compatible matrix file
`-nmr3`	`nmr3.xpm`	Output, Opt.	X PixMap compatible matrix file
`-nmr6`	`nmr6.xpm`	Output, Opt.	X PixMap compatible matrix file
`-noe`	`noe.dat`	Output, Opt.	Generic data file

Other options

`-h`	bool	`no`	Print help info and quit
`-nice`	int	`19`	Set the nicelevel
`-b`	time	`-1`	First frame (ps) to read from trajectory
`-e`	time	`-1`	Last frame (ps) to read from trajectory
`-dt`	time	`-1`	Only use frame when t MOD dt = first time (ps)
`-w`	bool	`no`	View output xvg, xpm, eps and pdb files
`-nlevels`	int	`40`	Discretize rms in # levels
`-max`	real	`-1`	Maximum level in matrices
`-sumh`	bool	`yes`	average distance over equivalent hydrogens

E.40 g_rmsf

g_rmsf computes the root mean square fluctuation (RMSF, i.e. standard deviation) of atomic positions after first fitting to a reference frame.

With option `-oq` the RMSF values are converted to B-factor values, which are written to a pdb file with the coordinates, of the structure file, or of a pdb file when `-q` is specified. Option `-ox` writes the B-factors to a file with the average coordinates.

With the option `-od` the root mean square deviation with respect to the reference structure is calculated.

With the option `aniso` g_rmsf will compute anisotropic temperature factors and then it will also output average coordinates and a pdb file with ANISOU records (corresonding to the `-oq` or `-ox` option). Please note that the U values are orientation dependent, so before comparison with experimental data you should verify that you fit to the experimental coordinates.

When a pdb input file is passed to the program and the `-aniso` flag is set a correlation plot of the Uij will be created, if any anisotropic temperature factors are present in the pdb file.

With option `-dir` the average MSF (3x3) matrix is diagonalized. This shows the directions in which the atoms fluctuate the most and the least.

Files

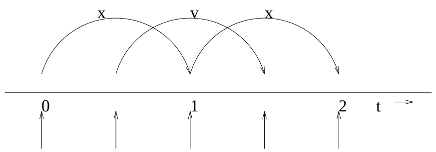


Figure 3.6: The Leap-Frog integration method. The algorithm is called Leap-Frog because r and v are leaping like frogs over each others back.

where V is the volume of the computational box. The scalar pressure P, which can be used for pressure coupling in the case of isotropic systems, is computed as:

$$P = \mathrm{trace}(\mathbf{P})/3 \tag{3.17}$$

The virial $\boldsymbol{\Xi}$ tensor is defined as

$$\boldsymbol{\Xi} = -\frac{1}{2}\sum_{i<j} \boldsymbol{r}_{ij} \otimes \boldsymbol{F}_{ij} \tag{3.18}$$

In sec. B.1 the implementation in GROMACS of the virial computation is described.

3.4.4 Update configuration

The GROMACS MD program utilizes the so-called *leap-frog* algorithm [14] for the integration of the equations of motion. The leap-frog algorithm uses positions $\boldsymbol{r}$ at time t and velocities $\boldsymbol{v}$ at time $t - \frac{\Delta t}{2}$; it updates positions and velocities using the forces $\boldsymbol{F}(t)$ determined by the positions at time t:

$$\boldsymbol{v}(t + \frac{\Delta t}{2}) = \boldsymbol{v}(t - \frac{\Delta t}{2}) + \frac{\boldsymbol{F}(t)}{m}\Delta t \tag{3.19}$$

$$\boldsymbol{r}(t + \Delta t) = \boldsymbol{r}(t) + \boldsymbol{v}(t + \frac{\Delta t}{2})\Delta t \tag{3.20}$$

The algorithm is visualized in Fig. 3.6. It is equivalent to the Verlet [15] algorithm:

$$\boldsymbol{r}(t + \Delta t) = 2\boldsymbol{r}(t) - \boldsymbol{r}(t - \Delta t) + \frac{\boldsymbol{F}(t)}{m}\Delta t^2 + O(\Delta t^4) \tag{3.21}$$

The algorithm is of third order in $\boldsymbol{r}$ and is time-reversible. See ref. [16] for the merits of this algorithm and comparison with other time integration algorithms.

The equations of motion are modified for temperature coupling and pressure coupling, and extended to include the conservation of constraints, all of which are described below.

3.4.5 Temperature coupling

For several reasons (drift during equilibration, drift as a result of force truncation and integration errors, heating due to external or frictional forces), it is necessary to control the temperature of the system. GROMACS can use either the *weak coupling* scheme of Berendsen [17] or the extended ensemble Nosé-Hoover scheme [18, 19].

Berendsen temperature coupling

The Berendsen algorithm mimics weak coupling with first-order kinetics to an external heat bath with given temperature T_0. See ref. [20] for a comparison with the Nosé-Hoover scheme. The effect of this algorithm is that a deviation of the system temperature from T_0 is slowly corrected according to

$$\frac{\mathrm{d}T}{\mathrm{d}t} = \frac{T_0 - T}{\tau} \tag{3.22}$$

which means that a temperature deviation decays exponentially with a time constant τ. This method of coupling has the advantage that the strength of the coupling can be varied and adapted to the user requirement: for equilibration purposes the coupling time can be taken quite short (e.g. 0.01 ps), but for reliable equilibrium runs it can be taken much longer (e.g. 0.5 ps) in which case it hardly influences the conservative dynamics.

The heat flow into or out of the system is effected by scaling the velocities of each particle every step with a time-dependent factor λ, given by

$$\lambda = \left[1 + \frac{\Delta t}{\tau_T}\left\{\frac{T_0}{T(t-\frac{\Delta t}{2})} - 1\right\}\right]^{1/2} \tag{3.23}$$

The parameter τ_T is close to, but not exactly equal to the time constant τ of the temperature coupling (eqn. 3.22):

$$\tau = 2C_V\tau_T/N_{df}k \tag{3.24}$$

where C_V is the total heat capacity of the system, k is Boltzmann's constant, and N_{df} is the total number of degrees of freedom. The reason that $\tau \neq \tau_T$ is that the kinetic energy change caused by scaling the velocities is partly redistributed between kinetic and potential energy and hence the change in temperature is less than the scaling energy. In practice, the ratio τ/τ_T ranges from 1 (gas) to 2 (harmonic solid) to 3 (water). When we use the term 'temperature coupling time constant', we mean the parameter τ_T. **Note** that in practice the scaling factor λ is limited to the range of 0.8 $<=\lambda<=$ 1.25, to avoid scaling by very large numbers which may crash the simulation. In normal use, λ will always be much closer to 1.0.

Strictly, for computing the scaling factor the temperature T is needed at time t, but this is not available in the algorithm. In practice, the temperature at the previous time step is used (as indicated in eqn. 3.23), which is perfectly all right since the coupling time constant is much longer than one time step. The Berendsen algorithm is stable up to $\tau_T \approx \Delta t$.

Nosé-Hoover temperature coupling

The Berendsen weak coupling algorithm is extremely efficient for relaxing a system to the target temperature, but once your system has reached equilibrium it might be more important to probe a correct canonical ensemble. This is unfortunately not the case for the weak coupling scheme, although the difference is usually negligible.

To enable canonical ensemble simulations, GROMACSalso supports the extended-ensemble approach first proposed by Nosé[18] and later modified by Hoover[19]. The system Hamiltonian is extended by introducing a thermal reservoir and a friction term in the equations of motion. The

---

All the structures are fitted pairwise.

With `-f2`, the 'other structures' are taken from a second trajectory.

Option `-bin` does a binary dump of the comparison matrix.

Option `-bm` produces a matrix of average bond angle deviations analogously to the `-m` option. Only bonds between atoms in the comparison group are considered.

Files

`-s`	`topol.tpr`	Input	Structure+mass(db): tpr tpb tpa gro g96 pdb
`-f`	`traj.xtc`	Input	Generic trajectory: xtc trr trj gro g96 pdb
`-f2`	`traj.xtc`	Input, Opt.	Generic trajectory: xtc trr trj gro g96 pdb
`-n`	`index.ndx`	Input, Opt.	Index file
`-o`	`rmsd.xvg`	Output	xvgr/xmgr file
`-mir`	`rmsdmir.xvg`	Output, Opt.	xvgr/xmgr file
`-a`	`avgrp.xvg`	Output, Opt.	xvgr/xmgr file
`-dist`	`rmsd-dist.xvg`	Output, Opt.	xvgr/xmgr file
`-m`	`rmsd.xpm`	Output, Opt.	X PixMap compatible matrix file
`-bin`	`rmsd.dat`	Output, Opt.	Generic data file
`-bm`	`bond.xpm`	Output, Opt.	X PixMap compatible matrix file

Other options

`-h`	bool	`no`	Print help info and quit
`-nice`	int	`19`	Set the nicelevel
`-b`	time	`-1`	First frame (ps) to read from trajectory
`-e`	time	`-1`	Last frame (ps) to read from trajectory
`-dt`	time	`-1`	Only use frame when t MOD dt = first time (ps)
`-tu`	enum	`ps`	Time unit: `ps`, `fs`, `ns`, `us`, `ms`, `s`, `m` or `h`
`-w`	bool	`no`	View output xvg, xpm, eps and pdb files
`-what`	enum	`rmsd`	Structural difference measure: `rmsd`, `rho` or `rhosc`
`-pbc`	bool	`yes`	PBC check
`-fit`	bool	`yes`	Fit to reference structure
`-prev`	int	`0`	Compare with previous frame
`-split`	bool	`no`	Split graph where time is zero
`-skip`	int	`1`	Only write every nr-th frame to matrix
`-skip2`	int	`1`	Only write every nr-th frame to matrix
`-max`	real	`-1`	Maximum level in comparison matrix
`-min`	real	`-1`	Minimum level in comparison matrix
`-bmax`	real	`-1`	Maximum level in bond angle matrix
`-bmin`	real	`-1`	Minimum level in bond angle matrix
`-nlevels`	int	`80`	Number of levels in the matrices

E.39 g_rmsdist

g_rmsdist computes the root mean square deviation of atom distances, which has the advantage that no fit is needed like in standard RMS deviation as computed by g_rms. The reference structure is taken from the structure file. The rmsd at time t is calculated as the rms of the differences in distance between atom-pairs in the reference structure and the structure at time t.

g_rmsdist can also produce matrices of the rms distances, rms distances scaled with the mean distance and the mean distances and matrices with NMR averaged distances ($1/r^3$ and $1/r^6$ averaging). Finally, lists of atom pairs with $1/r^3$ and $1/r^6$ averaged distance below the maximum distance (`-max`, which will default to

If a run input file is supplied (`-s`), exclusions defined in that file are taken into account when calculating the rdf. The option `-cut` is meant as an alternative way to avoid intramolecular peaks in the rdf plot. It is however better to supply a run input file with a higher number of exclusions. For eg. benzene a topology with nrexcl set to 5 would eliminate all intramolecular contributions to the rdf. Note that all atoms in the selected groups are used, also the ones that don't have Lennard-Jones interactions.

Option `-cn` produces the cumulative number rdf.

To bridge the gap between theory and experiment structure factors can be computed (option `-sq`). The algorithm uses FFT, the gridspacing of which is determined by option `-grid`.

Files

`-f`	`traj.xtc`	Input	Generic trajectory: xtc trr trj gro g96 pdb
`-s`	`topol.tpr`	Input, Opt.	Structure+mass(db): tpr tpb tpa gro g96 pdb
`-n`	`index.ndx`	Input, Opt.	Index file
`-o`	`rdf.xvg`	Output, Opt.	xvgr/xmgr file
`-sq`	`sq.xvg`	Output, Opt.	xvgr/xmgr file
`-cn`	`rdf_cn.xvg`	Output, Opt.	xvgr/xmgr file
`-hq`	`hq.xvg`	Output, Opt.	xvgr/xmgr file
`-image`	`sq.xpm`	Output, Opt.	X PixMap compatible matrix file

Other options

`-h`	bool	`no`	Print help info and quit
`-nice`	int	`19`	Set the nicelevel
`-b`	time	`-1`	First frame (ps) to read from trajectory
`-e`	time	`-1`	Last frame (ps) to read from trajectory
`-dt`	time	`-1`	Only use frame when t MOD dt = first time (ps)
`-w`	bool	`no`	View output xvg, xpm, eps and pdb files
`-bin`	real	`0.001`	Binwidth (nm)
`-com`	bool	`no`	RDF with respect to the center of mass of first group
`-cut`	real	`0`	Shortest distance (nm) to be considered
`-fade`	real	`0`	From this distance onwards the RDF is tranformed by g'(r) = 1 + [g(r)-1] exp(-(r/fade-1)^2 to make it go to 1 smoothly. If fade is 0.0 nothing is done.
`-grid`	real	`0.05`	Grid spacing (in nm) for FFTs when computing structure factors
`-nlevel`	int	`20`	Number of different colors in the diffraction image
`-wave`	real	`0.1`	Wavelength for X-rays/Neutrons for scattering. 0.1 nm corresponds to roughly 12 keV

E.38 g_rms

g_rms compares two structures by computing the root mean square deviation (RMSD), the size-independent 'rho' similarity parameter (rho) or the scaled rho (rhosc), reference Maiorov & Crippen, PROTEINS 22, 273 (1995). This is selected by `-what`.

Each structure from a trajectory (`-f`) is compared to a reference structure from a run input file by least-squares fitting the structures on top of each other. The reference structure is taken from the structure file (`-s`).

With option `-mir` also a comparison with the mirror image of the reference structure is calculated.

Option `-prev` produces the comparison with a previous frame.

Option `-m` produces a matrix in `.xpm` format of comparison values of each structure in the trajectory with respect to each other structure. This file can be visualized with for instance `xv` and can be converted to postscript with `xpm2ps`.

---

friction force is proportional to the product of each particle's velocity and a friction parameter ξ. This friction parameter (or 'heat bath' variable) is a fully dynamic quantity with its own equation of motion; the time derivative is calculated from the difference between the current kinetic energy and the reference temperature.

In the formulation of Hoover, the particles' equations of motion in Fig. 3.3 are replaced by

$$\frac{\mathrm{d}^2\boldsymbol{r}_i}{\mathrm{d}t^2} = \frac{\boldsymbol{F}_i}{m_i} - \xi\frac{\mathrm{d}\boldsymbol{r}_i}{\mathrm{d}t}, \tag{3.25}$$

where the equation of motion for the heat bath parameter ξ is

$$\frac{\mathrm{d}\xi}{\mathrm{d}t} = \frac{1}{Q}\left(T - T_0\right). \tag{3.26}$$

The reference temperature is denoted T_0, while T is the current momentary temperature of the system. The strength of the coupling is determined by the constant Q (usually called the 'mass parameter' of the reservoir) in combination with the reference temperature.

In our opinion, the mass parameter is a somewhat awkward way of describing coupling strength, especially due to its dependence on reference temperature (and some implementations even include the number of degrees of freedom in your system when defining Q). To maintain the coupling strength, you would have to change Q proportional to your change in reference temperature. For this reason, we prefer to let the GROMACSuser work with the period τ_T of the oscillations of kinetic energy between the system and the reservoir instead. It is directly related to Q and T_0 as

$$Q = \frac{\tau_T^2 T_0}{4\pi^2}. \tag{3.27}$$

This provides a much more intuitive way of selecting the Nosé-Hoover coupling strength (similar to the weak coupling relaxation), and in addition τ_T is independent of system size and reference temperature.

It is however important to keep the difference between the weak coupling scheme and the Nosé-Hoover algorithm in mind: Using weak coupling you get a strongly damped *exponential relaxation*, while the Nosé-Hoover approach produces an *oscillatory relaxation*. The actual time it takes to relax with Nosé-Hoover coupling is several times larger than the period of the oscillations that you select. These oscillations (in contrast to exponential relaxation) also means that the time constant normally should be 4–5 times larger than the relaxation time used with weak coupling, but your mileage may vary.

3.4.6 Pressure coupling

In the same spirit as the temperature coupling, the system can also be coupled to a 'pressure bath'. GROMACS supports both the Berendsen algorithm [17] that scales coordinates and box vectors every step, and the extended ensemble Parrinello-Rahman approach. Both of these can be combined with any of the temperature coupling methods above.

Berendsen pressure coupling

The Berendsen algorithm rescales the coordinates and box vectors every step with a matrix $\boldsymbol{\mu}$, which has the effect of a first-order kinetic relaxation of the pressure towards a given reference pressure $\mathbf{P}_0$:

$$\frac{\mathrm{d}\mathbf{P}}{\mathrm{d}t} = \frac{\mathbf{P}_0 - \mathbf{P}}{\tau_p} \tag{3.28}$$

The scaling matrix $\boldsymbol{\mu}$ is given by

$$\mu_{ij} = \delta_{ij} - \frac{\Delta t}{3\,\tau_p}\beta_{ij}\{P_{0ij} - P_{ij}(t)\} \tag{3.29}$$

Here $\boldsymbol{\beta}$ is the isothermal compressibility of the system. In most cases this will be a diagonal matrix, with equal elements on the diagonal, the value of which is generally not known. It suffices to take a rough estimate because the value of $\boldsymbol{\beta}$ only influences the non-critical time constant of the pressure relaxation without affecting the average pressure itself. For water at 1 atm and 300 K $\beta = 4.6 \times 10^{-10}$ Pa^{-1} $= 4.6 \times 10^{-5}$ Bar^{-1}, which is 7.6×10^{-4} MD units (see chapter 2). Most other liquids have similar values. When scaling completely anisotropically, the system has to be rotated in order to obey the box restriction (3.1). This rotation is approximated in first order in the scaling, which is usually less than 10^{-4}. The actual scaling matrix $\boldsymbol{\mu}'$ is:

$$\boldsymbol{\mu}' = \begin{pmatrix} \mu_{xx} & \mu_{xy}+\mu_{yx} & \mu_{xz}+\mu_{zx} \\ 0 & \mu_{yy} & \mu_{yz}+\mu_{zy} \\ 0 & 0 & \mu_{zz} \end{pmatrix} \tag{3.30}$$

The velocities are neither scaled nor rotated.

In GROMACS, the Berendsen scaling can also be done isotropically, which means that instead of $\boldsymbol{P}$ a diagonal matrix with elements of size trace$(\boldsymbol{P})/3$ is used. For systems with interfaces, semi-isotropic scaling can be useful. In this case the x/y-directions are scaled isotropically and the z direction is scaled independently. The compressibility in the x/y or z-direction can be set to zero, to scale only in the other direction(s).

If you allow full anisotropic deformations and use constraints you might have to scale slower or decrease your timestep to avoid errors from the constraint algorithms.

Parrinello-Rahman pressure coupling

In cases where the fluctuations in pressure or volume are important *per se* (e.g. to calculate thermodynamic properties) it might at least theoretically be a problem that the exact ensemble is not well-defined for the weak coupling scheme.

For this reason, GROMACSalso supports constant-pressure simulations using the Parrinello-Rahman approach[21, 22], which is similar to the Nosé-Hoover temperature coupling. With the Parrinello-Rahman barostat, the box vectors as represented by the matrix $\boldsymbol{b}$ obey the matrix equation of motion[1]

---

[1] The box matrix representation $\boldsymbol{b}$ in GROMACScorresponds to the transpose of the box matrix representation $\boldsymbol{h}$ in the paper by Nosé and Klein. Because of this, some of our equations will look slightly different.

Files

`-f`	`traj.xtc`	Input	Generic trajectory: xtc trr trj gro g96 pdb
`-n`	`index.ndx`	Input	Index file
`-s`	`topol.tpr`	Input	Generic run input: tpr tpb tpa
`-o`	`potential.xvg`	Output	xvgr/xmgr file
`-oc`	`charge.xvg`	Output	xvgr/xmgr file
`-of`	`field.xvg`	Output	xvgr/xmgr file

Other options

`-h`	bool	`no`	Print help info and quit
`-nice`	int	`19`	Set the nicelevel
`-b`	time	`-1`	First frame (ps) to read from trajectory
`-e`	time	`-1`	Last frame (ps) to read from trajectory
`-dt`	time	`-1`	Only use frame when t MOD dt = first time (ps)
`-w`	bool	`no`	View output xvg, xpm, eps and pdb files
`-d`	string	`Z`	Take the normal on the membrane in direction X, Y or Z.
`-sl`	int	`10`	Calculate potential as function of boxlength, dividing the box in #nr slices.
`-cb`	int	`0`	Discard first #nr slices of box for integration
`-ce`	int	`0`	Discard last #nr slices of box for integration
`-tz`	real	`0`	Translate all coordinates <distance> in the direction of the box
`-spherical`	bool	`no`	Calculate spherical thingie

- Discarding slices for integration should not be necessary.

E.36 g_rama

g_rama selects the Phi/Psi dihedral combinations from your topology file and computes these as a function of time. Using simple Unix tools such as *grep* you can select out specific residues.

Files

`-f`	`traj.xtc`	Input	Generic trajectory: xtc trr trj gro g96 pdb
`-s`	`topol.tpr`	Input	Generic run input: tpr tpb tpa
`-o`	`rama.xvg`	Output	xvgr/xmgr file

Other options

`-h`	bool	`no`	Print help info and quit
`-nice`	int	`19`	Set the nicelevel
`-b`	time	`-1`	First frame (ps) to read from trajectory
`-e`	time	`-1`	Last frame (ps) to read from trajectory
`-dt`	time	`-1`	Only use frame when t MOD dt = first time (ps)
`-w`	bool	`no`	View output xvg, xpm, eps and pdb files

E.37 g_rdf

The structure of liquids can be studied by either neutron or X-ray scattering. The most common way to describe liquid structure is by a radial distribution function. However, this is not easy to obtain from a scattering experiment.

g_rdf calculates radial distribution functions in different ways. The normal method is around a (set of) particle(s), the other method is around the center of mass of a set of particles.

`-n`	`index.ndx`	Input, Opt.	Index file
`-o`	`ensemble.xtc`	Output	Generic trajectory: xtc trr trj gro g96 pdb

Other options

`-h`	bool	`no`	Print help info and quit
`-nice`	int	`19`	Set the nicelevel
`-temp`	real	`300`	Temperature in Kelvin
`-seed`	int	`-1`	Random seed, -1 generates a seed from time and pid
`-num`	int	`100`	Number of structures to generate
`-first`	int	`7`	First eigenvector to use (-1 is select)
`-last`	int	`-1`	Last eigenvector to use (-1 is till the last)

E.34 g_order

Compute the order parameter per atom for carbon tails. For atom i the vector i-1, i+1 is used together with an axis. The index file has to contain a group with all equivalent atoms in all tails for each atom the order parameter has to be calculated for. The program can also give all diagonal elements of the order tensor and even calculate the deuterium order parameter Scd (default). If the option -szonly is given, only one order tensor component (specified by the -d option) is given and the order parameter per slice is calculated as well. If -szonly is not selected, all diagonal elements and the deuterium order parameter is given.

Files

`-f`	`traj.xtc`	Input	Generic trajectory: xtc trr trj gro g96 pdb
`-n`	`index.ndx`	Input	Index file
`-s`	`topol.tpr`	Input	Generic run input: tpr tpb tpa
`-o`	`order.xvg`	Output	xvgr/xmgr file
`-od`	`deuter.xvg`	Output	xvgr/xmgr file
`-os`	`sliced.xvg`	Output	xvgr/xmgr file

Other options

`-h`	bool	`no`	Print help info and quit
`-nice`	int	`19`	Set the nicelevel
`-b`	time	`-1`	First frame (ps) to read from trajectory
`-e`	time	`-1`	Last frame (ps) to read from trajectory
`-dt`	time	`-1`	Only use frame when t MOD dt = first time (ps)
`-w`	bool	`no`	View output xvg, xpm, eps and pdb files
`-d`	enum	`z`	Direction of the normal on the membrane: `z`, `x` or `y`
`-sl`	int	`1`	Calculate order parameter as function of boxlength, dividing the box in #nr slices.
`-szonly`	bool	`no`	Only give Sz element of order tensor. (axis can be specified with -d)
`-unsat`	bool	`no`	Calculate order parameters for unsaturated carbons. Note that this cannot be mixed with normal order parameters.

E.35 g_potential

Compute the electrostatical potential across the box. The potential iscalculated by first summing the charges per slice and then integratingtwice of this charge distribution. Periodic boundaries are not taken into account. Reference of potential is taken to be the left side ofthe box. It's also possible to calculate the potential in sphericalcoordinates as function of r by calculating a charge distribution inspherical slices and twice integrating them. epsilon_r is taken as 1,2 is more appropriate in many cases

---

$$\frac{\mathrm{d}\boldsymbol{b}^2}{\mathrm{d}t^2} = V\boldsymbol{W}^{-1}\boldsymbol{b}'^{-1}\left(\boldsymbol{P}-\boldsymbol{P}_{ref}\right). \tag{3.31}$$

The volume of the box is denoted V, and $\boldsymbol{W}$ is a matrix parameter that determines the strength of the coupling. The matrices $\boldsymbol{P}$ and $\boldsymbol{P}_{ref}$ are the current and reference pressures, respectively.

The equations of motion for the particles are also changed, just as for the Nosé-Hoover coupling. In most cases you would combine the Parrinello-Rahman barostat with the Nosé-Hoover thermostat, but to keep it simple we only show the Parrinello-Rahman modification here:

$$\begin{aligned}
\frac{\mathrm{d}^2\boldsymbol{r}_i}{\mathrm{d}t^2} &= \frac{\boldsymbol{F}_i}{m_i} - \boldsymbol{M}\frac{\mathrm{d}\boldsymbol{r}_i}{\mathrm{d}t}, && (3.32)\\
\boldsymbol{M} &= \boldsymbol{b}^{-1}\left[\boldsymbol{b}\frac{\mathrm{d}\boldsymbol{b}'}{\mathrm{d}t} + \frac{\mathrm{d}\boldsymbol{b}}{\mathrm{d}t}\boldsymbol{b}'\right]\boldsymbol{b}'^{-1}. && (3.33)
\end{aligned}$$

The (inverse) mass parameter matrix $\boldsymbol{W}^{-1}$ determines the strength of the coupling, and how the box can be deformed. The box restriction (3.1) will be fulfilled automatically if the corresponding elements of $\boldsymbol{W}^{-1}$ are zero. Since the coupling strength also depends on the size of your box, we prefer to calculate it automatically in GROMACS. You only have to provide the approximate isothermal compressibilities $\boldsymbol{\beta}$ and the pressure time constant τ_p in the input file (L is the largest box matrix element):

$$\left(\boldsymbol{W}^{-1}\right)_{ij} = \frac{4\pi^2\beta_{ij}}{3\tau_p^2 L}. \tag{3.34}$$

Just as for the Nosé-Hoover thermostat, you should realize that the Parrinello-Rahman time constant is *not* equivalent to the relaxation time used in the Berendsen pressure coupling algorithm. In most cases you will need to use a 4–5 times larger time constant with Parrinello-Rahman coupling. If your pressure is very far from equilibrium, the Parrinello-Rahman coupling may result in very large box oscillations that could even crash your run. In that case you would have to increase the time constant, or (better) use the weak coupling scheme to reach the target pressure, and then switch to Parrinello-Rahman coupling once the system is in equilibrium.

Surface tension coupling

When a periodic system consists of more than one phase, separated by surfaces which are parallel to the xy-plane, the surface tension and the z-component of the pressure can be coupled to a pressure bath. Presently, this only works with the Berendsen pressure coupling algorithm in GROMACS. The average surface tension $\gamma(t)$ can be calculated from the difference between the normal and the lateral pressure:

$$\begin{aligned}
\gamma(t) &= \frac{1}{n}\int_0^{L_z}\left\{P_{zz}(z,t) - \frac{P_{xx}(z,t)+P_{yy}(z,t)}{2}\right\}\mathrm{d}z && (3.35)\\
&= \frac{L_z}{n}\left\{P_{zz}(t) - \frac{P_{xx}(t)+P_{yy}(t)}{2}\right\} && (3.36)
\end{aligned}$$

where L_z is the height of the box and n is the number of surfaces. The pressure in the z-direction is corrected by scaling the height of the box with μ_z:

$$\Delta P_{zz} = \frac{\Delta t}{\tau_p}\{P_{0zz} - P_{zz}(t)\} \tag{3.37}$$

$$\mu_{zz} = 1 + \beta_{zz}\Delta P_{zz} \tag{3.38}$$

This is similar to normal pressure coupling, except that the power of one third is missing. The pressure correction in the z-direction is then used to get the correct convergence for the surface tension to the reference value γ_0. The correction factor for the box-length in the x/y-direction is:

$$\mu_{x/y} = 1 + \frac{\Delta t}{2\,\tau_p}\beta_{x/y}\left(\frac{n\gamma_0}{\mu_{zz}L_z} - \left\{P_{zz}(t) + \Delta P_{zz} - \frac{P_{xx}(t) + P_{yy}(t)}{2}\right\}\right) \tag{3.39}$$

The value of β_{zz} is more critical than with normal pressure coupling. Normally an incorrect compressibility will just scale τ_p, but with surface tension coupling it affects the convergence of the surface tension. When β_{zz} is set to zero (constant box height), ΔP_z is also set to zero, which is necessary for obtaining the correct surface tension.

The complete update algorithm

The complete algorithm for the update of velocities and coordinates is given in Fig. 3.7. The SHAKE algorithm of step 4 is explained below.

GROMACS has a provision to ”freeze” (prevent motion of) selected particles, which must be defined as a ’freeze group’. This is implemented using a *freeze factor* $\boldsymbol{f}_g$, which is a vector, and differs for each *freezegroup* (see sec. 3.3). This vector contains only zero (freeze) or one (don’t freeze). When we take this freeze factor and the external acceleration $\boldsymbol{a}_h$ into account the update algorithm for the velocities becomes:

$$\boldsymbol{v}(t+\frac{\Delta t}{2}) \;=\; \boldsymbol{f}_g * \lambda * \left[\boldsymbol{v}(t-\frac{\Delta t}{2}) + \frac{\boldsymbol{F}(t)}{m}\Delta t + \boldsymbol{a}_h\Delta t\right] \tag{3.40}$$

where g and h are group indices which differ per atom.

3.4.7 Output step

The important output of the MD run is the *trajectory file* `name.trj` which contains particle coordinates and -optionally- velocities at regular intervals. Since the trajectory files are lengthy, one should not save every step! To retain all information it suffices to write a frame every 15 steps, since at least 30 steps are made per period of the highest frequency in the system, and Shannon’s sampling theorem states that two samples per period of the highest frequency in a band-limited signal contain all available information. But that still gives very long files! So, if the highest frequencies are not of interest, 10 or 20 samples per ps may suffice. Be aware of the distortion of high-frequency motions by the *stroboscopic effect*, called *aliasing*: higher frequencies are mirrored with respect to the sampling frequency and appear as lower frequencies.

Other options

`-h`	bool	`no`	Print help info and quit
`-nice`	int	`19`	Set the nicelevel
`-b`	time	`-1`	First frame (ps) to read from trajectory
`-e`	time	`-1`	Last frame (ps) to read from trajectory
`-dt`	time	`-1`	Only use frame when t MOD dt = first time (ps)
`-tu`	enum	`ps`	Time unit: `ps`, `fs`, `ns`, `us`, `ms`, `s`, `m` or `h`
`-w`	bool	`no`	View output xvg, xpm, eps and pdb files
`-type`	enum	`no`	Compute diffusion coefficient in one direction: `no`, `x`, `y` or `z`
`-lateral`	enum	`no`	Calculate the lateral diffusion in a plane perpendicular to: `no`, `x`, `y` or `z`
`-ngroup`	int	`1`	Number of groups to calculate MSD for
`-mw`	bool	`yes`	Mass weighted MSD
`-trestart`	time	`0`	Time between restarting points in trajectory (ps)
`-beginfit`	time	`0`	Start time for fitting the MSD (ps)
`-endfit`	time	`-1`	End time for fitting the MSD (ps), -1 is till end

E.32 g_nmeig

g_nmeig calculates the eigenvectors/values of a (Hessian) matrix, which can be calculated with `nmrun`. The eigenvectors are written to a trajectory file (`-v`). The structure is written first with t=0. The eigenvectors are written as frames with the eigenvector number as timestamp. The eigenvectors can be analyzed with `g_anaeig`. An ensemble of structures can be generated from the eigenvectors with `g_nmens`.

Files

`-f`	`hessian.mtx`	Input	Hessian matrix
`-s`	`topol.tpr`	Input	Structure+mass(db): tpr tpb tpa gro g96 pdb
`-o`	`eigenval.xvg`	Output	xvgr/xmgr file
`-v`	`eigenvec.trr`	Output	Full precision trajectory: trr trj

Other options

`-h`	bool	`no`	Print help info and quit
`-nice`	int	`19`	Set the nicelevel
`-m`	bool	`yes`	Divide elements of Hessian by product of sqrt(mass) of involved atoms prior to diagonalization. This should be used for ’Normal Modes’ analysis
`-first`	int	`1`	First eigenvector to write away
`-last`	int	`100`	Last eigenvector to write away

E.33 g_nmens

`g_nmens` generates an ensemble around an average structure in a subspace which is defined by a set of normal modes (eigenvectors). The eigenvectors are assumed to be mass-weighted. The position along each eigenvector is randomly taken from a Gaussian distribution with variance kT/eigenvalue.

By default the starting eigenvector is set to 7, since the first six normal modes are the translational and rotational degrees of freedom.

Files

`-v`	`eigenvec.trr`	Input	Full precision trajectory: trr trj
`-e`	`eigenval.xvg`	Input	xvgr/xmgr file
`-s`	`topol.tpr`	Input	Structure+mass(db): tpr tpb tpa gro g96 pdb

E.30 g_morph

g_morph does a linear interpolation of conformations in order to create intermediates. Of course these are completely unphysical, but that you may try to justify yourself. Output is in the form of a generic trajectory. The number of intermediates can be controlled with the -ninterm flag. The first and last flag correspond to the way of interpolating: 0 corresponds to input structure 1 while 1 corresponds to input strucutre 2. If you specify first < 0 or last > 1 extrapolation will be on the path from input structure x1 to x2. In general the coordinates of the intermediate x(i) out of N total intermidates correspond to:

x(i) = x1 + (first+(i/(N-1))*(last-first))*(x2-x1)

Finally the RMSD with respect to both input structures can be computed if explicitly selected (-or option). In that case an index file may be read to select what group RMS is computed from.

Files

`-f1`	`conf1.gro`	Input	Generic structure: gro g96 pdb tpr tpb tpa
`-f2`	`conf2.gro`	Input	Generic structure: gro g96 pdb tpr tpb tpa
`-o`	`interm.xtc`	Output	Generic trajectory: xtc trr trj gro g96 pdb
`-or`	`rms-interm.xvg`	Output, Opt.	xvgr/xmgr file
`-n`	`index.ndx`	Input, Opt.	Index file

Other options

`-h`	bool	`no`	Print help info and quit
`-nice`	int	`0`	Set the nicelevel
`-w`	bool	`no`	View output xvg, xpm, eps and pdb files
`-ninterm`	int	`11`	Number of intermediates
`-first`	real	`0`	Corresponds to first generated structure (0 is input x0, see above)
`-last`	real	`1`	Corresponds to last generated structure (1 is input x1, see above)
`-fit`	bool	`yes`	Do a least squares fit of the second to the first structure before interpolating

E.31 g_msd

g_msd computes the mean square displacement (MSD) of atoms from their initial positions. This provides an easy way to compute the diffusion constant using the Einstein relation. The diffusion constant is calculated by least squares fitting a straight line through the MSD from `-beginfit` to `-endfit`. An error estimate given, which is the difference of the diffusion coefficients obtained from fits over the two halfs of the fit interval.

Option `-mol` plots the MSD for molecules, this implies `-mw`, i.e. for each inidividual molecule an diffusion constant is computed. When using an index file, it should contain molecule numbers instead of atom numbers. Using this option one also gets an accurate error estimate based on the statistics between individual molecules. Since one usually is interested in self-diffusion at infinite dilution this is probably the most useful number.

Files

`-f`	`traj.xtc`	Input	Generic trajectory: xtc trr trj gro g96 pdb
`-s`	`topol.tpr`	Input	Structure+mass(db): tpr tpb tpa gro g96 pdb
`-n`	`index.ndx`	Input, Opt.	Index file
`-o`	`msd.xvg`	Output	xvgr/xmgr file
`-mol`	`diff_mol.xvg`	Output, Opt.	xvgr/xmgr file

THE UPDATE ALGORITHM

Given:
Positions $\boldsymbol{r}$ of all atoms at time t
Velocities $\boldsymbol{v}$ of all atoms at time $t - \frac{\Delta t}{2}$
Accelerations $\boldsymbol{F}/m$ on all atoms at time t.
(Forces are computed disregarding any constraints)
Total kinetic energy and virial
$\Downarrow$
1. Compute the scaling factors λ and μ
according to eqns. 3.23 and 3.29
$\Downarrow$
2. Update and scale velocities: $\boldsymbol{v}' = \lambda(\boldsymbol{v} + \boldsymbol{a}\Delta t)$
$\Downarrow$
3. Compute new unconstrained coordinates: $\boldsymbol{r}' = \boldsymbol{r} + \boldsymbol{v}'\Delta t$
$\Downarrow$
4. Apply constraint algorithm to coordinates: constrain($\boldsymbol{r}^{'} \rightarrow \boldsymbol{r}''; \boldsymbol{r}$)
$\Downarrow$
5. Correct velocities for constraints: $\boldsymbol{v} = (\boldsymbol{r}'' - \boldsymbol{r})/\Delta t$
$\Downarrow$
6. Scale coordinates and box: $\boldsymbol{r} = \mu\boldsymbol{r}''; \boldsymbol{b} = \mu\boldsymbol{b}$

Figure 3.7: The MD update algorithm

3.5 Shell molecular dynamics

GROMACS can simulate polarizability using the shell model of Dick and Overhauser [23]. In such models a shell particle representing the electronic degrees of freedom is attached to a nucleus by a spring. The potential energy is minimized with respect to the shell position at every step of the simulation (see below). Succesfull applications of shell models in GROMACS have been published for N_2 [24] and water [25].

3.5.1 Optimization of the shell positions

The force $\boldsymbol{F}_S$ on a shell particle S can be decomposed into two components:

$$\boldsymbol{F}_S \;=\; \boldsymbol{F}_{bond} + \boldsymbol{F}_{nb} \tag{3.41}$$

where $\boldsymbol{F}_{bond}$ denotes the component representing the polarization energy, usually represented by a harmonic potential and $\boldsymbol{F}_{nb}$ is the sum of Coulomb and Van der Waals interactions. If we assume that $\boldsymbol{F}_{nb}$ is almost constant we can analytically derive the optimal position of the shell, i.e. where $\boldsymbol{F}_S = 0$. If we have the shell S connected to atom A we have

$$\boldsymbol{F}_{bond} \;=\; k_b\left(\boldsymbol{x}_S - \boldsymbol{x}_A\right) \tag{3.42}$$

In an iterative solver, we have positions $\boldsymbol{x}_S(n)$ where n is the iteration count. We now have it iteration n:

$$\boldsymbol{F}_{nb} \;=\; \boldsymbol{F}_S - k_b\left(\boldsymbol{x}_S(n) - \boldsymbol{x}_A\right) \tag{3.43}$$

and the optimal position for the shells $x_S(n+1)$ thus follows from

$$\boldsymbol{F}_S - k_b\left(\boldsymbol{x}_S(n) - \boldsymbol{x}_A\right) + k_b\left(\boldsymbol{x}_S(n+1) - \boldsymbol{x}_A\right) = 0 \tag{3.44}$$

if we write

$$\Delta\boldsymbol{x}_S = \boldsymbol{x}_S(n+1) - \boldsymbol{x}_S(n) \tag{3.45}$$

we finally obtain

$$\Delta\boldsymbol{x}_S = \boldsymbol{F}_S/k_b \tag{3.46}$$

which then yields the algorithm to compute the next trial in the optimization of shell positions:

$$\boldsymbol{x}_S(n+1) \;=\; \boldsymbol{x}_S(n) + \boldsymbol{F}_S/k_b \tag{3.47}$$

3.6 Constraint algorithms

Constraints can be imposed in GROMACS using LINCS (default) or the traditional SHAKE method.

E.28 g_mdmat

g_mdmat makes distance matrices consisting of the smallest distance between residue pairs. With -frames these distance matrices can be stored as a function of time, to be able to see differences in tertiary structure as a funcion of time. If you choose your options unwise, this may generate a large output file. Default only an averaged matrix over the whole trajectory is output. Also a count of the number of different atomic contacts between residues over the whole trajectory can be made. The output can be processed with xpm2ps to make a PostScript (tm) plot.

Files

`-f`	`traj.xtc`	Input	Generic trajectory: xtc trr trj gro g96 pdb
`-s`	`topol.tpr`	Input	Structure+mass(db): tpr tpb tpa gro g96 pdb
`-n`	`index.ndx`	Input, Opt.	Index file
`-mean`	`dm.xpm`	Output	X PixMap compatible matrix file
`-frames`	`dmf.xpm`	Output, Opt.	X PixMap compatible matrix file
`-no`	`num.xvg`	Output, Opt.	xvgr/xmgr file

Other options

`-h`	bool	`no`	Print help info and quit
`-nice`	int	`19`	Set the nicelevel
`-b`	time	`-1`	First frame (ps) to read from trajectory
`-e`	time	`-1`	Last frame (ps) to read from trajectory
`-dt`	time	`-1`	Only use frame when t MOD dt = first time (ps)
`-t`	real	`1.5`	trunc distance
`-nlevels`	int	`40`	Discretize distance in # levels

E.29 g_mindist

g_mindist computes the distance between one group and a number of other groups. Both the minimum distance and the number of contacts within a given distance are written to two separate output files.

With option `-pi` the minimum distance of a group to its periodic image is plotted. This is useful for checking if a protein has seen its periodic image during a simulation. Only one shift in each direction is considered, giving a total of 26 shifts. It also plots the maximum distance within the group and the lengths of the three box vectors. This option is very slow.

Files

`-f`	`traj.xtc`	Input	Generic trajectory: xtc trr trj gro g96 pdb
`-s`	`topol.tpr`	Input, Opt.	Structure+mass(db): tpr tpb tpa gro g96 pdb
`-n`	`index.ndx`	Input, Opt.	Index file
`-od`	`mindist.xvg`	Output	xvgr/xmgr file
`-on`	`numcont.xvg`	Output	xvgr/xmgr file
`-o`	`atm-pair.out`	Output	Generic output file

Other options

`-h`	bool	`no`	Print help info and quit
`-nice`	int	`19`	Set the nicelevel
`-b`	time	`-1`	First frame (ps) to read from trajectory
`-e`	time	`-1`	Last frame (ps) to read from trajectory
`-dt`	time	`-1`	Only use frame when t MOD dt = first time (ps)
`-w`	bool	`no`	View output xvg, xpm, eps and pdb files
`-matrix`	bool	`no`	Calculate half a matrix of group-group distances
`-d`	real	`0.6`	Distance for contacts
`-pi`	bool	`no`	Calculate minimum distance with periodic images

8. Average Calpha-Calpha dihedral angle (file phi-ahx.xvg).
9. Average Phi and Psi angles (file phipsi.xvg).
10. Ellipticity at 222 nm according to *Hirst and Brooks*

Files

`-s`	`topol.tpr`	Input	Generic run input: tpr tpb tpa
`-n`	`index.ndx`	Input	Index file
`-f`	`traj.xtc`	Input	Generic trajectory: xtc trr trj gro g96 pdb
`-to`	`gtraj.g87`	Output, Opt.	Gromos-87 ASCII trajectory format
`-cz`	`zconf.gro`	Output	Generic structure: gro g96 pdb
`-co`	`waver.gro`	Output	Generic structure: gro g96 pdb

Other options

`-h`	bool	`no`	Print help info and quit
`-nice`	int	`19`	Set the nicelevel
`-b`	time	`-1`	First frame (ps) to read from trajectory
`-e`	time	`-1`	Last frame (ps) to read from trajectory
`-dt`	time	`-1`	Only use frame when t MOD dt = first time (ps)
`-w`	bool	`no`	View output xvg, xpm, eps and pdb files
`-r0`	int	`1`	The first residue number in the sequence
`-q`	bool	`no`	Check at every step which part of the sequence is helical
`-F`	bool	`yes`	Toggle fit to a perfect helix
`-db`	bool	`no`	Print debug info
`-ev`	bool	`no`	Write a new 'trajectory' file for ED
`-ahxstart`	int	`0`	First residue in helix
`-ahxend`	int	`0`	Last residue in helix

E.27 g_lie

g_lie computes a free energy estimate based on an energy analysis from. One needs an energy file with the following components: Coul (A-B) LJ-SR (A-B) etc.

Files

`-f`	`ener.edr`	Input	Generic energy: edr ene
`-o`	`lie.xvg`	Output	xvgr/xmgr file

Other options

`-h`	bool	`no`	Print help info and quit
`-nice`	int	`19`	Set the nicelevel
`-b`	time	`-1`	First frame (ps) to read from trajectory
`-e`	time	`-1`	Last frame (ps) to read from trajectory
`-dt`	time	`-1`	Only use frame when t MOD dt = first time (ps)
`-w`	bool	`no`	View output xvg, xpm, eps and pdb files
`-Elj`	real	`0`	Lennard-Jones interaction between ligand and solvent
`-Eqq`	real	`0`	Coulomb interaction between ligand and solvent
`-Clj`	real	`0.181`	Factor in the LIE equation for Lennard-Jones component of energy
`-Cqq`	real	`0`	Factor in the LIE equation for Coulomb component of energy
`-ligand`	string	`none`	Name of the ligand in the energy file

3.6.1 SHAKE

The SHAKE [26] algorithm changes a set of unconstrained coordinates $\boldsymbol{r}^{'}$ to a set of coordinates $\boldsymbol{r}''$ that fulfill a list of distance constraints, using a set $\boldsymbol{r}$ as reference:

$$\text{SHAKE}(\boldsymbol{r}^{'} \rightarrow \boldsymbol{r}''; \boldsymbol{r})$$

This action is consistent with solving a set of Lagrange multipliers in the constrained equations of motion. SHAKE needs a *tolerance* `TOL`; it will continue until all constraints are satisfied within a *relative* tolerance `TOL`. An error message is given if SHAKE cannot reset the coordinates because the deviation is too large, or if a given number of iterations is surpassed.

Assume the equations of motion must fulfill K holonomic constraints, expressed as

$$\sigma_k(\boldsymbol{r}_1 \dots \boldsymbol{r}_N) = 0; \quad k = 1 \dots K \tag{3.48}$$

(e.g. $(\boldsymbol{r}_1 - \boldsymbol{r}_2)^2 - b^2 = 0$). Then the forces are defined as

$$-\frac{\partial}{\partial \boldsymbol{r}_i}\left(V + \sum_{k=1}^{K} \lambda_k \sigma_k\right) \tag{3.49}$$

where λ_k are Lagrange multipliers which must be solved to fulfill the constraint equations. The second part of this sum determines the *constraint forces* $\boldsymbol{G}_i$, defined by

$$\boldsymbol{G}_i = -\sum_{k=1}^{K} \lambda_k \frac{\partial \sigma_k}{\partial \boldsymbol{r}_i} \tag{3.50}$$

The displacement due to the constraint forces in the leap frog or Verlet algorithm is equal to $(\boldsymbol{G}_i/m_i)(\Delta t)^2$. Solving the Lagrange multipliers (and hence the displacements) requires the solution of a set of coupled equations of the second degree. These are solved iteratively by SHAKE. For the special case of rigid water molecules, that often make up more than 80% of the simulation system we have implemented the SETTLE algorithm [27] (sec. 5.4).

3.6.2 LINCS

The LINCS algorithm

LINCS is an algorithm that resets bonds to their correct lengths after an unconstrained update [28]. The method is non-iterative, as it always uses two steps. Although LINCS is based on matrices, no matrix-matrix multiplications are needed. The method is more stable and faster than SHAKE, but it can only be used with bond constraints and isolated angle constraints, such as the proton angle in OH. Because of its stability LINCS is especially useful for Brownian dynamics. LINCS has two parameters, which are explained in the subsection parameters.

The LINCS formulas

We consider a system of N particles, with positions given by a $3N$ vector $\boldsymbol{r}(t)$. For molecular dynamics the equations of motion are given by Newton's law

$$\frac{\mathrm{d}^2\boldsymbol{r}}{\mathrm{d}t^2} = \boldsymbol{M}^{-1}\boldsymbol{F} \tag{3.51}$$

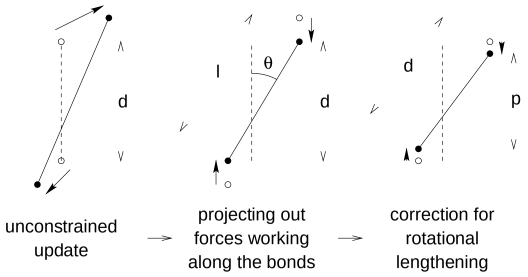


Figure 3.8: The three position updates needed for one time step. The dashed line is the old bond of length d, the solid lines are the new bonds. $l = d\cos\theta$ and $p = (2d^2 - l^2)^{\frac{1}{2}}$.

where $\boldsymbol{F}$ is the $3N$ force vector and $\boldsymbol{M}$ is a $3N \times 3N$ diagonal matrix, containing the masses of the particles. The system is constrained by K time-independent constraint equations

$$g_i(\boldsymbol{r}) = |\boldsymbol{r}_{i_1} - \boldsymbol{r}_{i_2}| - d_i = 0 \qquad i = 1,\dots,K \tag{3.52}$$

In a numerical integration scheme LINCS is applied after an unconstrained update, just like SHAKE. The algorithm works in two steps (see figure Fig. 3.8). In the first step the projections of the new bonds on the old bonds are set to zero. In the second step a correction is applied for the lengthening of the bonds due to rotation. The numerics for the first step and the second step are very similar. A complete derivation of the algorithm can be found in [28]. Only a short description of the first step is given here.

A new notation is introduced for the gradient matrix of the constraint equations which appears on the right hand side of the equation

$$B_{hi} = \frac{\partial g_h}{\partial r_i} \tag{3.53}$$

Notice that $\boldsymbol{B}$ is a $K \times 3N$ matrix, it contains the directions of the constraints. The following equation shows how the new constrained coordinates $\boldsymbol{r}_{n+1}$ are related to the unconstrained coordinates $\boldsymbol{r}_{n+1}^{unc}$

$$\begin{aligned} \boldsymbol{r}_{n+1} = (\boldsymbol{I} - \boldsymbol{T}_n\boldsymbol{B}_n)\boldsymbol{r}_{n+1}^{unc} + \boldsymbol{T}_n\boldsymbol{d} = \\ \boldsymbol{r}_{n+1}^{unc} - \boldsymbol{M}^{-1}\boldsymbol{B}_n(\boldsymbol{B}_n\boldsymbol{M}^{-1}\boldsymbol{B}_n^T)^{-1}(\boldsymbol{B}_n\boldsymbol{r}_{n+1}^{unc} - \boldsymbol{d}) \end{aligned} \tag{3.54}$$

where $\boldsymbol{T} = \boldsymbol{M}^{-1}\boldsymbol{B}^T(\boldsymbol{B}\boldsymbol{M}^{-1}\boldsymbol{B}^T)^{-1}$. The derivation of this equation from eqns. 3.51 and 3.52 can be found in [28].

This first step does not set the real bond lengths to the prescribed lengths, but the projection of the new bonds onto the old directions of the bonds. To correct for the rotation of bond i, the projection of the bond on the old direction is set to

$$p_i = \sqrt{2d_i^2 - l_i^2} \tag{3.55}$$

where l_i is the bond length after the first projection. The corrected positions are

$$\boldsymbol{r}_{n+1}^* = (\boldsymbol{I} - \boldsymbol{T}_n\boldsymbol{B}_n)\boldsymbol{r}_{n+1} + \boldsymbol{T}_n\boldsymbol{p} \tag{3.56}$$

---

insertion into hydrogen bonds. Ordering is identical to that in `-hbn` index file.
`-da`: write out the number of donors and acceptors analyzed for each timeframe. This is especially usefull when using `-shell`.

Files

`-f`	`traj.xtc`	Input	Generic trajectory: xtc trr trj gro g96 pdb
`-s`	`topol.tpr`	Input	Generic run input: tpr tpb tpa
`-n`	`index.ndx`	Input, Opt.	Index file
`-g`	`hbond.log`	Output, Opt.	Log file
`-sel`	`select.ndx`	Input, Opt.	Index file
`-num`	`hbnum.xvg`	Output	xvgr/xmgr file
`-ac`	`hbac.xvg`	Output, Opt.	xvgr/xmgr file
`-dist`	`hbdist.xvg`	Output, Opt.	xvgr/xmgr file
`-ang`	`hbang.xvg`	Output, Opt.	xvgr/xmgr file
`-hx`	`hbhelix.xvg`	Output, Opt.	xvgr/xmgr file
`-hbn`	`hbond.ndx`	Output, Opt.	Index file
`-hbm`	`hbmap.xpm`	Output, Opt.	X PixMap compatible matrix file
`-da`	`danum.xvg`	Output, Opt.	xvgr/xmgr file

Other options

`-h`	bool	`no`	Print help info and quit
`-nice`	int	`19`	Set the nicelevel
`-b`	time	`-1`	First frame (ps) to read from trajectory
`-e`	time	`-1`	Last frame (ps) to read from trajectory
`-dt`	time	`-1`	Only use frame when t MOD dt = first time (ps)
`-ins`	bool	`no`	Analyze solvent insertion
`-a`	real	`60`	Cutoff angle (degrees, Donor - Hydrogen - Acceptor)
`-r`	real	`0.25`	Cutoff radius (nm, Hydrogen - Acceptor)
`-abin`	real	`1`	Binwidth angle distribution (degrees)
`-rbin`	real	`0.005`	Binwidth distance distribution (nm)
`-nitacc`	bool	`yes`	Regard nitrogen atoms as acceptors
`-shell`	real	`-1`	when > 0, only calculate hydrogen bonds within # nm shell around one particle

E.26 g_helix

g_helix computes all kind of helix properties. First, the peptide is checked to find the longest helical part. This is determined by Hydrogen bonds and Phi/Psi angles. That bit is fitted to an ideal helix around the Z-axis and centered around the origin. Then the following properties are computed:

1. Helix radius (file radius.xvg). This is merely the RMS deviation in two dimensions for all Calpha atoms. it is calced as sqrt((SUM i(x^2(i)+y^2(i)))/N), where N is the number of backbone atoms. For an ideal helix the radius is 0.23 nm
2. Twist (file twist.xvg). The average helical angle per residue is calculated. For alpha helix it is 100 degrees, for 3-10 helices it will be smaller, for 5-helices it will be larger.
3. Rise per residue (file rise.xvg). The helical rise per residue is plotted as the difference in Z-coordinate between Ca atoms. For an ideal helix this is 0.15 nm
4. Total helix length (file len-ahx.xvg). The total length of the helix in nm. This is simply the average rise (see above) times the number of helical residues (see below).
5. Number of helical residues (file n-ahx.xvg). The title says it all.
6. Helix Dipole, backbone only (file dip-ahx.xvg).
7. RMS deviation from ideal helix, calculated for the Calpha atoms only (file rms-ahx.xvg).

Other options

`-h`	bool	`no`	Print help info and quit
`-nice`	int	`19`	Set the nicelevel
`-b`	time	`-1`	First frame (ps) to read from trajectory
`-e`	time	`-1`	Last frame (ps) to read from trajectory
`-dt`	time	`-1`	Only use frame when t MOD dt = first time (ps)
`-w`	bool	`no`	View output xvg, xpm, eps and pdb files
`-d`	string	`Z`	Take the normal on the membrane in direction X, Y or Z.
`-sl`	int	`0`	Calculate order parameter as function of boxlength, dividing the box in #nr slices.

- The program assigns whole water molecules to a slice, based on the firstatom of three in the index file group. It assumes an order O,H,H.Name is not important, but the order is. If this demand is not met,assigning molecules to slices is different.

E.25 g_hbond

g_hbond computes and analyzes hydrogen bonds. Hydrogen bonds are determined based on cutoffs for the angle Donor - Hydrogen - Acceptor (zero is extended) and the distance Hydrogen - Acceptor. OH and NH groups are regarded as donors, O is an acceptor always, N is an acceptor by default, but this can be switched using `-nitacc`. Dummy hydrogen atoms are assumed to be connected to the first preceding non-hydrogen atom.

You need to specify two groups for analysis, which must be either identical or non-overlapping. All hydrogen bonds between the two groups are analyzed.

If you set -shell, you will be asked for an additional index group which should contain exactly one atom. In this case, only hydrogen bonds between atoms within the shell distance from the one atom are considered.

It is also possible to analyse specific hydrogen bonds with `-sel`. This index file must contain a group of atom triplets Donor Hydrogen Acceptor, in the following way:

```
[ selected ]
20 21 24
25 26 29
1 3 6
```

Note that the triplets need not be on separate lines. Each atom triplet specifies a hydrogen bond to be analyzed, note also that no check is made for the types of atoms.

`-ins` turns on computing solvent insertion into hydrogen bonds. In this case an additional group must be selected, specifying the solvent molecules.

Output:
`-num`: number of hydrogen bonds as a function of time.
`-ac`: average over all autocorrelations of the existence functions (either 0 or 1) of all hydrogen bonds.
`-dist`: distance distribution of all hydrogen bonds.
`-ang`: angle distribution of all hydrogen bonds.
`-hx`: the number of n-n+i hydrogen bonds as a function of time where n and n+i stand for residue numbers and i ranges from 0 to 6. This includes the n-n+3, n-n+4 and n-n+5 hydrogen bonds associated with helices in proteins.
`-hbn`: all selected groups, donors, hydrogens and acceptors for selected groups, all hydrogen bonded atoms from all groups and all solvent atoms involved in insertion.
`-hbm`: existence matrix for all hydrogen bonds over all frames, this also contains information on solvent

---

This correction for rotational effects is actually an iterative process, but during MD only one iteration is applied. The relative constraint deviation after this procedure will be less than 0.0001 for every constraint. In energy minimization this might not be accurate enough, so the number of iterations is equal to the order of the expansion (see below).

Half of the CPU time goes to inverting the constraint coupling matrix $\boldsymbol{B}_n\boldsymbol{M}^{-1}\boldsymbol{B}_n^T$, which has to be done every time step. This $K \times K$ matrix has $1/m_{i_1} + 1/m_{i_2}$ on the diagonal. The off-diagonal elements are only non-zero when two bonds are connected, then the element is $\cos\phi/m_c$, where m_c is the mass of the atom connecting the two bonds and ϕ is the angle between the bonds.

The matrix $\boldsymbol{T}$ is inverted through a power expansion. A $K \times K$ matrix $\boldsymbol{S}$ is introduced which is the inverse square root of the diagonal of $\boldsymbol{B}_n\boldsymbol{M}^{-1}\boldsymbol{B}_n^T$. This matrix is used to convert the diagonal elements of the coupling matrix to one

$$\begin{aligned}(\boldsymbol{B}_n\boldsymbol{M}^{-1}\boldsymbol{B}_n^T)^{-1} &= \boldsymbol{S}\boldsymbol{S}^{-1}(\boldsymbol{B}_n\boldsymbol{M}^{-1}\boldsymbol{B}_n^T)^{-1}\boldsymbol{S}^{-1}\boldsymbol{S} \\ &= \boldsymbol{S}(\boldsymbol{S}\boldsymbol{B}_n\boldsymbol{M}^{-1}\boldsymbol{B}_n^T\boldsymbol{S})^{-1}\boldsymbol{S} = \boldsymbol{S}(\boldsymbol{I} - \boldsymbol{A}_n)^{-1}\boldsymbol{S}\end{aligned} \tag{3.57}$$

The matrix $\boldsymbol{A}_n$ is symmetric and sparse and has zeros on the diagonal. Thus a simple trick can be used to calculate the inverse

$$(\boldsymbol{I} - \boldsymbol{A}_n)^{-1} = \boldsymbol{I} + \boldsymbol{A}_n + \boldsymbol{A}_n^2 + \boldsymbol{A}_n^3 + \ldots \tag{3.58}$$

This inversion method is only valid if the absolute values of all the eigenvalues of $\boldsymbol{A}_n$ are smaller than one. In molecules with only bond constraints the connectivity is so low that this will always be true, even if ring structures are present. Problems can arise in angle-constrained molecules. By constraining angles with additional distance constraints multiple small ring structures are introduced. This gives a high connectivity, leading to large eigenvalues. Therefore LINCS should NOT be used with coupled angle-constraints.

The LINCS Parameters

The accuracy of LINCS depends on the number of matrices used in the expansion eqn. 3.58. For MD calculations a fourth order expansion is enough. For Brownian dynamics with large time steps an eighth order expansion may be necessary. The order is a parameter in the input file for `mdrun`. The implementation of LINCS is done in such a way that the algorithm will never crash. Even when it is impossible to to reset the constraints LINCS will generate a conformation which fulfills the constraints as well as possible. However, LINCS will generate a warning when in one step a bond rotates over more than a predefined angle. This angle is set by the user in the input file for `mdrun`.

3.7 Simulated Annealing

The well known simulated annealing (SA) protocol is implemented in a simple way into GROMACS. A modification of the temperature coupling scheme is used as a very basic implementation of the SA algorithm. The method works as follows: the reference temperature for coupling T_0 (eqn. 3.22) is not constant but can be varied linearly:

$$T_0(\text{step}) = T_0 * (\lambda_0 + \Delta\lambda * \text{step}) \tag{3.59}$$

if λ_0 = 1 and $\Delta\lambda$ is 0 this is the plain MD algorithm. Note that for standard SA $\Delta\lambda$ must be negative. When T_0(step) < 0 it is set to 0, as negative temperatures do not have a physical meaning. This "feature" allows for an annealing strategy in which at first the temperature is scaled down linearly until 0 K, and when more steps are taken the simulation proceeds at 0 K. Since the weak coupling scheme does not couple instantaneously, the actual temperature will always be slightly higher than 0 K.

3.8 Stochastic Dynamics

Stochastic or velocity Langevin dynamics adds a friction and a noise term to Newton's equations of motion:

$$m_i \frac{\mathrm{d}^2 \boldsymbol{r}_i}{\mathrm{d}t^2} = -m_i \xi_i \frac{\mathrm{d}\boldsymbol{r}_i}{\mathrm{d}t} + \boldsymbol{F}_i(\boldsymbol{r}) + \mathring{\boldsymbol{r}}_i \tag{3.60}$$

where ξ_i is the friction constant [1/ps] and $\mathring{\boldsymbol{r}}_i(t)$ is a noise process with $\langle \mathring{r}_i(t)\ \mathring{r}_j(t+s)\rangle = 2m_i\xi_i k_B T\delta(s)\delta_{ij}$. When $1/\xi_i$ is large compared to the time scales present in the system, one could see stochastic dynamics as molecular dynamics with stochastic temperature-coupling. The advantage compared to MD with Berendsen temperature-coupling is that in case of SD the generated ensemble is known. For vacuum simulations there is the additional advantage that there is no accumulation of errors for the overall translational and rotational degrees of freedom. When $1/\xi_i$ is small compared to the time scales present in the system, the dynamics will be completely different from MD, but the sampling is still correct.

GROMACS uses a complicated third-order leap-frog algorithm [29] to integrate equation (3.60). When constraints are present in the system, two constraint steps are performed per time step. The kinetic energy is computed at the whole time step, this is done by averaging the velocities at minus and plus a half time step, with a correction for the friction:

$$\boldsymbol{v}(t) = \frac{1}{2}\left(\boldsymbol{v}\left(t - \frac{\Delta t}{2}\right) + \boldsymbol{v}\left(t + \frac{\Delta t}{2}\right)\right)\left(e^{-\xi\Delta t} + \sqrt{2}\left(1 - e^{-\xi\Delta t}\right)\right) \tag{3.61}$$

Exact continuation of a stochastic dynamics simulation is not possible, since apart from the coordinates and the velocities one random term of the previous step in required, however, the error will be very small.

3.9 Brownian Dynamics

In the limit of high friction stochastic dynamics reduces to Brownian dynamics, also called position Langevin dynamics. This applies to over-damped systems, i.e. systems in which the inertia effects are negligible. The equation is:

$$\frac{\mathrm{d}\boldsymbol{r}_i}{\mathrm{d}t} = \frac{1}{\gamma_i}\boldsymbol{F}_i(\boldsymbol{r}) + \mathring{\boldsymbol{r}}_i \tag{3.62}$$

where γ_i is the friction coefficient [amu/ps] and $\mathring{\boldsymbol{r}}_i(t)$ is a noise process with $\langle \mathring{r}_i(t)\ \mathring{r}_j(t+s)\rangle = 2\delta(s)\delta_{ij}k_B T/\gamma_i$. In GROMACS the equations are integrated with a simple, explicit scheme:

$$\boldsymbol{r}_i(t+\Delta t) = \boldsymbol{r}_i(t) + \frac{\Delta t}{\gamma_i}\boldsymbol{F}_i(\boldsymbol{r}(t)) + \sqrt{2k_B T\frac{\Delta t}{\gamma_i}}\,\boldsymbol{r}_i^G \tag{3.63}$$

`-nmol`	int	`1`	Number of molecules in your sample: the energies are divided by this number
`-ndf`	int	`3`	Number of degrees of freedom per molecule. Necessary for calculating the heat capacity
`-fluc`	bool	`no`	Calculate autocorrelation of energy fluctuations rather than energy itself
`-acflen`	int	`-1`	Length of the ACF, default is half the number of frames
`-normalize`	bool	`yes`	Normalize ACF
`-P`	enum	`0`	Order of Legendre polynomial for ACF (0 indicates none): `0`, `1`, `2` or `3`
`-fitfn`	enum	`none`	Fit function: `none`, `exp`, `aexp`, `exp_exp` or `vac`
`-ncskip`	int	`0`	Skip N points in the output file of correlation functions
`-beginfit`	real	`0`	Time where to begin the exponential fit of the correlation function
`-endfit`	real	`-1`	Time where to end the exponential fit of the correlation function, -1 is till the end

E.23 g_gyrate

g_gyrate computes the radius of gyration of a group of atoms and the radii of gyration about the x, y and z axes,as a function of time. The atoms are explicitly mass weighted.

Files

`-f`	`traj.xtc`	Input	Generic trajectory: xtc trr trj gro g96 pdb
`-s`	`topol.tpr`	Input	Structure+mass(db): tpr tpb tpa gro g96 pdb
`-o`	`gyrate.xvg`	Output	xvgr/xmgr file
`-n`	`index.ndx`	Input, Opt.	Index file

Other options

`-h`	bool	`no`	Print help info and quit
`-nice`	int	`19`	Set the nicelevel
`-b`	time	`-1`	First frame (ps) to read from trajectory
`-e`	time	`-1`	Last frame (ps) to read from trajectory
`-dt`	time	`-1`	Only use frame when t MOD dt = first time (ps)
`-w`	bool	`no`	View output xvg, xpm, eps and pdb files
`-q`	bool	`no`	Use absolute value of the charge of an atom as weighting factor instead of mass
`-p`	bool	`no`	Calculate the radii of gyration about the principal axes.

E.24 g_h2order

Compute the orientation of water molecules with respect to the normal of the box. The program determines the average cosine of the angle between de dipole moment of water and an axis of the box. The box is divided in slices and the average orientation per slice is printed. Each water molecule is assigned to a slice, per time frame, based on the position of the oxygen. When -nm is used the angle between the water dipole and the axis from the center of mass to the oxygen is calculated instead of the angle between the dipole and a box axis.

Files

`-f`	`traj.xtc`	Input	Generic trajectory: xtc trr trj gro g96 pdb
`-n`	`index.ndx`	Input	Index file
`-nm`	`index.ndx`	Input, Opt.	Index file
`-s`	`topol.tpr`	Input	Generic run input: tpr tpb tpa
`-o`	`order.xvg`	Output	xvgr/xmgr file

`-lj`	bool	`yes`	calculate Lennard-Jones SR energies
`-lj14`	bool	`no`	calculate Lennard-Jones 1-4 energies
`-bham`	bool	`no`	calculate Buckingham energies
`-free`	bool	`yes`	calculate free energy
`-temp`	real	`300`	reference temperature for free energy calculation

E.22 g_energy

g_energy extracts energy components or distance restraint data from an energy file. The user is prompted to interactively select the energy terms she wants.

When the `-viol` option is set, the time averaged violations are plotted and the running time-averaged and instantaneous sum of violations are recalculated. Additionally running time-averaged and instantaneous distances between selected pairs can be plotted with the `-pairs` option.

Average and RMSD are calculated with full precision from the simulation (see printed manual). Drift is calculated by performing a LSQ fit of the data to a straight line. Total drift is drift multiplied by total time.

With `-fee` a free energy estimate is calculated using the formula: G = -ln < e ˆ(E/kT) > * kT, where k is Boltzmann's constant, T is set by `-fetemp` and the average is over the ensemble (or time in a trajectory). Note that this is in principle only correct when averaging over the whole (Boltzmann) ensemble and using the potential energy. This also allows for an entropy estimate using G = H - T S, where H is the enthalpy (H = U + p V) and S entropy.

When a second energy file is specified (`-f2`), a free energy difference is calculated dF = -kT ln < e ˆ(EB-EA)/kT >A , where EA and EB are the energies from the first and second energy files, and the average is over the ensemble A. **NOTE** that the energies must both be calculated from the same trajectory.

Files

`-f`	`ener.edr`	Input	Generic energy: edr ene
`-f2`	`ener.edr`	Input, Opt.	Generic energy: edr ene
`-s`	`topol.tpr`	Input, Opt.	Generic run input: tpr tpb tpa
`-o`	`energy.xvg`	Output	xvgr/xmgr file
`-viol`	`violaver.xvg`	Output, Opt.	xvgr/xmgr file
`-pairs`	`pairs.xvg`	Output, Opt.	xvgr/xmgr file
`-corr`	`enecorr.xvg`	Output, Opt.	xvgr/xmgr file
`-vis`	`visco.xvg`	Output, Opt.	xvgr/xmgr file
`-ravg`	`runavgdf.xvg`	Output, Opt.	xvgr/xmgr file

Other options

`-h`	bool	`no`	Print help info and quit
`-nice`	int	`19`	Set the nicelevel
`-b`	time	`-1`	First frame (ps) to read from trajectory
`-e`	time	`-1`	Last frame (ps) to read from trajectory
`-w`	bool	`no`	View output xvg, xpm, eps and pdb files
`-fee`	bool	`no`	Do a free energy estimate
`-fetemp`	real	`300`	Reference temperature for free energy calculation
`-zero`	real	`0`	Subtract a zero-point energy
`-sum`	bool	`no`	Sum the energy terms selected rather than display them all
`-dp`	bool	`no`	Print energies in high precision
`-mutot`	bool	`no`	Compute the total dipole moment from the components
`-skip`	int	`0`	Skip number of frames between data points
`-aver`	bool	`no`	Print also the X1,t and sigma1,t, only if only 1 energy is requested

where $\boldsymbol{r}_i^G$ is Gaussian distributed noise with $\mu = 0$, $\sigma = 1$. The friction coefficients γ_i can be chosen the same for all particles or as $\gamma_i = m_i/\xi_i$, where the friction constants ξ_i can be different for different groups of atoms. Because the system is assumed to be over damped, large time-steps can be used. LINCS should be used for the constraints since SHAKE will not converge for large atomic displacements. BD is an option of the `mdrun` program.

3.10 Energy Minimization

Energy minimization in GROMACS can be done using a steepest descent or conjugate gradient method. EM is just an option of the `mdrun` program.

3.10.1 Steepest Descent

Although steepest descent is certainly not the most efficient algorithm for searching, it is robust and easy to implement.

We define the vector $\boldsymbol{r}$ as the vector of all $3N$ coordinates. Initially a maximum displacement h_0 (e.g. 0.01 nm) must be given.

First the forces $\boldsymbol{F}$ and potential energy are calculated. New positions are calculated by

$$\boldsymbol{r}_{n+1} = \boldsymbol{r}_n + \frac{\boldsymbol{F}_n}{\max(|\boldsymbol{F}_n|)} h_n \tag{3.64}$$

where h_n is the maximum displacement and $\boldsymbol{F}_n$ is the force, or the negative gradient of the potential V. The notation $\max(|\boldsymbol{F}_n|)$ means the largest of the absolute values of the force components. The forces and energy are again computed for the new positions
If ($V_{n+1} < V_n$) the new positions are accepted and $h_{n+1} = 1.2 h_n$.
If ($V_{n+1} \geq V_n$) the new positions are rejected and $h_n = 0.2 h_n$.

The algorithm stops when either a user specified number of force evaluations has been performed (e.g. 100), or when the maximum of the absolute values of the force (gradient) components is smaller than a specified value ϵ. Since force truncation produces some noise in the energy evaluation, the stopping criterion should not be made too tight to avoid endless iterations. A reasonable value for ϵ can be estimated from the root mean square force f a harmonic oscillator would exhibit at a temperature T This value is

$$f = 2\pi\nu\sqrt{2mkT} \tag{3.65}$$

where ν is the oscillator frequency, m the (reduced) mass, and k Boltzmann's constant. For a weak oscillator with a wave number of 100 cm^{-1} and a mass of 10 atomic units, at a temperature of 1 K, $f = 7.7$ kJ mol^{-1} nm^{-1}. A value for ϵ between 1 and 10 is acceptable.

3.10.2 Conjugate Gradient

Conjugate gradient is slower than steepest descent in the early stages of the minimization, but becomes more efficient closer to the energy minimum. The parameters and stop criterion are the same as for steepest descent. Conjugate gradient can not be used with constraints or freeze groups.

3.11 Normal Mode Analysis

Normal mode analysis [30, 31, 32] can be performed using GROMACS, by diagonalization of the mass-weighted Hessian:

$$M^{-1/2}HM^{-1/2}Q \;=\; \omega^2 Q \tag{3.66}$$

where M contains the atomic masses, Q contains eigenvectors, and ω contains the corresponding eigenvalues (frequencies).

First, the Hessian matrix, which is a $3N \times 3N$ matrix where N is the number of atoms, has to be calculated:

$$H_{ij} \;=\; \frac{\partial^2 V}{\partial x_i \partial x_j} \tag{3.67}$$

where x_i and x_j denote the atomic x,y or z coordinates. In practice, these equations have not been developed analytically, but the force is used

$$F_i \;=\; \frac{\partial V}{\partial x_i} \tag{3.68}$$

from which the Hessian is computed numerically. It should be noted that for a usual Normal Mode calculation, it is necessary to completely minimize the energy prior to computation of the Hessian. This should be done with conjugate gradient in double precision. A number of GROMACS programs are involved in these calculations. First `nmrun`, which computes the Hessian, and secondly `g_nmeig` which does the diagonalization and sorting of normal modes according to frequencies. Both these programs should be run in double precision. An overview of normal mode analysis and the related principal component analysis (see sec. 8.9) can be found in [33].

3.12 Free energy calculations

Free energy calculations can be performed in GROMACS using slow-growth methods. An example problem might be: calculate the difference in free energy of binding of an inhibitor **I** to an enzyme **E** and to a mutated enzyme **E'**. It is not feasible with computer simulations to perform a docking calculation for such a large complex, or even releasing the inhibitor from the enzyme in a reasonable amount of computer time with reasonable accuracy. However, if we consider the free energy cycle in (Fig. 3.9A) we can write

$$\Delta G_1 - \Delta G_2 = \Delta G_3 - \Delta G_4 \tag{3.69}$$

If we are interested in the left-hand term we can equally well compute the right-hand term.

If we want to compute the difference in free energy of binding of two inhibitors **I** and **I'** to an enzyme **E** (Fig. 3.9B) we can again use eqn. 3.69 to compute the desired property.

Free energy differences between two molecular species can be calculated in GROMACS using the “slow-growth” method. In fact, such free energy differences between different molecular species are physically meaningless, but they can be used to obtain meaningful quantities employing a thermodynamic cycle. The method requires a simulation during which the Hamiltonian of the system changes slowly from that describing one system (A) to that describing the other system

---

Files

`-f`	`dyndom.pdb`	Input	Protein data bank file
`-o`	`rotated.xtc`	Output	Generic trajectory: xtc trr trj gro g96 pdb
`-n`	`domains.ndx`	Input	Index file

Other options

`-h`	bool	`no`	Print help info and quit
`-nice`	int	`0`	Set the nicelevel
`-firstangle`	real	`0`	Angle of rotation about rotation vector
`-lastangle`	real	`0`	Angle of rotation about rotation vector
`-nframe`	int	`11`	Number of steps on the pathway
`-maxangle`	real	`0`	DymDom dtermined angle of rotation about rotation vector
`-trans`	real	`0`	Translation (Aangstroem) along rotation vector (see DynDom info file)
`-head`	vector	`0 0 0`	First atom of the arrow vector
`-tail`	vector	`0 0 0`	Last atom of the arrow vector

E.21 g_enemat

g_enemat extracts an energy matrix from an energy file. With **-groups** a file must be supplied with on each line a group to be used. For these groups a matrices of interaction energies will be calculated. Also the total interaction energy energy per group is calculated.

An approximation of the free energy is calculated using: E(free) = E0 + kT log(<exp((E-E0)/kT)>), where ’<>’ stands for time-average. A file with reference free energies can be supplied to calculate the free energy difference with some reference state. Group names (e.g. residue names in the reference file should correspond to the group names as used in the **-groups** file, but a appended number (e.g. residue number)in the **-groups** will be ignored in the comparison.

Files

`-f`	`ener.edr`	Input, Opt.	Generic energy: edr ene
`-groups`	`groups.dat`	Input	Generic data file
`-eref`	`eref.dat`	Input, Opt.	Generic data file
`-emat`	`emat.xpm`	Output	X PixMap compatible matrix file
`-etot`	`energy.xvg`	Output	xvgr/xmgr file

Other options

`-h`	bool	`no`	Print help info and quit
`-nice`	int	`19`	Set the nicelevel
`-b`	time	`-1`	First frame (ps) to read from trajectory
`-e`	time	`-1`	Last frame (ps) to read from trajectory
`-dt`	time	`-1`	Only use frame when t MOD dt = first time (ps)
`-w`	bool	`no`	View output xvg, xpm, eps and pdb files
`-sum`	bool	`no`	Sum the energy terms selected rather than display them all
`-skip`	int	`0`	Skip number of frames between data points
`-mean`	bool	`yes`	with -groups calculates matrix of mean energies in stead of matrix for each timestep
`-nlevels`	int	`20`	number of levels for matrix colors
`-max`	real	`1e+20`	max value for energies
`-min`	real	`-1e+20`	min value for energies
`-coul`	bool	`yes`	calculate Coulomb SR energies
`-coulr`	bool	`no`	calculate Coulomb LR energies
`-coul14`	bool	`no`	calculate Coulomb 1-4 energies

`-dm`	`drmax.xvg`	Output	xvgr/xmgr file
`-dr`	`restr.xvg`	Output	xvgr/xmgr file
`-l`	`disres.log`	Output	Log file
`-n`	`viol.ndx`	Input, Opt.	Index file

Other options

`-h`	bool	`no`	Print help info and quit
`-nice`	int	`19`	Set the nicelevel
`-b`	time	`-1`	First frame (ps) to read from trajectory
`-e`	time	`-1`	Last frame (ps) to read from trajectory
`-dt`	time	`-1`	Only use frame when t MOD dt = first time (ps)
`-w`	bool	`no`	View output xvg, xpm, eps and pdb files
`-ntop`	int	`6`	Number of large violations that are stored in the log file every step

E.19 g_dist

g_dist can calculate the distance between the centers of mass of two groups of atoms as a function of time. The total distance and its x, y and z components are plotted.

Or when `-dist` is set, print all the atoms in group 2 that are closer than a certain distance to the center of mass of group 1.

Files

`-f`	`traj.xtc`	Input	Generic trajectory: xtc trr trj gro g96 pdb
`-s`	`topol.tpr`	Input	Generic run input: tpr tpb tpa
`-n`	`index.ndx`	Input, Opt.	Index file
`-o`	`dist.xvg`	Output, Opt.	xvgr/xmgr file

Other options

`-h`	bool	`no`	Print help info and quit
`-nice`	int	`19`	Set the nicelevel
`-b`	time	`-1`	First frame (ps) to read from trajectory
`-e`	time	`-1`	Last frame (ps) to read from trajectory
`-dt`	time	`-1`	Only use frame when t MOD dt = first time (ps)
`-dist`	real	`0`	Print all atoms in group 2 closer than dist to the center of mass of group 1

E.20 g_dyndom

g_dyndom reads a pdb file output from DynDom http://md.chem.rug.nl/ steve/DynDom/dyndom.home.html It reads the coordinates, and the coordinates of the rotation axis furthermore it reads an index file containing the domains. Furthermore it takes the first and last atom of the arrow file as command line arguments (head and tail) and finally it takes the translation vector (given in DynDom info file) and the angle of rotation (also as command line arguments). If the angle determined by DynDom is given, one should be able to recover the second structure used for generating the DynDom output. Because of limited numerical accuracy this should be verified by computing an all-atom RMSD (using `g_confrms`) rather than by file comparison (using diff).

The purpose of this program is to interpolate and extrapolate the rotation as found by DynDom. As a result unphysical structures with long or short bonds, or overlapping atoms may be produced. Visual inspection, and energy minimization may be necessary to validate the structure.

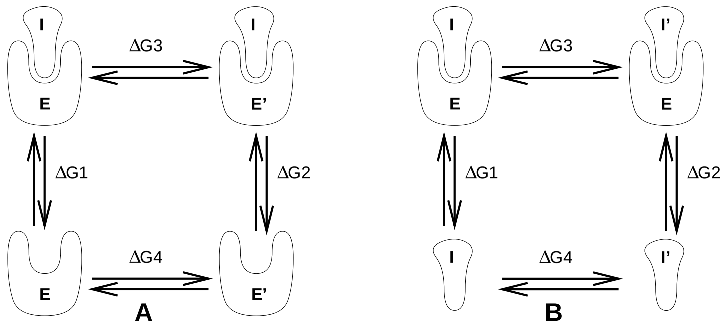


Figure 3.9: Free energy cycles. **A:** to calculate ΔG_{12} or the free energy difference between the binding of inhibitor **I** to enzymes **E** respectively **E'**. **B:** to calculate ΔG_{12} which is the free energy difference for binding of inhibitors **I** respectively **I'** to enzyme **E**.

(B). The change must be so slow that the system remains in equilibrium during the process; if that requirement is fulfilled, the change is reversible and a slow-growth simulation from B to A will yield the same results (but with a different sign) as a slow-growth simulation from A to B. This is a useful check, but the user should be aware of the danger that equality of forward and backward growth results does not guarantee correctness of the results.

The required modification of the Hamiltonian H is realized by making H a function of a *coupling parameter* $\lambda : H = H(p,q;\lambda)$ in such a way that $\lambda = 0$ describes system A and $\lambda = 1$ describes system B:

$$H(p,q;0) = H^{\mathrm{A}}(p,q); \quad H(p,q;1) = H^{\mathrm{B}}(p,q). \tag{3.70}$$

In GROMACS, the functional form of the λ-dependence is different for the various force-field contributions and is described in section sec. 4.3.

The Helmholtz free energy A is related to the partition function Q of an N,V,T ensemble, which is assumed to be the equilibrium ensemble generated by a MD simulation at constant volume and temperature. The generally more useful Gibbs free energy G is related to the partition function Δ of an N,p,T ensemble, which is assumed to be the equilibrium ensemble generated by a MD simulation at constant pressure and temperature:

$$\begin{aligned}
A(\lambda) &= -k_BT\ln Q && (3.71)\\
Q &= c\iint \exp[-\beta H(p,q;\lambda)]\,dp\,dq && (3.72)\\
G(\lambda) &= -k_BT\ln\Delta && (3.73)\\
\Delta &= c\iiint \exp[-\beta H(p,q;\lambda)-\beta pV]\,dp\,dq\,dV && (3.74)\\
G &= A+pV, && (3.75)
\end{aligned}$$

where $\beta = 1/(k_BT)$ and $c = (N!h^{3N})^{-1}$. These integrals over phase space cannot be evaluated from a simulation, but it is possible to evaluate the derivative to the parameter λ as an ensemble average:

$$\frac{dA}{d\lambda} = \frac{\iint (\partial H/\partial\lambda) \exp[-\beta H(p,q;\lambda)]\, dp\, dq}{\iint \exp[-\beta H(p,q;\lambda)]\, dp\, dq} = \left\langle \frac{\partial H}{\partial \lambda} \right\rangle_{NVT;\lambda}, \tag{3.76}$$

with a similar relation for $dG/d\lambda$ in the N,p,T ensemble. The difference in free energy between A and B can be found by integrating the derivative over λ:

$$A^{\rm B}(V,T) - A^{\rm A}(V,T) = \int_0^1 \left\langle \frac{\partial H}{\partial \lambda} \right\rangle_{NVT;\lambda} d\lambda \tag{3.77}$$

$$G^{\rm B}(p,T) - G^{\rm A}(p,T) = \int_0^1 \left\langle \frac{\partial H}{\partial \lambda} \right\rangle_{NpT;\lambda} d\lambda. \tag{3.78}$$

If one wishes to evaluate $G^{\rm B}(p,T) - G^{\rm A}(p,T)$, the natural choice is a constant-pressure simulation. However, this quantity can also be obtained from a slow-growth simulation at constant volume, starting with system A at pressure p and volume V and ending with system B at pressure p_B, by applying the following small correction:

$$G^{\rm B}(p) - G^{\rm A}(p) = A^{\rm B}(V) - A^{\rm A}(V) - \int_p^{p^{\rm B}} [V^{\rm B}(p') - V]\, dp' \tag{3.79}$$

Here we omitted the constant T from the notation. This correction is roughly equal to $-\frac{1}{2}(p^{\rm B} - p)\Delta V = (\Delta V)^2/(2\kappa V)$, where ΔV is the volume change at p and κ is the isothermal compressibility. This is usually negligible. For example, the growth of a water molecule from nothing in a bath of 1000 water molecules at constant volume would produce an additional pressure of 22 bar and a correction to the Helmholtz free energy of -20 J/mol.

In cartesian coordinates, the kinetic energy term in the Hamiltonian depends only on the momenta, and can be separately integrated and in fact removed from the equations. When masses do not change, there is no contribution from the kinetic energy at all; otherwise the integrated contribution to the free energy is $-\frac{3}{2}k_BT \ln(m^{\rm B}/m^{\rm A})$. This is no longer true in the presence of constraints.

GROMACS offers the possibility to integrate eq. 3.77 or eq. 3.78 in one simulation over the full range from A to B. However, if the change is large and sampling insufficiency can be expected, the user may prefer to determine the value of $\langle dG/d\lambda \rangle$ accurately at a number of well-chosen intermediate values of λ. This can be easily done by setting the stepsize **delta_lambda** to zero. Each simulation can be equilibrated first, and a proper error estimate can be made for each value of $dG/d\lambda$ from the fluctuation of $\partial H/\partial\lambda$. The total free energy change is then determined afterwards by an appropriate numerical integration procedure.

The λ-dependence for the force-field contributions is described in section sec. 4.3.

3.13 Essential Dynamics Sampling

The results from Essential Dynamics (see sec. 8.9) of a protein can be used to guide MD simulations. The idea is that from an initial MD simulation (or from other sources) a definition of the collective fluctuations with largest amplitude is obtained. The position along one or more of these collective modes can be constrained in a (second) MD simulation in a number of ways for several purposes. For example, the position along a certain mode may be kept fixed to monitor the average

---

`-s`	`topol.tpr`	Input	Generic run input: tpr tpb tpa
`-n`	`index.ndx`	Input, Opt.	Index file
`-o`	`Mtot.xvg`	Output	xvgr/xmgr file
`-e`	`epsilon.xvg`	Output	xvgr/xmgr file
`-a`	`aver.xvg`	Output	xvgr/xmgr file
`-d`	`dipdist.xvg`	Output	xvgr/xmgr file
`-c`	`dipcorr.xvg`	Output, Opt.	xvgr/xmgr file
`-g`	`gkr.xvg`	Output, Opt.	xvgr/xmgr file
`-q`	`quadrupole.xvg`	Output, Opt.	xvgr/xmgr file

Other options

`-h`	bool	`no`	Print help info and quit
`-nice`	int	`19`	Set the nicelevel
`-b`	time	`-1`	First frame (ps) to read from trajectory
`-e`	time	`-1`	Last frame (ps) to read from trajectory
`-dt`	time	`-1`	Only use frame when t MOD dt = first time (ps)
`-w`	bool	`no`	View output xvg, xpm, eps and pdb files
`-mu`	real	`-1`	dipole of a single molecule (in Debye)
`-mumax`	real	`5`	max dipole in Debye (for histrogram)
`-epsilonRF`	real	`0`	epsilon of the reaction field used during the simulation, needed for dieclectric constant calculation. WARNING: 0.0 means infinity (default)
`-skip`	int	`0`	Skip steps in the output (but not in the computations)
`-temp`	real	`300`	average temperature of the simulation (needed for dielectric constant calculation)
`-avercorr`	bool	`no`	calculate AC function of average dipole moment of the simulation box rather than average of AC function per molecule
`-gkratom`	int	`0`	Use the n-th atom of a molecule (starting from 1) to calculate the distance between molecules rather than the center of charge (when 0) in the calculation of distance dependent Kirkwood factors
`-acflen`	int	`-1`	Length of the ACF, default is half the number of frames
`-normalize`	bool	`yes`	Normalize ACF
`-P`	enum	`0`	Order of Legendre polynomial for ACF (0 indicates none): `0`, `1`, `2` or `3`
`-fitfn`	enum	`none`	Fit function: `none`, `exp`, `aexp`, `exp_exp` or `vac`
`-ncskip`	int	`0`	Skip N points in the output file of correlation functions
`-beginfit`	real	`0`	Time where to begin the exponential fit of the correlation function
`-endfit`	real	`-1`	Time where to end the exponential fit of the correlation function, -1 is till the end

E.18 g_disre

g_disre computes violations of distance restraints. If necessary all protons can be added to a protein molecule. The program allways computes the instantaneous violations rather than time-averaged, because this analysis is done from a trajectory file afterwards it does not make sense to use time averaging.

An index file may be used to select specific restraints for printing.

Files

`-s`	`topol.tpr`	Input	Generic run input: tpr tpb tpa
`-f`	`traj.xtc`	Input	Generic trajectory: xtc trr trj gro g96 pdb
`-ds`	`drsum.xvg`	Output	xvgr/xmgr file
`-da`	`draver.xvg`	Output	xvgr/xmgr file
`-dn`	`drnum.xvg`	Output	xvgr/xmgr file

The opther option is to discretize the dihedral space into a number of bins, and group each conformation in dihedral space in the appropriate bin. The output is then given as a number of dihedral conformations sorted according to occupancy.

Files

`-f`	`traj.xtc`	Input	Generic trajectory: xtc trr trj gro g96 pdb
`-s`	`topol.tpr`	Input	Generic run input: tpr tpb tpa
`-o`	`hello.out`	Output	Generic output file

Other options

`-h`	bool	`no`	Print help info and quit
`-nice`	int	`19`	Set the nicelevel
`-b`	time	`-1`	First frame (ps) to read from trajectory
`-e`	time	`-1`	Last frame (ps) to read from trajectory
`-dt`	time	`-1`	Only use frame when t MOD dt = first time (ps)
`-w`	bool	`no`	View output xvg, xpm, eps and pdb files
`-sa`	bool	`no`	Perform cluster analysis in dihedral space instead of analysing dihedral transitions.
`-mult`	int	`-1`	mulitiplicity for dihedral angles (by default read from topology)

- should not ask for number of frames

E.17 g_dipoles

g_dipoles computes the total dipole plus fluctuations of a simulation system. From this you can compute e.g. the dielectric constant for low dielectric media

The file dip.xvg contains the total dipole moment of a frame, the components as well as the norm of the vector. The file aver.xvg contains $< \mathrm{orMuor}^2 >$ and $< \mathrm{orMuor} >^2$ during the simulation. The file dip.xvg contains the distribution of dipole moments during the simulation The mu_max is used as the highest value in the distribution graph.

Furthermore the dipole autocorrelation function will be computed, when option -c is used. It can be averaged over all molecules, or (with option -avercorr) it can be computed as the autocorrelation of the total dipole moment of the simulation box.

At the moment the dielectric constant is calculated only correct if a rectangular or cubic simulation box is used.

Option `-g` produces a plot of the distance dependent Kirkwood G-factor, as well as the average cosine of the angle between the dipoles as a function of the distance. The plot also includes gOO and hOO according to Nymand & Linse, JCP 112 (2000) pp 6386-6395.

EXAMPLES

g_dipoles -P1 -n mols -o dip_sqr -mu 2.273 -mumax 5.0 -nofft

This will calculate the autocorrelation function of the molecular dipoles using a first order Legendre polynomial of the angle of the dipole vector and itself a time t later. For this calculation 1001 frames will be used. Further the dielectric constant will be calculated using an epsilonRF of infinity (default), temperature of 300 K (default) and an average dipole moment of the molecule of 2.273 (SPC). For the distribution function a maximum of 5.0 will be used.

Files

`-enx`	`ener.edr`	Input, Opt.	Generic energy: edr ene
`-f`	`traj.xtc`	Input	Generic trajectory: xtc trr trj gro g96 pdb

---

force (free-energy gradient) on that coordinate in that position. Another application is to enhance sampling efficiency with respect to usual MD [34, 35]. In this case, the system is encouraged to sample its available configuration space more systematically than in a diffusion-like path that proteins usually take.

All available constraint types are described in the appropriate chapter of the WHAT IF [36] manual.

3.14 Parallelization

The purpose of this section is to discuss the parallelization of the principle MD algorithm and not to describe the algorithms that are in practical use for molecular systems with their complex variety of atoms and terms in the force field descriptions. We shall therefore consider as an example a simple system consisting only of a single type of atoms with a simple form of the interaction potential. The emphasis will be on the special problems that arise when the algorithm is implemented on a parallel computer.

The simple model problem already contains the bottleneck of all MD simulations: the computationally intensive evaluation of the *non-bonded* forces between pairs of atoms, based on the distance between particles. Complex molecular systems will in addition involve many different kinds of *bonded* forces between designated atoms. Such interactions add to the complexity of the algorithm but do not modify the basic considerations concerning parallelization.

3.14.1 Methods of parallelization

There are a number of methods to parallelize the MD algorithm, each of them with their own advantages and disadvantages. The method to choose depends on the hardware and compilers available. We list them here:

1 *Message Passing.*
In this method, which is more or less the traditional way of parallel programming, all the parallelism is explicitly programmed by the user. The disadvantage is that it takes extra code and effort, the advantage is that the programmer keeps full control over the data flow and can do optimizations a compiler could not come up with.

The implementation is typically done by calling a set of library routines to send and receive data to and from other processors. Almost all hardware vendors support this way of parallelism in their C and Fortran compilers.

2 *Data Parallel.*
This method lets the user define arrays on which to operate in parallel. Programming this way is much like vectorizing: recurrence is not parallelized (e.g. `for(i=1; (i<MAX); i++) a[i] = a[i-1] + 1;` does not vectorize and not parallelize, because for every i the result from the previous step is needed).

The advantage of data parallelism is that it is easier for the user; the compiler takes care of the parallelism. The disadvantage is that it is supported by a small (though growing) number of hardware vendors, and that it is much harder to maintain a program that has to run on both parallel and sequential machines, because the only standard language that supports it is Fortran-90 which is not available on many platforms.

Both methods allow for the MD algorithm to be implemented without much trouble. Message passing MD algorithms have been published since the mid 80's ([37], [38]) and development is still continuing. Data parallel programming is newer, but starting from a well vectorized program it is not hard to do.

Our implementation of MD is a message passing one, the reason for which is partly historical: the project to develop a parallel MD program started when Fortran-90 was still in the making, and no compilers were expected to be available. At current, we still believe that message passing is the way to go, after having done some experiments with data parallel programming on a Connection Machine (CM-5), because of portability to other hardware, the poor performance of the code produced by the compilers and because this way of programming has the same drawback as vectorization: the part of the program that is not vectorized or parallelized determines the runtime of the program (Amdahl's law).

The approach we took to parallelism was a minimalist one: use as little non-standard elements in the software as possible, and use the simplest processor topology that does the job. We therefore decided to use a standard language (ANSI-C) with as little non-standard routines as possible. We only use 5 communication routines that are non-standard. It is therefore very easy to port our code to other machines.

For an $O(N^2)$ problem like MD, one of the best schemes for the interprocessor connections is a ring, so our software demands that a ring is present in the interprocessor connections. A ring can essentially always be mapped onto another network like a hypercube, a bus interface (Ethernet e.g. using Message Passing Interface MPI) or a tree (CM-5). Some hardware vendors have very luxurious connection schemes that connect every processor to every other processor, but we do not really need it and so do not use it even though it might come in handy at times. The advantage with this simplistic scheme is that GROMACS performs extremely well even on inexpensive workstation clusters.

When using a message passing scheme one has to divide the particles over processors, which can be done in two ways:

- *Space Decomposition.*
  An element of space is allocated to each processor, when dividing a cubic box with edge b over P processors this can be done by giving each processor a slab of length b/P. This method has the advantage that each processor has about the same number of interactions to calculate (at least when the simulated system has a homogeneous density, like a liquid or a gas). The disadvantage is that a lot of bookkeeping is necessary for particles that move over processor boundaries. When using more complex systems like macromolecules there are also 3- and 4-atom interactions that would complicate the bookkeeping so much that this method is not used in our program.

- *Particle Decomposition.*
  Every processor is allocated a number of particles. When dividing N particles over P processors each processor will get N/P particles. The implementation of this method is described in the next section.

E.15 g_dielectric

dielectric calculates frequency dependent dielectric constants from the autocorrelation function of the total dipole moment in your simulation. This ACF can be generated by g_dipoles. For an estimate of the error you can run g_statistics on the ACF, and use the output thus generated for this program. The functional forms of the available functions are:

One parmeter : y = Exp[-a1 x] Two parmeters : y = a2 Exp[-a1 x] Three parmeter: y = a2 Exp[-a1 x] + (1 - a2) Exp[-a3 x] Startvalues for the fit procedure can be given on the commandline. It is also possible to fix parameters at their start value, use -fix with the number of the parameter you want to fix.

Three output files are generated, the first contains the ACF, an exponential fit to it with 1, 2 or 3 parameters, and the numerical derivative of the combination data/fit. The second file contains the real and imaginary parts of the frequency-dependent dielectric constant, the last gives a plot known as the Cole-Cole plot, in which the imaginary component is plotted as a function of the real component. For a pure exponential relaxation (Debye relaxation) the latter plot should be one half of a circle

Files

`-f`	`Mtot.xvg`	Input	xvgr/xmgr file
`-d`	`deriv.xvg`	Output	xvgr/xmgr file
`-o`	`epsw.xvg`	Output	xvgr/xmgr file
`-c`	`cole.xvg`	Output	xvgr/xmgr file

Other options

`-h`	bool	`no`	Print help info and quit
`-nice`	int	`19`	Set the nicelevel
`-b`	time	`-1`	First frame (ps) to read from trajectory
`-e`	time	`-1`	Last frame (ps) to read from trajectory
`-dt`	time	`-1`	Only use frame when t MOD dt = first time (ps)
`-w`	bool	`no`	View output xvg, xpm, eps and pdb files
`-fft`	bool	`no`	use fast fourier transform for correlation function
`-x1`	bool	`yes`	use first column as X axis rather than first data set
`-eint`	real	`5`	Time were to end the integration of the data and start to use the fit
`-bfit`	real	`5`	Begin time of fit
`-efit`	real	`500`	End time of fit
`-tail`	real	`500`	Length of function including data and tail from fit
`-A`	real	`0.5`	Start value for fit parameter A
`-tau1`	real	`10`	Start value for fit parameter tau1
`-tau2`	real	`1`	Start value for fit parameter tau2
`-eps0`	real	`80`	Epsilon 0 of your liquid
`-epsRF`	real	`78.5`	Epsilon of the reaction field used in your simulation. A value of 0 means infinity.
`-fix`	int	`0`	Fix parameters at their start values, A (2), tau1 (1), or tau2 (4)
`-ffn`	enum	`none`	Fit function: `none`, `exp`, `aexp`, `exp_exp` or `vac`
`-nsmooth`	int	`3`	Number of points for smoothing

E.16 g_dih

g_dih can do two things. The default is to analyze dihedral transitions by merely computing all the dihedral angles defined in your topology for the whole trajectory. When a dihedral flips over to another minimum an angle/time plot is made.

`-l`	`covar.log`	Output	Log file
`-xpm`	`covar.xpm`	Output, Opt.	X PixMap compatible matrix file
`-xpma`	`covara.xpm`	Output, Opt.	X PixMap compatible matrix file

Other options

`-h`	bool	`no`	Print help info and quit
`-nice`	int	`19`	Set the nicelevel
`-b`	time	`-1`	First frame (ps) to read from trajectory
`-e`	time	`-1`	Last frame (ps) to read from trajectory
`-dt`	time	`-1`	Only use frame when t MOD dt = first time (ps)
`-tu`	enum	`ps`	Time unit: `ps`, `fs`, `ns`, `us`, `ms`, `s`, `m` or `h`
`-fit`	bool	`yes`	Fit to a reference structure
`-ref`	bool	`no`	Use the deviation from the conformation in the structure file instead of from the average
`-mwa`	bool	`no`	Mass-weighted covariance analysis
`-last`	int	`-1`	Last eigenvector to write away (-1 is till the last)

E.14 g_density

Compute partial densities across the box, using an index file. Densities in gram/cubic centimeter, number densities or electron densities can be calculated. For electron densities, each atom is weighed by its atomic partial charge.

Files

`-f`	`traj.xtc`	Input	Generic trajectory: xtc trr trj gro g96 pdb
`-n`	`index.ndx`	Input, Opt.	Index file
`-s`	`topol.tpr`	Input	Generic run input: tpr tpb tpa
`-ei`	`electrons.dat`	Output	Generic data file
`-o`	`density.xvg`	Output	xvgr/xmgr file

Other options

`-h`	bool	`no`	Print help info and quit
`-nice`	int	`19`	Set the nicelevel
`-b`	time	`-1`	First frame (ps) to read from trajectory
`-e`	time	`-1`	Last frame (ps) to read from trajectory
`-dt`	time	`-1`	Only use frame when t MOD dt = first time (ps)
`-w`	bool	`no`	View output xvg, xpm, eps and pdb files
`-d`	string	`Z`	Take the normal on the membrane in direction X, Y or Z.
`-sl`	int	`10`	Divide the box in #nr slices.
`-number`	bool	`no`	Calculate number density instead of mass density. Hydrogens are not counted!
`-ed`	bool	`no`	Calculate electron density instead of mass density
`-count`	bool	`no`	Only count atoms in slices, no densities. Hydrogens are not counted

- When calculating electron densities, atomnames are used instead of types. This is bad.
- When calculating number densities, atoms with names that start with H are not counted. This may be surprising if you use hydrogens with names like OP3.

3.14.2 MD on a ring of processors

When a neighbor list is not used the MD problem is in principle an $O(N^2)$ problem as each particle can interact with every other. This can be simplified using Newton's third law

$$F_{ij} \;=\; -\,F_{ji} \tag{3.80}$$

This implies that there is half a matrix of interactions (without diagonal, a particle does not interact with itself) to consider (Fig. 3.10). When we reflect the upper right triangle of interactions to the lower left triangle of the matrix, we still cover all possible interactions, but now every row in the matrix has almost the same number of points or possible interactions. We can now assign a (preferably equal) number of rows to each processor to compute the forces and at the same time a number of particles to do the update on, the *home* particles. The number of interactions per particle is dependent on the *total number* N of particles (see Fig. 3.11) and on the *particle number* i. The exact formulae are given in Table 3.2.

A flow chart of the algorithm is given in Fig. 3.12.

It is the same as the sequential algorithm, except for two communication steps. After the particles have been reset in the box, each processor sends its coordinates left and then starts computation of the forces. After this step each processor holds the *partial forces* for the available particles, e.g. processor 0 holds forces acting on home particles from processor 0, 1, 2 and 3. These forces must be accumulated and sent back (right) to the home processor. Finally the update of the velocity and coordinates is done on the home processor.

The `communicate_r` routine is given below in the full C-code:

```
void communicate_r(int nprocs,int pid,rvec vecs[],int start[],int homenr[])
/*
 * nprocs = number of processors
 * pid    = processor id (0..nprocs-1)
 * vecs   = vectors
 * start  = starting index in vecs for each processor
 * homenr = number of home particles for each processor
 */
{
  int i;        /* processor counter */
  int shift;    /* the amount of processors to communicate with */
  int cur;      /* current processor to send data from */
  int next;     /* next processor on a ring (using modulo) */

  cur   = pid;
  shift = nprocs/2;

  for (i=0; (i<shift); i++) {
    next=(cur+1) % nprocs;
    send   (left,  vecs[start[cur]],  homenr[cur]);
    receive(right, vecs[start[next]], homenr[next]);
    cur=next;
  }
}
```

The data flow around the ring is visualized in Fig. 3.13. Note that because of the ring topology each processor automatically gets the proper particles to interact with.

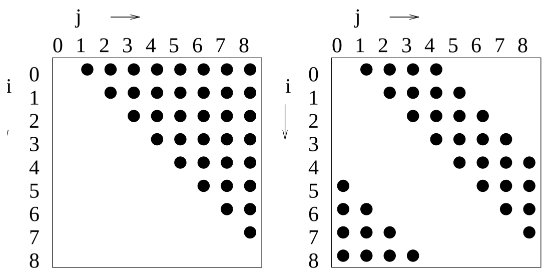
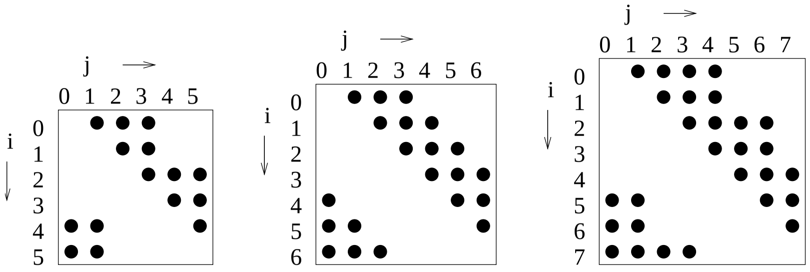


Figure 3.10: The interaction matrix (left) and the same using action $=$ $-$reaction (right).

	i mod 2 = 0 i < N/2	i mod 2 = 0 i ≥ N/2	i mod 2 = 1 i < N/2	i mod 2 = 1 i ≥ N/2
N mod 2 = 1	$N/2$	$N/2$	$N/2$	$N/2$
N mod 4 = 2	$N/2$	$N/2$	$N/2-1$	$N/2-1$
N mod 4 = 0	$N/2$	$N/2-1$	$N/2-1$	$N/2$

Table 3.2: The number of interactions between particles. The number of j particles per i particle is a function of the total number of particles N and particle number i. Note that here the $/$ operator is used for integer division, i.e. truncating the reminder.



N mod 4 = 2 **N mod 2 = 1** **N mod 4 = 0**

Figure 3.11: Interaction matrices for different N. The number of j-particles an i-particle interacts with depends on the *total* number of particles and on the *particle number*.

`-seed`	int	`1993`	Random number seed for Monte Carlo clustering algorithm
`-niter`	int	`10000`	Number of iterations for MC
`-kT`	real	`0.001`	Boltzmann weighting factor for Monte Carlo optimization (zero turns off uphill steps)

E.12 g_confrms

g_confrms computes the root mean square deviation (RMSD) of two structures after LSQ fitting the second structure on the first one. The two structures do NOT need to have the same number of atoms, only the two index groups used for the fit need to be identical.

The superimposed structures are written to file. In a `.pdb` file the two structures will be written as separate models (use `rasmol -nmrpdb`).

Files

`-f1`	`conf1.gro`	Input	Structure+mass(db): tpr tpb tpa gro g96 pdb
`-f2`	`conf2.gro`	Input	Generic structure: gro g96 pdb tpr tpb tpa
`-o`	`fit.pdb`	Output	Generic structure: gro g96 pdb
`-n1`	`fit1.ndx`	Input, Opt.	Index file
`-n2`	`fit2.ndx`	Input, Opt.	Index file

Other options

`-h`	bool	`no`	Print help info and quit
`-nice`	int	`19`	Set the nicelevel
`-one`	bool	`no`	Only write the fitted structure to file
`-pbc`	bool	`no`	Try to make molecules whole again

E.13 g_covar

`g_covar` calculates and diagonalizes the (mass-weighted) covariance matrix. All structures are fitted to the structure in the structure file. When this is not a run input file periodicity will not be taken into account. When the fit and analysis groups are identical and the analysis is non mass-weighted, the fit will also be non mass-weighted.

The eigenvectors are written to a trajectory file (`-v`). When the same atoms are used for the fit and the covariance analysis, the reference structure for the fit is written first with t=-1. The average (or reference when `-ref` is used) structure is written with t=0, the eigenvectors are written as frames with the eigenvector number as timestamp.

The eigenvectors can be analyzed with `g_anaeig`.

Option `-xpm` writes the whole covariance matrix to an xpm file.

Option `-xpma` writes the atomic covariance matrix to an xpm file, i.e. for each atom pair the sum of the xx, yy and zz covariances is written.

Files

`-f`	`traj.xtc`	Input	Generic trajectory: xtc trr trj gro g96 pdb
`-s`	`topol.tpr`	Input	Structure+mass(db): tpr tpb tpa gro g96 pdb
`-n`	`index.ndx`	Input, Opt.	Index file
`-o`	`eigenval.xvg`	Output	xvgr/xmgr file
`-v`	`eigenvec.trr`	Output	Full precision trajectory: trr trj
`-av`	`average.pdb`	Output	Generic structure: gro g96 pdb

the lower right half (depends on `-max` and `-keepfree`).
`-g` writes information on the options used and a detailed list of all clusters and their members.

Additionally, a number of optional output files can be written:
`-dist` writes the RMSD distribution.
`-ev` writes the eigenvectors of the RMSD matrix diagonalization.
`-sz` writes the cluster sizes.
`-tr` writes a matrix of the number transitions between cluster pairs.
`-ntr` writes the total number of transitions to or from each cluster.
`-clid` writes the cluster number as a function of time.
`-cl` writes average (with option `-av`) or central structure of each cluster or writes numbered files with cluster members for a selected set of clusters (with option `-wcl`, depends on `-nst` and `-rmsmin`).

Files

`-f`	`traj.xtc`	Input, Opt.	Generic trajectory: xtc trr trj gro g96 pdb
`-s`	`topol.tpr`	Input, Opt.	Structure+mass(db): tpr tpb tpa gro g96 pdb
`-n`	`index.ndx`	Input, Opt.	Index file
`-dm`	`rmsd.xpm`	Input, Opt.	X PixMap compatible matrix file
`-o`	`rmsd-clust.xpm`	Output	X PixMap compatible matrix file
`-g`	`cluster.log`	Output	Log file
`-dist`	`rmsd-dist.xvg`	Output, Opt.	xvgr/xmgr file
`-ev`	`rmsd-eig.xvg`	Output, Opt.	xvgr/xmgr file
`-sz`	`clust-size.xvg`	Output, Opt.	xvgr/xmgr file
`-tr`	`clust-trans.xpm`	Output, Opt.	X PixMap compatible matrix file
`-ntr`	`clust-trans.xvg`	Output, Opt.	xvgr/xmgr file
`-clid`	`clust-id.xvg`	Output, Opt.	xvgr/xmgr file
`-cl`	`clusters.pdb`	Output, Opt.	Generic trajectory: xtc trr trj gro g96 pdb

Other options

`-h`	bool	`no`	Print help info and quit
`-nice`	int	`19`	Set the nicelevel
`-b`	time	`-1`	First frame (ps) to read from trajectory
`-e`	time	`-1`	Last frame (ps) to read from trajectory
`-dt`	time	`-1`	Only use frame when t MOD dt = first time (ps)
`-tu`	enum	`ps`	Time unit: `ps`, `fs`, `ns`, `us`, `ms`, `s`, `m` or `h`
`-w`	bool	`no`	View output xvg, xpm, eps and pdb files
`-dista`	bool	`no`	Use RMSD of distances instead of RMS deviation
`-nlevels`	int	`40`	Discretize RMSD matrix in # levels
`-keepfree`	int	`-4`	if >0 # levels not to use when coloring clusters; if <0 nlevels/-keepfree+1 levels will not be used
`-cutoff`	real	`0.1`	RMSD cut-off (nm) for two structures to be neighbor
`-max`	real	`-1`	Maximum level in RMSD matrix
`-skip`	int	`1`	Only analyze every nr-th frame
`-av`	bool	`no`	Write average iso middle structure for each cluster
`-wcl`	int	`0`	Write all structures for first # clusters to numbered files
`-nst`	int	`1`	Only write all structures if more than # per cluster
`-rmsmin`	real	`0`	minimum rms difference with rest of cluster for writing structures
`-method`	enum	`linkage`	Method for cluster determination: `linkage`, `jarvis-patrick`, `monte-carlo`, `diagonalization` or `gromos`
`-binary`	bool	`no`	Treat the RMSD matrix as consisting of 0 and 1, where the cut-off is given by -cutoff
`-M`	int	`10`	Number of nearest neighbors considered for Jarvis-Patrick algorithm, 0 is use cutoff
`-P`	int	`3`	Number of identical nearest neighbors required to form a cluster

---

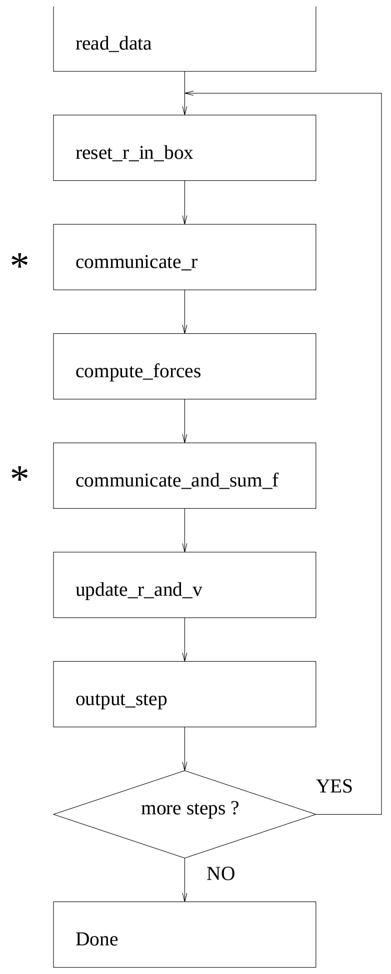


Figure 3.12: The Parallel MD algorithm. If the steps marked * are left out we have the sequential algorithm again.

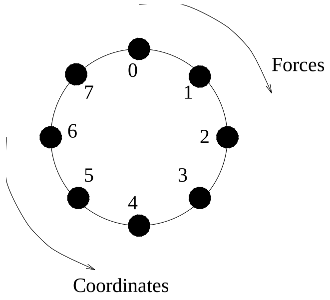


Figure 3.13: Data flow in a ring of processors.

3.15 Parallel Molecular Dynamics

In this chapter we describe some details of the parallel MD algorithm used in GROMACS. This also includes some other information on neighbor searching and a side excursion to parallel sorting. Please note the following which we use throughout this chapter:
definition: N: Number of particles, M number of processors.
GROMACS employs two different grids: the neighbor searching grid (NS grid) and the combined charge/potential grid (FFT grid), as will be described below. To maximize the confusion, these two grids are mapped onto a grid of processors when GROMACS runs on a parallel computer.

3.15.1 Domain decomposition

Modern day parallel computers, such as an IBM SP/2 or a Cray T3E consist of relatively small numbers of relatively fast scalar processors (typically 8 to 256). The communication channels that are available in hardware on these machine are not directly visible for the programmer, a software layer (usually MPI) hides this, and makes communication from all processors to all others possible. In contrast, in the GROMACS hardware [1] only communication in a ring was available, i.e. each processor could communicate with its direct neighbors only.

It seems logical to map the computational box of an MD simulation system to a 3D grid of processors (e.g. 4x4x4 for a 64 processor system). This ensures that most interactions that are local in space can be computed with information from neighboring processors only. However, this means that there have to be communication channels in 3 dimensions too, which is not necessarily the case. Although this may be overcome in software, such a mapping is complicated for the MD software as well, without clear benefits in terms of performance for most parallel computers.

Therefore we opt for a simple one-dimensional division scheme for the computational box. Each processor gets a slab of this box in the X-dimension. For the communication between processors this has two main advantages:

1. Simplicity of coding. Communication can only be to two neighbors (called *left* and *right* in GROMACS).

`-omega`	bool	`no`	Output for Omega dihedrals (peptide bonds)
`-rama`	bool	`no`	Generate Phi/Psi and Chi1/Chi2 ramachandran plots
`-viol`	bool	`no`	Write a file that gives 0 or 1 for violated Ramachandran angles
`-all`	bool	`no`	Output separate files for every dihedral.
`-shift`	bool	`no`	Compute chemical shifts from Phi/Psi angles
`-run`	int	`1`	perform running average over ndeg degrees for histograms
`-maxchi`	enum	`0`	calculate first ndih Chi dihedrals: `0`, `1`, `2`, `3`, `4`, `5` or `6`
`-normhisto`	bool	`yes`	Normalize histograms
`-ramomega`	bool	`no`	compute average omega as a function of phi/psi and plot it in an xpm plot
`-bfact`	real	`-1`	B-factor value for pdb file for atoms with no calculated dihedral order parameter
`-bmax`	real	`0`	Maximum B-factor on any of the atoms that make up a dihedral, for the dihedral angle to be considere in the statistics. Applies to database work where a number of X-Ray structures is analyzed. -bmax <= 0 means no limit.
`-acflen`	int	`-1`	Length of the ACF, default is half the number of frames
`-normalize`	bool	`yes`	Normalize ACF
`-P`	enum	`0`	Order of Legendre polynomial for ACF (0 indicates none): `0`, `1`, `2` or `3`
`-fitfn`	enum	`none`	Fit function: `none`, `exp`, `aexp`, `exp_exp` or `vac`
`-ncskip`	int	`0`	Skip N points in the output file of correlation functions
`-beginfit`	real	`0`	Time where to begin the exponential fit of the correlation function
`-endfit`	real	`-1`	Time where to end the exponential fit of the correlation function, -1 is till the end

- Produces MANY output files (up to about 4 times the number of residues in the protein, twice that if autocorrelation functions are calculated). Typically several hundred files are output.

E.11 g_cluster

g_cluster can cluster structures with several different methods. Distances between structures can be determined from a trajectory or read from an XPM matrix file with the `-dm` option. RMS deviation after fitting or RMS deviation of atom-pair distances can be used to define the distance between structures.

full linkage: add a structure to a cluster when its distance to any element of the cluster is less than `cutoff`.

Jarvis Patrick: add a structure to a cluster when this structure and a structure in the cluster have each other as neighbors and they have a least `P` neighbors in common. The neighbors of a structure are the M closest structures or all structures within `cutoff`.

Monte Carlo: reorder the RMSD matrix using Monte Carlo.

diagonalization: diagonalize the RMSD matrix.

gromos: use algorithm as described in Daura *et al.* (*Angew. Chem. Int. Ed.* **1999**, *38*, pp 236-240). Count number of neighbors using cut-off, take structure with largest number of neighbors with all its neighbors as cluster and eleminate it from the pool of clusters. Repeat for remaining structures in pool.

When the clustering algorithm assigns each structure to exactly one cluster (full linkage, Jarvis Patrick and gromos) and a trajectory file is supplied, the structure with the smallest average distance to the others or the average structure or all structures for each cluster will be written to a trajectory file. When writing all structures, separate numbered files are made for each cluster.

Two output files are always written:
`-o` writes the RMSD values in the upper left half of the matrix and a graphical depiction of the clusters in

`-otr`	`bun_tiltr.xvg`	Output	xvgr/xmgr file
`-otl`	`bun_tiltl.xvg`	Output	xvgr/xmgr file
`-ok`	`bun_kink.xvg`	Output, Opt.	xvgr/xmgr file
`-okr`	`bun_kinkr.xvg`	Output, Opt.	xvgr/xmgr file
`-okl`	`bun_kinkl.xvg`	Output, Opt.	xvgr/xmgr file
`-oa`	`axes.pdb`	Output, Opt.	Protein data bank file

Other options

`-h`	bool	`no`	Print help info and quit
`-nice`	int	`19`	Set the nicelevel
`-b`	time	`-1`	First frame (ps) to read from trajectory
`-e`	time	`-1`	Last frame (ps) to read from trajectory
`-dt`	time	`-1`	Only use frame when t MOD dt = first time (ps)
`-tu`	enum	`ps`	Time unit: `ps`, `fs`, `ns`, `us`, `ms`, `s`, `m` or `h`
`-na`	int	`0`	Number of axes
`-z`	bool	`no`	Use the Z-axis as reference iso the average axis

E.10 g_chi

g_chi computes phi, psi, omega and chi dihedrals for all your amino acid backbone and sidechains. It can compute dihedral angle as a function of time, and as histogram distributions. Output is in form of xvgr files, as well as a LaTeX table of the number of transitions per nanosecond.

Order parameters S2 for each of the dihedrals are calculated and output as xvgr file and optionally as a pdb file with the S2 values as B-factor.

If option `-c` is given, the program will calculate dihedral autocorrelation functions. The function used is C(t) = < cos(chi(tau)) cos(chi(tau+t)) >. The use of cosines rather than angles themselves, resolves the problem of periodicity. (Van der Spoel & Berendsen (1997), **Biophys. J. 72**, 2032-2041).

The option `-r` generates a contour plot of the average omega angle as a function of the phi and psi angles, that is, in a Ramachandran plot the average omega angle is plotted using color coding.

Files

`-c`	`conf.gro`	Input	Generic structure: gro g96 pdb tpr tpb tpa
`-f`	`traj.xtc`	Input	Generic trajectory: xtc trr trj gro g96 pdb
`-o`	`order.xvg`	Output	xvgr/xmgr file
`-p`	`order.pdb`	Output, Opt.	Protein data bank file
`-ss`	`ssdump.dat`	Input, Opt.	Generic data file
`-jc`	`Jcoupling.xvg`	Output	xvgr/xmgr file
`-corr`	`dihcorr.xvg`	Output, Opt.	xvgr/xmgr file
`-g`	`chi.log`	Output	Log file

Other options

`-h`	bool	`no`	Print help info and quit
`-nice`	int	`19`	Set the nicelevel
`-b`	time	`-1`	First frame (ps) to read from trajectory
`-e`	time	`-1`	Last frame (ps) to read from trajectory
`-dt`	time	`-1`	Only use frame when t MOD dt = first time (ps)
`-w`	bool	`no`	View output xvg, xpm, eps and pdb files
`-r0`	int	`1`	starting residue
`-phi`	bool	`no`	Output for Phi dihedral angles
`-psi`	bool	`no`	Output for Psi dihedral angles

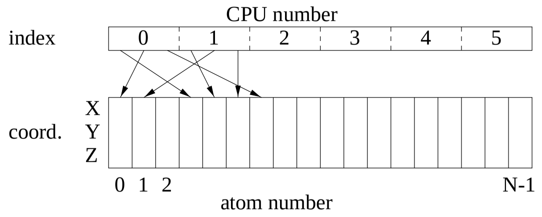


Figure 3.14: Index in the coordinate array. The division in slabs is indicated by dashed lines.

2. Communication can usually be done in large chunks, which makes it more efficient on most hardware platforms.

Most interactions in molecular dynamics have in principle a short ranged character. Bonds, angles and dihedrals are guaranteed to have the corresponding particles close in space.

3.15.2 Domain decomposition for non-bonded forces

For large parallel computers, domain decomposition is preferable over particle decomposition, since it is easier to do load balancing. Without load balancing the scaling of the code is rather poor... For this purpose, the computational box is divided in M slabs, where M is equal to the number of processors. There are multiple ways of dividing the box over processors, but since the GROMACS code assumes a ring topology for the processors, it is logical to cut the system in slabs in just one dimension, the X dimension. The algorithm for neighbor searching then becomes:

1. Make a list of charge group indices sorted on (increasing) X coordinate (Fig. 3.14). **Note** that care must be taken to parallelize the sorting algorithm as well. See sec. 3.15.4.
2. Divide this list into slabs, such that each slab has the same number of charge groups
3. Put the particles corresponding to the local slab on a 3D NS grid as described in sec. 3.4.2.
4. Communicate the NS grid to neighboring processors (not necessarily to all processors). The amount of neighboring NS grid cells (N_{gx}) to communicate is determined by the cut-off length r_c according to
$$N_{gx} \;=\; \frac{r_c M}{l_x} \tag{3.81}$$
where l_x is the box length in the slabbing direction.
5. On each processor compute the neighbor list for all charge groups in its slab using the normal grid neighbor-searching.

For homogeneous system, this is close to an optimal load balancing, without actually doing load balancing. For inhomogeneous system, such as membranes, or interfaces, the dimension for slabbing must be chosen such that it is perpendicular to the interface; in this fashion each processor has "a little bit of everything". The GROMACS utility program `editconf` has an option to rotate a whole computational box.

The following observations are important here:

- Particles may diffuse from one slab to the other, therefore each processor must hold coordinates for all particles all the time, and distribute forces back to all processors as well.
- Velocities are kept on the "home processor" for each particle, where the integration of Newton's equations is done.
- Fixed interaction lists (bonds, angles etc.) are kept each on a single processor. Since all processors have all coordinates, it does not matter where interactions are calculated. The division is actually done by the GROMACS preprocessor `grompp` and care is taken that, as far as possible, every processor gets the same number of bonded interactions.

In all, this makes for a mixed particle decomposition/domain decomposition scheme for parallelization of the MD code. The communication costs are four times higher than for the simple particle decomposition method described in sec. 3.14 (the whole coordinate and force array are communicated across the whole ring, rather than half the array over half the ring). However, for large numbers of processors the improved load balancing compensates this easily.

3.15.3 Parallel PPPM

A further reason for domain decomposition is the PPPM algorithm. This algorithm works with a 3D Fast Fourier Transform. It employs a discrete grid of dimensions (n_x,n_y,n_z), the FFT grid. The algorithm consist of five steps, each of which have to be parallelized:

1. Spreading charges on the FFT grid to obtain the charge distribution $\rho(\boldsymbol{r})$. This bit involves the following sub-steps:
   - **a.** put particle in the box
   - **b.** find the FFT grid cell in which the particle resides
   - **c.** add the charge of the particle times the appropriate weight factor (see sec. 4.6.3) to each of the 27 grid points (3 x 3 x 3).

   In the parallel case, the FFT grid must be filled on each processor with its share of the particles, and subsequently the FFT grids of all processors must be summed to find the total charge distribution. It may be clear that this induces a large amount of unnecessary work, unless we use domain decomposition. If each processor only has particles in a certain region of space, it only has to calculate the charge distribution for that region of space. Since GROMACS works with slabs, this means that each processor fills the FFT grid cells corresponding to it's slab in space and addition of FFT grids need only be done for neighboring slabs.

   To be more precise, the slab x for processor i is defined as:

$$i\,\frac{l_x}{M} \le x < \,(i+1)\,\frac{l_x}{M} \qquad (3.82)$$

E.8 g_bond

g_bond makes a distribution of bond lengths. If all is well a gaussian distribution should be made when using a harmonic potential. bonds are read from a single group in the index file in order i1-j1 i2-j2 thru in-jn.

`-tol` gives the half-width of the distribution as a fraction of the bondlength (`-blen`). That means, for a bond of 0.2 a tol of 0.1 gives a distribution from 0.18 to 0.22

Files

`-f`	`traj.xtc`	Input	Generic trajectory: xtc trr trj gro g96 pdb
`-n`	`index.ndx`	Input	Index file
`-o`	`bonds.xvg`	Output	xvgr/xmgr file
`-l`	`bonds.log`	Output, Opt.	Log file

Other options

`-h`	bool	`no`	Print help info and quit
`-nice`	int	`19`	Set the nicelevel
`-b`	time	`-1`	First frame (ps) to read from trajectory
`-e`	time	`-1`	Last frame (ps) to read from trajectory
`-dt`	time	`-1`	Only use frame when t MOD dt = first time (ps)
`-w`	bool	`no`	View output xvg, xpm, eps and pdb files
`-blen`	real	`-1`	Bond length. By default length of first bond
`-tol`	real	`0.1`	Half width of distribution as fraction of blen
`-aver`	bool	`yes`	Sum up distributions

- It should be possible to get bond information from the topology.

E.9 g_bundle

g_bundle analyzes bundles of axes. The axes can be for instance helix axes. The program reads two index groups and divides both of them in `-na` parts. The centers of mass of these parts define the tops and bottoms of the axes. Several quantities are written to file: the axis length, the distance and the z-shift of the axis mid-points with respect to the average center of all axes, the total tilt, the radial tilt and the lateral tilt with respect to the average axis.

With options `-ok`, `-okr` and `-okl` the total, radial and lateral kinks of the axes are plotted. An extra index group of kink atoms is required, which is also divided into `-na` parts. The kink angle is defined as the angle between the kink-top and the bottom-kink vectors.

With option `-oa` the top, mid (or kink when `-ok` is set) and bottom points of each axis are written to a pdb file each frame. The residue numbers correspond to the axis numbers. When viewing this file with `rasmol`, use the command line option `-nmrpdb`, and type `set axis true` to display the reference axis.

Files

`-f`	`traj.xtc`	Input	Generic trajectory: xtc trr trj gro g96 pdb
`-s`	`topol.tpr`	Input	Structure+mass(db): tpr tpb tpa gro g96 pdb
`-n`	`index.ndx`	Input, Opt.	Index file
`-ol`	`bun_len.xvg`	Output	xvgr/xmgr file
`-od`	`bun_dist.xvg`	Output	xvgr/xmgr file
`-oz`	`bun_z.xvg`	Output	xvgr/xmgr file
`-ot`	`bun_tilt.xvg`	Output	xvgr/xmgr file

E.7 g_angle

g_angle computes the angle distribution for a number of angles or dihedrals. This way you can check whether your simulation is correct. With option -ov you can plot the average angle of a group of angles as a function of time. With the -all option the first graph is the average, the rest are the individual angles.

With the -of option g_angle also calculates the fraction of trans dihedrals (only for dihedrals) as function of time, but this is probably only fun for a selected few.

With option -oc a dihedral correlation function is calculated.

It should be noted that the indexfile should contain atom-triples for angles or atom-quadruplets for dihedrals. If this is not the case, the program will crash.

Files

`-f`	`traj.xtc`	Input	Generic trajectory: xtc trr trj gro g96 pdb
`-s`	`topol.tpr`	Input	Generic run input: tpr tpb tpa
`-n`	`angle.ndx`	Input	Index file
`-od`	`angdist.xvg`	Output	xvgr/xmgr file
`-ov`	`angaver.xvg`	Output, Opt.	xvgr/xmgr file
`-of`	`dihfrac.xvg`	Output, Opt.	xvgr/xmgr file
`-ot`	`dihtrans.xvg`	Output, Opt.	xvgr/xmgr file
`-oh`	`trhisto.xvg`	Output, Opt.	xvgr/xmgr file
`-oc`	`dihcorr.xvg`	Output, Opt.	xvgr/xmgr file

Other options

`-h`	bool	`no`	Print help info and quit
`-nice`	int	`19`	Set the nicelevel
`-b`	time	`-1`	First frame (ps) to read from trajectory
`-e`	time	`-1`	Last frame (ps) to read from trajectory
`-dt`	time	`-1`	Only use frame when t MOD dt = first time (ps)
`-w`	bool	`no`	View output xvg, xpm, eps and pdb files
`-type`	enum	`angle`	Type of angle to analyse: `angle`, `dihedral`, `improper` or `ryckaert-bellemans`
`-all`	bool	`no`	Plot all angles separately in the averages file, in the order of appearance in the index file.
`-binwidth`	real	`1`	binwidth (degrees) for calculating the distribution
`-chandler`	bool	`no`	Use Chandler correlation function (N[trans] = 1, N[gauche] = 0) rather than cosine correlation function. Trans is defined as phi $<$ -60 or phi $>$ 60.
`-avercorr`	bool	`no`	Average the correlation functions for the individual angles/dihedrals
`-acflen`	int	`-1`	Length of the ACF, default is half the number of frames
`-normalize`	bool	`yes`	Normalize ACF
`-P`	enum	`0`	Order of Legendre polynomial for ACF (0 indicates none): `0`, `1`, `2` or `3`
`-fitfn`	enum	`none`	Fit function: `none`, `exp`, `aexp`, `exp_exp` or `vac`
`-ncskip`	int	`0`	Skip N points in the output file of correlation functions
`-beginfit`	real	`0`	Time where to begin the exponential fit of the correlation function
`-endfit`	real	`-1`	Time where to end the exponential fit of the correlation function, -1 is till the end

- Counting transitions only works for dihedrals with multiplicity 3

Particle with this x coordinate range will add to the charge distribution on the following range of of FFT grid slabs in the x direction:

$$\mathrm{trunc}\left(i\,\frac{l_x n_x}{M}\right) - 1 \le i_x \le \mathrm{trunc}\left((i+1)\,\frac{l_x n_x}{M}\right) + 2 \tag{3.83}$$

where trunc indicates the truncation of a real number to the largest integer smaller than or equal to that real number.

2. Doing the Fourier transform of the charge distribution $\rho(\boldsymbol{r})$ in parallel to obtain $\hat{\rho}(\boldsymbol{k})$. This is done using the FFTW library (see www.fftw.org) which employs the MPI library for message passing programs (note that there are also shared memory versions of the FFTW code).
This FFT algorithm actually use slabs as well (good thinking!). Each processor does 2D FFTS on its slab, and then the whole FFT grid is transposed *in place* (i.e. without using extra memory). This means that after the FFT the X and Y components are swapped. To complete the FFT, this swapping should be undone in principle (by transposing back). Happily the FFTW code has an option to omit this, which we use in the next step.

3. Convolute $\hat{\rho}(\boldsymbol{k})$ with the Fourier transform of the charge spread function $\hat{g}(\boldsymbol{k})$ (which we have tabulated before) to obtain the potential $\hat{\phi}(k)$. As an optimization, we store the $\hat{g}(\boldsymbol{k})$ in transposed form as well, matching the transposed form of $\hat{\rho}(\boldsymbol{k})$ which we get from the FFTW routine. After this step we have the potential $\hat{\phi}(k)$ in Fourier space, but still on the transposed FFT grid.

4. Do an inverse transform of $\hat{\phi}(k)$ to obtain $\phi(\boldsymbol{r})$. Since the algorithm must do a transpose of the data this step actually yields the wanted result: the un-transposed potential in real space.

5. Interpolate the potential $\phi(\boldsymbol{r})$ in real space at the particle positions to obtain forces and energy. For this bit the same considerations towards parallelism hold as for the charge spreading. However in this case more neighboring grid cells are needed, such that we need the following set of FFT grid slabs in the x direction:

$$\mathrm{trunc}\left(i\,\frac{l_x n_x}{M}\right) - 3 \le i_x \le \mathrm{trunc}\left((i+1)\,\frac{l_x n_x}{M}\right) + 4 \tag{3.84}$$

The algorithm as sketched above requires communication for spreading the charges, for the FFTW forward and backward, and for interpolating the forces. The GROMACS bits of the program use only left and right communication, i.e. using two communication channels. The FFTW routines actually use other forms of communication as well, and these routines are coded with MPI routines for message passing. This implies that GROMACS can only perform the PPPM algorithm on parallel computers computers that support MPI. However, most shared memory computers, such as the SGI Origin also support MPI using the shared memory for communication.

3.15.4 Parallel sorting

For the domain decomposition bit of GROMACS it is necessary to sort the coordinates (or rather the index to coordinates) every time a neighbor list is made. If we use brute force, and sort all

coordinates on each processor (which is technically possible since we have all the coordinates), then this sorting procedure will take a constant time (proportional to $N^2 \log N$, independent of the number of processors. We can however do a little better, if we assume that particles diffuse only slowly. A parallel sorting algorithm can be conceived as follows:
At the first step of the simulation

1. Do a full sort of all indices using e.g. the quick-sort algorithm that is built-in in the standard C-library
2. Divide the sorted array into slabs (as described above see Fig. 3.14).

At subsequent steps of the simulation:

1. Send the indices for each processor to the preceding processor (if not processor 0) and to the next processor (if not M-1). The communication associated with this operation is proportional to $2N/M$.
2. Sort the combined indices of the three (or two) processors. Note that the CPU time associated with sorting is now $(3N/M)^2 \log(3N/M)$.
3. On each processor, the indices belonging to it's slab can be determined from the order of the array (Fig. 3.14).

This is useful for principal components obtained from covariance analysis, since the principal components of random diffusion are pure cosines.

Option `-msd` produces the mean square displacement(s).

Option `-dist` produces distribution plot(s).

Option `-av` produces the average over the sets. Error bars can be added with the option `-errbar`. The errorbars can represent the standard deviation, the error (assuming the points are independent) or the interval containing 90% of the points, by discarding 5% of the points at the top and the bottom.

Option `-ee` produces error estimates using block averaging. A set is divided in a number of blocks and averages are calculated for each block. The error for the total average is calculated from the variance between averages of the m blocks B_i as follows: error^2 = Sum (B_i - <B>)^2 / (m*(m-1)). These errors are plotted as a function of the block size. Also an analytical block average curve is plotted, assuming that the autocorrelation is a sum of two exponentials. The analytical curve for the block average BA is:
BA(t) = sigma sqrt(2/T (a (tau1 ((exp(-t/tau1) - 1) tau1/t + 1)) +
(1-a) (tau2 ((exp(-t/tau2) - 1) tau2/t + 1)))),
where T is the total time. a, tau1 and tau2 are obtained by fitting BA(t) to the calculated block average. When the actual block average is very close to the analytical curve, the error is sigma*sqrt(2/T (a tau1 + (1-a) tau2)).

Option `-power` fits the data to b tâ, which is accomplished by fitting to a t + b on log-log scale. All points after the first zero or negative value are ignored.

Files

`-f`	`graph.xvg`	Input	xvgr/xmgr file
`-ac`	`autocorr.xvg`	Output, Opt.	xvgr/xmgr file
`-msd`	`msd.xvg`	Output, Opt.	xvgr/xmgr file
`-cc`	`coscont.xvg`	Output, Opt.	xvgr/xmgr file
`-dist`	`distr.xvg`	Output, Opt.	xvgr/xmgr file
`-av`	`average.xvg`	Output, Opt.	xvgr/xmgr file
`-ee`	`errest.xvg`	Output, Opt.	xvgr/xmgr file

Other options

`-h`	bool	`no`	Print help info and quit
`-nice`	int	`19`	Set the nicelevel
`-w`	bool	`no`	View output xvg, xpm, eps and pdb files
`-time`	bool	`yes`	Expect a time in the input
`-b`	real	`-1`	First time to read from set
`-e`	real	`-1`	Last time to read from set
`-n`	int	`1`	Read # sets seperated by &
`-d`	bool	`no`	Use the derivative
`-bw`	real	`0.1`	Binwidth for the distribution
`-errbar`	enum	`none`	Error bars for -av: `none`, `stddev`, `error` or `90`
`-power`	bool	`no`	Fit data to: b tâ
`-subav`	bool	`yes`	Subtract the average before autocorrelating
`-oneacf`	bool	`no`	Calculate one ACF over all sets
`-acflen`	int	`-1`	Length of the ACF, default is half the number of frames
`-normalize`	bool	`yes`	Normalize ACF
`-P`	enum	`0`	Order of Legendre polynomial for ACF (0 indicates none): `0`, `1`, `2` or `3`
`-fitfn`	enum	`none`	Fit function: `none`, `exp`, `aexp`, `exp_exp` or `vac`
`-ncskip`	int	`0`	Skip N points in the output file of correlation functions
`-beginfit`	real	`0`	Time where to begin the exponential fit of the correlation function
`-endfit`	real	`-1`	Time where to end the exponential fit of the correlation function, -1 is till the end

where M1 and M2 are the two covariance matrices and tr is the trace of a matrix. The numbers are proportional to the overlap of the square root of the fluctuations. The normalized overlap is the most useful number, it is 1 for identical matrices and 0 when the sampled subspaces are orthogonal.

Files

`-v`	`eigenvec.trr`	Input	Full precision trajectory: trr trj
`-v2`	`eigenvec2.trr`	Input, Opt.	Full precision trajectory: trr trj
`-f`	`traj.xtc`	Input, Opt.	Generic trajectory: xtc trr trj gro g96 pdb
`-s`	`topol.tpr`	Input, Opt.	Structure+mass(db): tpr tpb tpa gro g96 pdb
`-n`	`index.ndx`	Input, Opt.	Index file
`-eig1`	`eigenval1.xvg`	Input, Opt.	xvgr/xmgr file
`-eig2`	`eigenval2.xvg`	Input, Opt.	xvgr/xmgr file
`-disp`	`eigdisp.xvg`	Output, Opt.	xvgr/xmgr file
`-proj`	`proj.xvg`	Output, Opt.	xvgr/xmgr file
`-2d`	`2dproj.xvg`	Output, Opt.	xvgr/xmgr file
`-3d`	`3dproj.pdb`	Output, Opt.	Generic structure: gro g96 pdb
`-filt`	`filtered.xtc`	Output, Opt.	Generic trajectory: xtc trr trj gro g96 pdb
`-extr`	`extreme.pdb`	Output, Opt.	Generic trajectory: xtc trr trj gro g96 pdb
`-over`	`overlap.xvg`	Output, Opt.	xvgr/xmgr file
`-inpr`	`inprod.xpm`	Output, Opt.	X PixMap compatible matrix file

Other options

`-h`	bool	`no`	Print help info and quit
`-nice`	int	`19`	Set the nicelevel
`-b`	time	`-1`	First frame (ps) to read from trajectory
`-e`	time	`-1`	Last frame (ps) to read from trajectory
`-dt`	time	`-1`	Only use frame when t MOD dt = first time (ps)
`-tu`	enum	`ps`	Time unit: `ps`, `fs`, `ns`, `us`, `ms`, `s`, `m` or `h`
`-w`	bool	`no`	View output xvg, xpm, eps and pdb files
`-first`	int	`1`	First eigenvector for analysis (-1 is select)
`-last`	int	`8`	Last eigenvector for analysis (-1 is till the last)
`-skip`	int	`1`	Only analyse every nr-th frame
`-max`	real	`0`	Maximum for projection of the eigenvector on the average structure, max=0 gives the extremes
`-nframes`	int	`2`	Number of frames for the extremes output
`-split`	bool	`no`	Split eigenvector projections where time is zero

E.6 g_analyze

g_analyze reads an ascii file and analyzes data sets. A line in the input file may start with a time (see option `-time`) and any number of y values may follow. Multiple sets can also be read when they are seperated by & (option `-n`), in this case only one y value is read from each line. All lines starting with # and @ are skipped. All analyses can also be done for the derivative of a set (option `-d`).

All options, except for `-av` and `-power` assume that the points are equidistant in time.

g_analyze always shows the average and standard deviation of each set. For each set it also shows the relative deviation of the third and forth cumulant from those of a Gaussian distribution with the same standard deviation.

Option `-ac` produces the autocorrelation function(s).

Option `-cc` plots the resemblance of set i with a cosine of i/2 periods. The formula is:
2 (int0-T y(t) cos(pi t/i) dt)^2 / int0-T y(t) y(t) dt

Chapter 4

Force fields

A force field is built up from two distinct components:

- The set of equations (called the *potential functions*) used to generate the potential energies and their derivatives, the forces.
- The parameters used in this set of equations

Within one set of equations various sets of parameters can be used. Care must be taken that the combination of equations and parameters form a consistent set. It is in general dangerous to make *ad hoc* changes in a subset of parameters, because the various contributions to the total force are usually interdependent.

In GROMACS 3.0 the force field is based on GROMOS-87 [39], with a small modification concerning the interaction between water-oxygens and carbon atoms [40, 41], as well as 10 extra atom types [42, 43, 40, 41, 44]. However, the user is free to make her own modifications (beware!). This will be explained in details in chapter 5, which deals with the **Topology**.
To accommodate the potential functions used in some popular force fields, GROMACS offers a choice of functions, both for non-bonded interaction and for dihedral interactions. They are described in the appropriate subsections.

The potential functions can be subdivided into three parts

1. *Non-bonded*: Lennard-Jones or Buckingham, and Coulomb or modified Coulomb. The non-bonded interactions are computed on the basis of a neighbor list (a list of non-bonded atoms within a certain radius), in which exclusions are already removed.
2. *Bonded*: covalent bond-stretching, angle-bending, improper dihedrals, and proper dihedrals. These are computed on the basis of fixed lists.
3. *Special*: position restraints and distance restraints, based on fixed lists.

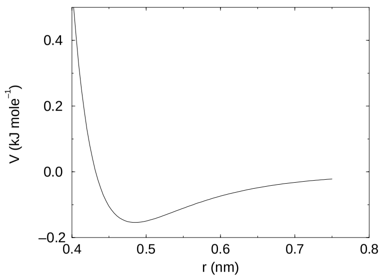


Figure 4.1: The Lennard-Jones interaction.

4.1 Non-bonded interactions

Non-bonded interactions in GROMACS are pair-additive and centro-symmetric:

$$V(\boldsymbol{r}_1, \ldots \boldsymbol{r}_N) = \sum_{i<j} V_{ij}(\boldsymbol{r}_{ij}); \tag{4.1}$$

$$\boldsymbol{F}_i = -\sum_j \frac{dV_{ij}(r_{ij})}{dr_{ij}} \frac{\boldsymbol{r}_{ij}}{r_{ij}} = -\boldsymbol{F}_j \tag{4.2}$$

The non-bonded interactions contain a repulsion term, a dispersion term, and a Coulomb term. The repulsion and dispersion term are combined in either the Lennard-Jones (or 6-12 interaction), or the Buckingham (or exp-6 potential). In addition, (partially) charged atoms act through the Coulomb term.

4.1.1 The Lennard-Jones interaction

The Lennard-Jones potential V_{LJ} between two atoms equals

$$V_{LJ}(r_{ij}) = \frac{C_{ij}^{(12)}}{r_{ij}^{12}} - \frac{C_{ij}^{(6)}}{r_{ij}^{6}} \tag{4.3}$$

see also Fig. 4.1 The parameters $C_{ij}^{(12)}$ and $C_{ij}^{(6)}$ depend on pairs of *atom types*; consequently they are taken from a matrix of LJ-parameters.

The force derived from this potential is:

$$\boldsymbol{F}_i(\boldsymbol{r}_{ij}) = \left(12\, \frac{C_{ij}^{(12)}}{r_{ij}^{12}} - 6\, \frac{C_{ij}^{(6)}}{r_{ij}^{6}} \right) \frac{\boldsymbol{r}_{ij}}{r_{ij}} \tag{4.4}$$

`-nice`	int	`19`	Set the nicelevel
`-b`	real	`-1`	First time to use
`-e`	real	`-1`	Last time to use
`-dt`	real	`0`	Only write out frame when t MOD dt = offset
`-offset`	real	`0`	Time offset for -dt option
`-settime`	bool	`no`	Change starting time interactively
`-sort`	bool	`yes`	Sort energy files (not frames)
`-scalefac`	real	`1`	Multiply energy component by this factor
`-error`	bool	`yes`	Stop on errors in the file

- When combining trajectories the sigma and E^2 (necessary for statistics) are not updated correctly. Only the actual energy is correct. One thus has to compute statistics in another way.

E.5 g_anaeig

`g_anaeig` analyzes eigenvectors. The eigenvectors can be of a covariance matrix (`g_covar`) or of a Normal Modes anaysis (`g_nmeig`).

When a trajectory is projected on eigenvectors, all structures are fitted to the structure in the eigenvector file, if present, otherwise to the structure in the structure file. When no run input file is supplied, periodicity will not be taken into account. Most analyses are performed on eigenvectors `-first` to `-last`, but when `-first` is set to -1 you will be prompted for a selection.

`-disp`: plot all atom displacements of eigenvectors `-first` to `-last`.

`-proj`: calculate projections of a trajectory on eigenvectors `-first` to `-last`. The projections of a trajectory on the eigenvectors of its covariance matrix are called principal components (pc's). It is often useful to check the cosine content the pc's, since the pc's of random diffusion are cosines with the number of periods equal to half the pc index. The cosine content of the pc's can be calculated with the program `g_analyze`.

`-2d`: calculate a 2d projection of a trajectory on eigenvectors `-first` and `-last`.

`-3d`: calculate a 3d projection of a trajectory on the first three selected eigenvectors.

`-filt`: filter the trajectory to show only the motion along eigenvectors `-first` to `-last`.

`-extr`: calculate the two extreme projections along a trajectory on the average structure and interpolate `-nframes` frames between them, or set your own extremes with `-max`. The eigenvector `-first` will be written unless `-first` and `-last` have been set explicitly, in which case all eigenvectors will be written to separate files. Chain identifiers will be added when writing a `.pdb` file with two or three structures (you can use `rasmol -nmrpdb` to view such a pdb file).

Overlap calculations between covariance analysis:
NOTE: the analysis should use the same fitting structure

`-over`: calculate the subspace overlap of the eigenvectors in file `-v2` with eigenvectors `-first` to `-last` in file `-v`.

`-inpr`: calculate a matrix of inner-products between eigenvectors in files `-v` and `-v2`. All eigenvectors of both files will be used unless `-first` and `-last` have been set explicitly.

When `-v`, `-eig1`, `-v2` and `-eig2` are given, a single number for the overlap between the covariance matrices is generated. The formulas are:
difference = sqrt(tr((sqrt(M1) - sqrt(M2))^2))
normalized overlap = 1 - difference/sqrt(tr(M1) + tr(M2))
shape overlap = 1 - sqrt(tr((sqrt(M1/tr(M1)) - sqrt(M2/tr(M2)))^2))

`-f`	`conf.gro`	Input	Generic structure: gro g96 pdb tpr tpb tpa
`-n`	`index.ndx`	Input, Opt.	Index file
`-o`	`out.gro`	Output	Generic structure: gro g96 pdb
`-bf`	`bfact.dat`	Input, Opt.	Generic data file

Other options

`-h`	bool	`no`	Print help info and quit
`-nice`	int	`0`	Set the nicelevel
`-w`	bool	`no`	View output xvg, xpm, eps and pdb files
`-ndef`	bool	`no`	Choose output from default index groups
`-bt`	enum	`tric`	Box type for -box and -d: `tric`, `cubic`, `dodecahedron` or `octahedron`
`-box`	vector	`0 0 0`	Box vector lengths (a,b,c)
`-angles`	vector	`90 90 90`	Angles between the box vectors (bc,ac,ab)
`-d`	real	`0`	Distance between the solute and the box
`-c`	bool	`no`	Center molecule in box (implied by -box and -d)
`-center`	vector	`0 0 0`	Coordinates of geometrical center
`-rotate`	vector	`0 0 0`	Rotation around the X, Y and Z axes in degrees
`-princ`	bool	`no`	Orient molecule(s) along their principal axes
`-scale`	vector	`1 1 1`	Scaling factor
`-density`	real	`1000`	Density (g/l) of the output box achieved by scaling
`-pbc`	bool	`no`	Remove the periodicity (make molecule whole again)
`-mead`	bool	`no`	Store the charge of the atom in the occupancy field and the radius of the atom in the B-factor field
`-grasp`	bool	`no`	Store the charge of the atom in the B-factor field and the radius of the atom in the occupancy field
`-rvdw`	real	`0.12`	Default Van der Waals radius if one can not be found in the database
`-atom`	bool	`no`	Force B-factor attachment per atom
`-legend`	bool	`no`	Make B-factor legend
`-label`	string	`A`	Add chain label for all residues

- For complex molecules, the periodicity removal routine may break down, in that case you can use trjconv

E.4 eneconv

When `-f` is *not* specified:
Concatenates several energy files in sorted order. In case of double time frames the one in the later file is used. By specifying `-settime` you will be asked for the start time of each file. The input files are taken from the command line, such that the command `eneconv -o fixed.edr *.edr` should do the trick.

With `-f` specified:
Reads one energy file and writes another, applying the `-dt`, `-offset`, `-t0` and `-settime` options and converting to a different format if necessary (indicated by file extentions).

`-settime` is applied first, then `-dt`/`-offset` followed by `-b` and `-e` to select which frames to write.

Files

`-f`	`ener.edr`	Input	Generic energy: edr ene
`-o`	`fixed.edr`	Output, Opt.	Generic energy: edr ene

Other options

`-h`	bool	`no`	Print help info and quit

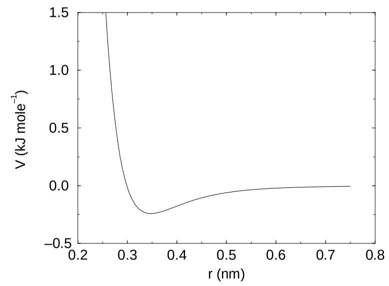


Figure 4.2: The Buckingham interaction.

The LJ potential may also be written in the following form :

$$V_{LJ}(\boldsymbol{r}_{ij}) = 4\epsilon_{ij}\left(\left(\frac{\sigma_{ij}}{r_{ij}}\right)^{12} - \left(\frac{\sigma_{ij}}{r_{ij}}\right)^{6}\right) \tag{4.5}$$

In constructing the parameter matrix for the non-bonded LJ-parameters, two types of combination rules can be used within GROMACS:

$$\begin{aligned} C_{ij}^{(6)} &= \left(C_{ii}^{(6)} * C_{jj}^{(6)}\right)^{1/2} \\ C_{ij}^{(12)} &= \left(C_{ii}^{(12)} * C_{jj}^{(12)}\right)^{1/2} \end{aligned} \tag{4.6}$$

or, alternatively,

$$\begin{aligned} \sigma_{ij} &= \tfrac{1}{2}(\sigma_{ii} + \sigma_{jj}) \\ \epsilon_{ij} &= (\epsilon_{ii}\epsilon_{jj})^{1/2} \end{aligned} \tag{4.7}$$

4.1.2 Buckingham potential

The Buckingham potential has a more flexible and realistic repulsion term than the Lennard-Jones interaction, but is also more expensive to compute. The potential form is:

$$V_{bh}(r_{ij}) = A_{ij}\exp(-B_{ij}r_{ij}) - \frac{C_{ij}}{r_{ij}^{6}} \tag{4.8}$$

see also Fig. 4.2, the force derived from this is:

$$\boldsymbol{F}_i(r_{ij}) = \left[-A_{ij}B_{ij}r_{ij}\exp(-B_{ij}r_{ij}) - 6\frac{C_{ij}}{r_{ij}^{6}}\right]\frac{\boldsymbol{r}_{ij}}{r_{ij}} \tag{4.9}$$

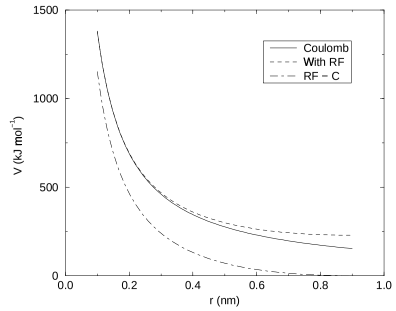


Figure 4.3: The Coulomb interaction (for particles with equal signed charge) with and without reaction field. In the latter case ε_{rf} was 78, and r_c was 0.9 nm. The dot-dashed line is the same as the dashed line, except for a constant.

4.1.3 Coulomb interaction

The Coulomb interaction between two charge particles is given by:

$$V_c(r_{ij}) = f\frac{q_i q_j}{\varepsilon_r r_{ij}} \tag{4.10}$$

see also Fig. 4.3, where $f = \frac{1}{4\pi\varepsilon_0} = 138.935\,485$ (see chapter 2)

The force derived from this potential is:

$$\boldsymbol{F}_i(\boldsymbol{r}_{ij}) = f\frac{q_i q_j}{\varepsilon_r r_{ij}^2}\frac{\boldsymbol{r}_{ij}}{r_{ij}} \tag{4.11}$$

In GROMACS the relative dielectric constant ε_r may be set in the in the input for `grompp`.

4.1.4 Coulomb interaction with reaction field

The coulomb interaction can be modified for homogeneous systems, by assuming a constant dielectric environment beyond the cut-off r_c with a dielectric constant of ε_{rf}. The interaction then reads:

$$V_{crf} \;=\; f\frac{q_i q_j}{r_{ij}}\left[1+\frac{\varepsilon_{rf}-1}{2\varepsilon_{rf}+1}\frac{r_{ij}^3}{r_c^3}\right] - f\frac{q_i q_j}{r_c}\frac{3\varepsilon_{rf}}{2\varepsilon_{rf}+1} \tag{4.12}$$

in which the constant expression on the right makes the potential zero at the cut-off r_c. We can rewrite this for simplicity as

$$V_{crf} \;=\; f q_i q_j\left[\frac{1}{r_{ij}} + k_{rf}\, r_{ij}^2 - c_{rf}\right] \tag{4.13}$$

E.3 editconf

editconf converts generic structure format to `.gro`, `.g96` or `.pdb`.

The box can be modified with options `-box`, `-d` and `-angles`. Both `-box` and `-d` will center the system in the box.

Option `-bt` determines the box type: `tric` is a triclinic box, `cubic` is a cubic box, `dodecahedron` is a rhombic dodecahedron and `octahedron` is a truncated octahedron. The last two are special cases of a triclinic box. The length of the three box vectors of the truncated octahedron is the shortest distance between two opposite hexagons. The volume of a dodecahedron is 0.71 and that of a truncated octahedron is 0.77 of that of a cubic box with the same periodic image distance.

Option `-box` requires only one value for a cubic box, dodecahedron and a truncated octahedron. With `-d` and `tric` the size of the system in the x, y and z directions is used. With `-d` and `cubic`, `dodecahedron` or `octahedron` the diameter of the system is used, which is the largest distance between two atoms.

Option `-angles` is only meaningful with option `-box` and a triclinic box and can not be used with option `-d`.

When `-n` or `-ndef` is set, a group can be selected for calculating the size and the geometric center, otherwise the whole system is used.

`-rotate` rotates the coordinates and velocities. `-princ` aligns the principal axes of the system along the coordinate axes, this may allow you to decrease the box volume, but beware that molecules can rotate significantly in a nanosecond.

Scaling is applied before any of the other operations are performed. Boxes can be scaled to give a certain density (option `-density`). A special feature of the scaling option, when the factor -1 is given in one dimension, one obtains a mirror image, mirrored in one of the plains, when one uses -1 in three dimensions a point-mirror image is obtained.

Groups are selected after all operations have been applied.

Periodicity can be removed in a crude manner. It is important that the box sizes at the bottom of your input file are correct when the periodicity is to be removed.

The program can optionally rotate the solute molecule to align the molecule along its principal axes (`-rotate`)

When writing `.pdb` files, B-factors can be added with the `-bf` option. B-factors are read from a file with with following format: first line states number of entries in the file, next lines state an index followed by a B-factor. The B-factors will be attached per residue unless an index is larger than the number of residues or unless the `-atom` option is set. Obviously, any type of numeric data can be added instead of B-factors. `-legend` will produce a row of CA atoms with B-factors ranging from the minimum to the maximum value found, effectively making a legend for viewing.

With the option -mead a special pdb file for the MEAD electrostatics program (Poisson-Boltzmann solver) can be made. A further prerequisite is that the input file is a run input file. The B-factor field is then filled with the Van der Waals radius of the atoms while the occupancy field will hold the charge.

The option -grasp is similar, but it puts the charges in the B-factor and the radius in the occupancy.

Finally with option `-label` editconf can add a chain identifier to a pdb file, which can be useful for analysis with e.g. rasmol.

To convert a truncated octrahedron file produced by a package which uses a cubic box with the corners cut off (such as Gromos) use:

```
editconf -f <in> -rotate 0 -45 -35.264 -bt o -box <veclen> -o <out>
```

where `veclen` is the size of the cubic box times sqrt(3)/2.

Files

- Most GROMACS programs can process a trajectory with less atoms than the run input or structure file, but only if the trajectory consists of the first n atoms of the run input or structure file.
- Many GROMACS programs will accept the `-tu` option to set the time units to use in output files (e.g. for `xmgr` graphs or `xpm` matrices) and in all time options.

E.2 do_dssp

do_dssp reads a trajectory file and computes the secondary structure for each time frame calling the dssp program. If you do not have the dssp program, get it. do_dssp assumes that the dssp executable is in /home/mdgroup/dssp/dssp. If that is not the case, then you should set an environment variable **DSSP** pointing to the dssp executable as in:

```
setenv DSSP /usr/local/bin/dssp
```

The structure assignment for each residue and time is written to an `.xpm` matrix file. This file can be visualized with for instance `xv` and can be converted to postscript with `xpm2ps`. The number of residues with each secondary structure type and the total secondary structure (`-sss`) count as a function of time are also written to file (`-sc`).

Solvent accessible surface (SAS) per residue can be calculated, both in absolute values (A^2) and in fractions of the maximal accessible surface of a residue. The maximal accessible surface is defined as the accessible surface of a residue in a chain of glycines. **Note** that the program `g_sas` can also compute SAS and that is more efficient.

Finally, this program can dump the secondary structure in a special file `ssdump.dat` for usage in the program `g_chi`. Together these two programs can be used to analyze dihedral properties as a function of secondary structure type.

Files

`-f`	`traj.xtc`	Input	Generic trajectory: xtc trr trj gro g96 pdb
`-s`	`topol.tpr`	Input	Structure+mass(db): tpr tpb tpa gro g96 pdb
`-n`	`index.ndx`	Input, Opt.	Index file
`-ssdump`	`ssdump.dat`	Output, Opt.	Generic data file
`-map`	`ss.map`	Input, Lib.	File that maps matrix data to colors
`-o`	`ss.xpm`	Output	X PixMap compatible matrix file
`-sc`	`scount.xvg`	Output	xvgr/xmgr file
`-a`	`area.xpm`	Output, Opt.	X PixMap compatible matrix file
`-ta`	`totarea.xvg`	Output, Opt.	xvgr/xmgr file
`-aa`	`averarea.xvg`	Output, Opt.	xvgr/xmgr file

Other options

`-h`	bool	`no`	Print help info and quit
`-nice`	int	`19`	Set the nicelevel
`-b`	time	`-1`	First frame (ps) to read from trajectory
`-e`	time	`-1`	Last frame (ps) to read from trajectory
`-dt`	time	`-1`	Only use frame when t MOD dt = first time (ps)
`-tu`	enum	`ps`	Time unit: `ps`, `fs`, `ns`, `us`, `ms`, `s`, `m` or `h`
`-w`	bool	`no`	View output xvg, xpm, eps and pdb files
`-sss`	string	`HEBT`	Secondary structures for structure count

- The program is very slow

---

with

$$k_{rf} = \frac{1}{r_c^3}\frac{\varepsilon_{rf}-1}{(2\varepsilon_{rf}+1)} \tag{4.14}$$

$$c_{rf} = \frac{1}{r_c}+k_{rf}\,r_c^2 = \frac{1}{r_c}\frac{3\varepsilon_{rf}}{(2\varepsilon_{rf}+1)} \tag{4.15}$$

for large ε_{rf} the k_{rf} goes to 0.5 r_c^{-3}, while for ε_{rf} = 1 the correction vanishes. This makes it possible to use the same expression with and without reaction field, albeit at some computational cost. In Fig. 4.3 the modified interaction is plotted, and it is clear that the derivative with respect to r_{ij} (= -force) goes to zero at the cut-off distance. The force derived from this potential reads:

$$\boldsymbol{F}_i(\boldsymbol{r}_{ij}) = f q_i q_j \left[\frac{1}{r_{ij}^2} - 2k_{rf}r_{ij}\right]\frac{\boldsymbol{r}_{ij}}{r_{ij}} \tag{4.16}$$

Tironi *et al.* have introduced a generalized reaction field in which the dielectric continuum beyond the cut-off r_c also has an ionic strength I [45]. In this case we can rewrite the constants k_{rf} and c_{rf} using the inverse Debye screening length κ:

$$\kappa = \frac{2I\,F^2}{\varepsilon_0\varepsilon_{rf}RT} = \frac{F^2}{\varepsilon_0\varepsilon_{rf}RT}\sum_{i=1}^{K} c_i z_i \tag{4.17}$$

$$k_{rf} = \frac{1}{r_c^3}\frac{(\varepsilon_{rf}-1)(1+\kappa r_c)+\varepsilon_{rf}(\kappa r_c)^2}{(2\varepsilon_{rf}+1)(1+\kappa r_c)+2\varepsilon_{rf}(\kappa r_c)^2} \tag{4.18}$$

$$c_{rf} = \frac{1}{r_c}\frac{3\varepsilon_{rf}(1+\kappa r_c+(\kappa r_c)^2)}{(2\varepsilon_{rf}+1)(1+\kappa r_c)+2\varepsilon_{rf}(\kappa r_c)^2} \tag{4.19}$$

where F is Faraday's constant, R is the ideal gas constant, T the absolute temperature, c_i the molar concentration for species i and z_i the charge number of species i where we have K different species. In the limit of zero ionic strength (κ = 0) eqns. 4.18 and 4.19 reduce to the simple forms of eqns. 4.14 and 4.15 respectively.

4.1.5 Modified non-bonded interactions

In the GROMACS force field the non-bonded potentials can be modified by a shift function. The purpose of this is to replace the truncated forces by forces that are continuous and have continuous derivatives at the cut-off radius. With such forces the time-step integration produces much smaller errors and there are no such complications as creating charges from dipoles by the truncation procedure. In fact, by using shifted forces there is no need for charge groups in the construction of neighbor lists. However, the shift function produces a considerable modification of the Coulomb potential. Unless the 'missing' long-range potential is properly calculated and added (through the use of PPPM, Ewald, or PME), the effect of such modifications must be carefully evaluated. The modification of the Lennard-Jones dispersion and repulsion is only minor, but it does remove the noise caused by cut-off effects.

There is *no* fundamental difference between a switch function (which multiplies the potential with a function) and a shift function (which adds a function to the force or potential). The switch function is a special case of the shift function, which we apply to the *force function* $F(r)$, related to the electrostatic or Van der Waals force acting on particle i by particle j as

$$\boldsymbol{F}_i = cF(r_{ij})\frac{\boldsymbol{r}_{ij}}{r_{ij}} \tag{4.20}$$

For pure Coulomb or Lennard-Jones interactions $F(r) = F_\alpha(r) = r^{-(\alpha+1)}$. The shifted force $F_s(r)$ can generally be written as:

$$\begin{array}{rclr} F_s(r) & = & F_\alpha(r) & r < r_1 \\ F_s(r) & = & F_\alpha(r) + S(r) & r_1 \le r < r_c \\ F_s(r) & = & 0 & r_c \le r \end{array} \tag{4.21}$$

When $r_1 = 0$ this is a traditional shift function, otherwise it acts as a switch function. The corresponding shifted coulomb potential then reads:

$$V_s(r_{ij}) = f\Phi_s(r_{ij})q_iq_j \tag{4.22}$$

where $\Phi(r)$ is the potential function

$$\Phi_s(r) = \int_r^\infty F_s(x)\,dx \tag{4.23}$$

The GROMACS shift function should be smooth at the boundaries, therefore the following boundary conditions are imposed on the shift function:

$$\begin{array}{rcl} S(r_1) & = & 0 \\ S'(r_1) & = & 0 \\ S(r_c) & = & -F_\alpha(r_c) \\ S'(r_c) & = & -F'_\alpha(r_c) \end{array} \tag{4.24}$$

A 3^{rd} degree polynomial of the form

$$S(r) = A(r-r_1)^2 + B(r-r_1)^3 \tag{4.25}$$

fulfills these requirements. The constants A and B are given by the boundary condition at r_c:

$$\begin{array}{rcl} A & = & -\dfrac{(\alpha+4)r_c \;-\; (\alpha+1)r_1}{r_c^{\alpha+2}\,(r_c-r_1)^2} \\ B & = & \dfrac{(\alpha+3)r_c \;-\; (\alpha+1)r_1}{r_c^{\alpha+2}\,(r_c-r_1)^3} \end{array} \tag{4.26}$$

Thus the total force function is

$$F_s(r) = \frac{1}{r^{\alpha+1}} + A(r-r_1)^2 + B(r-r_1)^3 \tag{4.27}$$

and the potential function reads

$$\Phi(r) = \frac{1}{r^\alpha} - \frac{A}{3}(r-r_1)^3 - \frac{B}{4}(r-r_1)^4 - C \tag{4.28}$$

Appendix E

Manual Pages

E.1 options

All GROMACS programs have 6 standard options, of which some are hidden by default:

Other options

`-h`	bool	`no`	Print help info and quit
`-X`	bool	`no`	Use dialog box GUI to edit command line options
`-nice`	int	`0`	Set the nicelevel

- If the configuration script found Motif or Lesstif on your system, you can use the graphical interface (if not, you will get an error):
  `-X` bool `no` Use dialog box GUI to edit command line options
- When compiled on an SGI-IRIX system, all GROMACS programs have an additional option:
  `-npri` int `0` Set non blocking priority (try 128)
- Optional files are not used unless the option is set, in contrast to non optional files, where the default file name is used when the option is not set.
- All GROMACS programs will accept file options without a file extension or filename being specified. In such cases the default filenames will be used. With multiple input file types, such as generic structure format, the directory will be searched for files of each type with the supplied or default name. When no such file is found, or with output files the first file type will be used.
- All GROMACS programs with the exception of `mdrun`, `nmrun` and `eneconv` check if the command line options are valid. If this is not the case, the program will be halted.
- Enumerated options (enum) should be used with one of the arguments listed in the option description, the argument may be abbreviated. The first match to the shortest argument in the list will be selected.
- Vector options can be used with 1 or 3 parameters. When only one parameter is supplied the two others are also set to this value.
- For many GROMACS programs, the time options can be supplied in different time units, depending on the setting of the `-tu` option.
- All GROMACS programs can read compressed or g-zipped files. There might be a problem with reading compressed `.xtc`, `.trr` and `.trj` files, but these will not compress very well anyway.

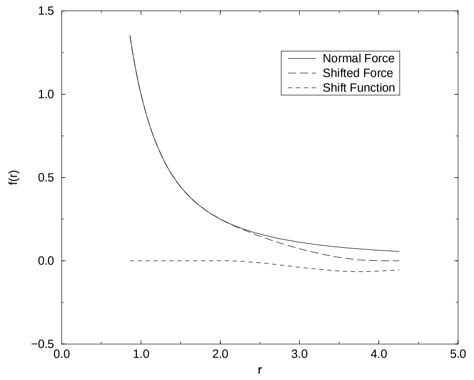


Figure 4.4: The Coulomb Force, Shifted Force and Shift Function $S(r)$, using r_1 = 2 and r_c = 4.

where

$$C = \frac{1}{r_c^\alpha} - \frac{A}{3}(r_c - r_1)^3 - \frac{B}{4}(r_c - r_1)^4 \tag{4.29}$$

When r_1 = 0, the modified Coulomb force function is

$$F_s(r) = \frac{1}{r^2} - \frac{5r^2}{r_c^4} + \frac{4r^3}{r_c^5} \tag{4.30}$$

identical to the *parabolic force* function recommended to be used as a short-range function in conjunction with a Poisson solver for the long-range part [13]. The modified Coulomb potential function is

$$\Phi(r) = \frac{1}{r} - \frac{5}{3r_c} + \frac{5r^3}{3r_c^4} - \frac{r^4}{r_c^5} \tag{4.31}$$

see also Fig. 4.4.

4.1.6 Modified short-range interactions with Ewald summation

When Ewald summation or particle-mesh Ewald is used to calculate the long-range interactions, the short-range coulomb potential must also be modified, similar to the switch function above. In this case the short range potential is given by

$$V(r) = f\frac{\mathrm{erfc}(\beta r_{ij})}{r_{ij}} q_i q_j, \tag{4.32}$$

where β is a parameter that determines the relative weight between the direct space sum and the reciprocal space sum and erfc(x) is the complementary error function. For further details on long-range electrostatics, see sec. 4.6.

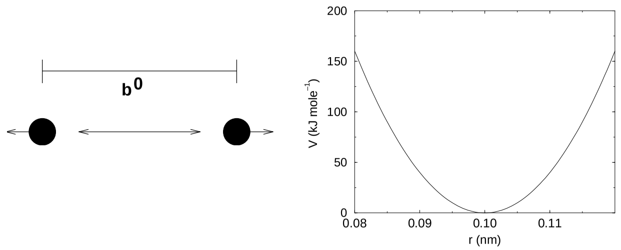


Figure 4.5: Principle of bond stretching (left), and the bond stretching potential (right).

4.2 Bonded interactions

Bonded interactions are based on a fixed list of atoms. They are not exclusively pair interactions, but include 3- and 4-body interactions as well. There are *bond stretching* (2-body), *bond angle* (3-body), and *dihedral angle* (4-body) interactions. A special type of dihedral interaction (called *improper dihedral*) is used to force atoms to remain in a plane or to prevent transition to a configuration of opposite chirality (a mirror image).

4.2.1 Bond stretching

Harmonic potential

The bond stretching between two covalently bonded atoms i and j is represented by a harmonic potential

$$V_b\,(r_{ij}) = \frac{1}{2}k^b_{ij}(r_{ij}-b_{ij})^2 \tag{4.33}$$

see also Fig. 4.5, with the force

$$\boldsymbol{F}_i(\boldsymbol{r}_{ij}) = k^b_{ij}(r_{ij}-b_{ij})\frac{\boldsymbol{r}_{ij}}{r_{ij}} \tag{4.34}$$

Fourth power potential

In the GROMOS-96 force field [46] the covalent bond potential is written for reasons of computational efficiency as:

$$V_b\,(r_{ij}) = \frac{1}{4}k^b_{ij}\left(r^2_{ij}-b^2_{ij}\right)^2 \tag{4.35}$$

the corresponding force is:

$$\boldsymbol{F}_i(\boldsymbol{r}_{ij}) = k^b_{ij}(r^2_{ij}-b^2_{ij})\;\boldsymbol{r}_{ij} \tag{4.36}$$

---

which we can expand to:

$$\sum_{i=m}^{n}\left[\sum_{s=1}^{S}(x_i^s)^2 + \left(\frac{X^S_{m,n}}{m-n+1}\right)^2 - 2\left(\frac{X^S_{m,n}}{m-n+1}\sum_{s=1}^{S}x_i^s + \sum_{s=1}^{S}\sum_{s'=s+1}^{S}x_i^s x_i^{s'}\right)\right]$$
$$-\sum_{s=1}^{S}\sum_{i=m}^{n}\left[(x_i^s)^2 - 2\frac{X^s_{m,n}}{m-n+1}x_i^s + \left(\frac{X^s_{m,n}}{m-n+1}\right)^2\right] \;=\; \Delta\sigma \tag{D.31}$$

the terms with $(x_i^s)^2$ cancel, so that we can simplify to:

$$\frac{\left(X^S_{m,n}\right)^2}{m-n+1} - 2\frac{X^S_{m,n}}{m-n+1}\sum_{i=m}^{n}\sum_{s=1}^{S}x_i^s - 2\sum_{i=m}^{n}\sum_{s=1}^{S}\sum_{s'=s+1}^{S}x_i^s x_i^{s'} -$$
$$\sum_{s=1}^{S}\sum_{i=m}^{n}\left[-2\frac{X^s_{m,n}}{m-n+1}x_i^s + \left(\frac{X^s_{m,n}}{m-n+1}\right)^2\right] \;=\; \Delta\sigma \tag{D.32}$$

or

$$-\frac{\left(X^S_{m,n}\right)^2}{m-n+1} - 2\sum_{i=m}^{n}\sum_{s=1}^{S}\sum_{s'=s+1}^{S}x_i^s x_i^{s'} + \sum_{s=1}^{S}\frac{\left(X^s_{m,n}\right)^2}{m-n+1} \;=\; \Delta\sigma \tag{D.33}$$

If we now expand the first term using eqn. D.28 we obtain:

$$-\frac{\left(\sum_{s=1}^{S}X^s_{m,n}\right)^2}{m-n+1} - 2\sum_{i=m}^{n}\sum_{s=1}^{S}\sum_{s'=s+1}^{S}x_i^s x_i^{s'} + \sum_{s=1}^{S}\frac{\left(X^s_{m,n}\right)^2}{m-n+1} \;=\; \Delta\sigma \tag{D.34}$$

which we can reformulate to:

$$-2\left[\sum_{s=1}^{S}\sum_{s'=s+1}^{S}X^s_{m,n}X^{s'}_{m,n} + \sum_{i=m}^{n}\sum_{s=1}^{S}\sum_{s'=s+1}^{S}x_i^s x_i^{s'}\right] \;=\; \Delta\sigma \tag{D.35}$$

or

$$-2\left[\sum_{s=1}^{S}X^s_{m,n}\sum_{s'=s+1}^{S}X^{s'}_{m,n} + \sum_{s=1}^{S}\sum_{i=m}^{n}x_i^s\sum_{s'=s+1}^{S}x_i^{s'}\right] \;=\; \Delta\sigma \tag{D.36}$$

which gives

$$-2\sum_{s=1}^{S}\left[X^s_{m,n}\sum_{s'=s+1}^{S}\sum_{i=m}^{n}x_i^{s'} + \sum_{i=m}^{n}x_i^s\sum_{s'=s+1}^{S}x_i^{s'}\right] \;=\; \Delta\sigma \tag{D.37}$$

Since we need all data points i to evaluate this, in general this is not possible. We can then make an estimate of $\sigma^S_{m,n}$ using only the data points that are available using the left hand side of eqn. D.30. While the average can be computed using all time steps in the simulation, the accuracy of the fluctuations is thus limited by the frequency with which energies are saved. Since this can be easily done with a program such as xmgr this is not built-in in GROMACS.

and thus

$$\sum_{i=1}^{n+m}\left[x_i^{AB}-\frac{X_{1,n+m}^{AB}}{n+m}\right]^2 = \sum_{i=1}^{n}\left[x_i^{A}-\frac{X_{1,n}^{A}}{n}\right]^2+\sum_{i=1}^{m}\left[x_i^{B}-\frac{X_{1,m}^{B}}{m}\right]^2+\Delta\sigma \tag{D.24}$$

or

$$\begin{aligned}\sum_{i=1}^{n+m}\left[(x_i^{AB})^2-2x_i^{AB}\frac{X_{1,n+m}^{AB}}{n+m}+\left(\frac{X_{1,n+m}^{AB}}{n+m}\right)^2\right] & - \\ \sum_{i=1}^{n}\left[(x_i^{A})^2-2x_i^{A}\frac{X_{1,n}^{A}}{n}+\left(\frac{X_{1,n}^{A}}{n}\right)^2\right] & - \\ \sum_{i=1}^{m}\left[(x_i^{B})^2-2x_i^{B}\frac{X_{1,m}^{B}}{m}+\left(\frac{X_{1,m}^{B}}{m}\right)^2\right] & = \Delta\sigma\end{aligned} \tag{D.25}$$

all the x_i^2 terms drop out, and the terms independent of the summation counter i can be simplified:

$$\begin{aligned}\frac{\left(X_{1,n+m}^{AB}\right)^2}{n+m}-\frac{\left(X_{1,n}^{A}\right)^2}{n}-\frac{\left(X_{1,m}^{B}\right)^2}{m} & - \\ 2\,\frac{X_{1,n+m}^{AB}}{n+m}\sum_{i=1}^{n+m}x_i^{AB}+2\,\frac{X_{1,n}^{A}}{n}\sum_{i=1}^{n}x_i^{A}+2\,\frac{X_{1,m}^{B}}{m}\sum_{i=1}^{m}x_i^{B} & = \Delta\sigma\end{aligned} \tag{D.26}$$

we recognize the three partial sums on the second line and use eqn. D.21 to obtain:

$$\Delta\sigma = \frac{\left(mX_{1,n}^{A}-nX_{1,m}^{B}\right)^2}{nm(n+m)} \tag{D.27}$$

if we check this by inserting $m=1$ we get back eqn. D.11

D.2.3 Summing energy terms

The g_energy program can also sum energy terms into one, e.g. potential + kinetic = total. For the partial averages this is again easy if we have S energy components s:

$$X_{m,n}^{S} = \sum_{i=m}^{n}\sum_{s=1}^{S}x_i^s = \sum_{s=1}^{S}\sum_{i=m}^{n}x_i^s = \sum_{s=1}^{S}X_{m,n}^{s} \tag{D.28}$$

For the fluctuations it is less trivial again, considering for example that the fluctuation in potential and kinetic energy should cancel. Nevertheless we can try the same approach as before by writing:

$$\sigma_{m,n}^{S} = \sum_{s=1}^{S}\sigma_{m,n}^{s}+\Delta\sigma \tag{D.29}$$

if we fill in eqn. D.6:

$$\sum_{i=m}^{n}\left[\left(\sum_{s=1}^{S}x_i^s\right)-\frac{X_{m,n}^{S}}{m-n+1}\right]^2 = \sum_{s=1}^{S}\sum_{i=m}^{n}\left[\left(x_i^s\right)-\frac{X_{m,n}^{s}}{m-n+1}\right]^2+\Delta\sigma \tag{D.30}$$

The force constants for this form of the potential is related to the usual harmonic force constant $k^{b,harm}$ (sec. 4.2.1) as

$$2k^b b_{ij}^2 = k^{b,harm} \tag{4.37}$$

The force constants are mostly derived from the harmonic ones used in GROMOS-87 [39]. Although this form is computationally more efficient (because no square root has to be evaluated), it is conceptually more complex. One particular disadvantage is that since the form is not harmonic, the average energy of a single bond is not equal to $\frac{1}{2}kT$ as it is for the normal harmonic potential.

4.2.2 Morse potential bond stretching

For some systems that require an anharmonic bond stretching potential, the Morse potential [47] between two atoms *i* and *j* is available in GROMACS. This potential differs from the harmonic potential in having an asymmetric potential well and a zero force at infinite distance The functional form is:

$$V_{morse}(r_{ij}) = D_{ij}[1-\exp(-\beta_{ij}(r_{ij}-b_{ij}))]^2, \tag{4.38}$$

see also Fig. 4.6, and the corresponding force is:

$$\begin{aligned}\mathbf{F}_{morse}(\mathbf{r}_{ij}) \;=\; & 2D_{ij}\beta_{ij}r_{ij}\exp(-\beta_{ij}(r_{ij}-b_{ij}))* \\ & [1-\exp(-\beta_{ij}(r_{ij}-b_{ij}))]\frac{\mathbf{r}_{ij}}{r_{ij}},\end{aligned} \tag{4.39}$$

where D_{ij} is the depth of the well in kJ/mol, β_{ij} defines the steepness of the well (in nm^{-1}), and b_{ij} is the equilibrium distance in nm. The steepness parameter β_{ij} can be expressed in terms of the reduced mass of the atoms *i* and *j*, the fundamental vibration frequency ω_{ij} and the well depth D_{ij}:

$$\beta_{ij} = \omega_{ij}\sqrt{\frac{\mu_{ij}}{2D_{ij}}} \tag{4.40}$$

and because $\omega = \sqrt{k/\mu}$, one can rewrite β_{ij} in terms of the harmonic force constant k_{ij}

$$\beta_{ij} = \sqrt{\frac{k_{ij}}{2D_{ij}}} \tag{4.41}$$

For small deviations $(r_{ij}-b_{ij})$, one can expand the exp-term to first-order in the Taylor expansion:

$$\exp(-x) \approx 1-x \tag{4.42}$$

Substituting this in the functional from;

$$\begin{aligned}V_{morse}(r_{ij}) &= D_{ij}[1-\exp(-\beta_{ij}(r_{ij}-b_{ij}))]^2 \\ &= D_{ij}[1-(1-\sqrt{\tfrac{k_{ij}}{2D_{ij}}}(r_{ij}-b_{ij}))]^2 \\ &= \tfrac{1}{2}k_{ij}(r_{ij}-b_{ij}))^2,\end{aligned} \tag{4.43}$$

one recovers the harmonic bond stretching potential.

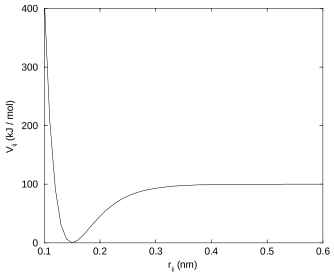


Figure 4.6: The Morse potential well, with bond length 0.15 nm.

4.2.3 Cubic bond stretching potential

Another anharmonic bond stretching potential that is slightly simpler than the Morse potential adds a cubic term in the distance to the simple harmonic form:

$$V_b\;(r_{ij}) = k^b_{ij}(r_{ij}-b_{ij})^2 + k^b_{ij}k^{cub}_{ij}(r_{ij}-b_{ij})^3 \tag{4.44}$$

A flexible water model (based on the SPC water model [48]) including a cubic bond stretching potential for the O-H bond was developed by Ferguson [49]. This model was found to yield a reasonable Infrared spectrum. The Ferguson water model is available in the GROMACS library. It should be noted that the potential is asymetric, overstretching leads to infinitely low energies. The integration timestep is therefore limited to 1 fs.

The force corresponding to this potential is:

$$\boldsymbol{F}_i(\boldsymbol{r}_{ij}) = 2k^b_{ij}(r_{ij}-b_{ij})\,\frac{\boldsymbol{r}_{ij}}{r_{ij}} + 3k^b_{ij}k^{cub}_{ij}(r_{ij}-b_{ij})^2\,\frac{\boldsymbol{r}_{ij}}{r_{ij}} \tag{4.45}$$

4.2.4 Harmonic angle potential

The bond angle vibration between a triplet of atoms i - j - k is also represented by a harmonic potential on the angle θ_{ijk}

$$V_a(\theta_{ijk}) = \frac{1}{2}k^{\theta}_{ijk}(\theta_{ijk}-\theta^0_{ijk})^2 \tag{4.46}$$

As the bond-angle vibration is represented by a harmonic potential the form is the same as the bond stretching (Fig. 4.5).

The force equations are given by the chain rule:

$$\begin{array}{rcl} \boldsymbol{F}_i & = & -\dfrac{dV_a(\theta_{ijk})}{d\boldsymbol{r}_i} \\ \boldsymbol{F}_k & = & -\dfrac{dV_a(\theta_{ijk})}{d\boldsymbol{r}_k} \\ \boldsymbol{F}_j & = & -\boldsymbol{F}_i - \boldsymbol{F}_k \end{array} \quad \text{where} \quad \theta_{ijk} = \arccos\frac{(\boldsymbol{r}_{ij}\cdot\boldsymbol{r}_{kj})}{r_{ij}r_{kj}} \tag{4.47}$$

D.2.1 Part of a Simulation

It is not uncommon to perform a simulation where the first part, e.g. 100 ps, is taken as equilibration. However, the averages and fluctuations as printed in the log file are computed over the whole simulation. The equilibration time, which is now part of the simulation, may in such a case invalidate the averages and fluctuations, because these numbers are now dominated by the initial drift towards equilibrium.

Using eqns. D.7 and D.8 the average and standard deviation over part of the trajectory can be computed as:

$$X_{m+1,m+k} \;=\; X_{1,m+k} - X_{1,m} \tag{D.15}$$

$$\sigma_{m+1,m+k} \;=\; \sigma_{1,m+k} - \sigma_{1,m} - \left[\,\frac{X_{1,m}}{m} - \frac{X_{1,m+k}}{m+k}\,\right]^2 \frac{m(m+k)}{k} \tag{D.16}$$

or, more generally (with $p \geq 1$ and $q \geq p$):

$$X_{p,q} \;=\; X_{1,q} - X_{1,p-1} \tag{D.17}$$

$$\sigma_{p,q} \;=\; \sigma_{1,q} - \sigma_{1,p-1} - \left[\,\frac{X_{1,p-1}}{p-1} - \frac{X_{1,q}}{q}\,\right]^2 \frac{(p-1)q}{q-p+1} \tag{D.18}$$

Note that implementation of this is not entirely trivial, since energies are not stored every time step of the simulation. We therefore have to construct $X_{1,p-1}$ and $\sigma_{1,p-1}$ from the information at time p using eqns. D.11 and D.12:

$$X_{1,p-1} \;=\; X_{1,p} - x_p \tag{D.19}$$

$$\sigma_{1,p-1} \;=\; \sigma_{1,p} - \frac{[\,X_{1,p-1} - (p-1)x_p\,]^2}{(p-1)p} \tag{D.20}$$

D.2.2 Combining two simulations

Another frequently occurring problem is, that the fluctuations of two simulations must be combined. Consider the following example: we have two simulations (A) of n and (B) of m steps, in which the second simulation is a continuation of the first. However, the second simulation starts numbering from 1 instead of from $n+1$. For the partial sum this is no problem, we have to add $X^A_{1,n}$ from run A:

$$X^{AB}_{1,n+m} \;=\; X^A_{1,n} + X^B_{1,m} \tag{D.21}$$

When we want to compute the partial variance from the two components we have to make a correction $\Delta\sigma$:

$$\sigma^{AB}_{1,n+m} \;=\; \sigma^A_{1,n} + \sigma^B_{1,m} + \Delta\sigma \tag{D.22}$$

if we define x^{AB}_i as the combined and renumbered set of data points we can write:

$$\sigma^{AB}_{1,n+m} \;=\; \sum_{i=1}^{n+m}\left[x^{AB}_i - \frac{X^{AB}_{1,n+m}}{n+m}\right]^2 \tag{D.23}$$

and the partial variance

$$\sigma_{n,m} \;=\; \sum_{i=n}^{m}\left[x_i - \frac{X_{n,m}}{m-n+1}\right]^2 \tag{D.6}$$

It can be shown that

$$X_{n,m+k} \;=\; X_{n,m} + X_{m+1,m+k} \tag{D.7}$$

and

$$\begin{aligned}\sigma_{n,m+k} \;&=\; \sigma_{n,m} + \sigma_{m+1,m+k} + \left[\frac{X_{n,m}}{m-n+1} - \frac{X_{n,m+k}}{m+k-n+1}\right]^2 * \\ &\quad \frac{(m-n+1)(m+k-n+1)}{k}\end{aligned} \tag{D.8}$$

For $n = 1$ one finds

$$\sigma_{1,m+k} \;=\; \sigma_{1,m} + \sigma_{m+1,m+k} \;+\; \left[\frac{X_{1,m}}{m} - \frac{X_{1,m+k}}{m+k}\right]^2 \frac{m(m+k)}{k} \tag{D.9}$$

and for $n = 1$ and $k = 1$ (eqn. D.8) becomes

$$\sigma_{1,m+1} \;=\; \sigma_{1,m} + \left[\frac{X_{1,m}}{m} - \frac{X_{1,m+1}}{m+1}\right]^2 m(m+1) \tag{D.10}$$

$$\;=\; \sigma_{1,m} + \frac{[\,X_{1,m} - m x_{m+1}\,]^2}{m(m+1)} \tag{D.11}$$

where we have used the relation

$$X_{1,m+1} \;=\; X_{1,m} + x_{m+1} \tag{D.12}$$

Using formulae (eqn. D.11) and (eqn. D.12) the average

$$\langle x\rangle \;=\; \frac{X_{1,N_x}}{N_x} \tag{D.13}$$

and the fluctuation

$$\left\langle(\Delta x)^2\right\rangle^{\frac{1}{2}} = \left[\frac{\sigma_{1,N_x}}{N_x}\right]^{\frac{1}{2}} \tag{D.14}$$

can be obtained by one sweep through the data.

D.2 Implementation

In GROMACS the instantaneous energies $E(m)$ are stored in the energy file, along with the values of $\sigma_{1,m}$ and $X_{1,m}$. Although the steps are counted from 0, for the energy and fluctuations steps are counted from 1. This means that the equations presented here are the ones that are implemented. We give somewhat lengthy derivations in this section to simplify checking of code and equations later on.

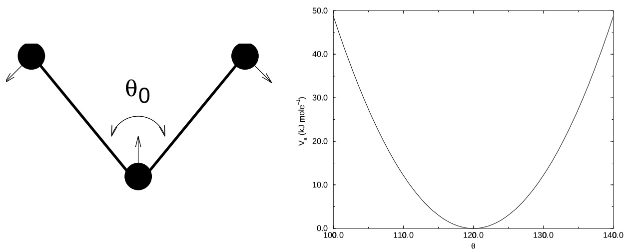


Figure 4.7: Principle of angle vibration (left) and the bond angle potential (right).

The numbering i, j, k is in sequence of covalently bonded atoms, with j denoting the middle atom (see Fig. 4.7).

4.2.5 Cosine based angle potential

In the GROMOS-96 force field a simplified function is used to represent angle vibrations:

$$V_a(\theta_{ijk}) = \frac{1}{2}k^{\theta}_{ijk}\left(\cos(\theta_{ijk}) - \cos(\theta^0_{ijk})\right)^2 \tag{4.48}$$

where

$$\cos(\theta_{ijk}) = \frac{\boldsymbol{r}_{ij}\cdot\boldsymbol{r}_{kj}}{r_{ij}r_{kj}} \tag{4.49}$$

The corresponding force can be derived by partial differentiation with respect to the atomic positions. The force constants in this function are related to the force constants in the harmonic form $k^{\theta,harm}$ (sec. 4.2.4) by:

$$k^{\theta}\sin^2(\theta^0_{ijk}) = k^{\theta,harm} \tag{4.50}$$

4.2.6 Improper dihedrals

Improper dihedrals are meant to keep planar groups planar (e.g. aromatic rings) or to prevent molecules from flipping over to their mirror images, see Fig. 4.8.

$$V_{id}(\xi_{ijkl}) = k_{\xi}(\xi_{ijkl} - \xi_0)^2 \tag{4.51}$$

This is also a harmonic potential, it is plotted in Fig. 4.9. Note that, since it is harmonic, periodicity is not taken into account, so it is best to define improper dihedrals to have a ξ_0 as far away from $\pm 180°$ as you can manage.

4.2.7 Proper dihedrals

For the normal dihedral interaction there is a choice of either the GROMOS periodic function or a function based on expansion in powers of $\cos\phi$ (the so-called Ryckaert-Bellemans potential). This

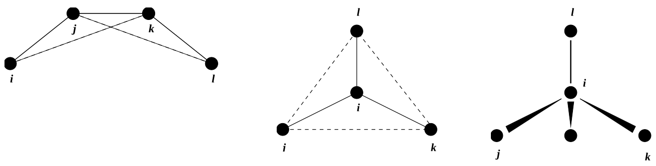


Figure 4.8: Principle of improper dihedral angles. Out of plane bending for rings (left), substituents of rings (middle), out of tetrahedral (right). The improper dihedral angle ξ is defined as the angle between planes (i,j,k) and (j,k,l) in all cases.

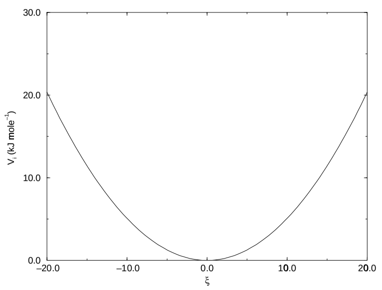


Figure 4.9: Improper dihedral potential.

Appendix D

Averages and fluctuations

D.1 Formulae for averaging

Note: this section was taken from ref [77].

When analyzing a MD trajectory averages $\langle x \rangle$ and fluctuations

$$\left\langle (\Delta x)^2 \right\rangle^{\frac{1}{2}} \; = \; \left\langle [x - \langle x \rangle]^2 \right\rangle^{\frac{1}{2}} \tag{D.1}$$

of a quantity x are to be computed. The variance σ_x of a series of N$_x$ values, $\{$x$_i\}$, can be computed from

$$\sigma_x \; = \; \sum_{i=1}^{N_x} x_i^2 \; - \; \frac{1}{N_x} \left(\sum_{i=1}^{N_x} x_i \right)^2 \tag{D.2}$$

Unfortunately this formula is numerically not very accurate, especially when $\sigma_x^{\frac{1}{2}}$ is small compared to the values of x_i. The following (equivalent) expression is numerically more accurate

$$\sigma_x \; = \; \sum_{i=1}^{N_x} [x_i - \langle x \rangle]^2 \tag{D.3}$$

with

$$\langle x \rangle \; = \; \frac{1}{N_x} \sum_{i=1}^{N_x} x_i \tag{D.4}$$

Using eqns. D.2 and D.4 one has to go through the series of x_i values twice, once to determine $\langle x \rangle$ and again to compute σ_x, whereas eqn. D.1 requires only one sequential scan of the series $\{$x$_i\}$. However, one may cast eqn. D.2 in another form, containing partial sums, which allows for a sequential update algorithm. Define the partial sum

$$X_{n,m} \; = \; \sum_{i=n}^{m} x_i \tag{D.5}$$

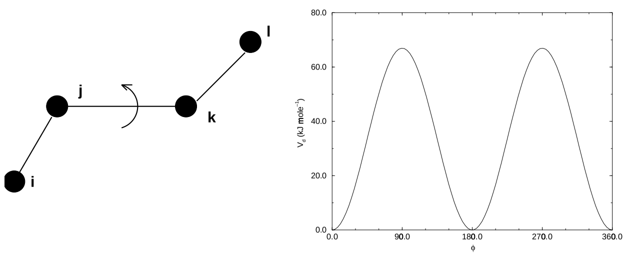


Figure 4.10: Principle of proper dihedral angle (left, in *trans* form) and the dihedral angle potential (right).

C_0	9.28	C_2	-13.12	C_4	26.24
C_1	12.16	C_3	-3.06	C_5	-31.5

Table 4.1: Constants for Ryckaert-Bellemans potential (kJ mol^{-1}).

choice has consequences for the inclusion of special interactions between the first and the fourth atom of the dihedral quadruple. With the periodic GROMOS potential a special 1-4 LJ-interaction must be included; with the Ryckaert-Bellemans potential the 1-4 interactions must be excluded from the non-bonded list.

Proper dihedrals: periodic type

Proper dihedral angles are defined according to the IUPAC/IUB convention, where ϕ is the angle between the ijk and the jkl planes, with **zero** corresponding to the *cis* configuration (i and l on the same side).

$$V_d(\phi_{ijkl}) = k_\phi(1 + \cos(n\phi - \phi_0)) \tag{4.52}$$

Proper dihedrals: Ryckaert-Bellemans function

For alkanes, the following proper dihedral potential is often used (see Fig. 4.11)

$$V_{rb}(\phi_{ijkl}) = \sum_{n=0}^{5} C_n(\cos(\psi))^n, \tag{4.53}$$

where $\psi = \phi - 180°$.
Note: A conversion from one convention to another can be achieved by multiplying every coefficient C_n by $(-1)^n$.

An example of constants for C is given in Table 4.1.

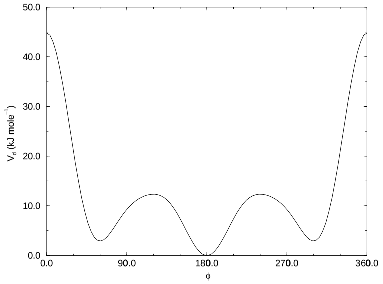


Figure 4.11: Ryckaert-Bellemans dihedral potential.

(**Note:** The use of this potential implies exclusions of LJ-interactions between the first and the last atom of the dihedral, and ψ is defined according to the 'polymer convention' ($\psi_{trans} = 0$).)

The RB dihedral function can also be used to include the OPLS dihedral potential [50]. The OPLS potential function is given as the first four terms of a Fourier series:

$$V_{rb}(\phi_{ijkl}) \; = \; V_0 + \frac{1}{2}(V_1(1+\cos(\psi)) + V_2(1-\cos(2\psi)) + V_3(1+\cos(3\psi))), \tag{4.54}$$

with $\psi = \phi$ (protein convention). Because of the equalities $\cos(2\phi) = 2(\cos(\phi))^2 - 1$ and $\cos(3\phi) = 4(\cos(\phi))^3 - 3\cos(\phi)$, one can translate the OPLS parameters to Ryckaert-Bellemans parameters as follows:

$$\begin{array}{rcl} C_0 & = & V_0 + V_2 + \frac{1}{2}(V_1 + V_3) \\ C_1 & = & \frac{1}{2}(3V_3 - V_1) \\ C_2 & = & -V_2 \\ C_3 & = & -2V_3 \\ C_4 & = & 0 \\ C_5 & = & 0 \end{array} \tag{4.55}$$

with OPLS parameters in protein convention and RB parameters in polymer convention.
Note: Mind the conversion from *kcal mol*$^{-1}$ for OPLS and RB parameters in literature to *kJ mol*$^{-1}$ in GROMACS.

4.2.8 Special interactions

Special potentials are used for imposing restraints on the motion of the system, either to avoid disastrous deviations, or to include knowledge from experimental data. In either case they are not really part of the force field and the reliability of the parameters is not important. The potential forms, as implemented in GROMACS, are mentioned just for the sake of completeness.

For homogeneous mixtures we can again use the average dispersion constant $\langle C_6 \rangle$ (eqn. C.6):

$$P_{lr} \; = \; -\frac{4}{3}\pi \left\langle C_6 \right\rangle \rho^2 r_c^{-3} \tag{C.12}$$

For inhomogeneous systems eqn. C.12 can be applied under the same restriction as holds for the energy (see sec. C.1.1).

If we consider for example a box of pure water, simulated with a cut-off of 0.9 nm and a density of 1 g cm^{-3} this correction is -0.25 kJ mol^{-1}.

For a homogeneous mixture of M components j with N_j particles each, we can write the long range contribution to the energy as:

$$V_{lr} \;=\; \sum_{i \neq j}^{M} -\frac{2 N_i N_j}{3V} \pi C_6(ij) r_c^{-3} \tag{C.5}$$

This can be rewritten if we define an *average dispersion constant* $\langle C_6 \rangle$:

$$\begin{aligned} \langle C_6 \rangle &= \sum_{i \neq j} \frac{N_i N_j}{N^2} C_6(ij) && \text{(C.6)} \\ V_{lr} &= -\frac{2}{3} N \rho \pi \, \langle C_6 \rangle \, r_c^{-3} && \text{(C.7)} \end{aligned}$$

A special form of a non-homogeneous system in this respect, is a pure liquid in which the atoms have different C_6 values. In practice this definition encompasses almost every molecule, except mono-atomic molecules and symmetric molecules like N_2 or O_2. Therefore we always have to determine the average dispersion constant $\langle C_6 \rangle$ in simulations.

In the case of inhomogeneous simulation systems, e.g. a system with a lipid interface, the energy correction can be applied if $\langle C_6 \rangle$ for both components is comparable.

C.1.2 Virial and pressure

The scalar virial of the system due to the dispersion interaction between two particles i and j is given by:

$$\Xi \;=\; -\,\boldsymbol{r}_{ij} \cdot \boldsymbol{F}_{ij} \;=\; 6 C_6 r_{ij}^{-6} \tag{C.8}$$

The pressure is given by:

$$P \;=\; \frac{2}{3\,V} \left(E_{kin} - \Xi \right) \tag{C.9}$$

We can again integrate the long range contribution to the virial [65]:

$$\begin{aligned} \Xi_{lr} &= \frac{1}{2} N \rho \int_{r_c}^{\infty} 4 \pi r^2 \, \Xi \mathrm{d}r \\ &= 12 N \pi \rho C_6 \int_{r_c}^{\infty} r_{ij}^{-4} \mathrm{d}r \\ &= 4 \pi C_6 N \rho r_c^{-3} && \text{(C.10)} \end{aligned}$$

The corresponding correction to the pressure is

$$P_{lr} \;=\; -\,\frac{4}{3} \pi C_6 \rho^2 r_c^{-3} \tag{C.11}$$

Using the same example of a water box, the correction to the virial is 3 kJ mol^{-1} the corresponding correction to the pressure for SPC water at liquid density is approx. -280 bar.

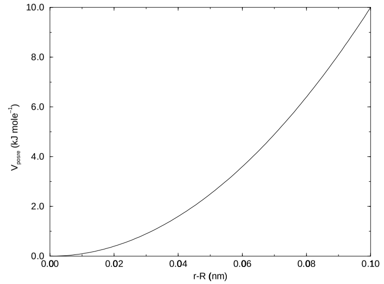


Figure 4.12: Position restraint potential.

4.2.9 Position restraints

These are used to restrain particles to fixed reference positions $\boldsymbol{R}_i$. They can be used during equilibration in order to avoid too drastic rearrangements of critical parts (e.g. to restrain motion in a protein that is subjected to large solvent forces when the solvent is not yet equilibrated). Another application is the restraining of particles in a shell around a region that is simulated in detail, while the shell is only approximated because it lacks proper interaction from missing particles outside the shell. Restraining will then maintain the integrity of the inner part. For spherical shells it is a wise procedure to make the force constant depend on the radius, increasing from zero at the inner boundary to a large value at the outer boundary. This application has not been implemented in GROMACS however.

The following form is used:

$$V_{pr}(\boldsymbol{r}_i) = \frac{1}{2} k_{pr} |\boldsymbol{r}_i - \boldsymbol{R}_i|^2 \tag{4.56}$$

The potential is plotted in Fig. 4.12.

The potential form can be rewritten without loss of generality as:

$$V_{pr}(\boldsymbol{r}_i) = \frac{1}{2} \left[k_{pr}^x (x_i - X_i)^2 \,\hat{\mathbf{x}} + k_{pr}^y (y_i - Y_i)^2 \,\hat{\mathbf{y}} + k_{pr}^z (z_i - Z_i)^2 \,\hat{\mathbf{z}} \right] \tag{4.57}$$

Now the forces are:

$$\begin{array}{rcl} F_i^x &=& -k_{pr}^x \, (x_i - X_i) \\ F_i^y &=& -k_{pr}^y \, (y_i - Y_i) \\ F_i^z &=& -k_{pr}^z \, (z_i - Z_i) \end{array} \tag{4.58}$$

Using three different force constants the position restraints can be turned on or off in each spatial dimension; this means that atoms can be harmonically restrained to a plane or a line. Position restraints are applied to a special fixed list of atoms. Such a list is usually generated by the pdb2gmx program.

4.2.10 Angle restraints

These are used to restrain the angle between two pairs of particles or between one pair of particles and the Z-axis. The functional form is similar to that of a proper dihedral. For two pairs of atoms:

$$V_{ar}(\boldsymbol{r}_i,\boldsymbol{r}_j,\boldsymbol{r}_k,\boldsymbol{r}_l) = k_{ar}(1-\cos(n(\theta-\theta_0))), \quad \text{where } \theta = \arccos\left(\frac{\boldsymbol{r}_j-\boldsymbol{r}_i}{\|\boldsymbol{r}_j-\boldsymbol{r}_i\|}\cdot\frac{\boldsymbol{r}_l-\boldsymbol{r}_k}{\|\boldsymbol{r}_l-\boldsymbol{r}_k\|}\right) \tag{4.59}$$

For one pair of atoms and the Z-axis:

$$V_{ar}(\boldsymbol{r}_i,\boldsymbol{r}_j) = k_{ar}(1-\cos(n(\theta-\theta_0))), \quad \text{where } \theta = \arccos\left(\frac{\boldsymbol{r}_j-\boldsymbol{r}_i}{\|\boldsymbol{r}_j-\boldsymbol{r}_i\|}\cdot\begin{pmatrix}0\\0\\1\end{pmatrix}\right) \tag{4.60}$$

A multiplicity (n) of 2 is useful when you do not want to distinguish between parallel and anti-parallel vectors.

4.2.11 Distance restraints

Distance restraints add a penalty to the potential when the distance between specified pairs of atoms exceeds a threshold value. They are normally used to impose experimental restraints, as from experiments in nuclear magnetic resonance (NMR), on the motion of the system. Thus MD can be used for structure refinement using NMR data. The potential form is quadratic below a specified lower bound and between two specified upper bounds and linear beyond the largest bound (see Fig. 4.13).

$$V_{dr}(r_{ij}) = \begin{cases} \frac{1}{2}k_{dr}(r_{ij}-r_0)^2 & \text{for} \quad r_{ij} < r_0 \\ 0 & \text{for} \quad r_0 \le r_{ij} < r_1 \\ \frac{1}{2}k_{dr}(r_{ij}-r_1)^2 & \text{for} \quad r_1 \le r_{ij} < r_2 \\ \frac{1}{2}k_{dr}(r_2-r_1)(2r_{ij}-r_2-r_1) & \text{for} \quad r_2 \le r_{ij} \end{cases} \tag{4.61}$$

The forces are

$$\boldsymbol{F}_i = \begin{cases} -k_{dr}(r_{ij}-r_0)\frac{\boldsymbol{r}_{ij}}{r_{ij}} & \text{for} \quad r_{ij} < r_0 \\ 0 & \text{for} \quad r_0 \le r_{ij} < r_1 \\ -k_{dr}(r_{ij}-r_1)\frac{\boldsymbol{r}_{ij}}{r_{ij}} & \text{for} \quad r_1 \le r_{ij} < r_2 \\ -k_{dr}(r_2-r_1)\frac{\boldsymbol{r}_{ij}}{r_{ij}} & \text{for} \quad r_2 \le r_{ij} \end{cases} \tag{4.62}$$

Time averaging

Distance restraints based on instantaneous distances can potentially reduce the fluctuations in a molecule significantly. This problem can be overcome by restraining to a *time averaged* dis-

---

Appendix C

Long range corrections

C.1 Dispersion

In this section we derive long range corrections due to the use of a cut-off for Lennard Jones interactions. We assume that the cut-off is so long that the repulsion term can safely be neglected, and therefore only the dispersion term is taken into account. Due to the nature of the dispersion interaction, energy and pressure corrections both are negative. While the energy correction is usually small, it may be important for free energy calculations. The pressure correction in contrast is very large and can not be neglected. Although it is in principle possible to parameterize a force field such that the pressure is close to 1 bar even without correction, such a method makes the parameterization dependent on the cut-off and is therefore undesirable. Please note that it is not consistent to use the long range correction to the dispersion without using either a reaction field method or a proper long range electrostatics method such as Ewald summation or PPPM.

C.1.1 Energy

The long range contribution of the dispersion interaction to the virial can be derived analytically, if we assume a homogeneous system beyond the cut-off distance r_c. The dispersion energy between two particles is written as:

$$V(r_{ij}) \;=\; -C_6 r_{ij}^{-6} \tag{C.1}$$

and the corresponding force is

$$\boldsymbol{F}_{ij} \;=\; -6C_6 r_{ij}^{-8}\boldsymbol{r}_{ij} \tag{C.2}$$

The long range contribution to the dispersion energy in a system with N particles and particle density $\rho = N/V$, where V is the volume, is [65]:

$$V_{lr} \;=\; \frac{1}{2}N\rho\int_{r_c}^{\infty} 4\pi r^2 g(r)V(r)\mathrm{d}r \tag{C.3}$$

which we can integrate assuming that the radial distribution function $g(r)$ is 1 beyond the cut-off r_c

$$V_{lr} \;=\; -\frac{2}{3}N\rho\pi C_6 r_c^{-3} \tag{C.4}$$

electrostatic, dispersion and repulsion interactions, but for the sake of caching performance these have been combined into a single array. The cubic spline interpolation looks like this:

$$y(x) \ = \ \eta y_i + \epsilon y_{i+1} + \frac{h^2}{6}\left[(\eta^3-\eta)y_i^{''} + (\epsilon^3-\epsilon)y_{i+1}^{''}\right] \tag{B.58}$$

where ϵ = 1-η, and y_i and $y_i^{''}$ are the tabulated values of a function $y(x)$ and its second derivative respectively. Furthermore,

$$h \ = \ x_{i+1} - x_i \tag{B.59}$$

$$\epsilon \ = \ (x - x_i)/h \tag{B.60}$$

so that $0 \leq \epsilon < 1$. eqn. B.58 can be rewritten as

$$y(x) \ = \ y_i + \epsilon\left(y_{i+1} - y_i - \frac{h^2}{6}\left(2y_i^{''} + y_{i+1}^{''}\right)\right) + \epsilon^2\left(\frac{h^2}{2}y_i^{''}\right) + \epsilon^3\frac{h^2}{6}\left(y_{i+1}^{''} - y_i^{''}\right) \tag{B.61}$$

Note that the x-dependence is completely in ϵ. This can abbreviated to

$$y(x) \ = \ y_i + \epsilon F_i + \epsilon^2 G_i + \epsilon^3 H_i \tag{B.62}$$

From this we can calculate the derivative in order to determine the forces:

$$\frac{\mathrm{d}y(x)}{\mathrm{d}x} \ = \ \frac{\mathrm{d}y(x)}{\mathrm{d}\epsilon}\frac{\mathrm{d}\epsilon}{\mathrm{d}x} \ = \ (F_i + 2\epsilon G_i + 3\epsilon^2 H_i)/h \tag{B.63}$$

If we store in the table y_i, F_i, G_i and H_i we need a table of length 4n. The number of points per nanometer should be on the order of 500 to 1000, for accurate representation (relative error $< 10^{-4}$ when n = 500 points/nm). The force routines get a scaling factor s as a parameter that is equal to the number of points per nm. (Note that h is s^{-1}).

The algorithm goes a little something like this:

1. Calculate distance vector ($\boldsymbol{r}_{ij}$) and distance r$_{ij}$
2. Multiply r$_{ij}$ by s and truncate to an integer value n_0 to get a table index
3. Calculate fractional component (ϵ = sr$_{ij} - n_0$) and ϵ^2
4. Do the interpolation to calculate the potential V and the the scalar force f
5. Calculate the vector force $\boldsymbol{F}$ by multiplying f with $\boldsymbol{r}_{ij}$

The tables are stored internally as y$_i$, F$_i$, G$_i$, H$_i$ in the order coulomb, dispersion, repulsion. In total there are 12 values in each table entry. Note that table lookup is significantly *slower* than computation of the most simple Lennard-Jones and Coulomb interaction. However, it is much faster than the shifted coulomb function used in conjunction with the PPPM method. Finally it is much easier to modify a table for the potential (and get a graphical representation of it) than to modify the inner loops of the MD program.

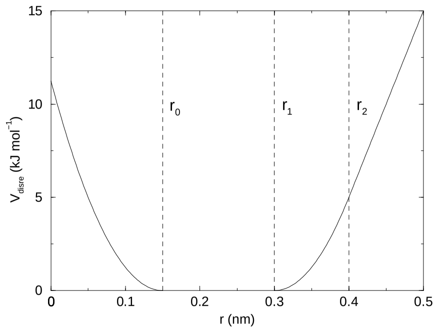


Figure 4.13: Distance Restraint potential.

tance [51]. The forces with time averaging are:

$$\boldsymbol{F}_i \ = \ \begin{cases} -k_{dr}(\bar{r}_{ij} - r_0)\frac{\boldsymbol{r}_{ij}}{r_{ij}} & \text{for} \quad \bar{r}_{ij} < r_0 \\ 0 & \text{for} \quad r_0 \leq \bar{r}_{ij} < r_1 \\ -k_{dr}(\bar{r}_{ij} - r_1)\frac{\boldsymbol{r}_{ij}}{r_{ij}} & \text{for} \quad r_1 \leq \bar{r}_{ij} < r_2 \\ -k_{dr}(r_2 - r_1)\frac{\boldsymbol{r}_{ij}}{r_{ij}} & \text{for} \quad r_2 \leq \bar{r}_{ij} \end{cases} \tag{4.63}$$

where $\bar{r}_{ij}$ is given by:

$$\bar{r}_{ij} \ = \ < r_{ij}^{-3} >^{-1/3} \tag{4.64}$$

Because of the time averaging we can no longer speak of a distance restraint potential.

This way an atom can satisfy two incompatible distance restraints *on average* by moving between two positions. An example would be an amino-acid side-chain which is rotating around its χ dihedral angle, thereby coming close to various other groups. Such a mobile side chain may give rise to multiple NOEs, which can not be fulfilled in a single structure.

The computation of the time averaged distance in the `mdrun` program is done in the following fashion:

$$\begin{array}{rcl} \overline{r^{-3}}_{ij}(0) & = & r_{ij}(0)^{-3} \\ \overline{r^{-3}}_{ij}(t) & = & \overline{r^{-3}}_{ij}(t-\Delta t)\ \exp\left(-\frac{\Delta t}{\tau}\right) + r_{ij}(t)^{-3}\left[1 - \exp\left(-\frac{\Delta t}{\tau}\right)\right] \end{array} \tag{4.65}$$

When a pair is within the bounds it can still feel a force, because the time averaged distance can still be beyond a bound. To prevent the protons from being pulled too close together a mixed approach can be used. In this approach the penalty is zero when the instantaneous distance is within the bounds, otherwise the violation is the square root of the product of the instantaneous violation and the time averaged violation.

Averaging over multiple pairs

Sometimes it is unclear from experimental data which atom pair gives rise to a single NOE, in other occasions it can be obvious that more than one pair contributes due to the symmetry of the system, e.g. a methyl group with three protons. For such a group it is not possible to distinguish between the protons, therefore they should all be taken into account when calculating the distance between this methyl group and another proton (or group of protons). Due to the physical nature of magnetic resonance, the intensity of the NOE signal is proportional to the distance between atoms to the power of -6. Thus, when combining atom pairs, a fixed list of N restraints may be taken together, where the apparent "distance" is given by:

$$r_N(t) = \left[\sum_{n=1}^{N} \bar{r}_n(t)^{-6}\right]^{-1/6} \tag{4.66}$$

where we use r_{ij} or eqn. 4.64 for the $\bar{r}_n$. The r_N of the instantaneous and time-averaged distances can be combined to do a mixed restraining as indicated above. As more pairs of protons contribute to the same NOE signal, the intensity will increase, and the summed "distance" will be shorter than any of its components due to the reciprocal summation.

There are two options for distributing the forces over the atom pairs. In the conservative option the force is defined as the derivate of the restraint potential with respect to the coordinates. This results in a conservative potential when no time averaging is used. The force distribution over the pairs is proportional to r^{-6}. This means that a close pair feels a much larger force than a distant pair, which might lead to a 'too rigid' molecule. The other option is an equal force distribution. In this case each pair feels $1/N$ of the derivative of the restraint potential with respect to r_N. The advantage of this method is that more conformations might be sampled, but the non-conservative nature of the forces can lead to local heating of the protons.

It is also possible to use *ensemble averaging* using multiple (protein) molecules. In this case the bounds should be lowered as in:

$$\begin{aligned} r_1 &= r_1 * M^{-1/6} \\ r_2 &= r_2 * M^{-1/6} \end{aligned} \tag{4.67}$$

where M is the number of molecules. The GROMACS preprocessor `grompp` can do this automatically when the appropriate option is given. The resulting "distance" is then used to calculate the scalar force according to:

$$\begin{array}{rcll} \boldsymbol{F}_i & = & 0 & r_N < r_1 \\ & = & -k_{dr}(r_N - r_1)\frac{\boldsymbol{r}_{ij}}{r_{ij}} & r_1 \le r_N < r_2 \\ & = & -k_{dr}(r_2 - r_1)\frac{\boldsymbol{r}_{ij}}{r_{ij}} & r_N \ge r_2 \end{array} \tag{4.68}$$

where i and j denote the atoms of all the pairs that contribute to the NOE signal.

Using distance restraints

A list of distance restrains based on NOE data can be added to a molecule definition in your topology file, like in the following example:

For $(127 - E) = odd$ equation (eqn. B.45) can be rewritten as

$$y(x) = (2^{\frac{127-E-1}{2}})(\frac{1.F}{2})^{-1/2} \tag{B.52}$$

thus

$$E' = \frac{126 - E}{2} + 127 \tag{B.53}$$

which also can be calculated exactly in integer arithmetic. Note that the fraction is automatically corrected for its range earlier mentioned, so the exponent does not need an extra correction.

The conclusions from this are:

- The fraction and exponent lookup table are independent. The fraction lookup table exists of two tables (odd and even exponent) so the odd/even information of the exponent (lsb bit) has to be used to select the right table.
- The exponent table is an 256 x 8 bit table, initialized for odd and $even$.

B.3.6 Implementation

The lookup tables can be generated by a small C program, which uses floating point numbers and operations with IEEE 32 bit single precision format. Note that because of the $odd/even$ information that is needed, the fraction table is twice the size earlier specified (13 bit i.s.o. 12 bit).

The function according to equation (eqn. B.29) has to be implemented. Applied to the $1/\sqrt{x}$ function, equation (eqn. B.28) leads to

$$f = a - \frac{1}{y^2} \tag{B.54}$$

and so

$$f' = \frac{2}{y^3} \tag{B.55}$$

so

$$y_{n+1} = y_n - \frac{a - \frac{1}{y_n^2}}{\frac{2}{y_n^3}} \tag{B.56}$$

or

$$y_{n+1} = \frac{y_n}{2}(3 - a y_n^2) \tag{B.57}$$

Where y_0 can be found in the lookup tables, and y_1 gives the result to the maximum accuracy. It is clear that only one iteration extra (in double precision) is needed for a double precision result.

B.4 Tabulated functions

In some of the inner loops of GROMACS lookup tables are used for computation of potential and forces. The tables are interpolated using a cubic spline algorithm. There are separate tables for

B.3.5 Separate exponent and fraction computation

The used IEEE 32 bit single precision floating point format specifies that a number is represented by a exponent and a fraction. The previous section specifies for every possible floating point number the lookup table length and width. Only the size of the fraction of a floating point number defines the accuracy. The conclusion from this can be that the size of the lookup table is length of lookup table, earlier specified, times the size of the exponent ($2^{12}2^8, 1Mb$). The $1/\sqrt{x}$ function has the property that the exponent is independent of the fraction. This becomes clear if the floating point representation is used. Define

$$x \equiv (-1)^S (2^{E-127})(1.F) \tag{B.42}$$

see Fig. B.1 where $0 \leq S \leq 1$, $0 \leq E \leq 255$, $1 \leq 1.F < 2$ and S, E, F integer (normalization conditions). The sign bit (S) can be omitted because $1/\sqrt{x}$ is only defined for $x > 0$. The $1/\sqrt{x}$ function applied to x results in

$$y(x) = \frac{1}{\sqrt{x}} \tag{B.43}$$

or

$$y(x) = \frac{1}{\sqrt{(2^{E-127})(1.F)}} \tag{B.44}$$

this can be rewritten as

$$y(x) = (2^{E-127})^{-1/2}(1.F)^{-1/2} \tag{B.45}$$

Define

$$(2^{E'-127}) \equiv (2^{E-127})^{-1/2} \tag{B.46}$$

$$1.F' \equiv (1.F)^{-1/2} \tag{B.47}$$

then $\frac{1}{\sqrt{2}} < 1.F' \leq 1$ holds, so the condition $1 \leq 1.F' < 2$ which is essential for normalized real representation is not valid anymore. By introducing an extra term this can be corrected. Rewrite the $1/\sqrt{x}$ function applied to floating point numbers, equation (eqn. B.45) as

$$y(x) = (2^{\frac{127-E}{2}-1})(2(1.F)^{-1/2}) \tag{B.48}$$

and

$$(2^{E'-127}) \equiv (2^{\frac{127-E}{2}-1}) \tag{B.49}$$

$$1.F' \equiv 2(1.F)^{-1/2} \tag{B.50}$$

then $\sqrt{2} < 1.F \leq 2$ holds. This is not the exact valid range as defined for normalized floating point numbers in equation (eqn. B.42). The value 2 causes the problem. By mapping this value on the nearest representation < 2 this can be solved. The small error that is introduced by this approximation is within the allowable range.

The integer representation of the exponent is the next problem. Calculating $(2^{\frac{127-E}{2}-1})$ introduces a fractional result if $(127 - E) = odd$. This is again easily accounted for by splitting up the calculation into an odd and an even part. For $(127 - E) = even$ E' in equation (eqn. B.49) can be exactly calculated in integer arithmetic as a function of E.

$$E' = \frac{127 - E}{2} + 126 \tag{B.51}$$

```
[ distance_restraints ]
; ai   aj    type  index  type'   low    up1    up2    fac
10     16     1      0      1     0.0    0.3    0.4    1.0
10     28     1      1      1     0.0    0.3    0.4    1.0
10     46     1      1      1     0.0    0.3    0.4    1.0
16     22     1      2      1     0.0    0.3    0.4    2.5
16     34     1      3      1     0.0    0.5    0.6    1.0
```

In this example a number of features can be found. In columns `ai` and `aj` you find the atom numbers of the particles to be restrained. The `type` column should always be 1. As explained in sec. 4.2.11, multiple distances can contribute to a single NOE signal. In the topology this can be set using the `index` column. In our example, the restraints 10-28 and 10-46 both have index 1, therefore they are treated simultaneously. An extra requirement for treating restraints together, is that the restraints should be on successive lines, without any other intervening restraint. The `type'` column will usually be 1, but can be set to 2 to obtain a distance restraint which will never be time and ensemble averaged, this can be useful for restraining hydrogen bonds. The columns `low`, `up1` and `up2` hold the values of r_0, r_1 and r_2 from eqn. 4.61. In some cases it can be useful to have different force constants for some restraints, this is controlled by the column `fac`. The force constant in the parameter file is multiplied by the value in the column `fac` for each restraint.

Some parameters for NMR refinement can be specified in the `grompp.mdp` file:

`disre`**: type of distance restraining.** The `disre` variable sets the type of distance restraining. `no/simple` turns the distance restraining off/on. When multiple proteins or peptides are used in the simulation ensemble averaging can be turned on by setting `disre = ensemble`.

`disre_weighting`**: force-weighting in restraints with multiple pairs.** By default, the force due to the distance restraint is distributed equally over all the pairs involved in the restraint. This can also be explicitly selected with `disre_weighting = equal`. If you instead set this option to `disre_weighting = conservative` you get conservative forces when `disre_tau = 0`.

`disre_mixed`**: how to calculate the violations.** `disre_mixed = no` gives normal time averaged violations. When `disre_mixed = yes` the square root of the product of the time averaged and the instantaneous violations is used.

`disre_fc`**: force constant k_{dr} for distance restraints.** k_{dr} (eqn. 4.61) can be set as variable `disre_fc = 1000` for a force constant of 1000 kJ mol^{-1} nm^{-2}. This value is multiplied by the value in the `fac` column in the distance restraint entries in the topology file.

`disre_tau`**: time constant for restraints.** τ (eqn. 4.65) can be set as variable `disre_tau = 10` for a time constant of 10 ps. Time averaging can be turned off by setting `disre_tau` to 0.

`nstdisreout`**: pair distance output frequency.** Determines how often the time averaged and instantaneous distances of all atom pairs involved in distance restraints are written to the energy file.

4.3 Free energy interactions

This section describes the λ-dependence of the potentials used for free energy calculations (see sec. 3.12). All common types of potentials and constraints can be interpolated smoothly from state A ($\lambda = 0$) to state B ($\lambda = 1$) and vice versa. All bonded interactions are interpolated by linear interpolation of the interaction parameters. Non-bonded interactions can be interpolated linearly or via soft-core interactions.

Harmonic potentials

The example given here is for the bond potential which is harmonic in GROMACS. However, these equations apply to the angle potential and the improper dihedral potential as well.

$$V_b = \frac{1}{2}((1-\lambda)k_b^A + \lambda k_b^B)(b - (1-\lambda)b_0^A - \lambda b_0^B)^2 \tag{4.69}$$

$$\frac{\partial V_b}{\partial \lambda} = \frac{1}{2}(k_b^B - k_b^A)\left[b - (1-\lambda)b_0^A + \lambda b_0^B)^2 + (b_0^A - b_0^B)(b - (1-\lambda)b_0^A - \lambda b_0^B)\right] \tag{4.70}$$

GROMOS-96 bonds and angles

Fourth power bond stretching and cosine based angle potentials are interpolated by linear interpolation of the force constant and the equilibrium position. Formulas are not given here.

Proper dihedrals

For the proper dihedrals, the equations are somewhat more complicated:

$$V_d = ((1-\lambda)k_d^A + \lambda k_d^B)(1 + \cos(n_\phi \phi - ((1-\lambda)\phi_0^A + \lambda\phi_0^B)) \tag{4.71}$$

$$\begin{aligned}\frac{\partial V_d}{\partial \lambda} = {} & (k_d^B - k_d^A)\Big[1 + \cos(n_\phi\phi - [(1-\lambda)\phi_0^A + \lambda\phi_0^B]) - \\ & ((1-\lambda)k_d^A + \lambda k_d^B)(\phi_0^A - \phi_0^B)\sin(n_\phi\phi - [(1-\lambda)\phi_0^A + \lambda\phi_0^B]\Big]\end{aligned} \tag{4.72}$$

Note: that the multiplicity n_ϕ can not be parameterized because the function should remain periodic on the interval $[0, 2\pi]$.

Coulomb interaction

The Coulomb interaction between two particles of which the charge varies with λ is:

$$V_c = \frac{f}{\varepsilon_{rf} r_{ij}}\left[((1-\lambda)q_i^A + \lambda q_i^B)\cdot((1-\lambda)q_j^A + \lambda q_i^B)\right] \tag{4.73}$$

$$\frac{\partial V_c}{\partial \lambda} = \frac{f}{\varepsilon_{rf} r_{ij}}\left[(q_j^B - q_j^A)((1-\lambda)q_i^A + \lambda q_i^B) + (q_i^B - q_i^A)((1-\lambda)q_j^A + \lambda q_j^B)\right] \tag{4.74}$$

where $f = \frac{1}{4\pi\varepsilon_0} = 138.935\,485$ (see chapter 2)

B.3.4 Specification of the lookup table

To calculate the function $1/\sqrt{x}$ using the previously mentioned iteration scheme, it is clear that the first estimation of the solution must be accurate enough to get precise results. The requirements for the calculation are

- Maximum possible accuracy with the used IEEE format
- Use only one iteration step for maximum speed

The first requirement states that the result of $1/\sqrt{x}$ may have a relative error ε_r equal to the ε_r of a IEEE 32 bit single precision floating point number. From this the $1/\sqrt{x}$ of the initial approximation can be derived, rewriting the definition of the relative error for succeeding steps, equation (eqn. B.34)

$$\frac{\varepsilon_n}{y} = \sqrt{\varepsilon_{r_{n+1}}\frac{2f'}{yf''}} \tag{B.35}$$

So for the lookup table the needed accuracy is

$$\frac{\Delta Y}{Y} = \sqrt{\frac{2}{3}2^{-24}} \tag{B.36}$$

which defines the width of the table that must be ≥ 13 bit.

At this point the relative error ε_{r_n} of the lookup table is known. From this the maximum relative error in the argument can be calculated as follows. The absolute error Δx is defined as

$$\Delta x \equiv \frac{\Delta Y}{Y'} \tag{B.37}$$

and thus

$$\frac{\Delta x}{Y} = \frac{\Delta Y}{Y}(Y')^{-1} \tag{B.38}$$

and thus

$$\Delta x = constant\frac{Y}{Y'} \tag{B.39}$$

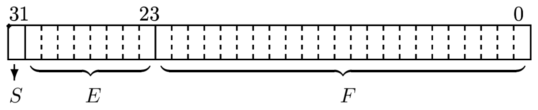
for the $1/\sqrt{x}$ function $Y/Y' \sim x$ holds, so $\Delta x/x = constant$. This is a property of the used floating point representation as earlier mentioned. The needed accuracy of the argument of the lookup table follows from

$$\frac{\Delta x}{x} = -2\frac{\Delta Y}{Y} \tag{B.40}$$

so, using the floating point accuracy, equation (eqn. B.36)

$$\frac{\Delta x}{x} = -2\sqrt{\frac{2}{3}2^{-24}} \tag{B.41}$$

This defines the length of the lookup table which should be ≥ 12 bit.



$$Value = (-1)^S (2^{E-127})(1.F)$$

Figure B.1: IEEE single precision floating point format

can now be solved using Newton-Raphson. An iteration is performed by calculating

$$y_{n+1} = y_n - \frac{f(y_n)}{f'(y_n)} \tag{B.29}$$

The absolute error ε, in this approximation is defined by

$$\varepsilon \equiv y_n - q \tag{B.30}$$

using Taylor series expansion to estimate the error results in

$$\varepsilon_{n+1} = -\frac{\varepsilon_n^2}{2} \frac{f''(y_n)}{f'(y_n)} \tag{B.31}$$

according to [76] equation (3.2). This is an estimation of the absolute error.

B.3.3 Applied to floating point numbers

Floating point numbers in IEEE 32 bit single precision format have a nearly constant relative error of $\Delta x/x = 2^{-24}$. As seen earlier in the Taylor series expansion equation (eqn. B.31), the error in every iteration step is absolute and in general dependent of y. If the error is expressed as a relative error ε_r the following holds

$$\varepsilon_{r_{n+1}} \equiv \frac{\varepsilon_{n+1}}{y} \tag{B.32}$$

and so

$$\varepsilon_{r_{n+1}} = -(\frac{\varepsilon_n}{y})^2 y \frac{f''}{2f'} \tag{B.33}$$

for the function $f(y) = y^{-2}$ the term $yf''/2f'$ is constant (equal to $-3/2$) so the relative error ε_{r_n} is independent of y.

$$\varepsilon_{r_{n+1}} = \frac{3}{2}(\varepsilon_{r_n})^2 \tag{B.34}$$

The conclusion of this is that the function $1/\sqrt{x}$ can be calculated with a specified accuracy.

Coulomb interaction with Reaction Field

The coulomb interaction including a reaction field, between two particles of which the charge varies with λ is:

$$V_c = f\left[\frac{1}{r_{ij}} + k_{rf}\, r_{ij}^2 - c_{rf}\right]\left[((1-\lambda)q_i^A + \lambda q_i^B) \cdot ((1-\lambda)q_j^A + \lambda q_i^B)\right] \tag{4.75}$$

$$\frac{\partial V_c}{\partial \lambda} = f\left[\frac{1}{r_{ij}} + k_{rf}\, r_{ij}^2 - c_{rf}\right] \cdot \left[(q_j^B - q_j^A)((1-\lambda)q_i^A + \lambda q_i^B) + (q_i^B - q_i^A)((1-\lambda)q_j^A + \lambda q_j^B)\right] \tag{4.76}$$

Note that the constants k_{rf} and c_{rf} are defined using the dielectric constant ε_{rf} of the medium (see sec. 4.1.4).

Lennard-Jones interaction

For the Lennard-Jones interaction between two particles of which the *atom type* varies with λ we can write:

$$V_{LJ} = \frac{((1-\lambda)C_{12}^A + \lambda C_{12}^B)}{r_{ij}^{12}} - \frac{(1-\lambda)C_6^A + \lambda C_6^B}{r_{ij}^6} \tag{4.77}$$

$$\frac{\partial V_{LJ}}{\partial \lambda} = \frac{C_{12}^B - C_{12}^A}{r_{ij}^{12}} - \frac{C_6^B - C_6^A}{r_{ij}^6} \tag{4.78}$$

It should be noted that it is also possible to express a pathway from state A to state B using σ and ϵ (see eqn. 4.5). It may seem to make sense physically, to vary the forcefield parameters σ and ϵ rather than the derived parameters C_{12} and C_6. However, the difference between the pathways in parameter space is not large, and the free energy itself does not depend on the pathway, therefore we use the simple formulation presented above.

Kinetic Energy

When the mass of a particle changes there is also a contribution of the kinetic energy to the free energy (note that we can not write the momentum $\boldsymbol{p}$ as m$\boldsymbol{v}$ since that would result in the sign of $\frac{\partial Ek}{\partial \lambda}$ being incorrect [52]):

$$Ek = \frac{1}{2} \frac{\boldsymbol{p}^2}{(1-\lambda)m^A + \lambda m^B} \tag{4.79}$$

$$\frac{\partial Ek}{\partial \lambda} = -\frac{1}{2} \frac{\boldsymbol{p}^2 (m^B - m^A)}{((1-\lambda)m^A + \lambda m^B)^2} \tag{4.80}$$

after taking the derivative, we *can* insert $\boldsymbol{p}$ = m$\boldsymbol{v}$, such that:

$$\frac{\partial Ek}{\partial \lambda} = -\frac{1}{2}\boldsymbol{v}^2 (m^B - m^A) \tag{4.81}$$

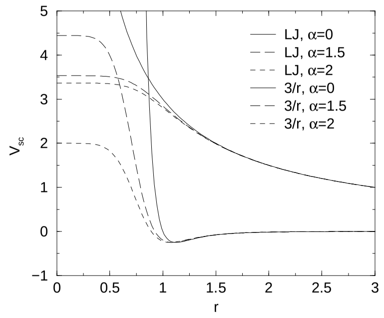


Figure 4.14: Soft-core interactions at $\lambda = 0.5$, with $C_6^A = C_{12}^A = C_6^B = C_{12}^B = 1$.

Constraints

The constraints are formally part of the Hamiltonian, and therefore they give a contribution to the free energy. In GROMACS this can be calculated using the LINCS or the SHAKE algorithm. If we have a number of constraint equations g_k:

$$g_k = r_k - d_k \tag{4.82}$$

where $\boldsymbol{r}_k$ is the distance vector between two particles and d_k is the constraint distance between the two particles we can write this using a λ dependent distance as

$$g_k = r_k - \left((1-\lambda)d_k^A + \lambda d_k^B\right) \tag{4.83}$$

the contribution C_λ to the Hamiltonian using Lagrange multipliers λ:

$$C_\lambda = \sum_k \lambda_k g_k \tag{4.84}$$

$$\frac{\partial C_\lambda}{\partial \lambda} = \sum_k \lambda_k \left(d_k^B - d_k^A\right) \tag{4.85}$$

4.3.1 Soft-core interactions

The linear interpolation of the Lennard-Jones and Coulomb potentials gives problems when growing particles out of nothing or when making particles disappear (λ close to 0 or 1). To circumvent these problems, the singularities in the potentials need to be removed. This is done with soft-core potentials. In GROMACS the soft-core potential V_{sc} is:

$$V_{sc}(r) = (1-\lambda)V^A(r_A) + \lambda V^B(r_B) \tag{4.86}$$

$$r_A = \left(\alpha\sigma_A^6\lambda^2 + r^6\right)^{\frac{1}{6}} \tag{4.87}$$

Note that this loop provides much less optimization than the water loop, but it is slightly better than the default routine.

The gain of these implementations is that there are more floating point operations in a single loop, which implies that some compilers can schedule the code better. However, it turns out that even some of the most advanced compilers have problems with scheduling, implying that manual tweaking is necessary to get optimum performance. This may include common-subexpression elimination, or moving code around.

B.2.2 Fortran Code

Unfortunately, Fortran compilers are still better than C-compilers, for most machines anyway. For some machines (e.g. SGI Power Challenge) the difference may be up to a factor of 3, in the case of vector computers this may be even larger. Therefore, some of the routines that take up a lot of computer time have been translated into Fortran and even assembly code for Intel and AMD x86 processors. In most cases, the Fortran or assembly loops should be selected automatically by the configure script when appropriate, but you can also tweak this by setting options to the configure script.

B.3 Computation of the 1.0/sqrt function.

B.3.1 Introduction.

The GROMACS project started with the development of a $1/\sqrt{x}$ processor which calculates

$$Y(x) = \frac{1}{\sqrt{x}} \tag{B.25}$$

As the project continued, the Intel *i*860 processor was used to implement GROMACS, which now turned into almost a full software project. The $1/\sqrt{x}$ processor was implemented using a Newton-Raphson iteration scheme for one step. For this it needed lookup tables to provide the initial approximation. The $1/\sqrt{x}$ function makes it possible to use two almost independent tables for the exponent seed and the fraction seed with the IEEE floating point representation.

B.3.2 General

According to [76] the $1/\sqrt{x}$ can be calculated using the Newton-Raphson iteration scheme. The inverse function is

$$X(y) = \frac{1}{y^2} \tag{B.26}$$

So instead of calculating

$$Y(a) = q \tag{B.27}$$

the equation

$$X(q) - a = 0 \tag{B.28}$$

B.1.5 Virial from Shake.

An important contribution to the virial comes from shake. Satisfying the constraints a force **G** is exerted on the particles shaken. If this force does not come out of the algorithm (as in standard shake) it can be calculated afterwards (when using *leap-frog*) by:

$$\begin{aligned}\Delta \boldsymbol{r}_i &= \boldsymbol{r}_i(t+\Delta t) - [\boldsymbol{r}_i(t) + \mathbf{v}_i(t-\frac{\Delta t}{2})\Delta t + \frac{\boldsymbol{F}_i}{m_i}\Delta t^2] && \text{(B.22)}\\ \mathbf{G}_i &= \frac{m_i \Delta \boldsymbol{r}_i}{\Delta t^2} && \text{(B.23)}\end{aligned}$$

but this does not help us in the general case. Only when no periodicity is needed (like in rigid water) this can be used, otherwise we must add the virial calculation in the inner loop of shake.

When it *is* applicable the virial can be calculated in the single sum way:

$$\Xi = -\frac{1}{2}\sum_i^{N_c} \boldsymbol{r}_i \otimes \boldsymbol{F}_i \qquad \text{(B.24)}$$

where N_c is the number of constrained atoms.

B.2 Optimizations

Here we describe some of the algorithmic optimizations used in GROMACS, apart from parallelism. One of these, the implementation of the 1.0/sqrt(x) function is treated separately in sec. B.3. The most important other optimizations are described below.

B.2.1 Inner Loops for Water

GROMACS users special inner loop to calculate non-bonded interactions for water molecules with other atoms, and yet another set of loops for interactions between pairs of water molecules. This very optimized loop assumes a water model similar to SPC [48], i.e.:

1. There are three atoms in the molecule.
2. The first atom has Lennard-Jones (sec. 4.1.1) and coulomb (sec. 4.1.3) interactions.
3. Atoms two and three have only coulomb interactions, and equal charges.

The loop also works for the SPC/E [75] and TIP3P [42] water models. For more complicated molecules there is a general solvent loop assuming (note the order):

1. At the beginning of the molecule topology there is an arbitrary number of atoms with Lennard-Jones and coulomb interactions.
2. Then we have an arbitrary number of atoms with coulomb interactions only.
3. And finally there can be an arbitrary number of atoms with Lennard-Jones interactions only.

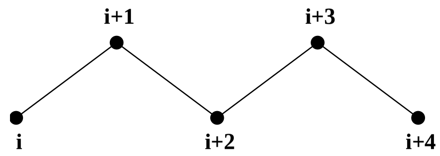


Figure 4.15: Atoms along an alkane chain.

$$r_B = \left(\alpha\sigma_B^6(1-\lambda)^2 + r^6\right)^{\frac{1}{6}} \qquad (4.88)$$

where V^A and V^B are the normal 'hard core' Van der Waals or Electrostatic potentials in state A ($\lambda = 0$) and state B ($\lambda = 1$) respectively, α is the soft-core parameter, which mainly controls the height of the potential around $r = 0$, σ is the radius of the interaction, which is $(C_{12}/C_6)^{1/6}$ or a predefined value when C_6 or C_{12} is zero. For intermediate λ, r_A and r_B alter the interactions very little when $r > \alpha^{1/6}\sigma$ and they quickly switch the soft-core interaction to an almost constant value when r becomes smaller (Fig. 4.14). The force is:

$$F_{sc}(r) = -\frac{\partial V_{sc}(r)}{\partial r} = (1-\lambda)F^A(r_A)\left(\frac{r}{r_A}\right)^5 + \lambda F^B(r_B)\left(\frac{r}{r_B}\right)^5 \qquad (4.89)$$

where F^A and F^B are the 'hard core' forces. The contribution to the derivative of the free energy is:

$$\frac{\partial V_{sc}(r)}{\partial \lambda} = -V^A(r_A) + V^B(r_B) + \frac{1}{3}\alpha\lambda(1-\lambda)\left(-F^A(r_A)\sigma_A^6 r_A^{-5} + F^B(r_B)\sigma_B^6 r_B^{-5}\right) \quad (4.90)$$

4.4 Methods

4.4.1 Exclusions and 1-4 Interactions.

Atoms within a molecule that are close by in the chain, i.e. atoms that are covalently bonded, or linked by one respectively two atoms are so-called *first neighbors*, *second neighbors* and *third neighbors*, (see Fig. 4.15). Since the interactions of atom **i** with **i+1**

and the interaction of atom **i** with atom **i+2** are mainly quantum mechanical, they can not be modeled by a Lennard-Jones potential. Instead it is assumed that these interactions are adequately modeled by a harmonic bond term or constraint (**i,i+1**) and a harmonic angle term (**i,i+2**). The first and second neighbors (atoms **i+1**and **i+2**) are therefore *excluded* from the Lennard-Jones interaction list of atom **i**; atoms **i+1** and **i+2** are called *exclusions* of atom **i**.

For third neighbors the normal Lennard-Jones repulsion is sometimes still too strong, which means that when applied to a molecule the molecule would deform or break due to the internal strain. This is especially the case for Carbon-Carbon interactions in a *cis*-conformation (e.g. *cis*-butane). Therefore for some of these interactions the Lennard-Jones repulsion has been reduced in the GROMOS force field, which is implemented by keeping a separate list of 1-4 and normal Lennard-Jones parameters. In other force fields, such as OPLS [50], the standard Lennard-Jones parameters are reduced by a factor of two, but in that case also the dispersion (r^{-6}) and the coulomb interaction are scaled. GROMACS can use either of these methods.

4.4.2 Charge Groups.

In principle the force calculation in MD is an $O(N^2)$ problem. Therefore we apply a cut-off for non-bonded force (NBF) calculations: only the particles within a certain distance of each other are interacting. This reduces the cost to $O(N)$ (typically $100N$ to $200N$) of the NBF. It also introduces an error, which is, in most cases, acceptable, except when applying the cut-off implies the creation of charges, in which case you should consider using the lattice sum methods provided by GROMACS.

Consider a water molecule interacting with another atom. When we would apply the cut-off on an atom-atom basis we might include the atom-Oxygen interaction (with a charge of -0.82) without the compensating charge of the Hydrogens and so induce a large dipole moment over the system. Therefore we have to keep groups of atoms with total charge 0 together, the so-called *charge groups*.

4.4.3 Treatment of cut-offs

GROMACS is quite flexible in treating cut-offs, which implies there can be quite a number of parameters to set. These parameters are set in the input file for grompp. There are two sort of parameters that affect the cut-off interactions; you can select which type of interaction to use in each case, and which cut-offs should be used in the neighborsearching.

For both Coulomb and van der Waals interactions there are interaction type selectors (termed `vdwtype` and `coulombtype`) and two parameters, for a total of six nonbonded interaction parameters. See sec. 7.3.1 for a complete description of these parameters.

The neighbor searching (NS) can be performed using a single-range, or a twin-range approach. Since the former is merely a special case of the latter we will discuss the more general twin-range. In this case NS is described by two radii `rlist` and max(`rcoulomb`,`rvdw`). Usually one builds the neighbor list every 10 time steps or every 20 fs (parameter `nstlist`). In the neighbor list all interaction pairs that fall within `rlist` are stored. Furthermore, the interactions between pairs that do not fall within `rlist` but do fall within and max(`rcoulomb`,`rvdw`) are computed during NS, and the forces and energy are stored separately, and added to short-range forces at every time step between successive NS. If `rlist` = max(`rcoulomb`,`rvdw`) no forces are evaluated during neighbor list generation. The virial is calculated from the sum of the short- and long-range forces. This means that the virial can be slightly asymmetrical at non-NS steps. In single precision the virial is almost always asymmetrical, because the off-diagonal elements are about as large as each element in the sum. In most cases this is not really a problem, since the fluctuations in de virial can be 2 orders of magnitude larger than the average.

Except for the plain cut-off, all of the interaction functions in Table 4.2 require that neighbor searching is done with a larger radius than the r_c specified for the functional form, because of the use of charge groups. The extra radius is typically of the order of 0.25 nm (roughly the largest distance between two atoms in a charge group plus the distance a charge group can diffuse within neighbor list updates).

The algorithm to generate such a list can be derived from graph theory, considering each particle in a molecule as a bead in a graph, the bonds as edges.

1 represent the bonds and atoms as bidirectional graph

2 make all atoms white

3 make one of the white atoms black (atom i) and put it in the central box

4 make all of the neighbors of i that are currently white, grey

5 pick one of the grey atoms (atom j), give it the correct periodicity with respect to any of its black neighbors and make it black

6 make all of the neighbors of j that are currently white, grey

7 if any grey atom remains, goto [5]

8 if any white atom remains, goto [3]

Using this algorithm we can

- optimize the bonded force calculation as well as shake
- calculate the virial from the bonded forces in the single sum way again

Find a representation of the bonds as a bidirectional graph.

B.1.4 Virial from Covalent Bonds.

The covalent bond force gives a contribution to the virial, we have

$$b = \|\boldsymbol{r}_{ij}^n\| \tag{B.15}$$

$$V_b = \frac{1}{2}k_b(b-b_0)^2 \tag{B.16}$$

$$\boldsymbol{F}_i = -\nabla V_b \tag{B.17}$$

$$= k_b(b-b_0)\frac{\boldsymbol{r}_{ij}^n}{b} \tag{B.18}$$

$$\boldsymbol{F}_j = -\boldsymbol{F}_i \tag{B.19}$$

The virial contribution from the bonds then is

$$\Xi_b = -\frac{1}{2}(\boldsymbol{r}_i^n \otimes \boldsymbol{F}_i \; + \; \boldsymbol{r}_j \otimes \boldsymbol{F}_j) \tag{B.20}$$

$$= -\frac{1}{2}\boldsymbol{r}_{ij}^n \otimes \boldsymbol{F}_i \tag{B.21}$$

In a triclinic system there are 27 possible images of i, when truncated octahedron is used there are 15 possible images.

B.1.2 Virial from non-bonded forces.

Here the derivation for the single sum virial in the *non-bonded force* routine is given. $i \neq j$ in all formulae below.

$$\begin{aligned}
\Xi &= -\frac{1}{2}\sum_{i<j}^{N} \boldsymbol{r}_{ij}^{n} \otimes \boldsymbol{F}_{ij} && \text{(B.5)}\\
&= -\frac{1}{4}\sum_{i=1}^{N}\sum_{j=1}^{N} (\boldsymbol{r}_i + \delta_i - \boldsymbol{r}_j) \otimes \boldsymbol{F}_{ij} && \text{(B.6)}\\
&= -\frac{1}{4}\sum_{i=1}^{N}\sum_{j=1}^{N} (\boldsymbol{r}_i + \delta_i) \otimes \boldsymbol{F}_{ij} - \boldsymbol{r}_j \otimes \boldsymbol{F}_{ij} && \text{(B.7)}\\
&= -\frac{1}{4}\left(\sum_{i=1}^{N}\sum_{j=1}^{N} (\boldsymbol{r}_i + \delta_i) \otimes \boldsymbol{F}_{ij} \; - \; \sum_{i=1}^{N}\sum_{j=1}^{N} \boldsymbol{r}_j \otimes \boldsymbol{F}_{ij}\right) && \text{(B.8)}\\
&= -\frac{1}{4}\left(\sum_{i=1}^{N} (\boldsymbol{r}_i + \delta_i) \otimes \sum_{j=1}^{N} \boldsymbol{F}_{ij} \; - \; \sum_{j=1}^{N} \boldsymbol{r}_j \otimes \sum_{i=1}^{N} \boldsymbol{F}_{ij}\right) && \text{(B.9)}\\
&= -\frac{1}{4}\left(\sum_{i=1}^{N} (\boldsymbol{r}_i + \delta_i) \otimes \boldsymbol{F}_{i} \; + \; \sum_{j=1}^{N} \boldsymbol{r}_j \otimes \boldsymbol{F}_{j}\right) && \text{(B.10)}\\
&= -\frac{1}{4}\left(2\sum_{i=1}^{N} \boldsymbol{r}_i \otimes \boldsymbol{F}_{i} + \sum_{i=1}^{N} \delta_i \otimes \boldsymbol{F}_{i}\right) && \text{(B.11)}
\end{aligned}$$

In these formulae we introduced

$$\begin{aligned}
\boldsymbol{F}_i &= \sum_{j=1}^{N} \boldsymbol{F}_{ij} && \text{(B.12)}\\
\boldsymbol{F}_j &= \sum_{i=1}^{N} \boldsymbol{F}_{ji} && \text{(B.13)}
\end{aligned}$$

which is the total force on i resp. j. Because we use Newton's third law

$$\boldsymbol{F}_{ij} \; = \; -\boldsymbol{F}_{ji} \qquad \text{(B.14)}$$

we must in the implementation double the term containing the shift δ_i.

B.1.3 The intramolecular shift (mol-shift).

For the bonded-forces and shake it is possible to make a *mol-shift* list, in which the periodicity is stored. We simple have an array `mshift` in which for each atom an index in the `shiftvec` array is stored.

	Type	Parameters
Coulomb	Plain cut-off	r_c, ε_r
	Reaction field	r_c, ε_{rf}
	Shift function	r_1, r_c, ε_r
	Switch function	r_1, r_c, ε_r
VdW	Plain cut-off	r_c
	Shift function	r_1, r_c
	Switch function	r_1, r_c

Table 4.2: Parameters for the different functional forms of the non-bonded interactions.

4.5 Dummy atoms.

Dummy atoms can be used in GROMACS in a number of ways. We write the position of the dummy particle $\boldsymbol{r}_d$ as a function of the positions of other particles $\boldsymbol{r}_i$: $\boldsymbol{r}_d = f(\boldsymbol{r}_1..\boldsymbol{r}_n)$. The dummy, which may carry charge, or can be involved in other interactions can now be used in the force calculation. The force acting on the dummy particle must be redistributed over the atoms in a consistent way. A good way to do this can be found in ref. [53]. We can write the potential energy as

$$V = V(\boldsymbol{r}_d, \boldsymbol{r}_1..\boldsymbol{r}_n) = V^*(\boldsymbol{r}_1..\boldsymbol{r}_n) \qquad (4.91)$$

The force on the particle i is then

$$\boldsymbol{F}_i = -\frac{\partial V^*}{\partial \boldsymbol{r}_i} = -\frac{\partial V}{\partial \boldsymbol{r}_i} - \frac{\partial \boldsymbol{r}_d}{\partial \boldsymbol{r}_i}\frac{\partial V}{\partial \boldsymbol{r}_d} = \boldsymbol{F}_i^{direct} + \boldsymbol{F}_i' \qquad (4.92)$$

the first term of which is the normal force. The second term is the force on particle i due to the dummy particle, which can be written in tensor notation:

$$\boldsymbol{F}_i' = \begin{bmatrix} \dfrac{\partial x_d}{\partial x_i} & \dfrac{\partial y_d}{\partial x_i} & \dfrac{\partial z_d}{\partial x_i} \\ \dfrac{\partial x_d}{\partial y_i} & \dfrac{\partial y_d}{\partial y_i} & \dfrac{\partial z_d}{\partial y_i} \\ \dfrac{\partial x_d}{\partial z_i} & \dfrac{\partial y_d}{\partial z_i} & \dfrac{\partial z_d}{\partial z_i} \end{bmatrix} \boldsymbol{F}_d \qquad (4.93)$$

where $\boldsymbol{F}_d$ is the force on the dummy particle and x_d, y_d and z_d are the coordinates of the dummy particle. In this way the total force and the total torque are conserved [53].

As a further note, the computation of the virial (eqn. 3.18) virial is non-trivial when dummy atoms are used. Since the virial involves a summation over all the atoms (rather than virtual particles) the forces most be redistributed from the dummies to the atoms (using eqn. 4.93) *before* computation of the virial. In some special cases where the forces on the atoms can be written as a linear combination of the forces on the dummies (types 2 and 3 below) there is no difference between computing the virial before and after the redistribution of forces. However, in the general case redistribution should be done first.

There are six ways to construct dummies from surrounding atoms in GROMACS, which we categorize based on the number of constructing atoms. Note that all dummies types mentioned can

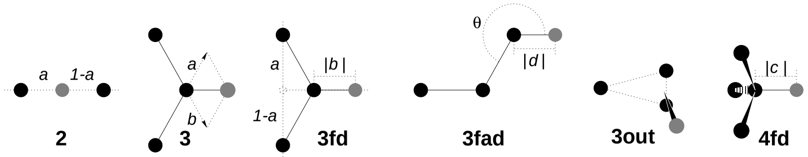


Figure 4.16: The six different types of dummy atom construction in GROMACS, the constructing atoms are shown as black circles, the dummy atoms in grey.

be constructed from types 3fd (normalized, in-plane) and 3out (non-normalized, out of plane). However, the amount of computation involved increases sharply along this list, so it is strongly recommended to always use the first dummy type that will be sufficient for a certain purpose. An overview of the dummy constructions is given in Fig. 4.16.

2. As a linear combination of two atoms (Fig. 4.16 2):

$$\boldsymbol{r}_d \;=\; \boldsymbol{r}_i + a\boldsymbol{r}_{ij} \tag{4.94}$$

in this case the dummy is on the line through atoms i and j. The force on particles i and j due to the force on the dummy can be computed as:

$$\begin{aligned} \boldsymbol{F}'_i &= (1-a)\boldsymbol{F}_d \\ \boldsymbol{F}'_j &= a\,\boldsymbol{F}_d \end{aligned} \tag{4.95}$$

3. As a linear combination of three atoms (Fig. 4.16 3):

$$\boldsymbol{r}_d \;=\; \boldsymbol{r}_i + a\boldsymbol{r}_{ij} + b\boldsymbol{r}_{ik} \tag{4.96}$$

in this case the dummy is in the plane of the other three particles. The force on particles i, j and k due to the force on the dummy can be computed as:

$$\begin{aligned} \boldsymbol{F}'_i &= (1-a-b)\boldsymbol{F}_d \\ \boldsymbol{F}'_j &= a\,\boldsymbol{F}_d \\ \boldsymbol{F}'_k &= b\,\boldsymbol{F}_d \end{aligned} \tag{4.97}$$

3fd. In the plane of three atoms, with a fixed distance (Fig. 4.16 3fd):

$$\boldsymbol{r}_d \;=\; \boldsymbol{r}_i + b\frac{\boldsymbol{r}_{ij} + a\boldsymbol{r}_{jk}}{|\boldsymbol{r}_{ij} + a\boldsymbol{r}_{jk}|} \tag{4.98}$$

in this case the dummy is in the plane of the other three particles at a distance of $|b|$ from i. The force on particles i, j and k due to the force on the dummy can be computed as:

$$\begin{aligned} \boldsymbol{F}'_i &= \boldsymbol{F}_d - \gamma(\boldsymbol{F}_d - \boldsymbol{p}) \\ \boldsymbol{F}'_j &= (1-a)\gamma(\boldsymbol{F}_d - \boldsymbol{p}) \quad \text{where} \quad \begin{array}{l} \gamma = \dfrac{b}{|\boldsymbol{r}_{ij} + a\boldsymbol{r}_{jk}|} \\ \boldsymbol{p} = \dfrac{\boldsymbol{r}_{id}\cdot\boldsymbol{F}_d}{\boldsymbol{r}_{id}\cdot\boldsymbol{r}_{id}}\boldsymbol{r}_{id} \end{array} \\ \boldsymbol{F}'_k &= a\gamma(\boldsymbol{F}_d - \boldsymbol{p}) \end{aligned} \tag{4.99}$$

Appendix B

Some implementation details

In this chapter we will present some implementation details. This is far from complete, but we deemed it necessary to clarify some things that would otherwise be hard to understand.

B.1 Single Sum Virial in GROMACS.

The virial Ξ can be written in full tensor form as:

$$\Xi \;=\; -\frac{1}{2}\sum_{i<j}^{N} \boldsymbol{r}_{ij} \otimes \boldsymbol{F}_{ij} \tag{B.1}$$

where $\otimes$ denotes the *direct product* of two vectors[1]. When this is computed in the inner loop of an MD program 9 multiplications and 9 additions are needed[2].

Here it is shown how it is possible to extract the virial calculation from the inner loop [74].

B.1.1 Virial.

In a system with Periodic Boundary Conditions, the periodicity must be taken into account for the virial:

$$\Xi \;=\; -\frac{1}{2}\sum_{i<j}^{N} \boldsymbol{r}^n_{ij} \otimes \boldsymbol{F}_{ij} \tag{B.2}$$

where $\boldsymbol{r}^n_{ij}$ denotes the distance vector of the *nearest image* of atom i from atom j. In this definition we add a *shift vector* δ_i to the position vector $\boldsymbol{r}_i$ of atom i. The difference vector $\boldsymbol{r}^n_{ij}$ is thus equal to:

$$\boldsymbol{r}^n_{ij} \;=\; \boldsymbol{r}_i + \delta_i - \boldsymbol{r}_j \tag{B.3}$$

or in shorthand:

$$\boldsymbol{r}^n_{ij} \;=\; \boldsymbol{r}^n_i - \boldsymbol{r}_j \tag{B.4}$$

---

[1] $(\mathbf{u} \otimes \mathbf{v})^{\alpha\beta} \;=\; \mathbf{u}_\alpha \mathbf{v}_\beta$

[2] The calculation of Lennard-Jones and Coulomb forces is about 50 floating point operations.

2. `IAMCOOL`, if this is explicitly set to NO your GROMACS life will be dull and boring. (i.e., no cool quotes).
3. `WHERE`, when set print debugging info on line numbers.
4. `LOG_BUFS`, the size of the buffer for file I/O. When set to 0, all file I/O will be unbuffered and therefore very slow. This can be handy for debugging purposes, because it ensures that all files are always totally up-to-date.
5. `GMXNPRI`, for SGI systems only. When set, gives the default non-degrading priority (npri) for `mdrun`, `nmrun`, `g_covar` and `g_nmeig`, e.g. setting `setenv GMXNPRI 250` causes all runs to be performed at near-lowest priority by default.
6. `GMX_VIEW_XPM`, `GMX_VIEW_XVG`, `GMX_VIEW_EPS` and `GMX_VIEW_PDB`, commands used to automatically view resp. `.xvg`, `.xpm`, `.eps` and `.pdb` file types. Default to `xv`, `xmgr`, `ghostview` and `rasmol`. Set to empty to disable automatic viewing of a particular file type. The command will be forked off and run in the background at the same priority as the GROMACS tool (which might not be what you want). Be careful not to use a command which blocks the terminal (e.g. `vi`), since multiple instances might be run.

Some other environment variables are specific to one program, such as TOTAL for the `do_shift` program, and DSPP for the `do_dssp` program.

3fad. In the plane of three atoms, with a fixed angle and distance (Fig. 4.16 3fad):

$$\boldsymbol{r}_d \;=\; \boldsymbol{r}_i + d\cos\theta\frac{\boldsymbol{r}_{ij}}{|\boldsymbol{r}_{ij}|} + d\sin\theta\frac{\boldsymbol{r}_\perp}{|\boldsymbol{r}_\perp|} \quad \text{where} \quad \boldsymbol{r}_\perp \;=\; \boldsymbol{r}_{jk} - \frac{\boldsymbol{r}_{ij}\cdot\boldsymbol{r}_{jk}}{\boldsymbol{r}_{ij}\cdot\boldsymbol{r}_{ij}}\boldsymbol{r}_{ij} \qquad (4.100)$$

in this case the dummy is in the plane of the other three particles at a distance of $|d|$ from i at an angle of α with $\boldsymbol{r}_{ij}$. Atom k defines the plane and the direction of the angle. Note that in this case b and α must be specified in stead of a and b (see also sec. 5.2.2). The force on particles i, j and k due to the force on the dummy can be computed as (with $\boldsymbol{r}_\perp$ as defined in eqn. 4.100):

$$\begin{aligned}
\boldsymbol{F}'_i &= \boldsymbol{F}_d - \frac{d\cos\theta}{|\boldsymbol{r}_{ij}|}\boldsymbol{F}_1 + \frac{d\sin\theta}{|\boldsymbol{r}_\perp|}\left(\frac{\boldsymbol{r}_{ij}\cdot\boldsymbol{r}_{jk}}{\boldsymbol{r}_{ij}\cdot\boldsymbol{r}_{ij}}\boldsymbol{F}_2 + \boldsymbol{F}_3\right) \\
\boldsymbol{F}'_j &= \frac{d\cos\theta}{|\boldsymbol{r}_{ij}|}\boldsymbol{F}_1 - \frac{d\sin\theta}{|\boldsymbol{r}_\perp|}\left(\boldsymbol{F}_2 + \frac{\boldsymbol{r}_{ij}\cdot\boldsymbol{r}_{jk}}{\boldsymbol{r}_{ij}\cdot\boldsymbol{r}_{ij}}\boldsymbol{F}_2 + \boldsymbol{F}_3\right) \\
\boldsymbol{F}'_k &= \frac{d\sin\theta}{|\boldsymbol{r}_\perp|}\boldsymbol{F}_2
\end{aligned}$$

$$\text{where} \;\; \boldsymbol{F}_1 = \boldsymbol{F}_d - \frac{\boldsymbol{r}_{ij}\cdot\boldsymbol{F}_d}{\boldsymbol{r}_{ij}\cdot\boldsymbol{r}_{ij}}\boldsymbol{r}_{ij}\,, \;\; \boldsymbol{F}_2 = \boldsymbol{F}_1 - \frac{\boldsymbol{r}_\perp\cdot\boldsymbol{F}_d}{\boldsymbol{r}_\perp\cdot\boldsymbol{r}_\perp}\boldsymbol{r}_\perp \;\text{and}\;\; \boldsymbol{F}_3 = \frac{\boldsymbol{r}_{ij}\cdot\boldsymbol{F}_d}{\boldsymbol{r}_{ij}\cdot\boldsymbol{r}_{ij}}\boldsymbol{r}_\perp \qquad (4.101)$$

3out. As a non-linear combination of three atoms, out of plane (Fig. 4.16 3out):

$$\boldsymbol{r}_d \;=\; \boldsymbol{r}_i + a\boldsymbol{r}_{ij} + b\boldsymbol{r}_{ik} + c(\boldsymbol{r}_{ij}\times\boldsymbol{r}_{ik}) \qquad (4.102)$$

this enables the construction of dummies out of the plane of the other atoms. The force on particles i, j and k due to the force on the dummy can be computed as:

$$\begin{aligned}
\boldsymbol{F}'_j &= \begin{bmatrix} a & -c\,z_{ik} & c\,y_{ik} \\ c\,z_{ik} & a & -c\,x_{ik} \\ -c\,y_{ik} & c\,x_{ik} & a \end{bmatrix}\boldsymbol{F}_d \\
\boldsymbol{F}'_k &= \begin{bmatrix} b & c\,z_{ij} & -c\,y_{ij} \\ -c\,z_{ij} & b & c\,x_{ij} \\ c\,y_{ij} & -c\,x_{ij} & b \end{bmatrix}\boldsymbol{F}_d \qquad (4.103) \\
\boldsymbol{F}'_i &= \boldsymbol{F}_d - \boldsymbol{F}'_j - \boldsymbol{F}'_k
\end{aligned}$$

4fd. From four atoms, with a fixed distance (Fig. 4.16 4fd):

$$\boldsymbol{r}_d \;=\; \boldsymbol{r}_i + c\frac{\boldsymbol{r}_{ij} + a\boldsymbol{r}_{jk} + b\boldsymbol{r}_{jl}}{|\boldsymbol{r}_{ij} + a\boldsymbol{r}_{jk} + b\boldsymbol{r}_{jl}|} \qquad (4.104)$$

in this case the dummy is at a distance of $|c|$ from i. The force on particles i, j, k and l due to the force on the dummy can be computed as:

$$\begin{aligned}
\boldsymbol{F}'_i &= \boldsymbol{F}_d - \gamma(\boldsymbol{F}_d - \boldsymbol{p}) \\
\boldsymbol{F}'_j &= (1-a-b)\gamma(\boldsymbol{F}_d - \boldsymbol{p}) \\
\boldsymbol{F}'_k &= a\gamma(\boldsymbol{F}_d - \boldsymbol{p}) \\
\boldsymbol{F}'_l &= b\gamma(\boldsymbol{F}_d - \boldsymbol{p})
\end{aligned}
\quad \text{where} \quad
\begin{aligned}
\gamma &= \frac{c}{|\boldsymbol{r}_{ij} + a\boldsymbol{r}_{jk} + b\boldsymbol{r}_{jl}|} \\
\boldsymbol{p} &= \frac{\boldsymbol{r}_{id}\cdot\boldsymbol{F}_d}{\boldsymbol{r}_{id}\cdot\boldsymbol{r}_{id}}\boldsymbol{r}_{id}
\end{aligned} \qquad (4.105)$$

4.6 Long Range Electrostatics

4.6.1 Ewald summation

The total electrostatic energy of N particles and the periodic images are given by

$$V = \frac{f}{2} \sum_{n_x} \sum_{n_y} \sum_{n_z*} \sum_{i}^{N} \sum_{j}^{N} \frac{q_i q_j}{\mathbf{r}_{ij,\mathbf{n}}}. \tag{4.106}$$

$(n_x, n_y, n_z) = \mathbf{n}$ is the box index vector, and the star indicates that terms with $i = j$ should be omitted when $(n_x, n_y, n_z) = (0, 0, 0)$. The distance $\mathbf{r}_{ij,\mathbf{n}}$ is the real distance between the charges and not the minimum-image. This sum is conditionally convergent, but very slow.

Ewald summation was first introduced as a method to calculate long-range interactions of the periodic images in crystals [54]. The idea is to convert the single slowly converging sum eqn. 4.106 into two fast converging terms and a constant term:

$$\begin{aligned}
V &= V_{dir} + V_{rec} + V_0 && (4.107)\\
V_{dir} &= \frac{f}{2} \sum_{i,j}^{N} \sum_{n_x} \sum_{n_y} \sum_{n_z*} q_i q_j \frac{\mathrm{erfc}(\beta r_{ij,\mathbf{n}})}{r_{ij,\mathbf{n}}} && (4.108)\\
V_{rec} &= \frac{f}{2\pi V} \sum_{i,j}^{N} q_i q_j \sum_{m_x} \sum_{m_y} \sum_{m_z*} \frac{\exp\left(-(\pi \mathbf{m}/\beta)^2 + 2\pi i \mathbf{m} \cdot (\mathbf{r}_i - \mathbf{r}_j)\right)}{\mathbf{m}^2} && (4.109)\\
V_0 &= -\frac{f\beta}{\sqrt{\pi}} \sum_{i}^{N} q_i^2, && (4.110)
\end{aligned}$$

where β is a parameter that determines the relative weight of the direct and reciprocal sums and $\mathbf{m} = (m_x, m_y, m_z)$. In this way we can use a short cut-off (of the order of 1 nm) in the direct space sum and a short cut-off in the reciprocal space sum (e.g. 10 wave vectors in each direction). Unfortunately, the computational cost of the reciprocal part of the sum increases as N^2 (or $N^{3/2}$ with a slightly better algorithm) and it is therefore not realistic to use for any large systems.

Using Ewald

Don't use Ewald unless you are absolutely sure this is what you want - for almost all cases the PME method below will perform much better. If you still want to employ classical Ewald summation enter this in your `.mdp` file, if the side of your box is about 3 nm:

```
coulombtype      = Ewald
rvdw             = 0.9
rlist            = 0.9
rcoulomb         = 0.9
fourierspacing   = 0.6
ewald_rtol       = 1e-5
```

A.3.1 Multi-processor Optimization

If you want to, you could write your own optimized communication (perhaps using specific libraries for your hardware) instead of MPI. This should never be necessary for normal use (we haven't heard of a modern computer where it isn't possible to run MPI), but if you absolutely want to do it, here are some clues.

The interface between the communication routines and the rest of the GROMACS system is described in the file `$GMXHOME/src/include/network.h` We will give a short description of the different routines below.

extern void gmx_tx(int pid,void *buf,int bufsize);
This routine, when called with the destination processor number, a pointer to a (byte oriented) transfer buffer, and the size of the buffer will send the buffer to the indicated processor (in our case always the neighboring processor). The routine does **not** wait until the transfer is finished.

extern void gmx_tx_wait(int pid);
This routine waits until the previous, or the ongoing transmission is finished.

extern void gmx_txs(int pid,void *buf,int bufsize);
This routine implements a synchronous send by calling the a-synchronous routine and then the wait. It might come in handy to code this differently.

extern void gmx_rx(int pid,void *buf,int bufsize);

extern void gmx_rx_wait(int pid);

extern void gmx_rxs(int pid,void *buf,int bufsize);
The very same routines for receiving a buffer and waiting until the reception is finished.

extern void gmx_init(int pid,int nprocs);
This routine initializes the different devices needed to do the communication. In general it sets up the communication hardware (if it is accessible) or does an initialize call to the lower level communication subsystem.

extern void gmx_stat(FILE *fp,char *msg);
With this routine we can diagnose the ongoing communication. In the current implementation it prints the various contents of the hardware communication registers of the (Intel *i*860) multiprocessor boards to a file.

A.4 Environment Variables

GROMACS programs may be influenced by the use of environment variables. First of all, the variables set in the GMXRC file are essential for running and compiling GROMACS. Other variables are:

1. `DUMP_NL`, dump neighbor list. If set to a positive number the *entire* neighbor list is printed in the log file (may be many megabytes). Mainly for debugging purposes, but may also be handy for porting to other platforms.

The energies in single precision are accurate up to the last decimal, the last one or two decimals of the forces are non-significant. The virial is less accurate than the forces, since the virial is only one order of magnitude larger than the size of each element in the sum over all atoms (sec. B.1). In most cases this is not really a problem, since the fluctuations in de virial can be 2 orders of magnitude larger than the average. In periodic charged systems these errors are often negligible. Especially cut-off's for the Coulomb interactions cause large errors in the energies, forces and virial. Even when using a reaction-field or lattice sum method the errors are larger than or comparable to the errors due to the single precision. Since MD is chaotic, trajectories with very similar starting conditions will diverge rapidly, the divergence is faster in single precision than in double precision.

For most simulations single precision is accurate enough. In some cases double precision is required to get reasonable results:

- normal mode analysis, for the conjugate gradient minimization and the calculation and diagonalization of the Hessian
- calculation of the constraint force between two large groups of atoms
- energy conservation (this can only be done without temperature coupling and without cut-off's)

A.3 Porting GROMACS

The GROMACS system is designed with portability as a major design goal. However there are a number of things we assume to be present on the system GROMACS is being ported on. We assume the following features:

1. A UNIX-like operating system (BSD 4.x or SYSTEM V rev.3 or higher) or UNIX-like libraries running under e.g. CygWin
2. an ANSI C compiler
3. optionally a Fortran-77 compiler or Fortran-90 compiler for faster (on some computers) inner loop routines
4. optionally the Nasm assembler to use the assembly innerloops on x86 processors.

There are some additional features in the package that require extra stuff to be present, but it is checked for in the configuration script and you will be warned if anything important is missing.

That's the requirements for a single processor system. If you want to compile GROMACS for a multiple processor environment you also need a MPI library (Message-Passing Interface) to perform the parallel communication. This is always shipped with supercomputers, and for workstations you can find links to free MPI implementations through the GROMACS homepage at www.gromacs.org.

The `fourierspacing` parameter times the box dimensions determines the highest magnitude of wave vectors m_x, m_y, m_z to use in each direction. With a 3 nm cubic box this example would use 11 wave vectors (from -5 to 5) in each direction. The `ewald_rtol` parameter is the relative strength of the electrostatic interaction at the cut-off. Decreasing this gives you a more accurate direct sum, but a less accurate reciprocal sum.

4.6.2 PME

Particle-mesh Ewald is a method proposed by Tom Darden [55, 56] to improve the performance of the reciprocal sum. Instead of directly summing wave vectors, the charges are assigned to a grid using cardinal B-spline interpolation. This grid is then Fourier transformed with a 3D FFT algorithm and the reciprocal energy term obtained by a single sum over the grid in k-space.

The potential at the grid points is calculated by inverse transformation, and by using the interpolation factors we get the forces on each atom.

The PME algorithm scales as $N\log(N)$, and is substantially faster than ordinary Ewald summation on medium to large systems. On very small systems it might still be better to use Ewald to avoid the overhead in setting up grids and transforms.

Using PME

To use Particle-mesh Ewald summation in GROMACS, specify the following lines in your `.mdp` file:

```
coulombtype     = PME
rvdw            = 0.9
rlist           = 0.9
rcoulomb        = 0.9
fourierspacing  = 0.12
pme_order       = 4
ewald_rtol      = 1e-5
```

In this case the `fourierspacing` parameter determines the maximum spacing for the FFT grid and `pme_order` controls the interpolation order. Using 4th order (cubic) interpolation and this spacing should give electrostatic energies accurate to about $5\cdot10^{-3}$. Since the Lennard-Jones energies are not this accurate it might even be possible to increase this spacing slightly.

Pressure scaling works with PME, but be aware of the fact that anisotropic scaling can introduce artificial ordering in some systems.

4.6.3 PPPM

The Particle-Particle Particle-Mesh methods of Hockney & Eastwood can also be applied in GROMACS for the treatment of long range electrostatic interactions [57, 55, 58]. With this algorithm the charges of all particles are spread over a grid of dimensions (n_x,n_y,n_z) using a weighting function called the triangle-shaped charged distribution:

$$W(\boldsymbol{r}) \;=\; W(x)\,W(y)\,W(z)$$
$$W(\xi) \;=\; \begin{cases} \frac{3}{4} - \left(\frac{\xi}{h}\right)^2 & |\xi| \leq \frac{h}{2} \\ \frac{1}{2}\left(\frac{3}{2} - \frac{|\xi|}{h}\right)^2 & \frac{h}{2} < |\xi| < \frac{3h}{2} \\ 0 & \frac{3h}{2} \leq |\xi| \end{cases} \tag{4.111}$$

where ξ (is x, y or z) is the distance to a grid point in the corresponding dimension. Only the 27 closest grid points need to be taken into account for each charge.

Then, this charge distribution is Fourier transformed using a 3D inverse FFT routine. In Fourier space a convolution with function $\hat{G}$ is performed:

$$\hat{G}(k) \;=\; \frac{\hat{g}(k)}{\epsilon_0 k^2} \tag{4.112}$$

where $\hat{g}$ is the Fourier transform of the charge spread function g(r). This yield the long range potential $\hat{\phi}(k)$ on the mesh, which can be transformed using a forward FFT routine into the real space potential. Finally the potential and forces are retrieved using interpolation [58]. It is not easy to calculate the full long-range virial tensor with PPPM, but it is possible to obtain the trace. This means that the sum of the pressure components is correct (and therefore the isotropic pressure) but not necessarily the individual pressure components!

Using PPPM

To use the PPPM algorithm in GROMACS, specify the following lines in your `.mdp` file:

```
coulombtype     = PPPM
rlist           = 1.0
rcoulomb        = 0.85
rcoulomb_switch = 0.0
rvdw            = 1.0
fourierspacing  = 0.075
```

For details on the switch parameters see the section on modified long-range interactions in this manual. When using PPPM we recommend to take at most 0.075 nm per gridpoint (e.g. 20 gridpoints for 1.5 nm). PPPM does not provide the same accuracy as PME but can be slightly faster in some cases. Due to the problem with the pressure tensor you shouldn't use it with pressure coupling.

We're somewhat ambivalent about PPPM, so if you use it please contact us - otherwise it might be removed from future relases so we can concentrate our efforts on PME.

4.6.4 Optimizing Fourier transforms

To get the best possible performance you should try to avoid large prime numbers for grid dimensions. The FFT code used in GROMACS is optimized for grid sizes of the form $2^a 3^b 5^c 7^d 11^e 13^f$,

---

Appendix A

Technical Details

A.1 Installation

The entire GROMACS package is Free Software, licensed under the GNU General Public License. The main distribution site is our WWW server at www.gromacs.org.

The package is mainly distributed as source code, but we also provide RPM packages for Linux. On the home page you will find all the information you need to install the package, mailing lists with archives, and several additional online resources like contributed topologies, etc. The default installation action is simply to unpack the source code and the issue

```
./configure
make
make install
```

The configuration script should automatically determine the best options for your platform, and it will tell you if anything is missing on your system. You will also find detailed step-by-step installation instructions on the website.

A.2 Single or Double precision

GROMACS can be compiled in either single or double precision. The default choice is single precision, but it is easy to turn on double precision by selecting the `--enable-double` option to the configuration script. Double precision will be 0 to 50% slower than single precision depending on the architecture you are running on. Double precision will use somewhat more memory and run input, energy and full-precision trajectory files will be almost twice as large. Note that the assembly loops are only available in single precision; Although the Intel SSE2 instruction set (available on Pentium IV and later) supports double precision instructions the performance is much lower than single precision. It would also mean very much extra work for a feature that very few people use, so we will probably not provide double precision assembly loops in the future either.

where $e + f$ is 0 or 1 and the other exponents arbitrary. (See further the documentation of the FFT algorithms at www.fftw.org.

It is also possible to optimize the transforms for the current problem by performing some calculations at the start of the run. This is not done per default since it takes a couple of minutes, but for large runs it will save time. Turn it on by specifying

```
optimize_fft      = yes
```

in your `.mdp` file.

When running in parallel the grid must be communicated several times and thus hurting scaling performance. With PME you can improve this by increasing grid spacing while simultaneously increasing the interpolation to e.g. 6th order. Since the interpolation is entirely local a this will improve the scaling in most cases.

4.7 All-hydrogen forcefield

The GROMACS all-hydrogen forcefield is almost identical to the normal GROMACS forcefield, since the extra hydrogens have no Lennard-Jones interaction and zero charge. The only differences are in the bond angle and improper dihedral angle terms. This forcefield is only useful when you need the exact hydrogen positions, for instance for distance restraints derived from NMR measurements.

4.8 GROMOS-96 notes

4.8.1 The GROMOS-96 force field

GROMACS supports the GROMOS-96 force fields [46]. All parameters for the 43a1, 43a2 (development, improved alkane dihedrals) and 43b1 (vacuum) force fields are included. All standard building blocks are included and topologies can be build automatically by `pdb2gmx`. The GROMOS-96 force field is a further development of the GROMOS-87 force field on which the GROMACS forcefield is based. The GROMOS-96 force field has improvements over the GROMACS force field for proteins and small molecules. It is, however, not recommended to be used for long alkanes and lipids. The GROMOS-96 force field differs from the GROMACS force field in a few aspects:

- the force field parameters
- the parameters for the bonded interactions are not linked to atom types
- a fourth power bond stretching potential (sec. 4.2.1)
- an angle potential based on the cosine of the angle (sec. 4.2.4)

There are two differences in implementation between GROMACS and GROMOS-96 which can lead to slightly different results when simulating the same system with both packages:

- in GROMOS-96 neighbor searching for solvents is performed on the first atom of the solvent molecule, this is not implemented in GROMACS, but the difference with searching with centers of charge groups is very small
- the virial in GROMOS-96 is molecule based, this is not implemented in GROMACS, which uses atomic virials

The GROMOS-96 force field was parameterized with a Lennard-Jones cut-off of 1.4 nm, so be sure to use a Lennard-Jones cut-off of at least 1.4. A larger cut-off is possible, because the Lennard-Jones potential and forces are almost zero beyond 1.4 nm.

4.8.2 GROMOS-96 files

GROMACS can read and write GROMOS-96 coordinate and trajectory files. These files should have the extension `.g96`. Such a file can be a GROMOS-96 initial/final configuration file or a coordinate trajectory file or a combination of both. The file is fixed format, all floats are written as 15.9 (files can get huge). GROMACS supports the following data blocks in the given order:

- Header block:

  ```
  TITLE (mandatory)
  ```

- Frame blocks:

  ```
  TIMESTEP (optional)
  POSITION/POSITIONRED (mandatory)
  VELOCITY/VELOCITYRED (optional)
  BOX (optional)
  ```

See the GROMOS-96 manual [46] for a complete description of the blocks. Note that all GROMACS programs can read compressed or gzip:ed files.

---

the double integral of the charge density ($\rho(z)$):

$$\psi(z) - \psi(-\infty) = -\int_{-\infty}^{z} dz' \int_{-\infty}^{z'} \rho(z'')dz''/\epsilon_0 \tag{8.38}$$

where the position $z = -\infty$ is far enough in the bulk phase that the field is zero. With this method, it is possible to "split" the total potential into separate contributions from lipid and water molecules. The program `g_potential` divides the box in slices and sums all charges of the atoms in each slice. It then integrates this charge density, giving the electric field, and the electric field, giving the potential. Charge density, field and potential are written to `xvgr`-input files.

The program `g_coord` is a very simple analysis program. All it does is print the coordinates of selected atoms to three files, containing respectively the *x*-, *y*- and *z*-coordinates of those atoms. It can also calculate the center of mass of one or more molecules and print the coordinates of the center of mass to three files. By itself, this is probably not a very useful analysis, but having the coordinates of selected molecules or atoms can be very handy for further analysis, not only in interface systems.

The program `g_pvd` calculates a lot of properties, among which the density of a group in particles per unit of volume, but not a density that takes the mass of the atoms into account. The program `g_density` also calculates the density of a group, but takes the masses into account and gives a plot of the density against a box axis. This is useful for looking at the distribution of groups or atoms across the interface.

8.13 Chemical shifts

```
total
do_shift
```

You can compute the NMR chemical shifts of protons with the program `do_shift`. This is just an GROMACS interface to the public domain program `total` [73]. For further information, read the article.

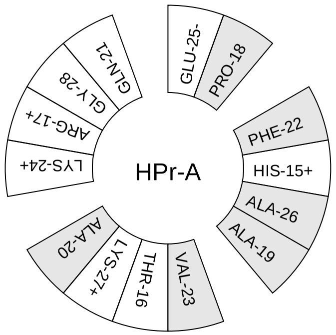


Figure 8.13: Helical wheel projection of the N-terminal helix of HPr.

8.12 Interface related items

```
g_order
g_density
g_potential
g_coord
```

When simulating molecules with long carbon tails, it can be interesting to calculate their average orientation. There are several flavors of order parameters, most of which are related. The program `g_order` can calculate order parameters using the equation

$$S_z = \frac{3}{2}\langle\cos^2\theta_z\rangle - \frac{1}{2} \tag{8.37}$$

where θ_z is the angle between the z-axis of the simulation box and the molecular axis under consideration. The latter is defined as the vector from C_{n-1} to C_{n+1}. The parameters S_x and S_y are defined in the same way. The brackets imply averaging over time and molecules. Order parameters can vary between 1 (full order along the interface normal) and $-1/2$ (full order perpendicular to the normal), with a value of zero in the case of isotropic orientation.

The program can do two things for you. It can calculate the order parameter for each CH_2 segment separately, for any of three axes, or it can divide the box in slices and calculate the average value of the order parameter per segment in one slice. The first method gives an idea of the ordering of a molecule from head to tail, the second method gives an idea of the ordering as function of the box length.

The electrostatic potential (ψ) across the interface can be computed from a trajectory by evaluating

---

Chapter 5

Topologies

5.1 Introduction

GROMACS must know on which atoms and combinations of atoms the various contributions to the potential functions (see chapter 4) must act. It must also know what parameters must be applied to the various functions. All this is described in the *topology* file `*.top`, which lists the *constant attributes* of each atom. There are many more atom types than elements, but only atom types present in biological systems are parameterized in the force field, plus some metals, ions and silicon. The bonded and special interactions are determined by fixed lists that are included in the topology file. Certain non-bonded interactions must be excluded (first and second neighbors), as these are already treated in bonded interactions. In addition there are *dynamic attributes* of atoms: their positions, velocities and forces, but these do not strictly belong to the molecular topology.

This Chapter describes the set up of the topology file, the `*.top` file: what the parameters stand for and how/where to change them if needed.

Note: if you construct your own topologies, we encourage you to upload them to our topology archive at www.gromacs.org! Just imagine how thankful you'd have been if your topology had been available there before you started. The same goes for new force field or modified versions of the standard force fields - contribute them to the force field archive!

The files are grouped per force field type (named e.g. `gmx` for the GROMACS force field or `G43a1` for the GROMOS96 force field). All files for one force field have names beginning with `ff???` where `???` stands for the force field name.

5.2 Particle type

In GROMACS there are 5 types of particles, see Table 5.1. Only regular atoms and dummy particles are used in GROMACS, nuclei, shells and bond shells are necessary for polarizable force fields, which we don't yet have.

Particle	Symbol
atoms	A
nucleuss	N
shells	S
bond shells	B
dummys	D

Table 5.1: Particle types in GROMACS

5.2.1 Atom types

GROMACS uses 47 different atom types, as listed below, with their corresponding masses (in a.m.u.). This is the same listing as in the file `ff???.atp` (.atp = **a**tom **t**ype **p**arameter file), therefore in this file you can change and/or add an atom type.

```
    O  15.99940 ;     carbonyl oxygen (C=O)
   OM  15.99940 ;     carboxyl oxygen (CO-)
   OA  15.99940 ;     hydroxyl oxygen (OH)
   OW  15.99940 ;     water oxygen
    N  14.00670 ;     peptide nitrogen (N or NH)
   NT  14.00670 ;     terminal nitrogen (NH2)
   NL  14.00670 ;     terminal nitrogen (NH3)
  NR5  14.00670 ;     aromatic N (5-ring,2 bonds)
 NR5*  14.00670 ;     aromatic N (5-ring,3 bonds)
   NP  14.00670 ;     porphyrin nitrogen
    C  12.01100 ;     bare carbon (peptide,C=O,C-N)
  CH1  13.01900 ;     aliphatic CH-group
  CH2  14.02700 ;     aliphatic CH2-group
  CH3  15.03500 ;     aliphatic CH3-group
 CR51  13.01900 ;     aromatic CH-group (5-ring), united
 CR61  13.01900 ;     aromatic CH-group (6-ring), united
   CB  12.01100 ;     bare carbon (5-,6-ring)
    H   1.00800 ;     hydrogen bonded to nitrogen
   HO   1.00800 ;     hydroxyl hydrogen
   HW   1.00800 ;     water hydrogen
   HS   1.00800 ;     hydrogen bonded to sulfur
    S  32.06000 ;     sulfur
   FE  55.84700 ;     iron
   ZN  65.37000 ;     zinc
   NZ  14.00670 ;     arg NH (NH2)
   NE  14.00670 ;     arg NE (NH)
    P  30.97380 ;     phosphor
   OS  15.99940 ;     sugar or ester oxygen
  CS1  13.01900 ;     sugar CH-group
  NR6  14.00670 ;     aromatic N (6-ring,2 bonds)
 NR6*  14.00670 ;     aromatic N (6-ring,3 bonds)
  CS2  14.02700 ;     sugar CH2-group
   SI  28.08000 ;     silicon
   NA  22.98980 ;     sodium (1+)
   CL  35.45300 ;     chlorine (1-)
```

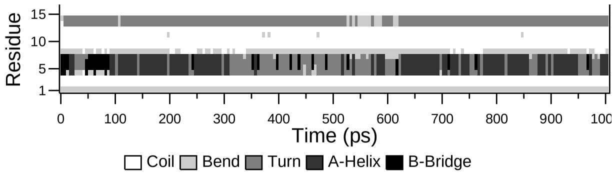


Figure 8.10: Analysis of the secondary structure elements of a peptide in time.

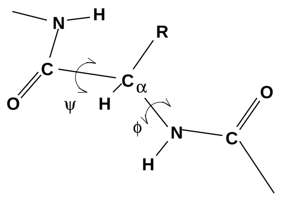


Figure 8.11: Definition of the dihedral angles ϕ and ψ of the protein backbone.

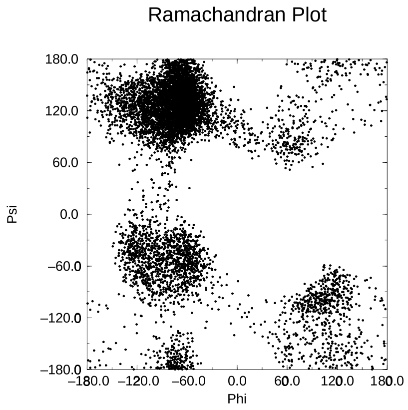


Figure 8.12: Ramachandran plot of a small protein.

- The lifetime of the H-bonds is calculated from the average over all autocorrelation functions of the existence functions (either 0 or 1) of all H-bonds:
$$C(\tau) \;=\; \langle s_i(t)\, s_i(t+\tau)\rangle \tag{8.35}$$
with $s_i(t) = \{0,1\}$ for H-bond i at time t. The integral of $C(\tau)$ gives a rough estimate of the average H-bond lifetime τ_{HB}:
$$\tau_{HB} \;=\; \int_0^{\infty} C(\tau)d\tau \tag{8.36}$$
Both the integral and the complete auto correlation function $C(\tau)$ will be output, so that more sophisticated analysis (e.g. using multi-exponential fits) can be used to get better estimates for τ_{HB}.
- An H-bond existence map can be generated of dimensions *# H-bonds*×*# frames*.
- Index groups are output containing the analyzed groups, all donor-hydrogen atom pairs and acceptor atoms in these groups, donor-hydrogen-acceptor triplets involved in hydrogen bonds between the analyzed groups and all solvent atoms involved in insertion.
- Solvent insertion into H-bonds can be analyzed, see Fig. 8.9. In this case an additional group identifying the solvent must be selected. The occurrence of insertion will be indicated in the existence map. Note that insertion into and existence of a specific H-bond can occur simultaneously and will also be indicated as such in the existence map.

8.11 Protein related items

```
do_dssp
g_rama
xrama
wheel
```

To analyze structural changes of a protein, you can calculate the radius of gyration or the minimum residue distances during time (see sec. 8.7), or calculate the RMSD (sec. 8.8).

You can also look at the changing of *secondary structure elements* during your run. For this you can use the program `do_dssp`, which is an interface for the commercial program `dssp` [72]. For further information, see the `dssp`-manual. A typical output plot of `do_dssp` is given in Fig. 8.10.

One other important analysis of proteins is the so called *Ramachandran plot*. This is the projection of the structure on the two dihedral angles ϕ and ψ of the protein backbone, see Fig. 8.11.

To evaluate this Ramachandran plot you can use the program `g_rama`. A typical output is given in Fig. 8.12.

It is also possible to generate an animation of the Ramachandran plot in time. This can be of help for analyzing certain dihedral transitions in your protein. You can use the program `xrama` for this.

When studying α-helices it is useful to have a *helical wheel* projection of your peptide, to see whether a peptide is amphipatic. This can be done using the `wheel` program. Two examples are plotted in Fig. 8.13.

```
  CA  40.08000 ;     calcium (2+)
  MG  24.30500 ;     magnesium (2+)
   F  18.99840 ;     fluorine (cov. bound)
 CP2  14.02700 ;     aliphatic CH2-group using Ryckaert-Bell.
 CP3  15.03500 ;     aliphatic CH3-group using Ryckaert-Bell.
 CR5  12.01100 ;     aromatic CH-group (5-ring)+H
 CR6  12.01100 ;     aromatic C- bonded to H (6-ring)+H
 HCR   1.00800 ;     H attached to aromatic C (5 or 6 ri
OWT3  15.99940 ;     TIP3P water oxygen
  SD  32.06000 ;     DMSO Sulphur
  OD  15.99940 ;     DMSO Oxygen
  CD  15.03500 ;     DMSO Carbon
```

Atomic detail is used except for hydrogen atoms bound to (aliphatic) carbon atoms, which are treated as *united atoms*. No special hydrogen-bond term is included.

The last 10 atom types are extra atom types with respect to the GROMOS-87 force field [39]:

- F was taken from ref. [43],
- CP2 and CP3 from ref. [40] and references cited therein,
- CR5, CR6 and HCR from ref. [59]
- OWT3 from ref. [42]
- SD, OD and CD from ref. [44]

Therefore, if you use the GROMACS force field as it is, make sure you use the references in your publications as mentioned above.

Note: GROMACS makes use of the atom types as a name, *not* as a number (as e.g. in GROMOS).

5.2.2 Dummy atoms

Some force fields use dummy atoms (virtual sites that are constructed from real atoms) on which certain interactions are located (e.g. on benzene rings, to reproduce the correct quadrupole). This is described in sec. 4.5.

To make dummy atoms in your system, you should include a section `[ dummies? ]` in your topology file, where the '`?`' stands for the number constructing atoms for the dummy atom. This will be '`2`' for type 2, '`3`' for types 3, 3fd, 3fad and 3out and '`4`' for type 4fd (the different types are explained in sec. 4.5).

Parameters for type 2 should look like this:

```
[ dummies2 ]
; Dummy from        funct  a
5       1     2     1      0.7439756
```

for type 3 like this:

```
[ dummies3 ]
; Dummy from               funct   a          b
5       1     2     3      1       0.7439756  0.128012
```

for type 3fd like this:

```
[ dummies3 ]
; Dummy from               funct   a          d
5       1     2     3      2       0.5        -0.105
```

for type 3fad like this:

```
[ dummies3 ]
; Dummy from               funct   theta      d
5       1     2     3      3       120        0.5
```

for type 3out like this:

```
[ dummies3 ]
; Dummy from               funct   a          b          c
5       1     2     3      4       -0.4       -0.4       6.9281
```

for type 4fd like this:

```
[ dummies4 ]
; Dummy from                      funct   a          b          d
5       1     2     3     4       1       0.33333    0.33333    -0.105
```

This will result in the construction of a dummy 'atom', number 5 (first column '`Dummy`'), based on the positions of 1 and 2 or 1, 2 and 3 or 1, 2, 3 and 4 (next two, three or four columns '`from`') following the rules determined by the function number (next column '`funct`') with the parameters specified (last one, two or three columns '`a b . .`').

Note that any bonds defined between dummy atoms and/or normal atoms will be removed by `grompp` after the exclusions have been generated. This way, exclusions will not be affected by an atom being defined as dummy atom or not, but by the bonding configuration of the atom.

5.3 Parameter files

5.3.1 Atoms

A number of *static* properties are assigned to the atom types in the GROMACS force field: Type, Mass, Charge, ϵ and σ (see Table 5.2 The mass is listed in `ff???.atp` (see 5.2.1), whereas the charge is listed in `ff???.rtp` (.rtp = **r**esidue **t**opology **p**arameter file, see 5.5.1). This implies that the charges are only defined in the building blocks of amino acids or user defined building blocks. When generating a topology (`*.top`) using the `pdb2gmx` program the information from these files is combined.

The following *dynamic* quantities are associated with an atom

---

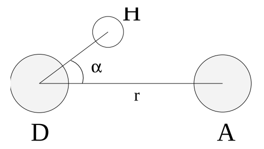


Figure 8.8: Geometrical Hydrogen bond criterion.

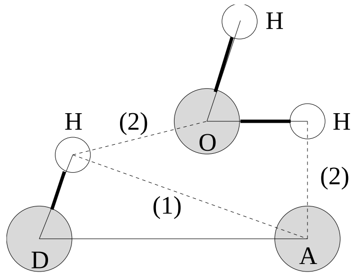


Figure 8.9: Insertion of water into an H-bond. (1) Normal H-bond between two residues. (2) H-bonding bridge via a water molecule.

8.10 Hydrogen bonds

`g_hbond`

The program `g_hbond` analyses the *hydrogen bonds* (H-bonds) between all possible donors D and acceptors A. To determine if an H-bond exists, a geometrical criterion is used, see also Fig. 8.8:

$$\begin{array}{rcrcl} r & \leq & r_{HB} & = & 0.35\text{nm} \\ \alpha & \leq & \alpha_{HB} & = & 60^o \end{array} \tag{8.34}$$

The value of $r_{HB} = 0.35$ nm corresponds to the first minimum of the rdf of SPC-water (see also Fig. 8.3).

The program `g_hbond` analyses all hydrogen bonds existing between two groups of atoms (which must be either identical or non-overlapping) or in specified Donor Hydrogen Acceptor triplets, in the following ways:

- Donor-Acceptor distance (r) distribution of all H-bonds
- Hydrogen-Donor-Acceptor angle (α) distribution of all H-bonds
- The total number of H-bonds in each time frame
- The number of H-bonds in time between residues, divided into groups n-n+i where n and n+i stand for residue numbers and i goes from 0 to 6. The group for $i = 6$ also includes all H-bonds for $i > 6$. These groups include the n-n+3, n-n+4 and n-n+5 H-bonds which provide a measure for the formation of α-helices or β-turns or strands.

be calculated from the eigenvalues λ_i and the eigenvectors, which are the columns of the rotation matrix R. For a symmetric and diagonally-dominant matrix A of size $3N \times 3N$ the square root can be calculated as:

$$A^{\frac{1}{2}} = R\,\mathrm{diag}(\lambda_1^{\frac{1}{2}}, \lambda_2^{\frac{1}{2}}, \ldots, \lambda_{3N}^{\frac{1}{2}})\,R^T \tag{8.27}$$

It can be verified easily that the product of this matrix with itself gives A. Now we can define a difference d between covariance matrices A and B as follows:

$$\begin{aligned} d(A,B) &= \sqrt{\mathrm{tr}\left(\left(A^{\frac{1}{2}} - B^{\frac{1}{2}}\right)^2\right)} & (8.28)\\ &= \sqrt{\mathrm{tr}\left(A + B - 2A^{\frac{1}{2}}B^{\frac{1}{2}}\right)} & (8.29)\\ &= \left(\sum_{i=1}^{N}\left(\lambda_i^A + \lambda_i^B\right) - 2\sum_{i=1}^{N}\sum_{j=1}^{N}\sqrt{\lambda_i^A\lambda_j^B}\left(R_i^A \cdot R_j^B\right)^2\right)^{\frac{1}{2}} & (8.30)\end{aligned}$$

where tr is the trace of a matrix. We can now define the overlap s as:

$$s(A,B) = 1 - \frac{d(A,B)}{\sqrt{\mathrm{tr}A + \mathrm{tr}B}} \tag{8.31}$$

The overlap is 1 if and only if matrices A and B are identical. It is 0 when the sampled subspaces are completely orthogonal.

A commonly used measure is the subspace overlap of the first few eigenvectors of covariance matrices. The overlap of the subspace spanned by m orthonormal vectors $\mathbf{w}_1, \ldots, \mathbf{w}_m$ with a reference subspace spanned by n orthonormal vectors $\mathbf{v}_1, \ldots, \mathbf{v}_n$ can be quantified as follows:

$$\mathrm{overlap}(\mathbf{v},\mathbf{w}) = \frac{1}{n}\sum_{i=1}^{n}\sum_{j=1}^{m}(\mathbf{v}_i \cdot \mathbf{w}_j)^2 \tag{8.32}$$

The overlap will increase with increasing m and will be 1 when set $\mathbf{v}$ is a subspace of set $\mathbf{w}$. The disadvantage of this method is that it does not take the eigenvalues into account. All eigenvectors are weighted equally and when degenerate subspaces are present (equal eigenvalues) the calculated overlap will be too low.

Another useful check is the cosine content. It has been proven the the principal components of random diffusion are cosines with the number of periods equal to half the principal component index[71]. The eigenvalues are proportional to the index to the power -2. The cosine content is defined as:

$$\frac{2}{T}\left(\int_0^T \cos(k\pi t)\,p_i(t)\mathrm{d}t\right)^2 \left(\int_0^T p_i^2(t)\mathrm{d}t\right)^{-1} \tag{8.33}$$

When the cosine content of the first few principal components is close to 1, the largest fluctuations are not connected with the potential, but with random diffusion.

The covariance matrix is built and diagonalized by `g_covar`. The principal components and overlap (any many more things) can be plotted and analyzed with `g_anaeig`. The cosine content can be calculated with `g_analyze`.

Property	Symbol	Unit
Type	-	-
Mass	m	a.m.u.
Charge	q	electron
epsilon	ϵ	kJ/mol
sigma	σ	nm

Table 5.2: Static atom type properties in GROMACS

- Position **x**
- Velocity **v**

These quantities are listed in the coordinate file, `*.gro` (see section File format, 5.6.6).

5.3.2 Bonded parameters

The bonded parameters (i.e. bonds, bond angles, improper and proper dihedrals) are listed in `ff???bon.itp`. The term `func` is 1 for harmonic and 2 for GROMOS-96 bond and angle potentials. For the dihedral, this is explained after this listing.

```
[ bondtypes ]
  ; i    j func        b0          kb
    C    O    1   0.12300     502080.
    C   OM    1   0.12500     418400.
    ......

[ angletypes ]
  ; i    j    k func       th0         cth
   HO   OA    C    1   109.500     397.480
   HO   OA  CH1    1   109.500     397.480
   ......

[ dihedraltypes ]
  ; i    l func        q0          cq
 NR5*  NR5    2     0.000     167.360
 NR5* NR5*    2     0.000     167.360
 ......

[ dihedraltypes ]
  ; j    k func      phi0          cp   mult
    C   OA    1   180.000      16.736      2
    C    N    1   180.000      33.472      2
    ......

[ dihedraltypes ]
;
; Ryckaert-Bellemans Dihedrals
;
```

```
; aj    ak      funct
CP2     CP2     3       9.2789  12.156  -13.120 -3.0597 26.240  -31.495
```

Also in this file are the Ryckaert-Bellemans [60] parameters for the CP2-CP2 dihedrals in alkanes or alkane tails with the following constants:

$$\begin{array}{ccccccccc} & & & & \text{(kJ/mol)} & & & & \\ C_0 & = & 9.28 & C_2 & = & -13.12 & C_4 & = & 26.24 \\ C_1 & = & 12.16 & C_3 & = & -3.06 & C_5 & = & -31.5 \end{array}$$

(**Note:** The use of this potential implies exclusions of LJ-interactions between the first and the last atom of the dihedral, and ψ is defined according to the 'polymer convention' ($\psi_{trans} = 0$)).

So there are three types of dihedrals in the GROMACS force field:

- proper dihedral : funct = 1, with mult = multiplicity, so the number of possible angles
- improper dihedral : funct = 2
- Ryckaert-Bellemans dihedral : funct = 3

In the file `ff???bon.itp` you can add bonded parameters. If you want to include parameters for new atom types, make sure you define this new atom type in `ff???.atp` as well.

5.3.3 Non-bonded parameters

The non-bonded parameters consist of the Van der Waals parameters A and C, as listed in the file `ff???nb.itp`, where `ptype` is the particle type (see Table 5.1):

```
[ atomtypes ]
;name        mass      charge   ptype            c6           c12
    O    15.99940       0.000       A   0.22617E-02   0.74158E-06
   OM    15.99940       0.000       A   0.22617E-02   0.74158E-06
   .....

[ nonbond_params ]
  ; i    j func          c6           c12
    O    O    1 0.22617E-02   0.74158E-06
    O   OA    1 0.22617E-02   0.13807E-05
    .....

[ pairtypes ]
  ; i    j func          cs6          cs12 ; THESE ARE 1-4 INTERACTIONS
    O    O    1 0.22617E-02   0.74158E-06
    O   OM    1 0.22617E-02   0.74158E-06
    .....
```

With A and C being defined as

$$A_{ii} = 4\epsilon_i\sigma_i^{12} \tag{5.1}$$

In stead of comparing the structures to the initial structure at time $t = 0$ (so for example a crystal structure), one can also calculate eqn. 8.20 using a time shift τ:

$$RMSD(t;\tau) \;=\; \left[\frac{1}{N}\sum_{i=1}^{N}\|\mathbf{r}_i(t) - \mathbf{r}_i(t-\tau)\|^2\right]^{\frac{1}{2}} \tag{8.22}$$

so comparing to a least-square structure at $t - \tau$. This gives some insight in the mobility as a function of τ. Use the program `g_run_rms`.

8.9 Covariance analysis

Covariance analysis, also called principal component analysis or essential dynamics [70], can find correlated motions. It uses the covariance matrix C of the atomic coordinates:

$$C_{ij} = \left\langle M_{ii}^{\frac{1}{2}}(x_i - \langle x_i\rangle)M_{jj}^{\frac{1}{2}}(x_j - \langle x_j\rangle)\right\rangle \tag{8.23}$$

where M is a diagonal matrix containing the masses of the atoms (mass-weighted analysis) or the unit matrix (non-mass weighted analysis). C is a symmetric $3N \times 3N$ matrix, which can be diagonalized with an orthonormal transformation matrix R:

$$R^T C R = \text{diag}(\lambda_1, \lambda_2, \ldots, \lambda_{3N}) \quad \text{where} \;\; \lambda_1 \geq \lambda_2 \geq \ldots \geq \lambda_{3N} \tag{8.24}$$

The columns of R are the eigenvectors, also called principal or essential modes. R defines a transformation to a new coordinate system. The trajectory can be projected on the principal modes to give the principal components $p_i(t)$:

$$\mathbf{p}(t) = R^T M^{\frac{1}{2}}(\mathbf{x}(t) - \langle\mathbf{x}\rangle) \tag{8.25}$$

The eigenvalue λ_i is the mean square fluctuation of principal component i. The first few principal modes often describe collective, global motions in the system. The trajectory can be filtered along one (or more) principal modes. For one principal mode i this goes as follows:

$$\mathbf{x}^f(t) = \langle\mathbf{x}\rangle + M^{-\frac{1}{2}}R_{*i}\,p_i(t) \tag{8.26}$$

When the analysis is performed on a macromolecule, one often wants to remove the overall rotation and translation to look at the internal motion only. This can be achieved by least square fitting to a reference structure. Care has to be taken that the reference structure is representative for the ensemble, since the choice of reference structure influences the covariance matrix. One should always check if the principal modes are well defined. If the first principal component resembles a half cosine and the second resembles a full cosine, you might be filtering noise. A good way to check the relevance of the first few principal modes is to calculate the overlap of the sampling between the first and second half of the simulation. Note that this can only be done when the same reference structure is used for the two halves.

The elements of the covariance matrix are proportional to the square of the displacement, so we need to take the square root of the matrix to examine the extent of sampling. The square root can

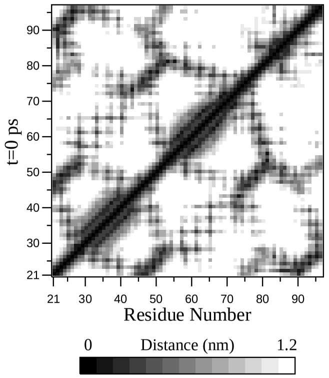


Figure 8.7: A minimum distance matrix for a peptide [3].

8.8 Root mean square deviations in structure

`g_rms`
`g_rmsdist`

The *root mean square deviation* ($RMSD$) of certain atoms in a molecule with respect to a reference structure can be calculated with the program `g_rms` by least-square fitting the structure to the reference structure ($t_2 = 0$) and subsequently calculating the $RMSD$ (eqn. 8.20).

$$RMSD(t_1,t_2) \;=\; \left[\frac{1}{N}\sum_{i=1}^{N}\|\mathbf{r}_i(t_1)-\mathbf{r}_i(t_2)\|^2\right]^{\frac{1}{2}} \tag{8.20}$$

where $\mathbf{r}_i(t)$ is the position of atom i at time t. **NOTE** that fitting does not have to use the same atoms as the calculation of the $RMSD$; e.g.: a protein is usually fitted on the backbone atoms (N,C$_\alpha$,C), but the $RMSD$ can be computed of the backbone or of the whole protein.

Instead of comparing the structures to the initial structure at time $t = 0$ (so for example a crystal structure), one can also calculate eqn. 8.20 with a structure at time $t_2 = t_1 - \tau$. This gives some insight in the mobility as a function of τ. Also a matrix can be made with the $RMSD$ as a function of t_1 and t_2, this gives a nice graphical impression of a trajectory. If there are transitions in a trajectory, they will clearly show up in such a matrix.

Alternatively the $RMSD$ can be computed using a fit-free method with the program `g_rmsdist`:

$$RMSD(t) \;=\; \left[\frac{1}{N^2}\sum_{i=1}^{N}\sum_{j=1}^{N}\|\mathbf{r}_{ij}(t)-\mathbf{r}_{ij}(0)\|^2\right]^{\frac{1}{2}} \tag{8.21}$$

where the *distance* $\mathbf{r}_{ij}$ between atoms at time t is compared with the distance between the same atoms at time 0.

$$C_{ii} = 4\epsilon_i\sigma_i^6 \tag{5.2}$$

and computed according to the combination rules :

$$A_{ij} = (A_{ii}A_{jj})^{\frac{1}{2}} \tag{5.3}$$

$$C_{ij} = (C_{ii}C_{jj})^{\frac{1}{2}} \tag{5.4}$$

It is also possible to use the combination rules where the σ's are averaged:

$$\sigma_{ij} = \frac{1}{2}(\sigma_{ii}+\sigma_{jj}) \tag{5.5}$$

$$\epsilon_{ij} = \sqrt{\epsilon_{ii}\epsilon_{jj}} \tag{5.6}$$

5.3.4 1-4 interactions

The 1-4 interactions for the atom types in `ff???nb.itp` are listed in the `[ pairtypes ]` section. The GROMACS and GROMOS force fields lists all these interactions explicitly, but this section might be empty for force fields like OPLS that calculate the 1-4 interactions by scaling.

5.3.5 Exclusions

The exclusions for bonded particles are generated by `grompp` for neighboring atoms up to a certain number of bonds away, as defined in the `[ moleculetype ]` section in the topology file (see 5.6.1). Particles are considered bonded when they are connected by bonds (`[ bonds ]` types 1 to 5) or constraints (`[ constraints ]` type 1). `[ bonds ]` type 5 can be used to create a connection between two atoms without creating an interaction. There is a harmonic interaction (`[ bonds ]` type 6) which does not connect the atoms by a chemical bond. There is also a second constraint type (`[ constraints ]` type 2) which fixes the distance, but does not connect the atoms by a chemical bond.

Extra exclusions within a molecule can be added manually in a `[ exclusions ]` section. Each line should start with one atom index, followed by one or more atom indices. All non-bonded interactions between the first atom and the other atoms will be excluded.

When all non-bonded interactions within or between groups of atoms need to be excluded, is it more convenient and much more efficient to use energy monitor group exclusions (see sec. 3.3).

5.4 Constraints

Constraints are defined in the `[ constraints ]` section. The format is two atom numbers followed by the function type, which can be 1 or 2 and the constraint distance. The only difference between the two types is that type 1 is used for generating exclusions and while type 2 is not (see 5.3.5). The distances are constrained using the LINCS or the SHAKE algorithm, which can be selected in the `*.mdp` file. Both types of constraints can be perturbed in free-energy calculations by adding a second constraint distance (see 5.6.5). Several types of bonds and angles (see Table 5.4) can be converted automatically to constraints by `grompp`. There are several options for this in the `*.mdp` file.

We have also implemented the SETTLE algorithm [27] which is an analytical solution of SHAKE specifically for water. SETTLE can be selected in the topology file. Check for instance the SPC molecule definition:

```
[ moleculetype ]
; molname       nrexcl
SOL             1

[ atoms ]
; nr    at type res nr  ren nm  at nm   cg nr   charge
1       OW      1       SOL     OW1     1       -0.82
2       HW      1       SOL     HW2     1        0.41
3       HW      1       SOL     HW3     1        0.41

[ settles ]
; OW    funct   doh     dhh
1       1       0.1     0.16333

[ exclusions ]
1       2       3
2       1       3
3       1       2
```

The section `[ settles ]` defines the first atom of the watery molecule, the settle funct is always one, and the distance between O and H, and distance between both H atoms must be given. Note that the algorithm can also be used for TIP3P and TIP4P [42]. TIP3P just has another geometry. TIP4P has a dummy atom, but since that is generated it does not need to be shaken (nor stirred).

5.5 Databases

5.5.1 Residue database

The file holding the residue database is `ff???.rtp`. Originally this file contained building blocks (amino acids) for proteins, and is the GROMACS interpretation of the `rt37c4.dat` file of GROMOS. So the residue file contains information (bonds, charge, charge groups and improper dihedrals) for a frequently used building block. It is better *not* to change this file because it is standard input for `pdb2gmx`, but if changes are needed make them in the `*.top` file (see section Topology file, 5.6.1). However, in the `ff???.rtp` file the user can define a new building block or molecule: see for example 2,2,2-trifluoroethanol (TFE) or *n*-decane (C10). But when defining new molecules (non-protein) it is preferable to create a `*.itp` file. This will be discussed in a next section (section 5.6.2).

The file `ff???.rtp` is only used by `pdb2gmx`. As mentioned before, the only extra information this program needs from `ff???.rtp` is bonds, charges of atoms, charge groups and improper

---

For planes it uses the normal vector perpendicular to the plane. It can also calculate the *distance* d between the geometrical center of two planes (see Fig. 8.6D), and the distances d_1 and d_2 between 2 atoms (of a vector) and the center of a plane defined by 3 atoms (see Fig. 8.6D). It further calculates the distance d between the center of the plane and the middle of this vector. Depending on the input groups (i.e. groups of 2 or 3 atom numbers), the program decides what angles and distances to calculate. For example, the index-file could look like this:

```
[ a_plane ]
 1     2     3
[ a_vector ]
 3     4     5
```

8.7 Radius of gyration and distances

```
g_gyrate
g_sgangle
g_mindist
g_mdmat
xpm2ps
```

To have a rough measure for the compactness of a structure, you can calculate the *radius of gyration* with the program `g_gyrate` as follows:

$$R_g = \left(\frac{\sum_i \|\mathbf{r}_i\|^2 m_i}{\sum_i m_i}\right)^{\frac{1}{2}} \tag{8.19}$$

where m_i is the mass of atom i and $\mathbf{r}_i$ the position of atom i with respect to the center of mass of the molecule. It is especially useful to characterize polymer solutions and proteins.

Sometimes it is interesting to plot the *distance* between two atoms, or the *minimum* distance between two groups of atoms (e.g.: protein side-chains in a salt bridge). To calculate these distances between certain groups there are several possibilities:

- The *distance between the geometrical centers* of two groups can be calculated with the program `g_sgangle`, as explained in sec. 8.6.
- The *minimum distance* between two groups of atoms during time can be calculated with the program `g_mindist`. It also calculates the *number of contacts* between these groups within a certain radius r_{max}.
- To monitor the *minimum distances between residues* (see chapter 5) within a (protein) molecule, you can use the program `g_mdmat`. This minimum distance between two residues A_i and A_j is defined as the smallest distance between any pair of atoms ($i \in A_i$, $j \in A_j$). The output is a symmetrical matrix of smallest distances between all residues. To visualize this matrix, you can use a program such as `xv`. If you want to view the axes and legend or if you want to print the matrix, you can convert it with `xpm2ps` into a Postscript picture, see Fig. 8.7.

  Plotting these matrices for different time-frames, one can analyze changes in the structure, and e.g. forming of salt bridges.

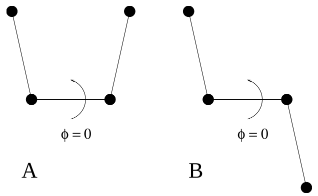


Figure 8.5: Dihedral conventions: A. “Biochemical convention”. B. “Polymer convention”.

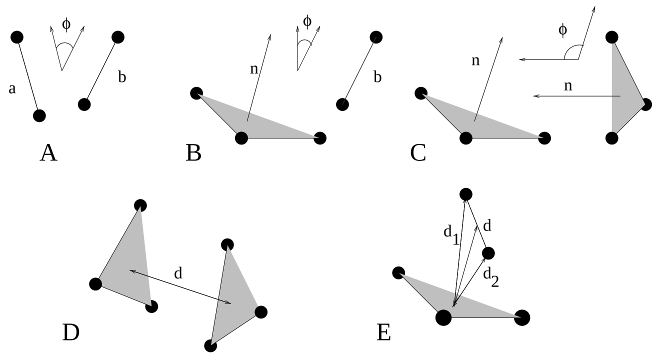


Figure 8.6: Options of `g_sgangle`: A. Angle between 2 vectors. B. Angle between a vector and the normal of a plane. C. Angle between two planes. D. Distance between the geometrical centers of 2 planes. E. Distances between a vector and the center of a plane.

---

dihedrals, because the rest is read from the coordinate input file (in the case of `pdb2gmx`, a pdb format file). Some proteins contain residues that are not standard, but are listed in the coordinate file. You have to construct a building block for this “strange” residue, otherwise you will not obtain a `*.top` file. This also holds for molecules in the coordinate file like phosphate or sulphate ions. The residue database is constructed in the following way:

```
[ bondedtypes ]  ; mandatory
; bonds  angles  dihedrals  impropers
     1       1          1          2  ; mandatory

[ GLY ]  ; mandatory

 [ atoms ]  ; mandatory
; name  type  charge  chargegroup
     N     N  -0.280     0
     H     H   0.280     0
    CA   CH2   0.000     1
     C     C   0.380     2
     O     O  -0.380     2

 [ bonds ]  ; optional
;atom1 atom2      b0      kb
     N     H
     N    CA
    CA     C
     C     O
    -C     N

 [ exclusions ]  ; optional
;atom1 atom2

 [ angles ]  ; optional
;atom1 atom2 atom3    th0    cth

 [ dihedrals ]  ; optional
;atom1 atom2 atom3 atom4   phi0     cp   mult

 [ impropers ]  ; optional
;atom1 atom2 atom3 atom4     q0     cq
     N    -C    CA     H
    -C   -CA     N    -O


[ ZN ]
 [ atoms ]
    ZN    ZN   2.000     0
```

The file is free format, the only restriction is that there can be at most one entry on a line. The first field in the file is the `[ bondedtypes ]` field, which is followed by four numbers, that indicate the interaction type for bonds, angles, dihedrals and improper dihedrals. The file contains residue entries, which consist of atoms and optionally bonds, angles dihedrals and impropers. The charge group codes denote the charge group numbers. Atoms in the same charge group should always be below each other. When using the hydrogen database with `pdb2gmx` for adding missing hydrogens, the atom names defined in the `.rtp` entry should correspond exactly to the naming convention used in the hydrogen database, see 5.5.2. The atom names in the bonded interaction can be preceded by a minus or a plus, indicating that the atom is in the preceding or following residue respectively. Parameters can be added to bonds, angles, dihedrals and impropers, these parameters override the standard parameters in the `.itp` files. This should only be used in special cases. Instead of parameters, a string can be added for each bonded interaction, this is used in GROMOS96 `.rtp` files. These strings are copied to the topology file and can be replaced by force field parameters by the C-preprocessor in `grompp` using `#define` statements.

`pdb2gmx` automatically generates all angles, this means that for the GROMACS force field the `[ angles ]` field is only useful for overriding `.itp` parameters. For the GROMOS-96 force field the interaction number off all angles need to be specified.

`pdb2gmx` automatically generates one proper dihedral for every rotatable bond, preferably on heavy atoms. When the `[ dihedrals ]` field is used, no other dihedrals will be generated for the bonds corresponding to the specified dihedrals. It is possible to put more than one dihedral on a rotatable bond.

`pdb2gmx` sets the number exclusions to 3, which means that interactions between atoms connected by at most 3 bonds are excluded. Pair interactions are generated for all pairs of atoms which are seperated by 3 bonds (except pairs of hydrogens). When more interactions need to be excluded, or some pair interactions should not be generated, an `[ exclusions ]` field can be added, followed by pairs of atom names on seperate lines. All non-bonded and pair interactions between these atoms will be excluded.

5.5.2 Hydrogen database

The hydrogen database is stored in `ff???.hdb`. It contains information for the `pdb2gmx` program on how to connect hydrogen atoms to existing atoms. Hydrogen atoms are named after the atom they are connected to: the first letter of the atom name is replaced by an 'H'. If more then one hydrogen atom is connected to the same atom, a number will be added to the end of the hydrogen atom name. For example, adding two hydrogen atoms to `ND2` (in asparagine), the hydrogen atoms will be named `HD21` and `HD22`. This is important since atom naming in the `.rtp` file (see 5.5.1) must be the same. The format of the hydrogen database is as follows:

```
; res   # additions
        # H add type    i       j       k
ALA     1
        1       1       N       -C      CA
ARG     4
        1       2       N       CA      C
        1       1       NE      CD      CZ
        2       3       NH1     CZ      NE
        2       3       NH2     CZ      NE
```

On the first line we see the residue name (ALA or ARG) and the number of additions. After that follows one line for each addition, on which we see:

- The number of H atoms added

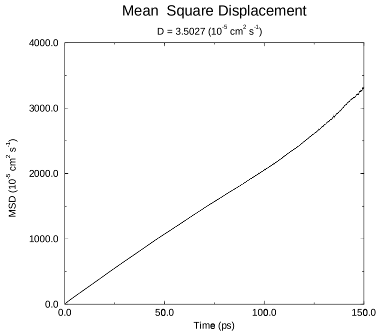


Figure 8.4: Mean Square Displacement of SPC-water.

To monitor specific *bonds* in your molecules during time, the program `g_bond` calculates the distribution of the bond length in time. The index file consists of pairs of atom numbers, for example

```
[ bonds_1 ]
 1     2
 3     4
 9    10
[ bonds_2 ]
12    13
```

The program `g_angle` calculates the distribution of *angles* and *dihedrals* in time. It also gives the average angle or dihedral. The index file consists of triplets or quadruples of atom numbers:

```
[ angles ]
 1     2     3
 2     3     4
 3     4     5
[ dihedrals ]
 1     2     3     4
 2     3     5     5
```

For the dihedral angles you can use either the "biochemical convention" ($\phi = 0 \equiv cis$) or "polymer convention" ($\phi = 0 \equiv trans$), see Fig. 8.5.

To follow specific *angles* in time between two vectors, a vector and a plane or two planes (defined by 2, resp. 3 atoms inside your molecule, see Fig. 8.6A, B, C), use the program `g_sgangle`.

The self diffusion coefficient can be calculated using the Green-Kubo relation [65]

$$D_A \; = \; \frac{1}{3} \int_0^\infty \langle \mathbf{v}_i(t) \cdot \mathbf{v}_i(0) \rangle_{i \in A} \, dt \tag{8.16}$$

which is just the integral of the velocity autocorrelation function. There is a widely held belief that the velocity ACF converges faster than the mean square displacement (sec. 8.5.5), which can also be used for the computation of diffusion constants. However, Allen & Tildesly [65] warn us that the long time contribution to the velocity ACF can not be ignored, so care must be taken.

Another important quantity is the dipole correlation time. The *dipole correlation function* for particles A is calculated as follows by `g_dipoles`:

$$C_\mu(\tau) \; = \; \langle \mu_i(\tau) \cdot \mu_i(0) \rangle_{i \in A} \tag{8.17}$$

with $\mu_i = \sum_{j \in i} \mathbf{r}_j q_j$. The dipole correlation time can be computed using eqn. 8.8. For some applications see [67].

The viscosity of a liquid can be related to the correlation time of the Pressure tensor $\boldsymbol{P}$ [68, 69]. `g_energy` can compute the viscosity, but in our experience this is not very accurate (actually the values do not converge...).

8.5.5 Mean Square Displacement

To determine the self diffusion coefficient D_A of particles A one can use the Einstein relation [65]

$$\lim_{t \to \infty} \langle \| \mathbf{r}_i(t) - \mathbf{r}_i(0) \|^2 \rangle_{i \in A} \; = \; 6 D_A t \tag{8.18}$$

This *Mean Square Displacement* and D_A are calculated by the program `g_msd`. For molecules consisting of more than one atom, $\mathbf{r}_i$ is the center of mass positions. In that case you should use an index file with molecule numbers! The program can also be used for calculating diffusion in one or two dimensions. This is useful for studying lateral diffusion on interfaces.

An example of the mean square displacement of SPC-water is given in Fig. 8.4.

8.6 Bonds, angles and dihedrals

```
g_bond
g_angle
g_sgangle
```

- The way of adding H atoms, can be any of

  1 *one planar hydrogen, e.g. rings or peptide bond*
  one hydrogen atom (n) is generated, lying in the plane of atoms (i,j,k) on the line bisecting angle (j-i-k) at a distance of 0.1 nm from atom i, such that the angles (n-i-j) and (n-i-k) are $>$ 90 degrees

  2 *one single hydrogen, e.g. hydroxyl*
  one hydrogen atom (n) is generated at a distance of 0.1 nm from atom i, such that angle (n-i-j)=109.5 degrees and dihedral (n-i-j-k)=trans

  3 *two planar hydrogens, e.g. -*NH_2
  two hydrogens (n1,n2) are generated at a distance of 0.1 nm from atom i, such that angle (n1-i-j)=(n2-i-j)=120 degrees and dihedral (n1-i-j-k)=cis and (n2-i-j-k)=trans, such that names are according to IUPAC standards [61]

  4 *two or three tetrahedral hydrogens, e.g. -*CH_3
  three (n1,n2,n3) or two (n1,n2) hydrogens are generated at a distance of 0.1 nm from atom i, such that angle (n1-i-j)=(n2-i-j)=(n3-i-j)=109.5, dihedral (n1-i-j-k)=trans, (n2-i-j-k)=trans+120 and (n3-i-j-k)=trans+240 degrees

  5 *one tetrahedral hydrogen, e.g.* C_3CH
  one hydrogen atom (n1) is generated at a distance of 0.1 nm from atom i in tetrahedral conformation such that angle (n1-i-j)=(n1-i-k)=(n1-i-l)=109.5 degrees

  6 *two tetrahedral hydrogens, e.g.* C-CH_2-C
  two hydrogen atoms (n1,n2) are generated at a distance of 0.1 nm from atom i in tetrahedral conformation on the plane bissecting angle i-j-k with angle (n1-i-n2)=(n1-i-j)=(n1-i-k)=109.5

  7 *two water hydrogens*
  two hydrogens are generated around atom i according to SPC [48] water geometry. The symmetry axis will alternate between three coordinate axes in both directions

- Three or four control atoms (i,j,k,l), where the first always is the atom to which the H atoms are connected. The other two or three depend on the code selected.

5.5.3 Termini database

The termini databases are stored in `ff???-n.tdb` and `ff???-c.tdb` for the N- and C-termini respectively. They contain information for the `pdb2gmx` program on how to connect new atoms to existing ones, which atoms should be removed or changed and which bonded interactions should be added. The format of the is as follows (this is an example from the `ffgmx-c.tdb`):

```
[ None ]

[ COO- ]
[ replace ]
C       C       C       12.011  0.27
[ add ]
2       8       C       CA      N
        O       OM      15.9994 -0.635
```

```
[ delete ]
O
[ impropers ]
C       O1      O2      CA
```

The file is organized in blocks, each with a header specifying the name of the block. These blocks correspond to different types of termini that can be added to a molecule. In this example `[ None ]` is the first block, corresponding to a terminus that leaves the molecule as it is; `[ COO- ]` is the second terminus type, corresponding to changing the terminal carbon atom into a deprotonated carboxyl group. Block names cannot be any of the following: `replace`, `add`, `delete`, `bonds`, `angles`, `dihedrals`, `impropers`; this would interfere with the parameters of the block, and would probably also be very confusing to human readers.

Per block the following options are present:

- `[ replace ]`
  replace an existing atom by one with a different atom type, atom name, charge and/or mass. For each atom to be replaced on line should be entered with the following fields:
  - name of the atom to be replaced
  - new atom name
  - new atom type
  - new mass
  - new charge
- `[ add ]`
  add new atoms. For each (group of) added atom(s), a two-line entry is necessary. The first line contains the same fields as an entry in the hydrogen database (number of atoms, type of addition, control atoms, see 5.5.1), but the possible types of addition are extended by two more, specifically for C-terminal additions:

  8 *two carboxyl oxygens, -COO*$^-$
  two oxygens (n1,n2) are generated according to rule 3, at a distance of 0.136 nm from atom i and an angle (n1-i-j)=(n2-i-j)=117 degrees

  9 *carboxyl oxygens and hydrogen, -COOH*
  two oxygens (n1,n2) are generated according to rule 3, at distances of 0.123 nm and 0.125 nm from atom i for n1 and n2 resp. and angles (n1-i-j)=121 and (n2-i-j)=115 degrees. One hydrogen (n') is generated around n2 according to rule 2, where n-i-j and n-i-j-k should be read as n'-n2-i and n'-n2-i-j resp.

  After this line another line follows which specifies the details of the added atom(s), in the same way as for replacing atoms, i.e.:
  - atom name
  - atom type
  - mass
  - charge

---

intervals ($j << M$), but it makes it easier to interpret the results. Another aspect that may not be neglected when computing ACFs from simulation, is that usually the time origins ξ (eqn. 8.6) are not statistically independent, which may introduce a bias in the results. This can be tested using a block-averaging procedure, where only time origins with a spacing at least the length of the time lag are included, e.g. using k time origins with spacing of $M\Delta t$ (where $kM \leq N$):

$$C_f(j\Delta t) \;=\; \frac{1}{k}\sum_{i=0}^{k-1} f(iM\Delta t)f((iM+j)\Delta t) \tag{8.11}$$

However, one needs very long simulations to get good accuracy this way, because there are many fewer points that contribute to the ACF.

8.5.2 Using FFT for computation of the ACF

The computational cost for calculating an ACF according to eqn. 8.9 is proportional to N^2, which is considerable. However, this can be improved by using fast Fourier transforms to do the convolution [65].

8.5.3 Special forms of the ACF

There are some important varieties on the ACF, e.g. the ACF of a vector $\boldsymbol{p}$:

$$C_{\boldsymbol{p}}(t) \;=\; \int_0^\infty P_n(\cos\angle\left(\boldsymbol{p}(t),\boldsymbol{p}(t+\xi)\right)\mathrm{d}\xi \tag{8.12}$$

where $P_n(x)$ is the n^{th} order Legendre polynomial [1]. Such correlation times can actually be obtained experimentally using e.g. NMR or other relaxation experiments. GROMACS can compute correlations using the 1st and 2nd order Legendre polynomial (eqn. 8.12). This can a.o. be used for rotational autocorrelation (`g_rotacf`), dipole autocorrelation (`g_dipoles`).

In order to study torsion angle dynamics we define a dihedral autocorrelation function as [66]:

$$C(t) \;=\; \left\langle \cos(\theta(\tau)-\theta(\tau+t))\right\rangle_\tau \tag{8.13}$$

Note that this is not a product of two functions as is generally used for correlation functions, but it may be rewritten as the sum of two products:

$$C(t) \;=\; \left\langle \cos(\theta(\tau))\cos(\theta(\tau+t))\,+\,\sin(\theta(\tau))\sin(\theta(\tau+t))\right\rangle_\tau \tag{8.14}$$

8.5.4 Some Applications

The program `g_velacc` calculates this *Velocity Auto Correlation Function*.

$$C_{\boldsymbol{v}}(\tau) \;=\; \langle \boldsymbol{v}_i(\tau)\cdot\boldsymbol{v}_i(0)\rangle_{i\in A} \tag{8.15}$$

---

[1] $P_0(x)=1$, $P_1(x)=x$, $P_2(x)=(3x^2-1)/2$

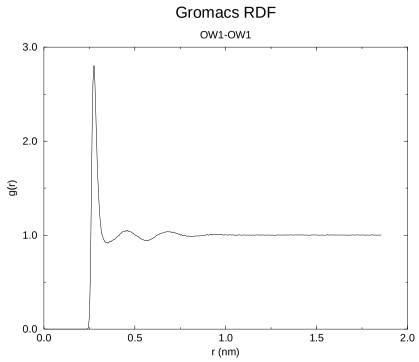


Figure 8.3: $g_{OO}(r)$ for Oxygen-Oxygen of SPC-water.

where the notation on the right hand side means averaging over ξ, i.e. over time origins. It is also possible to compute cross-correlation function from two properties $f(t)$ and $g(t)$:

$$C_{fg}(t) \;=\; \langle f(\xi) g(\xi + t) \rangle_{\xi} \tag{8.7}$$

however, in GROMACS there is no standard mechanism to do this (**note:** you can use the `xmgr` program to compute cross correlations). The integral of the correlation function over time is the correlation time τ_f:

$$\tau_f \;=\; \int_0^{\infty} C_f(t) \mathrm{d}t \tag{8.8}$$

In practice correlation functions are calculated based on data points with discrete time intervals Δt, so that the ACF from an MD simulation is:

$$C_f(j\Delta t) \;=\; \frac{1}{N-j} \sum_{i=0}^{N-1-j} f(i\Delta t) f((i+j)\Delta t) \tag{8.9}$$

where N is the number of available time frames for the calculation. The resulting ACF is obviously only available at time points with the same interval Δt. Since for many applications it is necessary to know the short time behavior of the ACF (e.g. the first 10 ps) this often means that we have to save the atomic coordinates with short intervals. Another implication of eqn. 8.9 is that in principle we can not compute all points of the ACF with the same accuracy, since we have $N-1$ data points for $C_f(\Delta t)$ but only 1 for $C_f((N-1)\Delta t)$. However, if we decide to compute only an ACF of length $M\Delta t$, where $M \leq N/2$ we can compute all points with the same statistical accuracy:

$$C_f(j\Delta t) \;=\; \frac{1}{M} \sum_{i=0}^{N-1-M} f(i\Delta t) f((i+j)\Delta t) \tag{8.10}$$

here of course $j < M$. M is sometimes referred to as the time lag of the correlation function. When we decide to do this, we intentionally do not use all the available points for very short time

---

Like in the hydrogen database (see 5.5.1), when more then one atom is connected to an existing one, a number will be appended to the end of the atom name.

- `[ delete ]`
  delete existing atoms. One atom name per line.
- `[ bonds ]`, `[ angles ]`, `[ dihedrals ]` and `[ impropers ]`
  add additional bonded parameters. The format is identical to that used in the `ff???.rtp`, see 5.5.1.

5.6 File formats

5.6.1 Topology file

The topology file is built following the GROMACS specification for a molecular topology. A `*.top` file can be generated by `pdb2gmx`.

Description of the file layout:

- semicolon (;) and newline surround comments
- on a line ending with \ the newline character is ignored.
- directives are surrounded by `[` and `]`
- the topology consists of three levels:
  - the parameter level (see Table 5.3)
  - the molecule level, which should contain one or more molecule definitions (see Table 5.4)
  - the system level: `[ system ]`, `[ molecules ]`
- items should be separated by spaces or tabs, not commas
- atoms in molecules should be numbered consecutively starting at 1
- the file is parsed once only which implies that no forward references can be treated: items must be defined before they can be used
- exclusions can be generated from the bonds or overridden manually
- the bonded force types can be generated from the atom types or overridden per bond
- descriptive comment lines and empty lines are highly recommended
- using one of the `[ atoms ]`, `[ bonds ]`, `[ pairs ]`, `[ angles ]`, etc. without having used `[ moleculetype ]` before is meaningless and generates a warning.
- using `[ molecules ]` without having used `[ system ]` before is meaningless and generates a warning.

Parameters

interaction type	directive	# at.	f. tp	parameters	pert
mandatory	`defaults`			non-bonded function type; combination rule; generate pairs (no/yes); fudge LJ (); fudge QQ ()	
mandatory	`atomtypes`			atom type; m (u); q (e); particle type; c_6 (kJ mol^{-1}nm^6); c_{12} (kJ mol^{-1}nm^{12})	
	`bondtypes`	(see Table 5.4, directive `bonds`)			
	`constrainttypes`	(see Table 5.4, directive `constraints`)			
	`pairtypes`	(see Table 5.4, directive `pairs`)			
	`angletypes`	(see Table 5.4, directive `angles`)			
proper dih.	`dihedraltypes`	$2^{(b)}$	1	θ_{max} (deg); f_c (kJ mol^{-1}); mult	X$^{(a)}$
improper dih.	`dihedraltypes`	$2^{(c)}$	2	θ_0 (deg); f_c (kJ mol^{-1}rad^{-2})	X
RB dihedral	`dihedraltypes`	$2^{(b)}$	3	C_0, C_1, C_2, C_3, C_4, C_5 (kJ mol^{-1})	
LJ	`nonbond_params`	2	1	c_6 (kJ mol^{-1}nm^6); c_{12} (kJ mol^{-1}nm^{12})	
Buckingham	`nonbond_params`	2	2	a (kJ mol^{-1}); b (nm^{-1}); c_6 (kJ mol^{-1}nm^6)	

Molecule definition(s)

one or more molecule definitions as described in Table 5.4 (next page)

System

mandatory	`system`	system name
mandatory	`molecules`	molecule name; number of molecules

'# at' is the number of atom types
'f. tp' is function type
'pert' indicates if this interaction type can be perturbed during free energy calculations
$^{(a)}$ multiplicities can not be perturbed
$^{(b)}$ the inner two atoms in the dihedral
$^{(c)}$ the outer two atoms in the dihedral
For free energy calculations, the parameters for topology 'B' (lambda = 1) should be added on the same line, after the normal parameters, in the same order as the normal parameters.

Table 5.3: The topology (`*.top`) file.

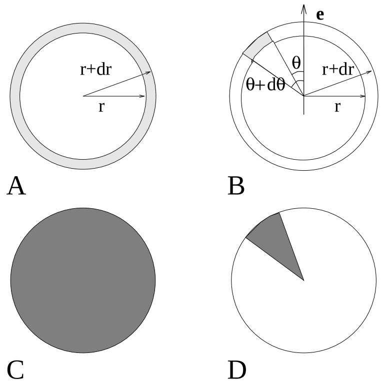


Figure 8.2: Definition of slices in `g_rdf`: A. $g_{AB}(r)$. B. $g_{AB}(r,\theta)$. The slices are colored grey. C. Normalization $\langle\rho_B\rangle_{local}$. D. Normalization $\langle\rho_B\rangle_{local,\,\theta}$. Normalization volumes are colored grey.

Usually the value of r_{max} is half of the box length. The averaging is also performed in time. In practice the analysis program `g_rdf` divides the system into spherical slices (from r to $r+dr$, see Fig. 8.2A) and makes a histogram in stead of the δ-function. An example of the rdf of Oxygen-Oxygen in SPC-water [48] is given in Fig. 8.3.

With `g_rdf` it is also possible to calculate an angle dependent rdf $g_{AB}(r,\theta)$, where the angle θ is defined with respect to a certain laboratory axis $\mathbf{e}$, see Fig. 8.2B.

$$g_{AB}(r,\theta) = \frac{1}{\langle\rho_B\rangle_{local,\,\theta}}\frac{1}{N_A}\sum_{i\in A}^{N_A}\sum_{j\in B}^{N_B}\frac{\delta(r_{ij}-r)\delta(\theta_{ij}-\theta)}{2\pi r^2 sin(\theta)} \tag{8.4}$$

$$cos(\theta_{ij}) = \frac{\mathbf{r}_{ij}\cdot\mathbf{e}}{\|r_{ij}\|\;\|e\|} \tag{8.5}$$

This $g_{AB}(r,\theta)$ is useful for analyzing anisotropic systems. Note that in this case the normalization $\langle\rho_B\rangle_{local,\,\theta}$ is the average density in all angle slices from θ to $\theta+d\theta$ up to r_{max}, so angle dependent, see Fig. 8.2D.

8.5 Correlation functions

8.5.1 Theory of correlation functions

The theory of correlation functions is well established [65]. However we want to describe here the implementation of the various correlation function flavors in the GROMACS code. The definition of the autocorrelation function (ACF) $C_f(t)$ for a property $f(t)$ is

$$C_f(t) = \langle f(\xi)f(\xi+t)\rangle_\xi \tag{8.6}$$

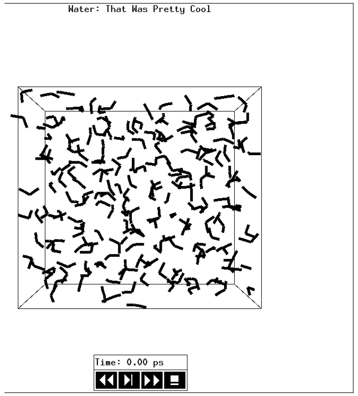


Figure 8.1: The window of `ngmx` showing a box of water.

a set of energies, like potential, kinetic or total energy, or individual contributions, like Lennard-Jones or dihedral energies.

The *center-of-mass velocity*, defined as

$$\mathbf{v}_{com} = \frac{1}{M}\sum_{i=1}^{N} m_i \mathbf{v}_i \tag{8.2}$$

with $M = \sum_{i=1}^{N} m_i$ the total mass of the system, can be monitored in time by the program `g_com`. It is however recommended to remove the center-of-mass velocity every step (see chapter 3)!

8.4 Radial distribution functions

`g_rdf`

The *radial distribution function* (rdf) or pair correlation function $g_{AB}(r)$ between particles of type A and B is defined in the following way:

$$\begin{aligned} g_{AB}(r) &= \frac{\langle \rho_B(r) \rangle}{\langle \rho_B \rangle_{local}} \\ &= \frac{1}{\langle \rho_B \rangle_{local}} \frac{1}{N_A} \sum_{i \in A}^{N_A} \sum_{j \in B}^{N_B} \frac{\delta(r_{ij} - r)}{4\pi r^2} \end{aligned} \tag{8.3}$$

with $\langle \rho_B(r) \rangle$ the particle density of type B at a distance r around particles A, and $\langle \rho_B \rangle_{local}$ the particle density of type B averaged over all spheres around particles A with radius r_{max} (see Fig. 8.2C).

Molecule definition

interaction type	directive	# at.	f. tp	parameters	pert
mandatory	`moleculetype`			molecule name; exclude neighbors # bonds away for non-bonded interactions	
mandatory	`atoms`	1		atom type; residue number; residue name; atom name; charge group number; q (e); m (u)	X$^{(b)}$
bond	`bonds`$^{(c,d)}$	2	1	b_0 (nm); f_c (kJ mol^{-1}nm^{-2})	X
G96 bond	`bonds`$^{(c,d)}$	2	2	b_0 (nm); f_c (kJ mol^{-1}nm^{-4})	X
morse	`bonds`$^{(c,d)}$	2	3	b_0 (nm); D (kJ mol^{-1}); β (nm^{-1})	
cubic bond	`bonds`$^{(c,d)}$	2	4	b_0 (nm); C_2 (kJ mol^{-1}nm^{-2}); C_3 (kJ mol^{-1}nm^{-3})	
connection	`bonds`$^{(c)}$	2	5		
harmonic pot.	`bonds`	2	6	b_0 (nm); f_c (kJ mol^{-1}nm^{-2})	X
LJ 1-4	`pairs`	2	1	c_6 (kJ mol^{-1}nm^{6}); c_{12} (kJ mol^{-1}nm^{12})	X
angle	`angles`$^{(d)}$	3	1	θ_0 (deg); f_c (kJ mol^{-1}rad^{-2})	X
G96 angle	`angles`$^{(d)}$	3	2	θ_0 (deg); f_c (kJ mol^{-1})	X
proper dih.	`dihedrals`	4	1	θ_{max} (deg); f_c (kJ mol^{-1}); mult	X$^{(a)}$
improper dih.	`dihedrals`	4	2	θ_0 (deg); f_c (kJ mol^{-1}rad^{-2})	X
RB dihedral	`dihedrals`	4	3	C_0, C_1, C_2, C_3, C_4, C_5 (kJ mol^{-1})	
constraint	`constraints`	2	1	b_0 (nm)	X
constr. n.c.	`constraints`	2	2	b_0 (nm)	X
settle	`settles`	3	1	d_{OH}, d_{HH} (nm)	
dummy2	`dummies2`	3	1	a ()	
dummy3	`dummies3`	4	1	a, b ()	
dummy3fd	`dummies3`	4	2	a (); d (nm)	
dummy3fad	`dummies3`	4	3	θ (deg); d (nm)	
dummy3out	`dummies3`	4	4	a, b (); c (nm^{-1})	
dummy4fd	`dummies4`	5	1	a, b (); d (nm);	
position res.	`position_restraints`	1	1	k_x, k_y, k_z (kJ mol^{-1}nm^{-2})	
distance res.	`distance_restraints`	2	1	type; index; low, up_1, up_2 (nm); factor ()	
angle res.	`angle_restraints`	4	1	θ_0 (deg); f_c (kJ mol^{-1}); mult	X$^{(a)}$
angle res. z	`angle_restraints_z`	2	1	θ_0 (deg); f_c (kJ mol^{-1}); mult	X$^{(a)}$
exclusions	`exclusions`	1		one or more atom indices	

'# at' is the number of atom indices
'f. tp' is function type
'pert' indicates if this interaction type can be perturbed during free energy calculations
(a) multiplicities can not be perturbed
(b) only the atom type, charge and mass can be perturbed
(c) used by `grompp` for generating exclusions
(d) can be converted to constraints by `grompp`
For free energy calculations, the parameters for topology 'B' (lambda = 1) should be added on the same line, after the normal parameters, in the same order as the normal parameters.

Table 5.4: The molecule definition.

- after `[ system ]` the only allowed directive is `[ molecules ]`
- using an unknown string in `[ ]` causes all the data until the next directive to be ignored, and generates a warning.

Here is an example of a topology file, `urea.top`:

```
;
;       Example topology file
;
; The force field files to be included
#include "ffgmx.itp"

[ moleculetype ]
; name  nrexcl
Urea         3

[ atoms ]
;   nr    type   resnr  residu    atom    cgnr  charge
     1       C       1    UREA      C1       1   0.683
     2       O       1    UREA      O2       1  -0.683
     3      NT       1    UREA      N3       2  -0.622
     4       H       1    UREA      H4       2   0.346
     5       H       1    UREA      H5       2   0.276
     6      NT       1    UREA      N6       3  -0.622
     7       H       1    UREA      H7       3   0.346
     8       H       1    UREA      H8       3   0.276

[ bonds ]
;  ai    aj funct           b0           kb
    3     4     1 1.000000e-01 3.744680e+05
    3     5     1 1.000000e-01 3.744680e+05
    6     7     1 1.000000e-01 3.744680e+05
    6     8     1 1.000000e-01 3.744680e+05
    1     2     1 1.230000e-01 5.020800e+05
    1     3     1 1.330000e-01 3.765600e+05
    1     6     1 1.330000e-01 3.765600e+05

[ pairs ]
;  ai    aj funct           c6          c12
    2     4     1 0.000000e+00 0.000000e+00
    2     5     1 0.000000e+00 0.000000e+00
    2     7     1 0.000000e+00 0.000000e+00
    2     8     1 0.000000e+00 0.000000e+00
    3     7     1 0.000000e+00 0.000000e+00
    3     8     1 0.000000e+00 0.000000e+00
    4     6     1 0.000000e+00 0.000000e+00
    5     6     1 0.000000e+00 0.000000e+00

[ angles ]
;  ai    aj    ak funct          th0          cth
    1     3     4     1 1.200000e+02 2.928800e+02
```

`SideChain`
protein side chain atoms; that is all atoms except N, C_α, C, O, backbone amide hydrogen, oxygens in C-terminus and hydrogens on the N-terminus

`SideChain-H`
protein side chain atoms excluding all hydrogens

`Prot-Masses`
protein atoms excluding dummy masses (as used in dummy atom constructions of NH_3 groups and Tryptophane sidechains), see also sec. 5.2.2; this group is only included when it differs from the ’`Protein`’ group

`Non-Protein`
all non-protein atoms

`DNA`
all DNA atoms

`molecule_name`
for all residues/molecules which are not recognized as protein or DNA, one group per residue/molecule name is generated

`Other`
all atoms which are neither protein nor DNA.

Empty groups will not be generated.

8.2 Looking at your trajectory

`ngmx`

Before analyzing your trajectory it is often informative to look at your trajectory first. Gromacs comes with a simple trajectory viewer `ngmx`; the advantage with this one is that it does not require OpenGL, which usually isn’t present e.g. on supercomputers. It is also possible to generate a hardcopy in Encapsulated Postscript format, see Fig. 8.1. If you want a faster and more fancy viewer there are several programs that can read the GROMACS trajectory formats – have a look at our homepage www.gromacs.org for updated links.

8.3 General properties

`g_energy`
`g_com`

To analyze some or all *energies* and other properties, such as *total pressure*, *pressure tensor*, *density*, *box-volume* and *box-sizes*, use the program `g_energy`. A choice can be made from a list

Groups can therefore consist of a series of *atom numbers*, but in some cases also of *molecule numbers*. It is also possible to specify a series of angles by *triples* of *atom numbers*, dihedrals by *quadruples* of *atom numbers* and bonds or vectors (in a molecule) by *pairs* of *atom numbers*. When appropriate the type of index file will be specified for the following analysis programs. To help creating such index files (`index.ndx`), there are a couple of programs to generate them, using either your input configuration or the topology. To generate an index file consisting of a series of *atom numbers* (as in the example of g_{AB}) use `make_ndx`. To generate an index file with angles or dihedrals, use `mk_angndx`. Of course you can also make them by hand. The general format is presented here:

```
[ Oxygen ]
   1       4       7
[ Hydrogen ]
   2       3       5       6
   8       9
```

First the group name is written between square brackets. The following atom numbers may be spread out over as many lines as you like. The atom numbering starts at 1.

8.1.1 Default Groups

When no index file is supplied to analysis tools, a number of default groups can be generated to choose from:

`System`
all atoms in the system

`Protein`
all protein atoms

`Protein-H`
protein atoms excluding hydrogens

`C-alpha`
C_α atoms

`Backbone`
protein backbone atoms; N, C_α and C

`MainChain`
protein main chain atoms: N, C_α, C and O, including oxygens in C-terminus

`MainChain+Cb`
protein main chain atoms including C_β

`MainChain+H`
protein main chain atoms including backbone amide hydrogen and hydrogens on the N-terminus

---

```
    1     3     5     1 1.200000e+02 2.928800e+02
    4     3     5     1 1.200000e+02 3.347200e+02
    1     6     7     1 1.200000e+02 2.928800e+02
    1     6     8     1 1.200000e+02 2.928800e+02
    7     6     8     1 1.200000e+02 3.347200e+02
    2     1     3     1 1.215000e+02 5.020800e+02
    2     1     6     1 1.215000e+02 5.020800e+02
    3     1     6     1 1.170000e+02 5.020800e+02

[ dihedrals ]
;  ai    aj    ak    al funct           phi           cp         mult
    2     1     3     4     1 1.800000e+02 3.347200e+01 2.000000e+00
    6     1     3     4     1 1.800000e+02 3.347200e+01 2.000000e+00
    2     1     3     5     1 1.800000e+02 3.347200e+01 2.000000e+00
    6     1     3     5     1 1.800000e+02 3.347200e+01 2.000000e+00
    2     1     6     7     1 1.800000e+02 3.347200e+01 2.000000e+00
    3     1     6     7     1 1.800000e+02 3.347200e+01 2.000000e+00
    2     1     6     8     1 1.800000e+02 3.347200e+01 2.000000e+00
    3     1     6     8     1 1.800000e+02 3.347200e+01 2.000000e+00

[ dihedrals ]
;  ai    aj    ak    al funct            q0           cq
    3     4     5     1     2 0.000000e+00 1.673600e+02
    6     7     8     1     2 0.000000e+00 1.673600e+02
    1     3     6     2     2 0.000000e+00 1.673600e+02

[ position_restraints ]
; you wouldn't normally use this for a molecule like Urea,
; but it's here for didactical purposes
; ai   funct    fc
   1     1     1000    1000    1000 ; Restrain to a point
   2     1     1000       0    1000 ; Restrain to a line (Y-axis)
   3     1     1000       0       0 ; Restrain to a plane (Y-Z-plane)

; Include SPC water topology
#include "spc.itp"

[ system ]
Urea in Water

[ molecules ]
;molecule name   nr.
Urea             1
SOL              1000
```

Here follows the explanatory text.

`[ defaults ]:`

- non-bond type = 1 (Lennard-Jones) or 2 (Buckingham)
  note: when using the Buckingham potential no combination rule can be used, and a full interaction matrix must be provided under the `nonbond_params` section.

- combination rule = 1 (supply $C^{(6)}$ and $C^{(12)}$, $\sigma_{ij} = \sqrt{\sigma_i \sigma_j}$), 2 (supply σ and ϵ, $\sigma_{ij} = \frac{1}{2}(\sigma_i + \sigma_j)$) or 3 (supply σ and ϵ, $\sigma_{ij} = \sqrt{\sigma_i \sigma_j}$)
- generate pairs = no (get 1-4 interactions from pair list) or yes (generate 1-4 interactions from normal Lennard-Jones parameters using FudgeLJ and FudgeQQ)
- FudgeLJ = factor to change Lennard-Jones 1-4 interactions
- FudgeQQ = factor to change electrostatic 1-4 interactions

note: FudgeLJ and FudgeQQ only need to be specified when generate pairs is set to ’yes’.

`#include "ffgmx.itp"`**:** this includes the bonded and non-bonded GROMACS parameters, the `gmx` in `ffgmx` will be replaced by the name of the force field you are actually using.

`[ moleculetype ]`**:** defines the name of your molecule in this `*.top` and nrexcl = 3 stands for excluding non-bonded interactions between atoms that are no further than 3 bonds away.

`[ atoms ]`**:** defines the molecule, where `nr` and `type` are fixed, the rest is user defined. So `atom` can be named as you like, `cgnr` made larger or smaller (if possible, the total charge of a charge group should be zero), and charges can be changed here too.

`[ bonds ]`**:** no comment.

`[ pairs ]`**:** 1-4 interactions

`[ angles ]`**:** no comment

`[ dihedrals ]`**:** in this case there are 9 proper dihedrals (funct = 1), 3 improper (funct = 2) and no Ryckaert-Bellemans type dihedrals. If you want to include Ryckaert-Bellemans type dihedrals in a topology, do the following (in case of e.g. decane):

```
[ dihedrals ]
;  ai    aj    ak    al funct       c0       c1       c2
    1     2     3     4     3
    2     3     4     5     3
```

and do not forget to *erase the 1-4 interaction* in `[ pairs ]`!!

`[ position_restraints ]`**:** harmonically restrain the selected particles to reference positions (sec. 4.2.9). The reference positions are read from a separate coordinate file by grompp.

`#include "spc.itp"`**:** includes a topology file that was already constructed (see next section, molecule.itp).

`[ system ]`**:** title of your system, user defined

`[ molecules ]`**:** this defines the total number of (sub)molecules in your system that are defined in this `*.top`. In this example file it stands for 1 urea molecules dissolved in 1000 water molecules. The molecule type SOL is defined in the `spc.itp` file.

5.6.2 Molecule.itp file

If you construct a topology file you will use more often (like a water molecule, `spc.itp`) it is better to make a `molecule.itp` file, which only lists the information of the molecule:

Chapter 8

Analysis

In this chapter different ways of analyzing your trajectory are described. The names of the corresponding analysis programs are given. Specific info on the in- and output of these programs can be found in the on-line manual at www.gromacs.org. The output files are often produced as finished Grace/Xmgr graphs.

First in sec. 8.1 the group concept in analysis is explained. Then the different analysis tools are presented.

8.1 Groups in Analysis.

```
make_ndx
mk_angndx
```

In chapter 3 it was explained how *groups of atoms* can be used in the MD-program. In most analysis programs groups of atoms are needed to work on. Most programs can generate several default index groups, but groups can always be read from an index file. Let’s consider a simulation of a binary mixture of components A and B. When we want to calculate the radial distribution function (rdf) $g_{AB}(r)$ of A with respect to B, we have to calculate

$$4\pi r^2 g_{AB}(r) \;=\; V \sum_{i\in A}^{N_A} \sum_{j\in B}^{N_B} P(r) \tag{8.1}$$

where V is the volume and $P(r)$ is the probability to find a B atom at a distance r from an A atom.

By having the user define the *atom numbers* for groups A and B in a simple file we can calculate this g_{AB} in the most general way, without having to make any assumptions in the rdf-program about the type of particles.

do_dssp	assigns secondary structure and calculates solvent accessible surface area
g_chi	calculates everything you want to know about chi and other dihedrals
g_helix	calculates everything you want to know about helices
g_rama	computes Ramachandran plots
xrama	shows animated Ramachandran plots
wheel	plots helical wheels

Interfaces

g_potential	calculates the electrostatic potential across the box
g_density	calculates the density of the system
g_order	computes the order parameter per atom for carbon tails
g_h2order	computes the orientation of water molecules
g_bundle	analyzes bundles of axes, e.g. transmembrane helices

Covariance analysis

g_covar	calculates and diagonalizes the covariance matrix
g_anaeig	analyzes the eigenvectors

Normal modes

grompp	makes a run input file
mdrun	finds a potential energy minimum
nmrun	calculates the Hessian
g_nmeig	diagonalizes the Hessian
g_anaeig	analyzes the normal modes
g_nmens	generates an ensemble of structures from the normal modes

---

```
[ moleculetype ]
; name  nrexcl
Urea       3

[ atoms ]
;   nr    type   resnr  residu    atom    cgnr  charge
     1       C       1    UREA      C1       1   0.683
     .................
     .................
     8       H       1    UREA      H8       3   0.276

[ bonds ]
;  ai    aj funct           c0           c1
    3     4     1 1.000000e-01 3.744680e+05
     .................
     .................
    1     6     1 1.330000e-01 3.765600e+05

[ pairs ]
;  ai    aj funct           c0           c1
    2     4     1 0.000000e+00 0.000000e+00
     .................
     .................
    5     6     1 0.000000e+00 0.000000e+00

[ angles ]
;  ai    aj    ak funct           c0           c1
    1     3     4     1 1.200000e+02 2.928800e+02
     .................
     .................
    3     1     6     1 1.170000e+02 5.020800e+02

[ dihedrals ]
;  ai    aj    ak    al funct           c0           c1           c2
    2     1     3     4     1 1.800000e+02 3.347200e+01 2.000000e+00
     .................
     .................
    3     1     6     8     1 1.800000e+02 3.347200e+01 2.000000e+00

[ dihedrals ]
;  ai    aj    ak    al funct           c0           c1
    3     4     5     1     2 0.000000e+00 1.673600e+02
    6     7     8     1     2 0.000000e+00 1.673600e+02
    1     3     6     2     2 0.000000e+00 1.673600e+02
```

This results in a very short `*.top` file as described in the previous section, but this time you only need to include files:

```
; The force field files to be included
#include "ffgmx.itp"

; Include urea topology
```

```
#include "urea.itp"

; Include SPC water topology
#include "spc.itp"

[ system ]
Urea in Water

[ molecules ]
;molecule name  number
Urea              1
SOL               1000
```

5.6.3 Ifdef option

A very powerful feature in GROMACS is the use of `#ifdef` statements in your `*.top` file. By making use of this statement, different parameters for one molecule can be used in the same `*.top` file. An example is given for TFE, where there is an option to use different charges on the atoms: charges derived by De Loof *et al.* [62] or by Van Buuren and Berendsen [43]. In fact you can use all the options of the C-Preprocessor, `cpp`, because this is used to scan the file. The way to make use of the `#ifdef` option is as follows:

- in `grompp.mdp` (the GROMACS preprocessor input parameters) use the option
  `define = -DDeloof`
  or
  `define =`
- put the `#ifdef` statements in your `*.top`, as shown below:

```
...

[ atoms ]
; nr   type   resnr  residu   atom    cgnr    charge      mass
#ifdef DeLoof
; Use Charges from DeLoof
   1      C      1    TFE       C       1       0.74
   2      F      1    TFE       F       1      -0.25
   3      F      1    TFE       F       1      -0.25
   4      F      1    TFE       F       1      -0.25
   5    CH2      1    TFE     CH2       1       0.25
   6     OA      1    TFE      OA       1      -0.65
   7     HO      1    TFE      HO       1       0.41
#else
; Use Charges from VanBuuren
   1      C      1    TFE       C       1       0.59
   2      F      1    TFE       F       1      -0.2
   3      F      1    TFE       F       1      -0.2
   4      F      1    TFE       F       1      -0.2
   5    CH2      1    TFE     CH2       1       0.26
   6     OA      1    TFE      OA       1      -0.55
```

Distances in structures over time

g_mindist	calculates the minimum distance between two groups
g_dist	calculates the distances between the centers of mass of two groups
g_mdmat	calculates residue contact maps
g_rmsdist	calculates atom pair distances averaged with power 2, -3 or -6

Mass distribution properties over time

g_traj	plots x, v, f, box, temperature and rotational energy
g_gyrate	calculates the radius of gyration
g_msd	calculates mean square displacements
g_rotacf	calculates the rotational correlation function for molecules
g_rdf	calculates radial distribution functions

Analyzing bonded interactions

g_bond	calculates bond length distributions
mk_angndx	generates index files for g_angle
g_angle	calculates distributions and correlations for angles and dihedrals
g_dih	analyzes dihedral transitions

Structural properties

g_hbond	computes and analyzes hydrogen bonds
g_saltbr	computes salt bridges
g_sas	computes solvent accessible surface area
g_order	computes the order parameter per atom for carbon tails
g_sgangle	computes the angle and distance between two groups
g_sorient	analyzes solvent orientation around solutes
g_bundle	analyzes bundles of axes, e.g. helices
g_disre	analyzes distance restraints

Kinetic properties

g_traj	plots x, v, f, box, temperature and rotational energy
g_velacc	calculates velocity autocorrelation functions
g_tcaf	calculates viscosities of liquids

Electrostatic properties

genion	generates mono atomic ions on energetically favorable positions
g_potential	calculates the electrostatic potential across the box
g_dipoles	computes the total dipole plus fluctuations
g_dielectric	calculates frequency dependent dielectric constants

Protein specific analysis

genion	generates mono atomic ions on energetically favorable positions
genconf	multiplies a conformation in 'random' orientations
genpr	generates position restraints for index groups
protonate	protonates structures

Running a simulation

grompp	makes a run input file
tpbconv	makes a run input file for restarting a crashed run
mdrun	performs a simulation
xmdrun	performs simulations with extra experimental features

Viewing trajectories

ngmx	displays a trajectory
trjconv	converts trajectories to e.g. pdb which can be viewed with e.g. rasmol

Processing energies

g_energy	writes energies to xvg files and displays averages
g_enemat	extracts an energy matrix from an energy file
mdrun	with -rerun (re)calculates energies for trajectory frames

Converting files

editconf	converts and manipulates structure files
trjconv	converts and manipulates trajectory files
trjcat	concatenates trajectory files
eneconv	converts energy files
xmp2ps	converts XPM matrices to encapsulated postscript (or XPM)

Tools

make_ndx	makes index files
mk_angndx	generates index files for g_angle
gmxcheck	checks and compares files
gmxdump	makes binary files human readable
g_traj	plots x, v and f of selected atoms/groups (and more) from a trajectory
g_analyze	analyzes data sets
trjorder	orders molecules according to their distance to a group

Distances between structures

g_rms	calculates rmsd's with a reference structure and rmsd matrices
g_confrms	fits two structures and calculates the rmsd
g_cluster	clusters structures
g_rmsf	calculates atomic fluctuations

```
   7     HO      1    TFE      HO       1       0.3
#endif

[ bonds ]
;  ai    aj funct           c0           c1
    6     7     1 1.000000e-01 3.138000e+05
    1     2     1 1.360000e-01 4.184000e+05
    1     3     1 1.360000e-01 4.184000e+05
    1     4     1 1.360000e-01 4.184000e+05
    1     5     1 1.530000e-01 3.347000e+05
    5     6     1 1.430000e-01 3.347000e+05

...
```

5.6.4 Free energy calculations

Free energy differences between two systems A and B can be calculated as described in sec. 3.12. The systems A and B are described by topologies consisting of the same number of molecules with the same number of atoms. Masses and non-bonded interactions can be perturbed by adding B parameters in the `[ atoms ]` field. Bonded interactions can be perturbed by adding B parameters to the bonded types or the bonded interactions. The parameters that can be perturbed are listed in Table 5.3 and Table 5.4. The λ-dependence of the interactions is described in section sec. 4.3. Below is an example of a topology which changes from 200 propanols to 200 pentanes using the GROMOS-96 force field.

```
; Include forcefield parameters
#include "ffG43a1.itp"

[ moleculetype ]
; Name            nrexcl
PropPent          3

[ atoms ]
; nr type resnr residue atom cgnr  charge    mass  typeB chargeB  massB
  1    H    1     PROP    PH    1   0.398    1.008  CH3     0.0  15.035
  2   OA    1     PROP    PO    1  -0.548  15.9994  CH2     0.0  14.027
  3  CH2    1     PROP   PC1    1   0.150   14.027  CH2     0.0  14.027
  4  CH2    1     PROP   PC2    2   0.000   14.027
  5  CH3    1     PROP   PC3    2   0.000   15.035

[ bonds ]
;  ai    aj funct    par_A  par_B
    1     2     2    gb_1   gb_26
    2     3     2    gb_17  gb_26
    3     4     2    gb_26  gb_26
    4     5     2    gb_26

[ pairs ]
;  ai    aj funct
    1     4     1
    2     5     1
```

```
[ angles ]
;  ai    aj    ak funct    par_A   par_B
    1     2     3     2    ga_11   ga_14
    2     3     4     2    ga_14   ga_14
    3     4     5     2    ga_14   ga_14

[ dihedrals ]
;  ai    aj    ak    al funct    par_A   par_B
    1     2     3     4     1    gd_12   gd_17
    2     3     4     5     1    gd_17   gd_17

[ system ]
; Name
Propanol to Pentane

[ molecules ]
; Compound        #mols
PropPent            200
```

Atoms that are not perturbed, `PC2` and `PC3`, do not need B parameter specifications, the B parameters will be copied from the A parameters. Bonded interactions between atoms that are not perturbed do not need B parameter specifications, here this is the case for the last bond. Topologies using the GROMACS force field need no bonded parameters at all, since both the A and B parameters are determined by the atom types. Non-bonded interactions involving one or two perturbed atoms use the free-energy perturbation functional forms. Non-bonded interaction between two non-perturbed atoms use the normal functional forms. This means that when, for instance, only the charge of a particle is perturbed, its Lennard-Jones interactions will also be affected when lambda is not equal to zero or one.

Note that this topology uses the GROMOS-96 force field, in which the bonded interactions are not determined by the atom types. The bonded interaction strings are converted by the C-preprocessor. The force field parameter files contain lines like:

```
#define gb_26       0.1530  7.1500e+06

#define gd_17     0.000       5.86          3
```

5.6.5 Constraint force

The constraint force between two atoms in one molecule can be calculated with the free energy perturbation code by adding a constraint between the two atoms, with a different length in the A and B topology. When the B length is 1 nanometer longer than the A length and lambda is kept constant at zero, the derivative of the Hamiltonian with respect to lambda is the constraint force. For constraints between molecules the pull code can be used, see sec. 6.1. Below is an example for calculating the constraint force at 0.7 nanometer between two methanes in water, by combining the two methanes into one molecule. The added constraint is of function type 2, which means that it is not used for generating exclusions (see 5.3.5).

```
; Include forcefield parameters
```

---

freezegrps:

Groups that are to be frozen (i.e. their X, Y, and/or Z position will not be updated; e.g. `Lipid SOL`). **freezedim** specifies for which dimension the freezing applies. You might want to use energy group exclusions for completely frozen groups.

freezedim:

dimensions for which groups in **freezegrps** should be frozen, specify `Y` or `N` for X, Y and Z and for each group (e.g. `Y Y N N N N` means that particles in the first group can move only in Z direction. The particles in the second group can move in any direction).

cos_acceleration: (0) [nm ps^{-2}]

the amplitude of the acceleration profile for calculating the viscosity. The acceleration is in the X-direction and the magnitude is **cos_acceleration** cos(2 pi z/boxheight). Two terms are added to the energy file: the amplitude of the velocity profile and 1/viscosity.

7.3.19 Electric fields

E_x ; E_y ; E_z:

If you want to use an electric field in a direction, enter 3 numbers after the appropriate E_*, the first number: the number of cosines, only 1 is implemented (with frequency 0) so enter 1, the second number: the strength of the electric field in V nm^{-1}, the third number: the phase of the cosine, you can enter any number here since a cosine of frequency zero has no phase.

E_xt ; E_yt ; E_zt:

not implemented yet

7.3.20 User defined thingies

user1_grps ; user2_grps:

userint1 (0); userint2 (0); userint3 (0); userint4: (0)

userreal1 (0); userreal2 (0); userreal3 (0); userreal4: (0)

These you can use if you hack out code. You can pass integers and reals to your subroutine. Check the inputrec definition in `src/include/types/inputrec.h`

7.4 Programs by topic

Generating topologies and coordinates

pdb2gmx	converts pdb files to topology and coordinate files
x2top	generates a primitive topology from coordinates
editconf	edits the box and writes subgroups
genbox	solvates a system

disre_fc: (1000) [kJ mol^{-1} nm^{-2}]
force constant for distance restraints, which is multiplied by a (possibly) different factor for each restraint

disre_tau: (0) [ps]
time constant for distance restraints running average

nstdisreout: (100) [steps]
frequency to write the running time averaged and instantaneous distances of all atom pairs involved in restraints to the energy file (can make the energy file very large)

7.3.17 Free Energy Perturbation

free_energy:

no
Only use topology A.

yes
Interpolate between topology A (lambda=0) to topology B (lambda=1) and write the derivative of the Hamiltonian with respect to lambda to the energy file and to `dgdl.xvg`. The potentials, bond-lengths and angles are interpolated linearly as described in the manual. When **sc_alpha** is larger than zero, soft-core potentials are used for the LJ and Coulomb interactions.

init_lambda: (0)
starting value for lambda

delta_lambda: (0)
increase per time step for lambda

sc_alpha: (0)
the soft-core parameter, a value of 0 results in linear interpolation of the LJ and Coulomb interactions

sc_sigma: (0.3) [nm]
the soft-core sigma for particles which have a C6 or C12 parameter equal to zero

7.3.18 Non-equilibrium MD

acc_grps:
groups for constant acceleration (e.g.: `Protein Sol`) all atoms in groups Protein and Sol will experience constant acceleration as specified in the **accelerate** line

accelerate: (0) [nm ps^{-2}]
acceleration for **acc_grps**; x, y and z for each group (e.g. `0.1 0.0 0.0 -0.1 0.0 0.0` means that first group has constant acceleration of 0.1 nm ps^{-2} in X direction, second group the opposite).

```
#include "ffG43a1.itp"

[ moleculetype ]
; Name            nrexcl
Methanes               1

[ atoms ]
; nr   type   resnr  residu   atom    cgnr     charge    mass
   1    CH4     1     CH4      C1       1          0    16.043
   2    CH4     1     CH4      C2       2          0    16.043

[ constraints ]
;  ai    aj funct   length_A  length_B
    1     2     2        0.7       1.7

#include "spc.itp"

[ system ]
; Name
Methanes in Water

[ molecules ]
; Compound        #mols
Methanes              1
SOL                2002
```

5.6.6 Coordinate file

Files with the `.gro` file extension contain a molecular structure in GROMOS87 format. A sample piece is included below:

```
MD of 2 waters, reformat step, PA aug-91
    6
    1WATER  OW1    1   0.126   1.624   1.679  0.1227 -0.0580  0.0434
    1WATER  HW2    2   0.190   1.661   1.747  0.8085  0.3191 -0.7791
    1WATER  HW3    3   0.177   1.568   1.613 -0.9045 -2.6469  1.3180
    2WATER  OW1    4   1.275   0.053   0.622  0.2519  0.3140 -0.1734
    2WATER  HW2    5   1.337   0.002   0.680 -1.0641 -1.1349  0.0257
    2WATER  HW3    6   1.326   0.120   0.568  1.9427 -0.8216 -0.0244
   1.82060   1.82060   1.82060
```

This format is fixed, i.e. all columns are in a fixed position. If you want to read such a file in your own program without using the GROMACS libraries you can use the following formats:

C-format: `"%5i%5s%5s%5i%8.3f%8.3f%8.3f%8.4f%8.4f%8.4f"`

Or to be more precise, with title etc., it looks like this:

```
  "%s\n", Title
  "%5d\n", natoms
  for (i=0; (i<natoms); i++) {
```

```
    "%5d%5s%5s%5d%8.3f%8.3f%8.3f%8.4f%8.4f%8.4f\n",
      residuenr,residuename,atomname,atomnr,x,y,z,vx,vy,vz
  }
  "%10.5f%10.5f%10.5f%10.5f%10.5f%10.5f%10.5f%10.5f%10.5f\n",
    box[X][X],box[Y][Y],box[Z][Z],
    box[X][Y],box[X][Z],box[Y][X],box[Y][Z],box[Z][X],box[Z][Y]
```

Fortran format: `(i5,2a5,i5,3f8.3,3f8.4)`

So `confin.gro` is the GROMACS coordinate file and is almost the same as the GROMOS-87 file (for GROMOS users: when used with ntx=7). The only difference is the box for which GROMACS uses a tensor, not a vector.

morse:

no
bonds are represented by a harmonic potential

yes
bonds are represented by a Morse potential

7.3.15 Energy group exclusions

energygrp_excl:
Pairs of energy groups for which all non-bonded interactions are excluded. An example: if you have two energy groups `Protein` and `SOL`, specifying

`energy_excl = Protein Protein  SOL SOL`

would give only the non-bonded interactions between the protein and the solvent. This is especially useful for speeding up energy calculations with `mdrun -rerun` and for excluding interactions within frozen groups.

7.3.16 NMR refinement

disre:

no
no distance restraints (ignore distance restraints information in topology file)

simple
simple (per-molecule) distance restraints

ensemble
distance restraints over an ensemble of molecules

disre_weighting:

equal
divide the restraint force equally over all atom pairs in the restraint

conservative
the forces are the derivative of the restraint potential, this results in an r^{-7} weighting of the atom pairs

disre_mixed:

no
the violation used in the calculation of the restraint force is the time averaged violation

yes
the violation used in the calculation of the restraint force is the square root of the time averaged violation times the instantaneous violation

7.3.14 Bonds

constraints:

none
No constraints, i.e. bonds are represented by a harmonic or a Morse potential (depending on the setting of **morse**) and angles by a harmonic potential.

hbonds
Only constrain the bonds with H-atoms.

all-bonds
Constrain all bonds.

h-angles
Constrain all bonds and constrain the angles that involve H-atoms by adding bond-constraints.

all-angles
Constrain all bonds and constrain all angles by adding bond-constraints.

constraint_algorithm:

lincs
LINear Constraint Solver. The accuracy in set with **lincs_order**, which sets the number of matrices in the expansion for the matrix inversion, 4 is enough for a ”normal” MD simulation, 8 is needed for BD with large time-steps. The accuracy of the constraints is printed to the log file every **nstlog** steps. If a bond rotates more than **lincs_warnangle** [degrees] in one step, a warning will be printed both to the log file and to `stderr`. Lincs should not be used with coupled angle constraints.

shake
Shake is slower and less stable than Lincs, but does work with angle constraints. The relative tolerance is set with **shake_tol**, 0.0001 is a good value for ”normal” MD.

unconstrained_start:

no
apply constraints to the start configuration

yes
do not apply constraints to the start configuration

shake_tol: (0.0001)
relative tolerance for shake

lincs_order: (4)
Highest order in the expansion of the constraint coupling matrix. **lincs_order** is also used for the number of Lincs iterations during energy minimization, only one iteration is used in MD.

lincs_warnangle: (30) [degrees]
maximum angle that a bond can rotate before Lincs will complain

Chapter 6

Special Topics

6.1 Calculating potentials of mean force: the pull code

There are a number of options to calculate potentials of mean force and related topics. In the current version of GROMACS this is implemented through some extra files for `mdrun`.

6.1.1 Overview

Four different types of calculation are supported:

1. **Constraint forces** The distance between the centers of mass of two groups of atoms can be constrained and the constraint force monitored. The distance can be in 1, 2, or 3 dimensions. This method uses the SHAKE algorithm but only needs 1 iteration to be exact if only two groups are constrained.
2. **Umbrella sampling** A simple umbrella sampling with an harmonic umbrella potential that acts on the center of mass of a group of atoms.
3. **AFM pulling** A spring is connected to an atom and slowly retracted. This has the effect of pulling an atom or group of atoms away from its initial location. The rate constant and spring constant for the spring can be varied to study e.g. the unbinding of a protein and a ligand (see figure 6.1).
4. **Starting structures** This option creates a number of starting structures for potential of mean force calculations, moving 1 or 2 groups of atoms at a specified rate towards or away from a reference group, writing out a coordinate file at specified intervals. Note that the groups given in the index file are translated a specified distance each step, but in addition also undergo the normal MD, subject to definitions of *e.g.* temperature coupling groups, freeze groups and the like.

In the calculations, there has to be 1 reference group and 1 or 2 other groups of atoms. For constrained runs, the distance between the reference group and the other groups is kept constant

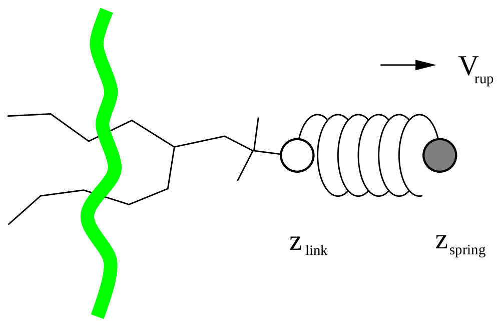


Figure 6.1: Schematic picture of pulling a lipid out of a lipid bilayer with AFM pulling. V_{rup} is the velocity at which the spring is retracted, Z_{link} is the atom to which the spring is attached and Z_{spring} is the location of the spring.

at the distance they have in the input coordinate file (`.tpr`) file.

6.1.2 Usage

Input files

The `mdrun` programs needs 4 additional files: 2 input files and 2 output files.

`-pi pull.ppa`
If this file is specified the pull code will be used. It contains the parameters that control what type of calculation is done. A full explanation of all the options is given below.

`-pn index.ndx`
This file defines the different groups for use in all pull calculations. The groups are referred to by name, so the index file can contain other groups that are not used as well.

`-po pullout.ppa`
A formatted copy of the input parameter file with the parameters that were actually used in the run.

`-pdo pull.pdo`
The data file with the calculated forces (AFM pulling, constraint force) or positions (umbrella sampling).

Definition of groups

The way the reference groups and different reference types work is summarized in figure 6.2. There are four different possibilities for the reference group.

---

compressibility: [bar^{-1}]
compressibility (NOTE: this is now really in bar^{-1}) For water at 1 atm and 300 K the compressibility is 4.5e-5 [bar^{-1}].

ref_p: [bar]
reference pressure for coupling

7.3.12 Simulated annealing

annealing:

no
No simulated annealing.

yes
Simulated annealing to 0 [K] at time **zero_temp_time** (ps). Reference temperature for the Berendsen-thermostat is **ref_t** x (1 - time / **zero_temp_time**), time constant is **tau_t** [ps]. Note that the reference temperature will not go below 0 [K], i.e. after **zero_temp_time** (if it is positive) the reference temperature will be 0 [K]. Negative **zero_temp_time** results in heating, which will go on indefinitely.

zero_temp_time: (0) [ps]
time at which temperature will be zero (can be negative). Temperature during the run can be seen as a straight line going through T=**ref_t** [K] at t=0 [ps], and T=0 [K] at t=**zero_temp_time** [ps]. Look in our FAQ for a schematic graph of temperature versus time.

7.3.13 Velocity generation

gen_vel:

no
Do not generate velocities at startup. The velocities are set to zero when there are no velocities in the input structure file.

yes
Generate velocities according to a Maxwell distribution at temperature **gen_temp** [K], with random seed **gen_seed**. This is only meaningful with integrator **md**.

gen_temp: (300) [K]
temperature for Maxwell distribution

gen_seed: (173529) [integer]
used to initialize random generator for random velocities, when **gen_seed** is set to -1, the seed is calculated as `(time() + getpid()) % 1000000`

no

No pressure coupling. This means a fixed box size.

berendsen

Exponential relaxation pressure coupling with time constant **tau_p** [ps]. The box is scaled every timestep. It has been argued that this does not yield a correct thermodynamic ensemble, but it is the most efficient way to scale a box at the beginning of a run.

Parinello-Rahman

Extended-ensemble pressure coupling where the box vectors are subject to an equation of motion. The equation of motion for the atoms is coupled to this. No instantaneous scaling takes place. As for Nose-Hoover temperature coupling the time constant **tau_p** [ps] is the period of pressure fluctuations at equilibrium. This is probably a better method when you want to apply pressure scaling during data collection, but beware that you can get very large oscillations if you are starting from a different pressure.

pcoupltype:

isotropic

Isotropic pressure coupling with time constant **tau_p** [ps]. The compressibility and reference pressure are set with **compressibility** [bar^{-1}] and **ref_p** [bar], one value is needed.

semiisotropic

Pressure coupling which is isotropic in the x and y direction, but different in the z direction. This can be useful for membrane simulations. 2 values are needed for x/y and z directions respectively.

anisotropic

Idem, but 6 values are needed for xx, yy, zz, xy/yx, xz/zx and yz/zy components respectively. When the off-diagonal compressibilities are set to zero, a rectangular box will stay rectangular. Beware that anisotropic scaling can lead to extreme deformation of the simulation box.

surface-tension

Surface tension coupling for surfaces parallel to the xy-plane. Uses normal pressure coupling for the z-direction, while the surface tension is coupled to the x/y dimensions of the box. The first **ref_p** value is the reference surface tension times the number of surfaces [bar nm], the second value is the reference z-pressure [bar]. The two **compressibility** [bar^{-1}] values are the compressibility in the x/y and z direction respectively. The value for the z-compressibility should be reasonably accurate since it influences the converge of the surface-tension, it can also be set to zero to have a box with constant height.

triclinic

Fully dynamic box - supported, but experimental. You should provide six values for the compressibility and reference pressure.

tau_p: (1) [ps]

time constant for coupling

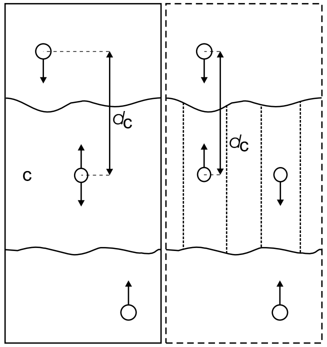


Figure 6.2: Overview of the different reference group possibilities, applied to interface systems. C is the reference group. The circles represent the center of mass of 2 groups plus the reference group, and d_c is the reference distance.

`com`

The center of mass of the group given under `reference_group`, calculated each step from the current coordinates.

`com_t0`

The center of mass of the group given under `reference_group`, calculated each step from the current coordinates, but corrected for atoms that have crossed the box. If the reference group consists of all the water molecules in the system, and a single water molecule moves across the box and enters from the other side, the c.o.m. will show a slight jump. This is simply due to the periodic boundary conditions, and shows that the center of mass in a simulation in periodic boundary conditions is ill defined if the group used to calculate it is *e.g.* a slab of liquid. If the 'real' positions are used instead of the coordinates that have been reset to be inside the box, the center of mass of the **whole** system is conserved.

`dynamic`

In a phospholipid bilayer system it may be of interest to calculate the pmf of a lipid as function of its distance from the whole bilayer. The whole bilayer can be taken as reference group in that case, but it might also be of interest to define the reaction coordinate for the pmf more locally. `dynamic` does not use all the atoms of the `reference_group`, but instead only those within a cylinder with radius `r` below the main group. This only works for distances defined in 1 dimension, and the cylinder is oriented with its long axis along this 1 dimension. A second cylinder can be defined with `rc`, with a linear switch function that weighs the contribution of atoms between `r` and `rc` with distance. This smoothes the effects of atoms moving in and out of the cylinder (which causes jumps in the constraint forces).

`dynamic_t0`

The same as `dynamic`, but the coordinates are corrected for boxcrossings like in `com_t0`.

Note that strictly speaking this is not correct if the reference group is not the whole system, including the groups defined with `group_1` and `group_2`.

To further smooth rapidly fluctuating distances between the reference group and the other groups, the average distance can be constrained instead of the instantaneous distance. This is defined by setting `reflag` to the number of steps to average over. However, using this option is not strictly correct for calculating potentials of mean force from the average constraint force.

The parameter file

`verbose = no`
If this is set to `yes`, a large amount of detailed information is sent to `stderr`, which is only useful for diagnostic purposes. The `.pdo` file also becomes more detailed, which is not necessary for normal use.

`runtype = constraint`
Options are `start, afm, constraint, umbrella`. This selects the type of calculation: making starting structures, AFM pulling, constraint force calculation or umbrella sampling.

`group_1 = MB21_1`

`group_2 = MB21_2`
The groups with the atoms to act on. The first group is mandatory, the second optional.

`reference_group = OCTA`
The reference group. Distances are calculated betweeen `group_1` (and `group_2` if specified) and this group. If *e.g.* the constraint force between two ions is needed, you would specifiy `group_1` as a group with 1 ion, and `reference_group` as the other ion.

`reftype = com`
The type of reference group. Options are `com, com_t0, dynamic, dynamic_t0` as explained above.

`reflag = 1`
The position of the reference group can be taken as average over a number of steps, specified by `reflag` (see above).

`direction = 0.0 0.0 1.0`
Distances are calculated weighted by x, y, z as specified in `direction`. Setting them all to 1.0 calculates the distance between two groups, setting the first two to 0.0 and the third to 1.0 calculates the distance in the z direction only.

`reverse = to_reference`
This option selects the direction in which the groups are moved with respect to the reference group for AFM pulling and starting structure calculations. Choices are `to_reference, from_reference`.

---

pme_order (4)
Interpolation order for PME. 4 equals cubic interpolation. You might try 6/8/10 when running in parallel and simultaneously decrease grid dimension.

ewald_rtol (1e-5)
The relative strength of the Ewald-shifted direct potential at the cutoff is given by **ewald_rtol**. Decreasing this will give a more accurate direct sum, but then you need more wave vectors for the reciprocal sum.

optimize_fft:

no
Don't calculate the optimal FFT plan for the grid at startup.

yes
Calculate the optimal FFT plan for the grid at startup. This saves a few percent for long simulations, but takes a couple of minutes at start.

7.3.10 Temperature coupling

tcoupl:

no
No temperature coupling.

berendsen
Temperature coupling with a Berendsen-thermostat to a bath with temperature **ref_t** [K], with time constant **tau_t** [ps]. Several groups can be coupled separately, these are specified in the **tc_grps** field separated by spaces.

nose-hoover
Temperature coupling with a by using a Nose-Hoover extended ensemble. The reference temperature and coupling groups are selected as above, but in this case **tau_t** [ps] controls the period of the temperature fluctuations at equilibrium, which is slightly different from a relaxation time.

tc_grps:
groups to couple separately to temperature bath

tau_t: [ps]
time constant for coupling (one for each group in tc_grps)

ref_t: [K]
reference temperature for coupling (one for each group in tc_grps)

7.3.11 Pressure coupling

pcoupl:

vdwtype:

Cut-off
Twin range cut-off's with neighbor list cut-off **rlist** and VdW cut-off **rvdw**, where **rvdw** > **rlist**.

Shift
The LJ (not Buckingham) potential is decreased over the whole range and the forces decay smoothly to zero between **rvdw_switch** and **rvdw**. The neighbor search cut-off **rlist** should be 0.1 to 0.3 nm larger than **rvdw** to accommodate for the size of charge groups and diffusion between neighbor list updates.

User
`mdrun` will now expect to find two files with user-defined functions: `rtab.xvg` for Repulsion, `dtab.xvg` for Dispersion. These files should contain 5 columns: the `x` value, `f(x)`, `-f`$^{(1)}$`(x)`, `f`$^{(2)}$`(x)` and `-f`$^{(3)}$`(x)`, where `f`$^{(n)}$`(x)` denotes the `n`th derivative of function `f(x)` with respect to `x`. The `x` should run from 0 [nm] to **rlist**+`0.5` [nm], with a spacing of `0.002` [nm] when you run in single precision, or `0.0005` [nm] when you run in double precision. The function value at `x=0` is not important. When you want to use LJ correction, make sure that **rvdw** corresponds to the cut-off in the user-defined function.

rvdw_switch: (0) [nm]
where to start switching the LJ potential

rvdw: (1) [nm]
distance for the LJ or Buckingham cut-off

DispCorr:

no
don't apply any correction

EnerPres
apply long range dispersion corrections for Energy and Pressure

Ener
apply long range dispersion corrections for Energy only

fourierspacing: (0.12) [nm]
The maximum grid spacing for the FFT grid when using PPPM or PME. For ordinary Ewald the spacing times the box dimensions determines the highest magnitude to use in each direction. In all cases each direction can be overridden by entering a non-zero value for **fourier_n***.

fourier_nx (0) ; fourier_ny (0) ; fourier_nz: (0)
Highest magnitude of wave vectors in reciprocal space when using Ewald. Grid size when using PPPM or PME. These values override **fourierspacing** per direction. The best choice is powers of 2, 3, 5 and 7. Avoid large primes.

`r = 0`
If dynamic reference groups are selected (`dynamic, dynamic_t0`), `r` is the radius of the cylinder used to define which atoms are part of the reference group (see above).

`rc = 0`
With dynamic reference groups, the cylinder can be smoothly switched so that atoms that fall between `r` and `rc` are weighted linearly from 1 to 0 going from `r` to `rc`. As reasonable initial values we suggest `r = 1.0` and `rc = 1.5` but this will depend strongly on the exact system of interest.

`update = 1`
The frequency with which the dynamic reference groups are recalculated. Usually there is no reason to use anything other than 1.

`pullrate = 0.00005`
The pull rate in nm/timestep for AFM pulling.

`forceconstant = 100`
The force constant for the spring in AFM pulling, in kJ mol^{-1} nm^{-2}.

`width = 0`
Width of the umbrella sampling potential in kJ mol^{-1} nm^{-2}.

`r0_group2 = 0.0 0.0 3.300`
The initial location of the groups with respect to the reference group. Only coordinates selected with `direction` are taken into account. The groups are moved to these initial positions before the actual creation of a series of starting structures commences.

`tolerance = 0.001`
The accuracy with which the actual position of the groups must match the calculated ideal positions for a starting structure (in nm).

`translation_rate = 0.00001`
The rate of translation in all directions (nm/step). As mentioned above, normal MD force calculations and position updates also act on the groups.

`transstep = 0.2`
The interval in nm at which structures are written out.

6.1.3 Output

The output file is a text file with forces or positions, one per line. If there are two groups they alternate in the output file. Currently there is no supported analysis program to read this file, but it is simple to parse.

6.1.4 Limitations

Apart from obvious limitations that are simply not implemented (*e.g.* a better umbrella sampling and analysis scheme), there is one important limitation: constraint forces can **only** be calculated between molecules or groups of molecules. If a group contains part of a molecule of which the bondlengths are constrained, SHAKE or LINCS and the constraint force calculation here will interfere with each other, making the results unreliable. If a constraint force is wanted between two atoms, this can be done through the free energy perturbation code. In summary:

- **pull code:** between molecules or groups of molecules.
- **free energy perturbation code:** between single atoms.
- **not possible currently:** between groups of atoms that are part of a larger molecule for which the bonds are constrained with SHAKE or LINCS.

6.1.5 Implementation

The code for the options described above can be found in the files `pull.c, pullinit.c, pullio.c, pullutil.c` and the headerfiles `pull.h` and `pulls.h`. This last file defines a few datatypes, `pull.h` explains the main functions.

6.1.6 Future development

There are several additional features that would be useful, including more advanced umbrella sampling, an analysis tool to analyse the output of the pull code, incorporation of the input parameters and index file into the `grompp` program input files, extension to more groups, more flexible definition of a reaction coordinate, extension to groups that are parts of molecules that use SHAKE or LINCS, and a combination of the starting structure calculation with constraints for faster convergence of starting structures.

6.2 Removing fastest degrees of freedom

The maximum time step in MD simulations is limited by the smallest oscillation period that can be found in the simulated system. Bond-stretching vibrations are in their quantum-mechanical ground state and are therefore better represented by a constraint than by a harmonic potential.

For the remaining degrees of freedom, the shortest oscillation period as measured from a simulation is 13 fs for bond-angle vibrations involving hydrogen atoms. Taking as a guideline that with a Verlet (leap-frog) integration scheme a minimum of 5 numerical integration steps should be performed per period of a harmonic oscillation in order to integrate it with reasonable accuracy, the maximum time step will be about 3 fs. Disregarding these very fast oscillations of period 13 fs the next shortest periods are around 20 fs, which will allow a maximum time step of about 4 fs

Removing the bond-angle degrees of freedom from hydrogen atoms can best be done by defining them as dummy atoms in stead of normal atoms. Where a normal atoms is connected to the molecule with bonds, angles and dihedrals, a dummy atom's position is calculated from the position of three nearby heavy atoms in a predefined manner (see also sec. 4.5). For the hydrogens in water and in hydroxyl, sulfhydryl or amine groups, no degrees of freedom can be removed, because rotational freedom should be preserved. The only other option available to slow down these

---

PME

Fast Particle-Mesh Ewald electrostatics. Direct space is similar to the Ewald sum, while the reciprocal part is performed with FFTs. Grid dimensions are controlled with **fourierspacing** and the interpolation order with **pme_order**. With a grid spacing of 0.1 nm and cubic interpolation the electrostatic forces have an accuracy of 2-3e-4. Since the error from the vdw-cutoff is larger than this you might try 0.15 nm. When running in parallel the interpolation parallelizes better than the FFT, so try decreasing grid dimensions while increasing interpolation.

PPPM

Particle-Particle Particle-Mesh algorithm for long range electrostatic interactions. Use for example **rlist**=`1.0`, **rcoulomb_switch**=`0.0`, **rcoulomb**=`0.85`, **rvdw_switch**=`1.0` and **rvdw**=`1.0`. The grid dimensions are controlled by **fourierspacing**. Reasonable grid spacing for PPPM is 0.05-0.1 nm. See `Shift` for the details of the particle-particle potential.

NOTE: the pressure in incorrect when using PPPM.

Reaction-Field

Reaction field with Coulomb cut-off **rcoulomb**, where **rcoulomb** > **rvdw** > **rlist**. The dielectric constant beyond the cut-off is **epsilon_r**. The dielectric constant can be set to infinity by setting **epsilon_r**=0.

Generalized-Reaction-Field

Generalized reaction field with Coulomb cut-off **rcoulomb**, where **rcoulomb** > **rvdw** > **rlist**. The dielectric constant beyond the cut-off is **epsilon_r**. The ionic strength is computed from the number of charged (i.e. with non zero charge) charge groups. The temperature for the GRF potential is set with **ref_t** [K].

Shift

The Coulomb potential is decreased over the whole range and the forces decay smoothly to zero between **rcoulomb_switch** and **rcoulomb**. The neighbor search cut-off **rlist** should be 0.1 to 0.3 nm larger than **rcoulomb** to accommodate for the size of charge groups and diffusion between neighbor list updates.

User

Specify **rshort** and **rlong** to the same value, `mdrun` will now expect to find a file `ctab.xvg` with user-defined functions. This file should contain 5 columns: the `x` value, `f(x)`, $\texttt{-f}^{(1)}\texttt{(x)}$, $\texttt{f}^{(2)}\texttt{(x)}$ and $\texttt{-f}^{(3)}\texttt{(x)}$, where $\texttt{f}^{(n)}\texttt{(x)}$ denotes the $\texttt{n}^{th}$ derivative of function `f(x)` with respect to `x`. The `x` should run from 0 [nm] to **rlist**+`0.5` [nm], with a spacing of `0.002` [nm] when you run in single precision, or `0.0005` [nm] when you run in double precision. The function value at `x=0` is not important.

rcoulomb_switch: (0) [nm]

where to start switching the Coulomb potential

rcoulomb: (1) [nm]

distance for the Coulomb cut-off

epsilon_r: (1)

dielectric constant

xtc_grps:

group(s) to write to xtc trajectory, default the whole system is written (if **nstxtcout** is larger than zero)

energygrps:

group(s) to write to energy file

7.3.8 Neighbor searching

nstlist: (10) [steps]

Frequency to update the neighbor list (and the long-range forces, when using twin-range cut-off's). When this is 0, the neighbor list is made only once.

ns_type:

grid

Make a grid in the box and only check atoms in neighboring grid cells when constructing a new neighbor list every **nstlist** steps. In large systems grid search is much faster than simple search.

simple

Check every atom in the box when constructing a new neighbor list every **nstlist** steps.

pbc:

xyz

Use periodic boundary conditions in all directions.

no

Use no periodic boundary conditions, ignore the box. To simulate without cut-offs, set all cut-offs to 0 and **nstlist**=0.

rlist: (1) [nm]

cut-off distance for the short-range neighbor list

7.3.9 Electrostatics and VdW

coulombtype:

Cut-off

Twin range cut-off's with neighborlist cut-off **rlist** and Coulomb cut-off **rcoulomb**, where **rlist** $<$ **rvdw** $<$ **rcoulomb**. The dielectric constant is set with **epsilon_r**.

Ewald

Classical Ewald sum electrostatics. Use e.g. **rlist**=0.9, **rvdw**=0.9, **rcoulomb**=0.9. The highest magnitude of wave vectors used in reciprocal space is controlled by **fourierspacing**. The relative accuracy of direct/reciprocal space is controlled by **ewald_rtol**. NOTE: Ewald scales as O($N^{3/2}$) and is thus extremely slow for large systems. It is included mainly for reference - in most cases PME will perform much better.

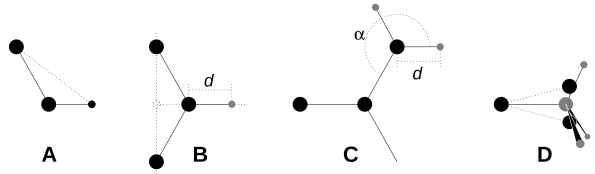


Figure 6.3: The different types of dummy atom constructions used for hydrogen atoms. The atoms used in the construction of the dummy atom(s) are depicted as black circles, dummy atoms as grey ones. Hydrogens are smaller than heavy atoms. A: fixed bond angle, note that here the hydrogen is not a dummy atom; B: in the plane of three atoms, with fixed distance; C: in the plane of three atoms, with fixed angle and distance; D: construction for amine groups (-NH_2 or -NH_3^+), see text for details.

motions, is to increase the mass of the hydrogen atoms at the expense of the mass of the connected heavy atom. This will increase the moment of inertia of the water molecules and the hydroxyl, sulfhydryl or amine groups, without affecting the equilibrium properties of the system and without affecting the dynamical properties too much. These constructions will shortly be described in sec. 6.2.1 and have previously been described in full detail [63].

Using both dummy atoms and modified masses, the next bottleneck is likely to be formed by the improper dihedrals (which are used to preserve planarity or chirality of molecular groups) and the peptide dihedrals. The peptide dihedral cannot be changed without affecting the physical behavior of the protein. The improper dihedrals that preserve planarity, mostly deal with aromatic residues. Bonds, angles and dihedrals in these residues can also be replaced with somewhat elaborate dummy atom constructions, as will be described in sec. 6.2.2 [64].

All modifications described in this section can be performed using the GROMACS topology building tool `pdb2gmx`. Separate options exist to increase hydrogen masses, dummify all hydrogen atoms or also dummify all aromatic residues. Note that when all hydrogen atoms are dummified, also those inside the aromatic residues will be dummified, i.e. hydrogens in the aromatic residues are treated differently depending on the treatment of the aromatic residues.

Parameters for the dummy constructions for the hydrogen atoms are inferred from the forcefield parameters (*vis.* bond lengths and angles) directly by `grompp` while processing the topology file. The constructions for the aromatic residues are based on the bond lengths and angles for the geometry as described in the forcefields, but these parameters are hard-coded into `pdb2gmx` due to the complex nature of the construction needed for a whole aromatic group.

6.2.1 Hydrogen bond-angle vibrations

Construction of Dummy Atoms

The goal of defining hydrogen atoms as dummy atoms is to remove all high-frequency degrees of freedom from them. In some cases not all degrees of freedom of a hydrogen atom should be removed, e.g. in the case of hydroxyl or amine groups the rotational freedom of the hydrogen atom(s) should be preserved. Care should be taken that no unwanted correlations are introduced by the construction of dummy atoms, e.g. bond-angle vibration between the constructing atoms could translate into hydrogen bond-length vibration. Additionally, since dummy atoms are by definition mass-less, in order to preserve total system mass, the mass of each hydrogen atom that is treated as dummy atom should be added to the bonded heavy atom.

Taking into account these considerations, the hydrogen atoms in a protein naturally fall into several categories, each requiring a different approach, see also Fig. 6.3:

- *hydroxyl (*-OH*) or sulfhydryl (*-SH*) hydrogen:* The only internal degree of freedom in a hydroxyl group that can be constrained is the bending of the C-O-H angle. This angle is fixed by defining an additional bond of appropriate length, see Fig. 6.3A. This removes the high frequency angle bending, but leaves the dihedral rotational freedom. The same goes for a sulfhydryl group. Note that in these cases the hydrogen is not treated as a dummy atom.
- *single amine or amide (*-NH-*) and aromatic hydrogens (*-CH-*):* The position of these hydrogens cannot be constructed from a linear combination of bond vectors, because of the flexibility of the angle between the heavy atoms. In stead, the hydrogen atom is positioned at a fixed distance from the bonded heavy atom on a line going through the bonded heavy atom and a point on the line through both second bonded atoms, see Fig. 6.3B.
- *planar amine (*-NH_2*) hydrogens:* The method used for the single amide hydrogen is not well suited for planar amine groups, because no suitable two heavy atoms can be found to define the direction of the hydrogen atoms. In stead, the hydrogen is constructed at a fixed distance from the nitrogen atom, with a fixed angle to the carbon atom, in the plane defined by one of the other heavy atoms, see Fig. 6.3C.
- *amine group (umbrella* -NH_2 *or* -NH_3^+*) hydrogens:* Amine hydrogens with rotational freedom cannot be constructed as dummy atoms from the heavy atoms they are connected to, since this would result in loss of the rotational freedom of the amine group. To preserve the rotational freedom while removing the hydrogen bond-angle degrees of freedom, two "dummy masses" are constructed with the same total mass, moment of inertia (for rotation around the C-N bond) and center of mass as the amine group. These dummy masses have no interaction with any other atom, except for the fact that they are connected to the carbon and to each other, resulting in a rigid triangle. From these three particles the positions of the nitrogen and hydrogen atoms are constructed as linear combinations of the two carbon-mass vectors and their outer product, resulting in an amine group with rotational freedom intact, but without other internal degrees of freedom. See Fig. 6.3D.

6.2.2 Out-of-plane vibrations in aromatic groups

The planar arrangements in the side chains of the aromatic residues lends itself perfectly for a dummy-atom construction, giving a perfectly planar group without the inherently instable constraints that are necessary to keep normal atoms in a plane. The basic approach is to define three atoms or dummy masses with constraints between them to fix the geometry and create the rest of the atoms as simple dummy type 3 atoms (see sec. 4.5) from these three. Each of the aromatic residues require a different approach:

7.3.5 Energy minimization

emtol: (100.0) [kJ mol^{-1} nm^{-1}]
the minimization is converged when the maximum force is smaller than this value

emstep: (0.01) [nm]
initial step-size

nstcgsteep: (1000) [steps]
frequency of performing 1 steepest descent step while doing conjugate gradient energy minimization.

7.3.6 Shell Molecular Dynamics

The shell molecular dynamics program **xmdrun** optimizes the positions of the shells at every time step until either the RMS force on the shells is less than emtol, or a maximum number of iterations (niter) has been reached

emtol: (100.0) [kJ mol^{-1} nm^{-1}]
the minimization is converged when the maximum force is smaller than this value. For shell MD this value should be 1.0 at most, but since the variable is used for energy minimization as well the default is 100.0.

niter: (20)
maximum number of iterations for optimizing the shell positions.

7.3.7 Output control

nstxout: (100) [steps]
frequency to write coordinates to output trajectory file, the last coordinates are always written

nstvout: (100) [steps]
frequency to write velocities to output trajectory, the last velocities are always written

nstfout: (0) [steps]
frequency to write forces to output trajectory.

nstlog: (100) [steps]
frequency to write energies to log file, the last energies are always written

nstenergy: (100) [steps]
frequency to write energies to energy file, the last energies are always written

nstxtcout: (0) [steps]
frequency to write coordinates to xtc trajectory

xtc_precision: (1000) [real]
precision to write to xtc trajectory

sd

A leap-frog stochastic dynamics integrator. The temperature for one or more groups of atoms (**tc_grps**) is set with **ref_t** [K], the inverse friction constant for each group is set with **tau_t** [ps]. The parameter **tcoupl** is ignored. The random generator is initialized with **ld_seed**.

NOTE: temperature deviations decay twice as fast as with a Berendsen thermostat with the same **tau_t**.

bd

An Euler integrator for Brownian or position Langevin dynamics, the velocity is the force divided by a friction coefficient (**bd_fric** [amu ps^{-1}]) plus random thermal noise (**bd_temp** [K]). When **bd_fric**=0, the friction coefficient for each particle is calculated as mass/**tau_t**, as for the integrator `sd`. The random generator is initialized with **ld_seed**.

tinit: (0) [ps]

starting time for your run (only makes sense for integrators `md`, `sd` and `bd`)

dt: (0.001) [ps]

time step for integration (only makes sense for integrators `md`, `sd` and `bd`)

nsteps: (1)

maximum number of steps to integrate

nstcomm: (1) [steps]

if positive: frequency for center of mass motion removal if negative: frequency for center of mass motion and rotational motion removal (should only be used for vacuum simulations)

comm_grps:

group(s) for center of mass motion removal, default is the whole system, rotation removal can only be done on the whole system

7.3.4 Langevin dynamics

bd_temp: (300) [K]

temperature in Brownian dynamics run (controls thermal noise level). When **bd_fric**=0, **ref_t** is used instead.

bd_fric: (0) [amu ps^{-1}]

Brownian dynamics friction coefficient. When **bd_fric**=0, the friction coefficient for each particle is calculated as mass/**tau_t**.

ld_seed: (1993) [integer]

used to initialize random generator for thermal noise for stochastic and Brownian dynamics. When **ld_seed** is set to -1, the seed is calculated as `(time() + getpid()) % 1000000`

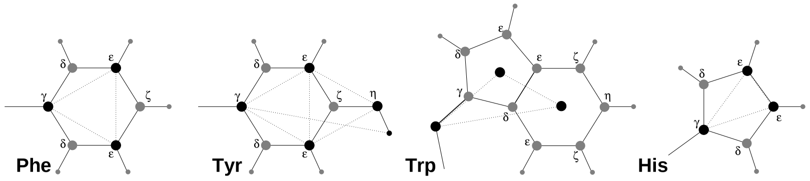


Figure 6.4: The different types of dummy atom constructions used for aromatic residues. The atoms used in the construction of the dummy atom(s) are depicted as black circles, dummy atoms as grey ones. Hydrogens are smaller than heavy atoms. A: phenylalanine; B: tyrosine (note that the hydroxyl hydrogen is *not* a dummy atom); C: tryptophane; D: histidine.

- *Phenylalanine:* C_γ, $C_{\epsilon1}$ and $C_{\epsilon2}$ are kept as normal atoms, but with each a mass of one third the total mass of the phenyl group. See Fig. 6.3A.
- *Tyrosine:* The ring is treated identical to the phenylalanine ring. Additionally, constraints are defined between $C_{\epsilon1}$ and $C_{\epsilon2}$ and O_η. The original improper dihedral angles will keep both triangles (one for the ring and one with O_η) in a plane, but due to the larger moments of inertia this construction will be much more stable. The bond angle in the hydroxyl group will be constrained by a constraint between C_γ and H_η, note that the hydrogen is not treated as a dummy atom. See Fig. 6.3B.
- *Tryptophane:* C_β is kept as a normal atom and two dummy masses are created at the center of mass of each of the rings, each with a mass equal to the total mass of the respective ring ($C_{\delta2}$ and $C_{\epsilon2}$ are each counted half for each ring). This keeps the overall center of mass and the moment of inertia almost (but not quite) equal to what it was. See Fig. 6.3C.
- *Histidine:* C_γ, $C_{\epsilon1}$ and $N_{\epsilon2}$ are kept as normal atoms, but with masses redistributed such that the center of mass of the ring is preserved. See Fig. 6.3D.

6.3 Viscosity calculation

The shear viscosity is a property of liquid which can be determined easily by experiment. It is useful for parameterizing the forcefield, because it is a kinetic property, while most other properties which are used for parameterization are thermodynamic. The viscosity is also an important property, since it is of influence on the rates of conformational changes of molecules solvated in the liquid.

The viscosity can be calculated from an equilibrium simulation using an Einstein relation:

$$\eta = \frac{1}{2}\frac{V}{k_B T}\lim_{t\to\infty}\frac{\mathrm{d}}{\mathrm{d}t}\left\langle\left(\int_{t_0}^{t_0+t} P_{xz}(t')\mathrm{d}t'\right)^2\right\rangle_{t_0} \tag{6.1}$$

This can be done with `g_energy`. This method converges very slowly. A nanosecond simulation might not be long enough for an accurate determinination of the viscoity. The result is very dependent on the treatment of the electrostatics. Using a (short) cut-off results in large noise on the off-diagonal pressure elements, which can increase the calculated viscosity by an order of magnitude.

GROMACS also has a non-equilibrium method for determining the viscosity. This makes use of the fact that energy, which is fed into system by external forces, is dissipated through viscous friction. The generated heat is removed by coupling to a heat bath. For a Newtonian liquid adding a small force will result in a velocity gradient according to the following equation:

$$a_x(z) + \frac{\eta}{\rho}\frac{\partial^2 v_x(z)}{\partial z^2} = 0 \tag{6.2}$$

here we have applied an acceleration $a_x(z)$ in the x-direction, which is a function of the z-coordinate. In GROMACS the acceleration profile is:

$$a_x(z) = A\cos\left(\frac{2\pi z}{l_z}\right) \tag{6.3}$$

where l_z is the height of the box. The generated velocity profile is:

$$v_x(z) = V\cos\left(\frac{2\pi z}{l_z}\right) \tag{6.4}$$

$$V = A\frac{\rho}{\eta}\left(\frac{l_z}{2\pi}\right)^2 \tag{6.5}$$

The viscosity can be calculated from A and V:

$$\eta = \frac{A}{V}\rho\left(\frac{l_z}{2\pi}\right)^2 \tag{6.6}$$

In the simulation V is defined as:

$$V = \frac{\displaystyle\sum_{i=1}^{N} m_i v_{i,x} 2\cos\left(\frac{2\pi z}{l_z}\right)}{\displaystyle\sum_{i=1}^{N} m_i} \tag{6.7}$$

The generated velocity profile is not coupled to the heat bath, also the velocity profile is excluded from the kinetic energy. One would like V to be as large as possible to get good statistics. However the shear rate should not be so high that the system gets too far from equilibrium. The maximum shear rate occurs where the cosine is zero, the rate is:

$$\mathrm{sh}_{\max} = \max_z \left|\frac{\partial v_x(z)}{\partial z}\right| = A\frac{\rho}{\eta}\frac{l_z}{2\pi} \tag{6.8}$$

For a simulation with: $\eta = 10^{-3}$ [kg m^{-1} s^{-1}], $\rho = 10^3$ [kg m^{-3}] and $l_z = 2\pi$ [nm], $\mathrm{sh}_{\max} = 1$ [ps nm^{-1}] A. This shear rate should be smaller than one over the longest correlation time in the system. For most liquids this will be the rotation correlation time, which is around 10 picoseconds. In this case A should be smaller than 0.1 [nm ps^{-2}]. When the shear rate is too high, the observed viscosity will be too low. Because V is proportional to the square of the box height, the optimal

---

7.3 Run Parameters

7.3.1 General

Default values are given in parentheses. The first option is always the default option. Units are given in square brackets The difference between a dash and an underscore is ignored. A sample `.mdp` file is available. This should be appropriate to start a normal simulation. Edit it to suit your specific needs and desires.

7.3.2 Preprocessing

title:
this is redundant, so you can type anything you want

cpp: (/lib/cpp)
your preprocessor

include:
directories to include in your topology. format:
`-I/home/john/my_lib -I../more_lib`

define: ()
defines to pass to the preprocessor, default is no defines. You can use any defines to control options in your customized topology files. Options that are already available by default are:

-DFLEX_SPC
Will tell grompp to include FLEX_SPC in stead of SPC into your topology, this is necessary to make **conjugate gradient** work and will allow **steepest descent** to minimize further.

-DPOSRE
Will tell grompp to include posre.itp into your topology, used for position restraints.

7.3.3 Run control

integrator:

md
A leap-frog algorithm for integrating Newton's equations.

steep
A steepest descent algorithm for energy minimization. The maximum step size is **emstep** [nm], the tolerance is **emtol** [kJ mol^{-1} nm^{-1}].

cg
A conjugate gradient algorithm for energy minimization, the tolerance is **emtol** [kJ mol^{-1} nm^{-1}]. CG is more efficient when a steepest descent step is done every once in a while, this is determined by **nstcgsteep**.

Default Name Ext.	Type	Default Option	Description
`atomtp.atp`	Asc		Atomtype file used by pdb2gmx
`eiwit.brk`	Asc	`-f`	Brookhaven data bank file
`nnnice.dat`	Asc		Generic data file
`user.dlg`	Asc		Dialog Box data for ngmx
`sam.edi`	Asc		ED sampling input
`sam.edo`	Asc		ED sampling output
`ener.edr`			Generic energy: `edr ene`
`ener.edr`	xdr		Energy file in portable xdr format
`ener.ene`	Bin		Energy file
`eiwit.ent`	Asc	`-f`	Entry in the protein date bank
`plot.eps`	Asc		Encapsulated PostScript (tm) file
`gtraj.g87`	Asc		Gromos-87 ASCII trajectory format
`conf.g96`	Asc	`-c`	Coordinate file in Gromos-96 format
`conf.gro`		`-c`	Generic structure: `gro g96 pdb tpr tpb tpa`
`out.gro`		`-o`	Generic structure: `gro g96 pdb`
`conf.gro`	Asc	`-c`	Coordinate file in Gromos-87 format
`polar.hdb`	Asc		Hydrogen data base
`topinc.itp`	Asc		Include file for topology
`run.log`	Asc	`-l`	Log file
`ps.m2p`	Asc		Input file for mat2ps
`ss.map`	Asc		File that maps matrix data to colors
`ss.mat`	Asc		Matrix Data file
`grompp.mdp`	Asc	`-f`	grompp input file with MD parameters
`hessian.mtx`	Bin	`-m`	Hessian matrix
`index.ndx`	Asc	`-n`	Index file
`hello.out`	Asc	`-o`	Generic output file
`eiwit.pdb`	Asc	`-f`	Protein data bank file
`pull.pdo`	Asc		Pull data output
`pull.ppa`	Asc		Pull parameters
`residue.rtp`	Asc		Residue Type file used by pdb2gmx
`doc.tex`	Asc	`-o`	LaTeX file
`topol.top`	Asc	`-p`	Topology file
`topol.tpa`	Asc	`-s`	Ascii run input file
`topol.tpb`	Bin	`-s`	Binary run input file
`topol.tpr`		`-s`	Generic run input: `tpr tpb tpa`
`topol.tpr`		`-s`	Structure+mass(db): `tpr tpb tpa gro g96 pdb`
`topol.tpr`	xdr	`-s`	Portable xdr run input file
`traj.trj`	Bin		Trajectory file (cpu specific)
`traj.trr`			Full precision trajectory: `trr trj`
`traj.trr`	xdr		Trajectory in portable xdr format
`root.xpm`	Asc		X PixMap compatible matrix file
`traj.xtc`		`-f`	Generic trajectory: `xtc trr trj gro g96 pdb`
`traj.xtc`	xdr		Compressed trajectory (portable xdr format)
`graph.xvg`	Asc	`-o`	xvgr/xmgr file

Table 7.1: The GROMACS file types and how they can be used for input/output.

box is elongated in the z-direction. In general a simulation length of 100 picoseconds is enough to obtain an accurate value for the viscosity.

The heat generated by the viscous friction is removed by coupling to a heat bath. Because this coupling is not instantaneous the real temperature of the liquid will be slightly lower than the observed temperature. Berendsen derived this temperature shift[20], which can be written in terms of the shear rate as:

$$T_s = \frac{\eta\,\tau}{2 \rho\,C_v} \mathrm{sh}_{\mathrm{max}}^2 \tag{6.9}$$

where τ is the coupling time for the Berendsen thermostat and C_v is the heat capacity. Using the values of the example above, $\tau = 10^{-13}$ [s] and $C_v = 2 \cdot 10^3$ [J kg^{-1} K^{-1}], we get: $T_s = 25$ [K ps^{-2}] $\mathrm{sh}_{\mathrm{max}}^2$. When we want the shear rate to be smaller than $1/10$ [ps^{-1}], T_s is smaller than 0.25 [K], which is negligible.

Note that the system has to build up the velocity profile when starting from an equilibrium state. This build-up time is of the order of the correlation time of the liquid.

Two quantities are written to the energy file, along with their averages and fluctuations: V and $1/\eta$ as obtained from (6.6).

6.4 User specified potential functions

You can also use your own potential functions without editing the GROMACS code. The potential function should be according to the following equation

$$V(r_{ij}) \;=\; \frac{q_i q_j}{4 \pi \epsilon_0} f(r_{ij}) + C_6 g(r_{ij}) + C_{12} h(r_{ij}) \tag{6.10}$$

with f,g,h user defined functions. Note that if g(r) represents a normal dispersion interaction, g(r) should be < 0. C_6, C_{12} and the charges are read from the topology. Also note that combination rules are only supported for Lennard Jones and Buckingham, and that your tables should match the parameters in the binary topology.

When you add the following lines in your `.mdp` file:

```
rlist           = 1.0
coulombtype     = User
rcoulomb        = 1.0
vdwtype         = User
rvdw            = 1.0
```

the MD program will read a single file (name can be changed with option `-table`) with seven columns of table lookup data in the order: x, f(x), f”(x), g(x), g”(x), h(x), h”(x). The x should run from 0 to r_c+0.5, with a spacing of 0.002 nm when you run in single precision, or 0.0005 when you run in double precision. In this context r_c denotes the maximum of the two cut-offs `rvdw` and `rcoulomb` (see above). These variables need not be the same (and need not be 1.0 either). Some functions used for potentials contain a singularity at x = 0, but since atoms are normally not closer to each other than 0.1 nm, the function value at x = 0 is not important and the tables can be started at e.g. 0.02 nm. Finally, it is also possible to combine a standard Coulomb with a modified LJ potential (or vice versa). One then specifies e.g. coulombtype = Cut-off or coulombtype = PME, combined with vdwtype = User. The table file must always contain the 7 columns however, and meaningful data (i.e. not zeroes) must be entered in all columns. A number of pre-built table files can be found in the GMXLIB directory, for 6-8, 6-9, 6-10, 6-11, 6-12 Lennard Jones potentials combined with a normal Coulomb.

6.5 Running GROMACS in parallel

If you have installed the MPI (Message Passing Interface) on your computer(s) you can compile GROMACS with this library to run simulations in parallel. All supercomputers are shipped with MPI libraries optimized for that particular platform, and if you are using a cluster of workstations there are several good free MPI implementations. You can find updated links to these on the gromacs homepage www.gromacs.org. Once you have an MPI library installed it's trivial to compile GROMACS with MPI support: Just set the option `--enable-mpi` to the configure script and recompile. (But don't forget to make distclean before running configure if you have previously compiled with a different configuration.) If you are using a supercomputer you might also want to turn of the default nicing of the mdrun process with the `--disable-nice` option.

There is usually a program called `mpirun` with which you can fire up the parallel processes. A typical command line looks like:

```
% mpirun -p goofus,doofus,fred 10 mdrun -s topol -v -N 30
```

this runs on each of the machines goofus,doofus,fred with 10 processes on each[1].

If you have a single machine with multiple processors you don't have to use the `mpirun` command, but you can do with an extra option to `mdrun`:

```
% mdrun -np 8 -s topol -v -N 8
```

In this example MPI reads the first option from the command line. Since `mdrun` also wants to know the number of processes you have to type it twice.

Check your local manuals (or online manual) for exact details of your MPI implementation.

If you are interested in programming MPI yourself, you can find manuals and reference litterature on the web at www.mcs.anl.gov/mpi/index.html.

---

[1] Example taken from Silicon Graphics manual

Chapter 7

Run parameters and Programs

7.1 Online and html manuals

All the information in this chapter can also be found in HTML format in your GROMACS data directory. The path depends on where your files are installed, but the default location is

`/usr/local/gromacs/share/html/online.html`

Or, if you installed from Linux packages it can be found as

`/usr/local/share/gromacs/html/online.html`

You can also use the online from our web site,

www.gromacs.org/documentation/reference_3.0/online.html

In addition, we install standard UNIX manuals for all the programs. If you have sourced the `GMXRC` script in the GROMACS binary directory for your host they should already be present in your `$MANPATH`, and you should be able to type e.g. `man grompp`.

The program manual pages can also be found in Appendix E in this manual.

7.2 File types

Table 7.1 lists the file types used by GROMACS along with a short description, and you can find a more detail description for each file in your HTML reference, or in our online version.

GROMACS files written in xdr format can be read on any architecture with GROMACS version 1.6 or later if the configuration script found the XDR libraries on your system. They should always be present on UNIX since they are necessary for the network file system support.