GROMACS

Groningen Machine for Chemical Simulations

USER MANUAL

Version 5.0

GROMACS
USER MANUAL

Version 5.0

Contributions from

Emile Apol, Rossen Apostolov, Herman J.C. Berendsen,
Aldert van Buuren, Pär Bjelkmar, Rudi van Drunen,
Anton Feenstra, Sebastian Fritsch, Gerrit Groenhof,
Christoph Junghans, Jochen Hub, Peter Kasson,
Carsten Kutzner, Brad Lambeth, Per Larsson,
Justin A. Lemkul, Erik Marklund, Peiter Meulenhoff,
Teemu Murtola, Szilárd Páll, Sander Pronk,
Roland Schulz, Michael Shirts, Alfons Sijbers,
Peter Tieleman, Christian Wennberg and Maarten Wolf.

Mark Abraham, Berk Hess, David van der Spoel, and Erik Lindahl.

More information can be found on our website: www.gromacs.org.

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 worked on continuously, which in some cases might mean the information is not entirely correct.

Comments on form and content are welcome, please send them to one of the mailing lists (see www.gromacs.org), or open an issue at redmine.gromacs.org. Corrections can also be made in the GROMACS git source repository and uploaded to gerrit.gromacs.org.

We 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.

On-line Resources

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

Citation information

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

> M.J. Abraham, D. van der Spoel, E. Lindahl, B. Hess, and the GROMACS development team, *GROMACS User Manual version 5.0*, www.gromacs.org (2014)

However, we prefer that you cite (some of) the GROMACS papers [1, 2, 3, 4, 5, 6] when you publish your results. Any future development depends on academic research grants, since the package is distributed as free software!

GROMACS is *Free Software*

The GROMACS source code and and selected set of binary packages are available on our homepage, www.gromacs.org. Have fun.

Contents

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 goal being 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 by (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 of 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 necessary; the higher the complexity of a system and the longer the time span of the processes of interest is, the more severe the required approximations are. 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, 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 [7], to *mesoscopic dynamics* describing densities rather than atoms and fluxes as response to thermodynamic gradients rather than velocities or accelerations as response to forces [8].

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

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$\cdots$O	libration	400– 700
O-H$\cdots$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} may misbehave 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 one 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: [9]

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

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

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 equations of motion. The rationale 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 [10]. GROMACS has this option for the bonds and bond angles. The flexibility of the latter is rather essential to allow for the realistic motion and coverage of configurational space [11].

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 instantly 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 modified by the user 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 biologically-relevant macromolecules 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

and sometimes for the Coulomb interactions as well. The "minimum-image convention" used by GROMACS requires that only one image of each particle in the periodic boundary conditions is considered for a pair interaction, so the cut-off radius cannot 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 truly bad things can 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, such as particle-mesh Ewald [12, 13].

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). We must avoid this condition if we wish to simulate a bulk system. As such, we use periodic boundary conditions to avoid real phase boundaries. Since liquids are not crystals, something unnatural remains. This item is mentioned last because it is the least of the evils. 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 of, 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 number of methods for local energy minimization, as detailed in sec. 3.10.

The potential energy function of a (macro)molecular system is a very complex landscape (or *hypersurface*) 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 non-negative. The matrix of second derivatives, which is called the *Hessian matrix*, has non-negative 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 in any practical amount of time. Impractical methods exist, some much faster than others [14]. 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 imagination 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 [15]), 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 [16], 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 that build 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.* [17]) 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. GROMACS also supports the L-BFGS minimizer, which is mostly comparable to *conjugate gradient method*, but in some cases converges faster.

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 version 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

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

Quantity	Symbol	Unit
length	r	nm $= 10^{-9}$ m
mass	m	u (atomic mass unit) $= 1.6605402(10)\times10^{-27}$ kg
		($1/12$ the mass of a ^{12}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^{-1}
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^{-1}

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 strongly recommend following the usual practice of including 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 kcal mol^{-1} (= 4.184 kJ mol^{-1}) is used instead of kJ mol^{-1} 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^{-1}, 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}.

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^{-1}

Table 2.3: Some Physical Constants

Quantity	Symbol	Relation to SI
Length	r*	r σ^{-1}
Mass	m*	m M^{-1}
Time	t*	t σ^{-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. The one exception is the *temperature*, which is expressed in $0.008\,314\,51$ reduced units. This is a consequence of using 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 $0.008\ldots$ 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:

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

Chapter 3

Algorithms

3.1 Introduction

In this chapter we first give describe some 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.

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 the system is crystalline, 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, such 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.

There are several possible shapes for space-filling unit cells. Some, like the *rhombic dodecahedron* and the *truncated octahedron* [18] are closer to being a sphere than a cube is, and are therefore

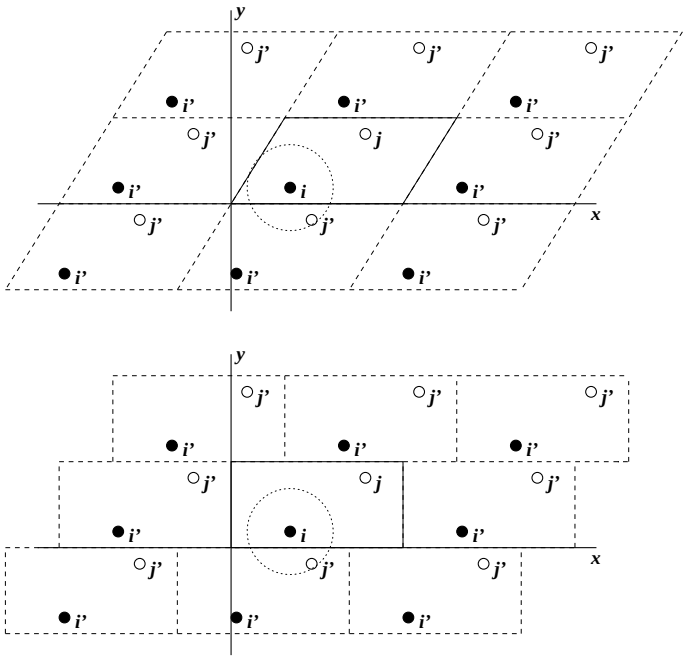


Figure 3.1: Periodic boundary conditions in two dimensions.

better suited to the study of an approximately spherical macromolecule in solution, since fewer solvent molecules are required to fill the box given a minimum distance between macromolecular images. At the same time, rhombic dodecahedra and truncated octahedra are special cases of *triclinic* unit cells; the most general space-filling unit cells that comprise all possible space-filling shapes [19]. For this reason, 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 such as Ewald Sum, PME and PPPM.

GROMACS supports triclinic boxes of any shape. The simulation box (unit cell) 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 satisfied by rotating the box. Inequalities (3.2) and (3.3) can always be satisfied by adding and subtracting box vectors.

Even when simulating using a triclinic box, GROMACS always keeps the particles in a brick-shaped volume for efficiency, as illustrated in Fig. 3.1 for a 2-dimensional system. Therefore, from the output trajectory it might seem that the simulation was done in a rectangular box. The program `trjconv` can be used to convert the trajectory to a different unit-cell representation.

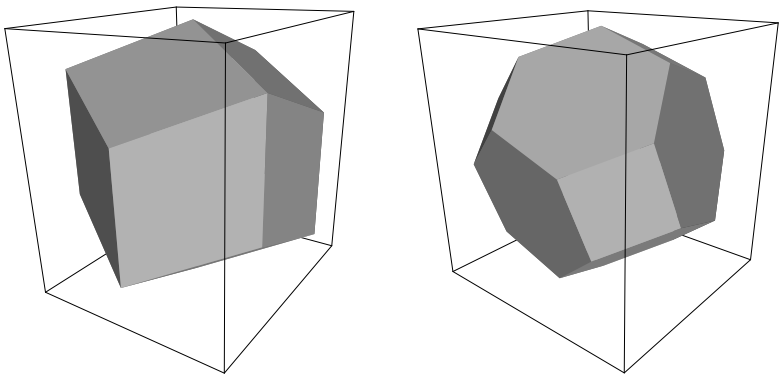


Figure 3.2: A rhombic dodecahedron and truncated octahedron (arbitrary orientations).

box type	image distance	box volume	box vectors			box vector angles		
			a	**b**	**c**	$\angle\mathbf{bc}$	$\angle\mathbf{ac}$	$\angle\mathbf{ab}$
cubic	d	d^3	d 0 0	0 d 0	0 0 d	90°	90°	90°
rhombic dodecahedron (xy-square)	d	$\frac{1}{2}\sqrt{2}\,d^3$ $0.707\,d^3$	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	$\frac{1}{2}\sqrt{2}\,d^3$ $0.707\,d^3$	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	$\frac{4}{9}\sqrt{3}\,d^3$ $0.770\,d^3$	d 0 0	$\frac{1}{3}\,d$ $\frac{2}{3}\sqrt{2}\,d$ 0	$-\frac{1}{3}\,d$ $\frac{1}{3}\sqrt{2}\,d$ $\frac{1}{3}\sqrt{6}\,d$	71.53°	109.47°	71.53°

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

It is also possible to simulate without periodic boundary conditions, but it is usually 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 cube having the same image distance. This saves about 29% of CPU-time when simulating a spherical or flexible molecule in solvent. There are two different orientations of a rhombic dodecahedron that satisfy equations 3.1, 3.2 and 3.3. The program `editconf` produces the orientation which has a square intersection 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 not only saves 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 may not exceed half the shortest box vector:

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

because 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 alone is not sufficient: 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, however, 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 triclinic boxes is more restricted. For grid search the extra restriction is weak:

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

For simple search the extra restriction is stronger:

$$R_c < \frac{1}{2}\min(a_x, b_y, c_z) \tag{3.6}$$

Each unit cell (cubic, rectangular or triclinic) is surrounded by 26 translated images. A particular image can therefore 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). Restriction (3.5) ensures that only 26 images need to be considered.

3.3 The group concept

The GROMACS MD and analysis programs use user-defined *groups* of atoms to perform certain actions on. The maximum number of groups is 256, but each atom can only belong to six different groups, one each of the following:

temperature-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 generate more heating by force and integration errors) can be coupled with a shorter time constant to a bath than is a macromolecule, or a surface can be kept cooler than an adsorbing molecule. Many different T-coupling groups may be defined. See also center of mass groups below.

freeze group Atoms that belong to a freeze group are kept stationary in the dynamics. This is useful during equilibration, *e.g.* to avoid badly placed solvent molecules giving 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, if desired, on just one or two coordinates of an atom, thereby freezing the atoms in a plane or on a line. When an atom is partially frozen, constraints will still be able to move it, even in a frozen direction. A fully frozen atom can not be moved by constraints. Many freeze groups can be defined. Frozen coordinates are unaffected by pressure scaling; in some cases this can produce unwanted results, particularly when constraints are also used (in this case you will get very large pressures). Accordingly, it is recommended to avoid combining freeze groups with constraints and pressure coupling. For the sake of equilibration it could suffice to start with freezing in a constant volume simulation, and afterward use position restraints in conjunction with constant pressure.

accelerate group On each atom in an "accelerate group" an acceleration $\boldsymbol{a}^g$ is imposed. This is equivalent to an external force. This feature makes it possible to drive the system into a non-equilibrium state and enables the performance of non-equilibrium MD and hence 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). 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.

center of mass group In GROMACS the center of mass (COM) motion can be removed, for either the complete system or for groups of atoms. The latter is useful, *e.g.* for systems where there is limited friction (*e.g.* gas systems) to prevent center of mass motion to occur. It makes sense to use the same groups for temperature coupling and center of mass motion removal.

Compressed position output group In order to further reduce the size of the compressed trajectory file (`.xtc` or `.tng`), it is possible to store only a subset of all particles. All x-compression groups that are specified are saved, the rest are not. If no such groups are specified, than all atoms are saved to the compressed trajectory file.

The use of groups in GROMACS tools is described in sec. 8.1.

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.

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 for the 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 may be 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

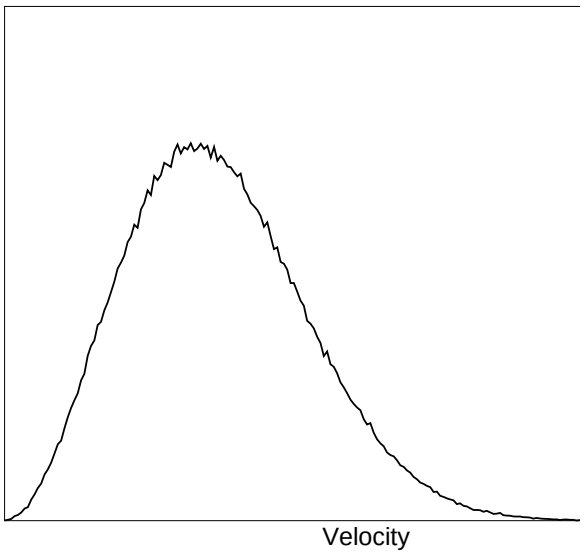


Figure 3.4: A Maxwell-Boltzmann velocity distribution, generated from random numbers.

3.4.1 Initial conditions

Topology and force field

The system topology, including a description of the force field, must be read in. Force fields and topologies are described in chapter 4 and 5, respectively. 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 and shape 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.

If the run starts at $t = t_0$, the coordinates at $t = t_0$ must be known. The *leap-frog algorithm*, the default algorithm used to update the time step with Δt (see 3.4.4), also requires that the velocities at $t = t_0 - \frac{1}{2}\Delta t$ are known. If velocities are not available, the program can generate initial atomic velocities $v_i, i = 1 \ldots 3N$ with a (Fig. 3.4) at a given absolute temperature T:

$$p(v_i) = \sqrt{\frac{m_i}{2\pi kT}} \exp\left(-\frac{m_i v_i^2}{2kT}\right) \tag{3.7}$$

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 $0 \leq R_k < 1$ 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 so that the total energy corresponds exactly to T (see eqn. 3.18).

Center-of-mass motion

The center-of-mass velocity is normally set to zero at every step; there is (usually) no net external force acting on the system and the center-of-mass velocity should remain constant. In practice, however, the update algorithm introduces a very slow change in the center-of-mass velocity, and therefore in the total kinetic energy of the system – especially when temperature coupling is used. If such changes are not quenched, an appreciable center-of-mass motion can develop in long runs, and the temperature will be significantly misinterpreted. Something similar 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 cause such problems.

3.4.2 Neighbor searching

As mentioned in chapter 4, internal forces are either generated from fixed (static) lists, or from dynamic lists. The latter consist of 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. Equation 3.1 ensures 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, 3.4 and 3.5 ensure that we can find the 14 nearest triclinic images within a linear combination that 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 atoms i, a displacement vector for atom i, and all particles j that are within `rlist` of this particular image of atom i. The list is updated every `nstlist` steps, where `nstlist` is typically 10. There is an option to calculate the total non-bonded force on each particle due to all particle in a shell around the list cut-off, *i.e.* at a distance between `rlist` and `rlistlong`. 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 neighbor list cut-off) to a given

particle must be found. This searching, usually called neighbor search (NS) or pair search, involves periodic boundary conditions and determining the *image* (see sec. 3.2). The search algorithm is $O(N)$, although a simpler $O(N^2)$ algorithm is still available under some conditions.

Cut-off schemes: group versus Verlet

From version 4.6, GROMACS supports two different cut-off scheme setups: the original one based on atom groups and one using a Verlet buffer. There are some important differences that affect results, performance and feature support. The group scheme can be made to work (almost) like the Verlet scheme, but this will lead to a decrease in performance. The group scheme is especially fast for water molecules, which are abundant in many simulations, but on the most recent x86 processors, this advantage is negated by the better instruction-level parallelism available in the Verlet-scheme implementation. The group scheme is deprecated in version 5.0, and will be removed in a future version.

In the group scheme, a neighbor list is generated consisting of pairs of groups of at least one atom. These groups were originally charge groups (see sec. 3.4.2), but with a proper treatment of long-range electrostatics, performance is their only advantage. A pair of groups is put into the neighbor list when their center of geometry is within the cut-off distance. Interactions between all atom pairs (one from each charge group) are calculated for a certain number of MD steps, until the neighbor list is updated. This setup is efficient, as the neighbor search only checks distance between charge group pair, not atom pairs (saves a factor of $3 \times 3 = 9$ with a three-atom water model) and the non-bonded force kernels can be optimized for, say, a water molecule "group". Without explicit buffering, this setup leads to energy drift as some atom pairs which are within the cut-off don't interact and some outside the cut-off do interact. This can be caused by

- atoms moving across the cut-off between neighbor search steps, and/or
- for charge groups consisting of more than one atom, atom pairs moving in/out of the cut-off when their charge group center of geometry distance is outside/inside of the cut-off.

Explicitly adding a buffer to the neighbor list will remove such artifacts, but this comes at a high computational cost. How severe the artifacts are depends on the system, the properties in which you are interested, and the cut-off setup.

The Verlet cut-off scheme uses a buffered pair list by default. It also uses clusters of atoms, but these are not static as in the group scheme. Rather, the clusters are defined spatially and consist of 4 or 8 atoms, which is convenient for stream computing, using e.g. SSE, AVX or CUDA on GPUs. At neighbor search steps, a pair list is created with a Verlet buffer, ie. the pair-list cut-off is larger than the interaction cut-off. In the non-bonded force kernels, forces are only added when an atom pair is within the cut-off distance at that particular time step. This ensures that as atoms move between pair search steps, forces between nearly all atoms within the cut-off distance are calculated. We say *nearly* all atoms, because GROMACS uses a fixed pair list update frequency for efficiency. An atom-pair, whose distance was outside the cut-off, could possibly move enough during this fixed number of steps that its distance is now within the cut-off. This small chance results in a small energy drift, and the size of the chance depends on the temperature. When temperature coupling is used, the buffer size can be determined automatically, given a certain tolerance on the energy drift.

The mdp file settings specific to the Verlet scheme are:

```
cutoff-scheme            = Verlet
verlet-buffer-tolerance = 0.005
```

The Verlet buffer size is determined from the latter option, which is by default set to 0.005 kJ/mol/ps pair energy error per atom. Note that errors in pair energies cancel and the effect on the total energy drift is usually at least an order of magnitude smaller than the tolerance. Furthermore, the drift of the total energy is affected by many other factors; often, the contribution from the constraint algorithm dominates.

For constant-energy (NVE) simulations, the buffer size will be inferred from the temperature that corresponds to the velocities (either those generated, if applicable, or those found in the input configuration). Alternatively, the tolerance can be set to -1 and a buffer set manually by specifying `rlist` > `max(rcoulomb, rvdw)`. The simplest way to get a reasonable buffer size is to use an NVT mdp file with the target temperature set to what you expect in your NVE simulation, and transfer the buffer size printed by `grompp` to your NVE mdp file.

The Verlet cut-off scheme is implemented in a very efficient fashion based on clusters of particles. The simplest example is a cluster size of 4 particles. The pair list is then constructed based on cluster pairs. The cluster-pair search is much faster searching based on particle pairs, because $4 \times 4 = 16$ particle pairs are put in the list at once. The non-bonded force calculation kernel can then calculate all 16 particle-pair interactions at once, which maps nicely to SIMD units which can perform multiple floating operations at once (e.g. SSE, AVX, CUDA on GPUs, BlueGene FPUs). These non-bonded kernels are much faster than the kernels used in the group scheme for most types of systems, except for water molecules on processors with short SIMD widths when not using a buffered pair list. This latter case is common for (bio-)molecular simulations, so for greatest speed, it is worth comparing the performance of both schemes.

As the Verlet cut-off scheme was introduced in version 4.6, not all features of the group scheme are supported yet. The Verlet scheme supports a few new features which the group scheme does not support. A list of features not (fully) supported in both cut-off schemes is given in Table 3.2.

Energy drift and pair-list buffering

For a canonical (NVT) ensemble, the average energy error caused by the finite Verlet buffer size can be determined from the atomic displacements and the shape of the potential at the cut-off. The displacement distribution along one dimension for a freely moving particle with mass m over time t at temperature T is Gaussian with zero mean and variance $\sigma^2 = t\,k_B T/m$. For the distance between two atoms, the variance changes to $\sigma^2 = \sigma_{12}^2 = t\,k_B T(1/m_1 + 1/m_2)$. Note that in practice particles usually interact with other particles over time t and therefore the real displacement distribution is much narrower. Given a non-bonded interaction cut-off distance of r_c and a pair-list cut-off $r_\ell = r_c + r_b$, we can then write the average energy error after time t for pair interactions between one particle of type 1 surrounded by particles of type 2 with number density ρ_2, when the inter particle distance changes from r_0 to r_t, as:

$$\langle \Delta V \rangle \;=\; \int_0^{r_c} \int_{r_\ell}^{\infty} 4\pi r_0^2 \rho_2 V(r_t) G\left(\frac{r_t - r_0}{\sigma}\right) dr_0\, dr_t \tag{3.8}$$

Non-bonded interaction feature	group	Verlet
unbuffered cut-off scheme	√	not by default
exact cut-off	shift/switch	√
shifted interactions	force+energy	energy
switched potential	√	√
switched forces	√	√
non-periodic systems	√	Z + walls
implicit solvent	√	
free energy perturbed non-bondeds	√	√
group energy contributions	√	CPU (not on GPU)
energy group exclusions	√	
AdResS multi-scale	√	
OpenMP multi-threading	only PME	√
native GPU support		√
Lennard-Jones PME	√	√

Table 3.2: Differences (only) in the support of non-bonded features between the group and Verlet cut-off schemes.

$$\approx \int_{-\infty}^{r_c}\int_{r_\ell}^{\infty} 4\pi r_0^2\rho_2\Big[V'(r_c)(r_t-r_c)+ V''(r_c)\frac{1}{2}(r_t-r_c)^2\Big]G\Big(\frac{r_t-r_0}{\sigma}\Big)\,dr_0\,dr_t \tag{3.9}$$

$$\approx 4\pi(r_\ell+\sigma)^2\rho_2\int_{-\infty}^{r_c}\int_{r_\ell}^{\infty}\Big[V'(r_c)(r_t-r_c)+ V''(r_c)\frac{1}{2}(r_t-r_c)^2+ V'''(r_c)\frac{1}{6}(r_t-r_c)^3\Big]G\Big(\frac{r_t-r_0}{\sigma}\Big)\,dr_0\,dr_t \tag{3.10}$$

$$= 4\pi(r_\ell+\sigma)^2\rho_2\Big\{\frac{1}{2}V'(r_c)\Big[r_b\sigma G\Big(\frac{r_b}{\sigma}\Big)-(r_b^2+\sigma^2)E\Big(\frac{r_b}{\sigma}\Big)\Big]+ \frac{1}{6}V''(r_c)\Big[\sigma(r_b^2+2\sigma^2)G\Big(\frac{r_b}{\sigma}\Big)-r_b(r_b^2+3\sigma^2)E\Big(\frac{r_b}{\sigma}\Big)\Big]+ \frac{1}{24}V'''(r_c)\Big[r_b\sigma(r_b^2+5\sigma^2)G\Big(\frac{r_b}{\sigma}\Big)-(r_b^4+6r_b^2\sigma^2+3\sigma^4)E\Big(\frac{r_b}{\sigma}\Big)\Big]\Big\} \tag{3.11}$$

where G is a Gaussian distribution with 0 mean and unit variance and $E(x) = \frac{1}{2}\mathrm{erfc}(x/\sqrt{2})$. We always want to achieve small energy error, so σ will be small compared to both r_c and r_ℓ, thus the approximations in the equations above are good, since the Gaussian distribution decays rapidly. The energy error needs to be averaged over all particle pair types and weighted with the particle counts. In GROMACS we don't allow cancellation of error between pair types, so we average the absolute values. To obtain the average energy error per unit time, it needs to be divided by the neighbor-list life time $t = (\texttt{nstlist}-1)\times\texttt{dt}$. This function can not be inverted analytically, so we use bisection to obtain the buffer size r_b for a target drift. Again we note that in practice the error we usually be much smaller than this estimate, as in the condensed phase particle

displacements will be much smaller than for freely moving particles, which is the assumption used here.

When (bond) constraints are present, some particles will have fewer degrees of freedom. This will reduce the energy errors. The displacement in an arbitrary direction of a particle with 2 degrees of freedom is not Gaussian, but rather follows the complementary error function:

$$\frac{\sqrt{\pi}}{2\sqrt{2}\sigma}\,\mathrm{erfc}\left(\frac{|r|}{\sqrt{2}\,\sigma}\right) \tag{3.12}$$

where σ^2 is again $k_B T/m$. This distribution can no longer be integrated analytically to obtain the energy error. But we can generate a tight upper bound using a scaled and shifted Gaussian distribution (not shown). This Gaussian distribution can then be used to calculate the energy error as described above. We consider particles constrained, i.e. having 2 degrees of freedom or fewer, when they are connected by constraints to particles with a total mass of at least 1.5 times the mass of the particles itself. For a particle with a single constraint this would give a total mass along the constraint direction of at least 2.5, which leads to a reduction in the variance of the displacement along that direction by at least a factor of 6.25. As the Gaussian distribution decays very rapidly, this effectively removes one degree of freedom from the displacement. Multiple constraints would reduce the displacement even more, but as this gets very complex, we consider those as particles with 2 degrees of freedom.

There is one important implementation detail that reduces the energy errors caused by the finite Verlet buffer list size. The derivation above assumes a particle pair-list. However, the GROMACS implementation uses a cluster pair-list for efficiency. The pair list consists of pairs of clusters of 4 particles in most cases, also called a 4×4 list, but the list can also be 4×8 (GPU CUDA kernels and AVX 256-bit single precision kernels) or 4×2 (SSE double-precision kernels). This means that the pair-list is effectively much larger than the corresponding 1×1 list. Thus slightly beyond the pair-list cut-off there will still be a large fraction of particle pairs present in the list. This fraction can be determined in a simulation and accurately estimated under some reasonable assumptions. The fraction decreases with increasing pair-list range, meaning that a smaller buffer can be used. For typical all-atom simulations with a cut-off of 0.9 nm this fraction is around 0.9, which gives a reduction in the energy errors of a factor of 10. This reduction is taken into account during the automatic Verlet buffer calculation and results in a smaller buffer size.

In Fig. 3.5 one can see that for small buffer sizes the drift of the total energy is much smaller than the pair energy error tolerance, due to cancellation of errors. For larger buffer size, the error estimate is a factor of 6 higher than drift of the total energy, or alternatively the buffer estimate is 0.024 nm too large. This is because the protons don't move freely over 18 fs, but rather vibrate.

Cut-off artifacts and switched interactions

With the Verlet scheme, the pair potentials are shifted to be zero at the cut-off, which makes the potential the integral of the force. This is only possible in the group scheme if the shape of the potential is such that its value is zero at the cut-off distance. However, there can still be energy drift when the forces are non-zero at the cut-off. This effect is extremely small and often not noticeable, as other integration errors (e.g. from constraints) may dominate. To completely avoid cut-off artifacts, the non-bonded forces can be switched exactly to zero at some distance smaller

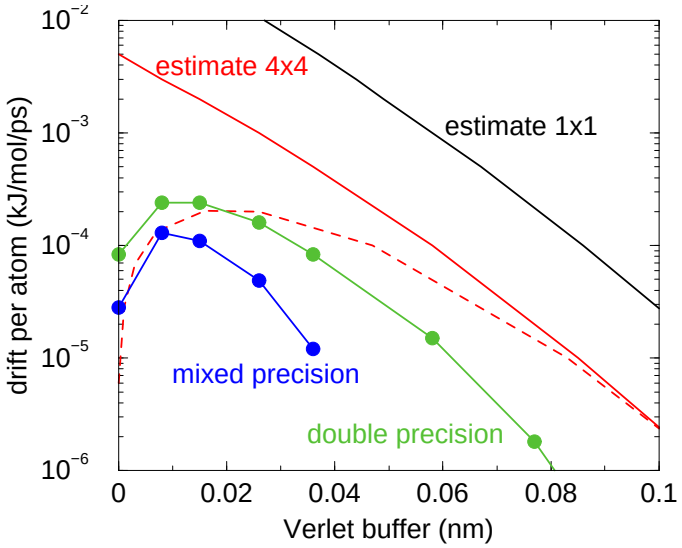


Figure 3.5: Energy drift per atom for an SPC/E water system at 300K with a time step of 2 fs and a pair-list update period of 10 steps (pair-list life time: 18 fs). PME was used with `ewald-rtol` set to 10^{-5}; this parameter affects the shape of the potential at the cut-off. Error estimates due to finite Verlet buffer size are shown for a 1×1 atom pair list and 4×4 atom pair list without and with (dashed line) cancellation of positive and negative errors. Real energy drift is shown for simulations using double- and mixed-precision settings. Rounding errors in the SETTLE constraint algorithm from the use of single precision causes the drift to become negative at large buffer size. Note that at zero buffer size, the real drift is small because positive (H-H) and negative (O-H) energy errors cancel.

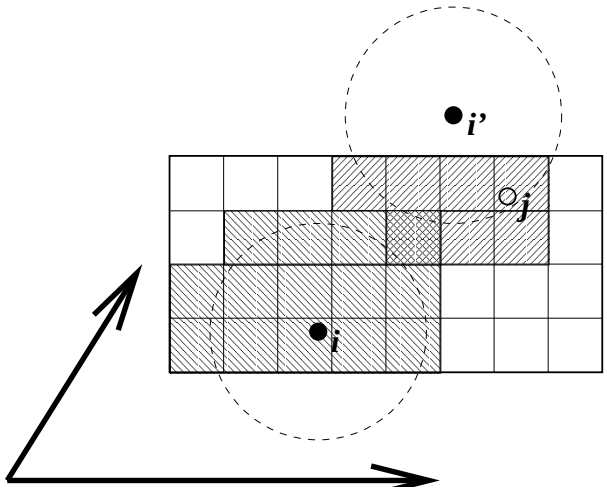


Figure 3.6: Grid search in two dimensions. The arrows are the box vectors.

than the neighbor list cut-off (there are several ways to do this in GROMACS, see sec. 4.1.5). One then has a buffer with the size equal to the neighbor list cut-off less the longest interaction cut-off.

With the group cut-off scheme, one can also choose to let `mdrun` only update the neighbor list when required. That is when one or more particles have moved more than half the buffer size from the center of geometry of the charge group to which they belong (see sec. 3.4.2), as determined at the previous neighbor search. This option guarantees that there are no cut-off artifacts. **Note** that for larger systems this comes at a high computational cost, since the neighbor list update frequency will be determined by just one or two particles moving slightly beyond the half buffer length (which not even necessarily implies that the neighbor list is invalid), while 99.99% of the particles are fine.

Simple search

Due to eqns. 3.1 and 3.6, the vector $\boldsymbol{r}_{ij}$ connecting images within the cut-off R_c can be found by constructing:

$$\begin{aligned} \boldsymbol{r}''' &= \boldsymbol{r}_j - \boldsymbol{r}_i & (3.13)\\ \boldsymbol{r}'' &= \boldsymbol{r}''' - \mathbf{c} * \texttt{round}(r'''_z/c_z) & (3.14)\\ \boldsymbol{r}' &= \boldsymbol{r}'' - \mathbf{b} * \texttt{round}(r''_y/b_y) & (3.15)\\ \boldsymbol{r}_{ij} &= \boldsymbol{r}' - \mathbf{a} * \texttt{round}(r'_x/a_x) & (3.16) \end{aligned}$$

When distances between two particles in a triclinic box are needed that do not obey eqn. 3.1, many shifts of combinations of box vectors need to be considered to find the nearest image.

Grid search

The grid search is schematically depicted in Fig. 3.6. 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 Fig. 3.6 shows, some grid cells may be searched more than once for different images of i. This is not a problem, since, due to the minimum image convention, at most one image will "see" the j-particle. For every particle, fewer than 125 (5^3) neighboring cells are searched. Therefore, the algorithm scales linearly 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 there are more than a few hundred particles. The grid search is equally fast for rectangular and triclinic boxes. Thus for most protein and peptide simulations the rhombic dodecahedron will be the preferred box shape.

Charge groups

Charge groups were originally introduced to reduce cut-off artifacts of Coulomb interactions. When a plain cut-off is used, significant jumps in the potential and forces arise when atoms with (partial) charges move in and out of the cut-off radius. When all chemical moieties have a net charge of zero, these jumps can be reduced by moving groups of atoms with net charge zero, called charge groups, in and out of the neighbor list. This reduces the cut-off effects from the charge-charge level to the dipole-dipole level, which decay much faster. With the advent of full range electrostatics methods, such as particle-mesh Ewald (sec. 4.8.2), the use of charge groups is no longer required for accuracy. It might even have a slight negative effect on the accuracy or efficiency, depending on how the neighbor list is made and the interactions are calculated.

But there is still an important reason for using "charge groups": efficiency with the group cut-off scheme. Where applicable, neighbor searching is carried out on the basis of charge groups which 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 cut-off radius, all atom pairs between the charge groups are included in the pair list. The neighbor searching for a water system, for instance, is $3^2 = 9$ times faster when each molecule is treated as a charge group. Also the highly optimized water force loops (see sec. B.2.1) only work when all atoms in a water molecule form a single charge group. Currently the name *neighbor-search group* would be more appropriate, but the name charge group is retained for historical reasons. When developing a new force field, the advice is to use charge groups of 3 to 4 atoms for optimal performance. For all-atom force fields this is relatively easy, as one can simply put hydrogen atoms, and in some case oxygen atoms, in the same charge group as the heavy atom they are connected to; for example: CH_3, CH_2, CH, NH_2, NH, OH, CO_2, CO.

With the Verlet cut-off scheme, charge groups are ignored.

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 *energy-monitor 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:

$$E_{kin} = \frac{1}{2}\sum_{i=1}^{N} m_i v_i^2 \tag{3.17}$$

From this the absolute temperature T can be computed using:

$$\frac{1}{2} N_{\mathrm{df}} kT = E_{\mathrm{kin}} \tag{3.18}$$

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

$$N_{\mathrm{df}} \;=\; 3N - N_c - N_{\mathrm{com}} \tag{3.19}$$

Here N_c is the number of *constraints* imposed on the system. When performing molecular dynamics $N_{\mathrm{com}} = 3$ 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 $N_{\mathrm{com}} = 6$. When more than one temperature-coupling group is used, the number of degrees of freedom for group i is:

$$N_{\mathrm{df}}^i \;=\; (3N^i - N_c^i)\frac{3N - N_c - N_{\mathrm{com}}}{3N - N_c} \tag{3.20}$$

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:

$$\mathbf{E}_{\mathrm{kin}} = \frac{1}{2}\sum_{i}^{N} m_i \boldsymbol{v}_i \otimes \boldsymbol{v}_i \tag{3.21}$$

Pressure and virial

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

$$\mathbf{P} = \frac{2}{V}(\mathbf{E}_{\mathrm{kin}} - \boldsymbol{\Xi}) \tag{3.22}$$

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:

$$P = \mathrm{trace}(\mathbf{P})/3 \tag{3.23}$$

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

$$\boldsymbol{\Xi} = -\frac{1}{2}\sum_{i<j} \boldsymbol{r}_{ij} \otimes \boldsymbol{F}_{ij} \tag{3.24}$$

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

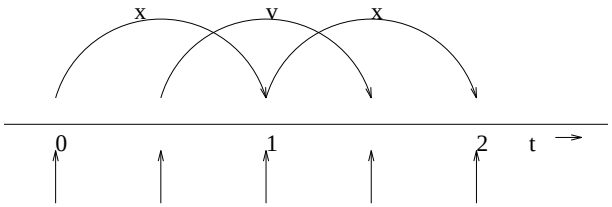


Figure 3.7: The Leap-Frog integration method. The algorithm is called Leap-Frog because $\boldsymbol{r}$ and $\boldsymbol{v}$ are leaping like frogs over each other's backs.

3.4.4 The leap-frog integrator

The default MD integrator in GROMACS is the so-called *leap-frog* algorithm [20] for the integration of the equations of motion. When extremely accurate integration with temperature and/or pressure coupling is required, the velocity Verlet integrators are also present and may be preferable (see 3.4.5). The leap-frog algorithm uses positions $\boldsymbol{r}$ at time t and velocities $\boldsymbol{v}$ at time $t - \frac{1}{2}\Delta t$; it updates positions and velocities using the forces $\boldsymbol{F}(t)$ determined by the positions at time t using these relations:

$$\boldsymbol{v}(t + \frac{1}{2}\Delta t) = \boldsymbol{v}(t - \frac{1}{2}\Delta t) + \frac{\Delta t}{m}\boldsymbol{F}(t) \tag{3.25}$$

$$\boldsymbol{r}(t + \Delta t) = \boldsymbol{r}(t) + \Delta t \boldsymbol{v}(t + \frac{1}{2}\Delta t) \tag{3.26}$$

The algorithm is visualized in Fig. 3.7. It produces trajectories that are identical to the Verlet [21] algorithm, whose position-update relation is

$$\boldsymbol{r}(t + \Delta t) = 2\boldsymbol{r}(t) - \boldsymbol{r}(t - \Delta t) + \frac{1}{m}\boldsymbol{F}(t)\Delta t^2 + O(\Delta t^4) \tag{3.27}$$

The algorithm is of third order in $\boldsymbol{r}$ and is time-reversible. See ref. [22] 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 The velocity Verlet integrator

The velocity Verlet algorithm [23] is also implemented in GROMACS, though it is not yet fully integrated with all sets of options. In velocity Verlet, positions $\boldsymbol{r}$ and velocities $\boldsymbol{v}$ at time t are used to integrate the equations of motion; velocities at the previous half step are not required.

$$\boldsymbol{v}(t + \frac{1}{2}\Delta t) = \boldsymbol{v}(t) + \frac{\Delta t}{2m}\boldsymbol{F}(t) \tag{3.28}$$

$$\boldsymbol{r}(t + \Delta t) = \boldsymbol{r}(t) + \Delta t\, \boldsymbol{v}(t + \frac{1}{2}\Delta t) \tag{3.29}$$

$$\boldsymbol{v}(t + \Delta t) = \boldsymbol{v}(t + \frac{1}{2}\Delta t) + \frac{\Delta t}{2m}\boldsymbol{F}(t + \Delta t) \tag{3.30}$$

or, equivalently,

$$\boldsymbol{r}(t+\Delta t) = \boldsymbol{r}(t) + \Delta t\,\boldsymbol{v} + \frac{\Delta t^2}{2m}\boldsymbol{F}(t) \tag{3.31}$$

$$\boldsymbol{v}(t+\Delta t) = \boldsymbol{v}(t) + \frac{\Delta t}{2m}\left[\boldsymbol{F}(t) + \boldsymbol{F}(t+\Delta t)\right] \tag{3.32}$$

With no temperature or pressure coupling, and with *corresponding* starting points, leap-frog and velocity Verlet will generate identical trajectories, as can easily be verified by hand from the equations above. Given a single starting file with the *same* starting point $\boldsymbol{x}(0)$ and $\boldsymbol{v}(0)$, leap-frog and velocity Verlet will *not* give identical trajectories, as leap-frog will interpret the velocities as corresponding to $t = -\frac{1}{2}\Delta t$, while velocity Verlet will interpret them as corresponding to the timepoint $t = 0$.

3.4.6 Understanding reversible integrators: The Trotter decomposition

To further understand the relationship between velocity Verlet and leap-frog integration, we introduce the reversible Trotter formulation of dynamics, which is also useful to understanding implementations of thermostats and barostats in GROMACS.

A system of coupled, first-order differential equations can be evolved from time $t = 0$ to time t by applying the evolution operator

$$\begin{aligned} \Gamma(t) &= \exp(iLt)\Gamma(0) \\ iL &= \dot{\Gamma}\cdot\nabla_{\Gamma}, \end{aligned} \tag{3.33}$$

where L is the Liouville operator, and Γ is the multidimensional vector of independent variables (positions and velocities). A short-time approximation to the true operator, accurate at time $\Delta t = t/P$, is applied P times in succession to evolve the system as

$$\Gamma(t) = \prod_{i=1}^{P} \exp(iL\Delta t)\Gamma(0) \tag{3.34}$$

For NVE dynamics, the Liouville operator is

$$iL = \sum_{i=1}^{N} \boldsymbol{v}_i \cdot \nabla_{\boldsymbol{r}_i} + \sum_{i=1}^{N} \frac{1}{m_i}\boldsymbol{F}(r_i)\cdot\nabla_{\boldsymbol{v}_i}. \tag{3.35}$$

This can be split into two additive operators

$$\begin{aligned} iL_1 &= \sum_{i=1}^{N} \frac{1}{m_i}\boldsymbol{F}(r_i)\cdot\nabla_{\boldsymbol{v}_i} \\ iL_2 &= \sum_{i=1}^{N} \boldsymbol{v}_i\cdot\nabla_{\boldsymbol{r}_i} \end{aligned} \tag{3.36}$$

Then a short-time, symmetric, and thus reversible approximation of the true dynamics will be

$$\exp(iL\Delta t) = \exp(iL_2\frac{1}{2}\Delta t)\exp(iL_1\Delta t)\exp(iL_2\frac{1}{2}\Delta t) + \mathcal{O}(\Delta t^3). \tag{3.37}$$

This corresponds to velocity Verlet integration. The first exponential term over $\frac{1}{2}\Delta t$ corresponds to a velocity half-step, the second exponential term over Δt corresponds to a full velocity step, and the last exponential term over $\frac{1}{2}\Delta t$ is the final velocity half step. For future times $t = n\Delta t$, this becomes

$$\begin{aligned}\exp(iLn\Delta t) &\approx \left(\exp(iL_2\frac{1}{2}\Delta t)\exp(iL_1\Delta t)\exp(iL_2\frac{1}{2}\Delta t)\right)^n \\ &\approx \exp(iL_2\frac{1}{2}\Delta t)\Big(\exp(iL_1\Delta t)\exp(iL_2\Delta t)\Big)^{n-1} \\ &\quad \exp(iL_1\Delta t)\exp(iL_2\frac{1}{2}\Delta t)\end{aligned} \tag{3.38}$$

This formalism allows us to easily see the difference between the different flavors of Verlet integrators. The leap-frog integrator can be seen as starting with Eq. 3.37 with the $\exp(iL_1\Delta t)$ term, instead of the half-step velocity term, yielding

$$\exp(iLn\Delta t) = \exp(iL_1\Delta t)\exp(iL_2\Delta t) + \mathcal{O}(\Delta t^3). \tag{3.39}$$

Here, the full step in velocity is between $t - \frac{1}{2}\Delta t$ and $t + \frac{1}{2}\Delta t$, since it is a combination of the velocity half steps in velocity Verlet. For future times $t = n\Delta t$, this becomes

$$\exp(iLn\Delta t) \approx \Big(\exp(iL_1\Delta t)\exp(iL_2\Delta t)\Big)^n. \tag{3.40}$$

Although at first this does not appear symmetric, as long as the full velocity step is between $t-\frac{1}{2}\Delta t$ and $t + \frac{1}{2}\Delta t$, then this is simply a way of starting velocity Verlet at a different place in the cycle.

Even though the trajectory and thus potential energies are identical between leap-frog and velocity Verlet, the kinetic energy and temperature will not necessarily be the same. Standard velocity Verlet uses the velocities at the t to calculate the kinetic energy and thus the temperature only at time t; the kinetic energy is then a sum over all particles

$$\begin{aligned}KE_{\text{full}}(t) &= \sum_i \left(\frac{1}{2m_i}\boldsymbol{v}_i(t)\right)^2 \\ &= \sum_i \frac{1}{2m_i}\left(\frac{1}{2}\boldsymbol{v}_i(t-\frac{1}{2}\Delta t) + \frac{1}{2}\boldsymbol{v}_i(t+\frac{1}{2}\Delta t)\right)^2,\end{aligned} \tag{3.41}$$

with the square on the *outside* of the average. Standard leap-frog calculates the kinetic energy at time t based on the average kinetic energies at the timesteps $t+\frac{1}{2}\Delta t$ and $t-\frac{1}{2}\Delta t$, or the sum over all particles

$$KE_{\text{average}}(t) = \sum_i \frac{1}{2m_i}\left(\frac{1}{2}\boldsymbol{v}_i(t-\frac{1}{2}\Delta t)^2 + \frac{1}{2}\boldsymbol{v}_i(t+\frac{1}{2}\Delta t)^2\right), \tag{3.42}$$

where the square is *inside* the average.

A non-standard variant of velocity Verlet which averages the kinetic energies $KE(t+\frac{1}{2}\Delta t)$ and $KE(t-\frac{1}{2}\Delta t)$, exactly like leap-frog, is also now implemented in GROMACS (as `.mdp` file option `md-vv-avek`). Without temperature and pressure coupling, velocity Verlet with half-step-averaged kinetic energies and leap-frog will be identical up to numerical precision. For

temperature- and pressure-control schemes, however, velocity Verlet with half-step-averaged kinetic energies and leap-frog will be different, as will be discussed in the section in thermostats and barostats.

The half-step-averaged kinetic energy and temperature are slightly more accurate for a given step size; the difference in average kinetic energies using the half-step-averaged kinetic energies (*md* and *md-vv-avek*) will be closer to the kinetic energy obtained in the limit of small step size than will the full-step kinetic energy (using *md-vv*). For NVE simulations, this difference is usually not significant, since the positions and velocities of the particles are still identical; it makes a difference in the way the the temperature of the simulations are *interpreted*, but *not* in the trajectories that are produced. Although the kinetic energy is more accurate with the half-step-averaged method, meaning that it changes less as the timestep gets large, it is also more noisy. The RMS deviation of the total energy of the system (sum of kinetic plus potential) in the half-step-averaged kinetic energy case will be higher (about twice as high in most cases) than the full-step kinetic energy. The drift will still be the same, however, as again, the trajectories are identical.

For NVT simulations, however, there *will* be a difference, as discussed in the section on temperature control, since the velocities of the particles are adjusted such that kinetic energies of the simulations, which can be calculated either way, reach the distribution corresponding to the set temperature. In this case, the three methods will not give identical results.

Because the velocity and position are both defined at the same time t the velocity Verlet integrator can be used for some methods, especially rigorously correct pressure control methods, that are not actually possible with leap-frog. The integration itself takes negligibly more time than leap-frog, but twice as many communication calls are currently required. In most cases, and especially for large systems where communication speed is important for parallelization and differences between thermodynamic ensembles vanish in the $1/N$ limit, and when only NVT ensembles are required, leap-frog will likely be the preferred integrator. For pressure control simulations where the fine details of the thermodynamics are important, only velocity Verlet allows the true ensemble to be calculated. In either case, simulation with double precision may be required to get fine details of thermodynamics correct.

3.4.7 Twin-range cut-offs

To save computation time, slowly varying forces can be calculated less often than rapidly varying forces. In GROMACS such a multiple time step splitting is possible between short and long range non-bonded interactions. In GROMACS versions up to 4.0, an irreversible integration scheme was used which is also used by the GROMOS simulation package: every n steps the long range forces are determined and these are then also used (without modification) for the next $n-1$ integration steps in eqn. 3.25. Such an irreversible scheme can result in bad energy conservation and, possibly, bad sampling. Since version 4.5, a leap-frog version of the reversible Trotter decomposition scheme [24] is used. In this integrator the long-range forces are determined every n steps and are then integrated into the velocity in eqn. 3.25 using a time step of $\Delta t_{\mathrm{LR}} = n\Delta t$:

$$\boldsymbol{v}(t+\frac{1}{2}\Delta t) = \begin{cases} \boldsymbol{v}(t-\frac{1}{2}\Delta t) + \frac{1}{m}\left[\boldsymbol{F}_{\mathrm{SR}}(t) + n\boldsymbol{F}_{\mathrm{LR}}(t)\right]\Delta t & , \quad \text{step } \% \; n = 0 \\ \boldsymbol{v}(t-\frac{1}{2}\Delta t) + \frac{1}{m}\boldsymbol{F}_{\mathrm{SR}}(t)\Delta t & , \quad \text{step } \% \; n \neq 0 \end{cases} \tag{3.43}$$

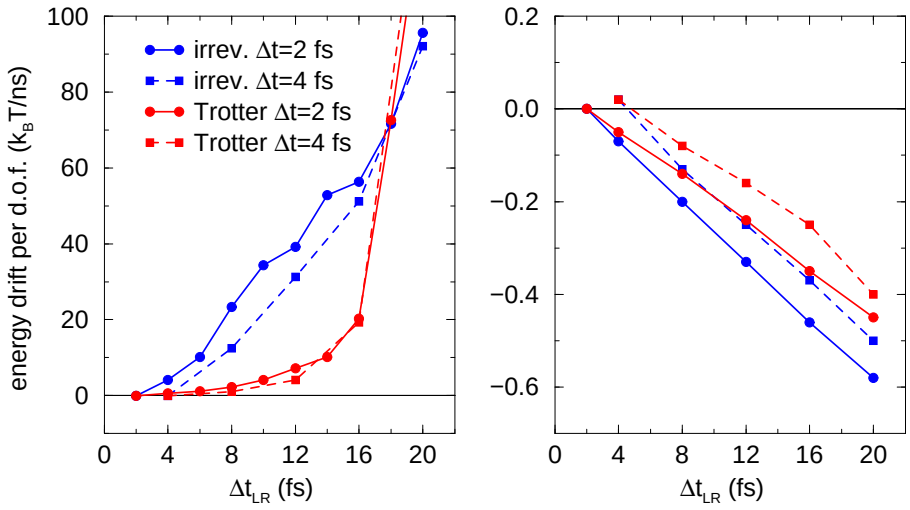


Figure 3.8: Energy drift per degree of freedom in SPC/E water with twin-range cut-offs for reaction field (left) and Lennard-Jones interaction (right) as a function of the long-range time step length for the irreversible "GROMOS" scheme and a reversible Trotter scheme.

The parameter n is equal to the neighbor list update frequency. In 4.5, the velocity Verlet version of multiple time-stepping is not yet fully implemented.

Several other simulation packages uses multiple time stepping for bonds and/or the PME mesh forces. In GROMACS we have not implemented this (yet), since we use a different philosophy. Bonds can be constrained (which is also a more sound approximation of a physical quantum oscillator), which allows the smallest time step to be increased to the larger one. This not only halves the number of force calculations, but also the update calculations. For even larger time steps, angle vibrations involving hydrogen atoms can be removed using virtual interaction sites (see sec. 6.8), which brings the shortest time step up to PME mesh update frequency of a multiple time stepping scheme.

As an example we show the energy conservation for integrating the equations of motion for SPC/E water at 300 K. To avoid cut-off effects, reaction-field electrostatics with $\epsilon_{RF} = \infty$ and shifted Lennard-Jones interactions are used, both with a buffer region. The long-range interactions were evaluated between 1.0 and 1.4 nm. In Fig. 3.7 one can see that for electrostatics the Trotter scheme does an order of magnitude better up to Δt_{LR} = 16 fs. The electrostatics depends strongly on the orientation of the water molecules, which changes rapidly. For Lennard-Jones interactions, the energy drift is linear in Δt_{LR} and roughly two orders of magnitude smaller than for the electrostatics. Lennard-Jones forces are smaller than Coulomb forces and they are mainly affected by translation of water molecules, not rotation.

3.4.8 Temperature coupling

While direct use of molecular dynamics gives rise to the NVE (constant number, constant volume, constant energy ensemble), most quantities that we wish to calculate are actually from a constant temperature (NVT) ensemble, also called the canonical ensemble. GROMACS can use

the *weak-coupling* scheme of Berendsen [25], stochastic randomization through the Andersen thermostat [26], the extended ensemble Nosé-Hoover scheme [27, 28], or a velocity-rescaling scheme [29] to simulate constant temperature, with advantages of each of the schemes laid out below.

There are several other reasons why it might be necessary to control the temperature of the system (drift during equilibration, drift as a result of force truncation and integration errors, heating due to external or frictional forces), but this is not entirely correct to do from a thermodynamic standpoint, and in some cases only masks the symptoms (increase in temperature of the system) rather than the underlying problem (deviations from correct physics in the dynamics). For larger systems, errors in ensemble averages and structural properties incurred by using temperature control to remove slow drifts in temperature appear to be negligible, but no completely comprehensive comparisons have been carried out, and some caution must be taking in interpreting the results.

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. [30] 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:

$$\frac{\mathrm{d}T}{\mathrm{d}t} = \frac{T_0 - T}{\tau} \tag{3.44}$$

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 Berendsen thermostat suppresses the fluctuations of the kinetic energy. This means that one does not generate a proper canonical ensemble, so rigorously, the sampling will be incorrect. This error scales with $1/N$, so for very large systems most ensemble averages will not be affected significantly, except for the distribution of the kinetic energy itself. However, fluctuation properties, such as the heat capacity, will be affected. A similar thermostat which does produce a correct ensemble is the velocity rescaling thermostat [29] described below.

The heat flow into or out of the system is affected by scaling the velocities of each particle every step, or every n_{TC} steps, with a time-dependent factor λ, given by:

$$\lambda = \left[1 + \frac{n_{\mathrm{TC}}\Delta t}{\tau_T}\left\{\frac{T_0}{T(t - \frac{1}{2}\Delta t)} - 1\right\}\right]^{1/2} \tag{3.45}$$

The parameter τ_T is close, but not exactly equal, to the time constant τ of the temperature coupling (eqn. 3.44):

$$\tau = 2C_V\tau_T/N_{df}k \tag{3.46}$$

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.

Velocity-rescaling temperature coupling

The velocity-rescaling thermostat [29] is essentially a Berendsen thermostat (see above) with an additional stochastic term that ensures a correct kinetic energy distribution by modifying it according to

$$\mathrm{d}K = (K_0 - K)\frac{\mathrm{d}t}{\tau_T} + 2\sqrt{\frac{KK_0}{N_f}}\frac{\mathrm{d}W}{\sqrt{\tau_T}}, \tag{3.47}$$

where K is the kinetic energy, N_f the number of degrees of freedom and $\mathrm{d}W$ a Wiener process. There are no additional parameters, except for a random seed. This thermostat produces a correct canonical ensemble and still has the advantage of the Berendsen thermostat: first order decay of temperature deviations and no oscillations. When an NVT ensemble is used, the conserved energy quantity is written to the energy and log file.

Andersen thermostat

One simple way to maintain a thermostatted ensemble is to take an NVE integrator and periodically re-select the velocities of the particles from a Maxwell-Boltzmann distribution. [26] This can either be done by randomizing all the velocities simultaneously (massive collision) every $\tau_T/\Delta t$ steps (`andersen-massive`), or by randomizing every particle with some small probability every timestep (`andersen`), equal to $\Delta t/\tau$, where in both cases Δt is the timestep and τ_T is a characteristic coupling time scale. Because of the way constraints operate, all particles in the same constraint group must be randomized simultaneously. Because of parallelization issues, the `andersen` version cannot currently (5.0) be used in systems with constraints. `andersen-massive` can be used regardless of constraints. This thermostat is also currently only possible with velocity Verlet algorithms, because it operates directly on the velocities at each timestep.

This algorithm completely avoids some of the ergodicity issues of other thermostatting algorithms, as energy cannot flow back and forth between energetically decoupled components of the system as in velocity scaling motions. However, it can slow down the kinetics of system by randomizing correlated motions of the system, including slowing sampling when τ_T is at moderate levels (less than 10 ps). This algorithm should therefore generally not be used when examining kinetics or transport properties of the system. [31]

Nosé-Hoover temperature coupling

The Berendsen weak-coupling algorithm is extremely efficient for relaxing a system to the target temperature, but once the 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.

To enable canonical ensemble simulations, GROMACS also supports the extended-ensemble approach first proposed by Nosé [27] and later modified by Hoover [28]. The system Hamiltonian is extended by introducing a thermal reservoir and a friction term in the equations of motion. The 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 momentum (p_ξ) and equation of motion; the time derivative is calculated from the difference between the current kinetic energy and the reference temperature.

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

$$\frac{\mathrm{d}^2 \boldsymbol{r}_i}{\mathrm{d}t^2} = \frac{\boldsymbol{F}_i}{m_i} - \frac{p_\xi}{Q}\frac{\mathrm{d}\boldsymbol{r}_i}{\mathrm{d}t}, \tag{3.48}$$

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

$$\frac{\mathrm{d}p_\xi}{\mathrm{d}t} = (T - T_0)\,. \tag{3.49}$$

The reference temperature is denoted T_0, while T is the current instantaneous 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. [1]

The conserved quantity for the Nosé-Hoover equations of motion is not the total energy, but rather

$$H = \sum_{i=1}^{N} \frac{\boldsymbol{p}_i}{2m_i} + U\left(\boldsymbol{r}_1, \boldsymbol{r}_2, \ldots, \boldsymbol{r}_N\right) + \frac{p_\xi^2}{2Q} + N_f kT\xi, \tag{3.50}$$

where N_f is the total number of degrees of freedom.

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, one would have to change Q in proportion to the change in reference temperature. For this reason, we prefer to let the GROMACS user work instead 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 via:

$$Q = \frac{\tau_T^2 T_0}{4\pi^2}. \tag{3.51}$$

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.

[1] Note that some derivations, an alternative notation $\xi_{\mathrm{alt}} = v_\xi = p_\xi/Q$ is used.

Nosé-Hoover dynamics in simple systems such as collections of harmonic oscillators, can be *nonergodic*, meaning that only a subsection of phase space is ever sampled, even if the simulations were to run for infinitely long. For this reason, the Nosé-Hoover chain approach was developed, where each of the Nosé-Hoover thermostats has its own Nosé-Hoover thermostat controlling its temperature. In the limit of an infinite chain of thermostats, the dynamics are guaranteed to be ergodic. Using just a few chains can greatly improve the ergodicity, but recent research has shown that the system will still be nonergodic, and it is still not entirely clear what the practical effect of this [32]. Currently, the default number of chains is 10, but this can be controlled by the user. In the case of chains, the equations are modified in the following way to include a chain of thermostatting particles [33]:

$$\begin{aligned}
\frac{\mathrm{d}^2\boldsymbol{r}_i}{\mathrm{d}t^2} &= \frac{\boldsymbol{F}_i}{m_i} - \frac{p_{\xi_1}}{Q_1}\frac{\mathrm{d}\boldsymbol{r}_i}{\mathrm{d}t} \\
\frac{\mathrm{d}p_{\xi_1}}{\mathrm{d}t} &= (T - T_0) - p_{\xi_1}\frac{p_{\xi_2}}{Q_2} \\
\frac{\mathrm{d}p_{\xi_{i=2\ldots N}}}{\mathrm{d}t} &= \left(\frac{p_{\xi_{i-1}}^2}{Q_{i-1}} - kT\right) - p_{\xi_i}\frac{p_{\xi_{i+1}}}{Q_{i+1}} \\
\frac{\mathrm{d}p_{\xi_N}}{\mathrm{d}t} &= \left(\frac{p_{\xi_{N-1}}^2}{Q_{N-1}} - kT\right)
\end{aligned} \tag{3.52}$$

The conserved quantity for Nosé-Hoover chains is

$$H = \sum_{i=1}^{N}\frac{\boldsymbol{p}_i}{2m_i} + U\left(\boldsymbol{r}_1, \boldsymbol{r}_2, \ldots, \boldsymbol{r}_N\right) + \sum_{k=1}^{M}\frac{p_{\xi_k}^2}{2Q_k'} + N_f kT\xi_1 + kT\sum_{k=2}^{M}\xi_k \tag{3.53}$$

The values and velocities of the Nosé-Hoover thermostat variables are generally not included in the output, as they take up a fair amount of space and are generally not important for analysis of simulations, but this can be overridden by defining the environment variable `GMX_NOSEHOOVER_CHAINS`, which will print the values of all the positions and velocities of all Nosé-Hoover particles in the chain to the `.edr` file. Leap-frog simulations currently can only have Nosé-Hoover chain lengths of 1, but this will likely be updated in later version.

As described in the integrator section, for temperature coupling, the temperature that the algorithm attempts to match to the reference temperature is calculated differently in velocity Verlet and leap-frog dynamics. Velocity Verlet (*md-vv*) uses the full-step kinetic energy, while leap-frog and *md-vv-avek* use the half-step-averaged kinetic energy.

We can examine the Trotter decomposition again to better understand the differences between these constant-temperature integrators. In the case of Nosé-Hoover dynamics (for simplicity, using a chain with $N = 1$, with more details in Ref. [34]), we split the Liouville operator as

$$iL = iL_1 + iL_2 + iL_{\mathrm{NHC}}, \tag{3.54}$$

where

$$iL_1 = \sum_{i=1}^{N}\left[\frac{\boldsymbol{p}_i}{m_i}\right]\cdot\frac{\partial}{\partial\boldsymbol{r}_i}$$

$$
\begin{aligned}
iL_2 &= \sum_{i=1}^{N} \boldsymbol{F}_i \cdot \frac{\partial}{\partial \boldsymbol{p}_i} \\
iL_{\mathrm{NHC}} &= \sum_{i=1}^{N} -\frac{p_\xi}{Q}\boldsymbol{v}_i \cdot \nabla_{\boldsymbol{v}_i} + \frac{p_\xi}{Q}\frac{\partial}{\partial \xi} + (T - T_0)\frac{\partial}{\partial p_\xi}
\end{aligned} \tag{3.55}
$$

For standard velocity Verlet with Nosé-Hoover temperature control, this becomes

$$
\begin{aligned}
\exp(iL\Delta t) &= \exp\left(iL_{\mathrm{NHC}}\Delta t/2\right)\exp\left(iL_2\Delta t/2\right) \\
&\quad \exp\left(iL_1\Delta t\right)\exp\left(iL_2\Delta t/2\right)\exp\left(iL_{\mathrm{NHC}}\Delta t/2\right) + \mathcal{O}(\Delta t^3).
\end{aligned} \tag{3.56}
$$

For half-step-averaged temperature control using *md-vv-avek*, this decomposition will not work, since we do not have the full step temperature until after the second velocity step. However, we can construct an alternate decomposition that is still reversible, by switching the place of the NHC and velocity portions of the decomposition:

$$
\begin{aligned}
\exp(iL\Delta t) &= \exp\left(iL_2\Delta t/2\right)\exp\left(iL_{\mathrm{NHC}}\Delta t/2\right)\exp\left(iL_1\Delta t\right) \\
&\quad \exp\left(iL_{\mathrm{NHC}}\Delta t/2\right)\exp\left(iL_2\Delta t/2\right) + \mathcal{O}(\Delta t^3)
\end{aligned} \tag{3.57}
$$

This formalism allows us to easily see the difference between the different flavors of velocity Verlet integrator. The leap-frog integrator can be seen as starting with Eq. 3.57 just before the $\exp\left(iL_1\Delta t\right)$ term, yielding:

$$
\begin{aligned}
\exp(iL\Delta t) &= \exp\left(iL_1\Delta t\right)\exp\left(iL_{\mathrm{NHC}}\Delta t/2\right) \\
&\quad \exp\left(iL_2\Delta t\right)\exp\left(iL_{\mathrm{NHC}}\Delta t/2\right) + \mathcal{O}(\Delta t^3)
\end{aligned} \tag{3.58}
$$

and then using some algebra tricks to solve for some quantities are required before they are actually calculated [35].

Group temperature coupling

In GROMACS temperature coupling can be performed on groups of atoms, typically a protein and solvent. The reason such algorithms were introduced is that energy exchange between different components is not perfect, due to different effects including cut-offs etc. If now the whole system is coupled to one heat bath, water (which experiences the largest cut-off noise) will tend to heat up and the protein will cool down. Typically 100 K differences can be obtained. With the use of proper electrostatic methods (PME) these difference are much smaller but still not negligible. The parameters for temperature coupling in groups are given in the `mdp` file. Recent investigation has shown that small temperature differences between protein and water may actually be an artifact of the way temperature is calculated when there are finite timesteps, and very large differences in temperature are likely a sign of something else seriously going wrong with the system, and should be investigated carefully [36].

One special case should be mentioned: it is possible to temperature-couple only part of the system, leaving other parts without temperature coupling. This is done by specifying -1 for the time constant τ_T for the group that should not be thermostatted. If only part of the system is thermostatted, the system will still eventually converge to an NVT system. In fact, one suggestion for minimizing errors in the temperature caused by discretized timesteps is that if constraints on the water

are used, then only the water degrees of freedom should be thermostatted, not protein degrees of freedom, as the higher frequency modes in the protein can cause larger deviations from the "true" temperature, the temperature obtained with small timesteps [36].

3.4.9 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 [25] that scales coordinates and box vectors every step, the extended-ensemble Parrinello-Rahman approach [37, 38], and for the velocity Verlet variants, the Martyna-Tuckerman-Tobias-Klein (MTTK) implementation of pressure control [34]. Parrinello-Rahman and Berendsen can be combined with any of the temperature coupling methods above; MTTK can only be used with Nosé-Hoover temperature control.

Berendsen pressure coupling

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

$$\frac{\mathrm{d}\mathbf{P}}{\mathrm{d}t} = \frac{\mathbf{P}_0 - \mathbf{P}}{\tau_p}. \tag{3.59}$$

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

$$\mu_{ij} = \delta_{ij} - \frac{n_{\mathrm{PC}}\Delta t}{3\,\tau_p}\beta_{ij}\{P_{0ij} - P_{ij}(t)\}. \tag{3.60}$$

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 eqn. 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

$$\boldsymbol{\mu'} = \begin{pmatrix} \mu_{xx} & \mu_{xy} + \mu_{yx} & \mu_{xz} + \mu_{zx} \\ 0 & \mu_{yy} & \mu_{yz} + \mu_{zy} \\ 0 & 0 & \mu_{zz} \end{pmatrix}. \tag{3.61}$$

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 more slowly or decrease your timestep to avoid errors from the constraint algorithms. It is important to note that although the Berendsen pressure control algorithm yields a simulation with the correct average pressure, it does not yield the exact NPT ensemble, and it is not yet clear exactly what errors this approximation may yield.

Parrinello-Rahman pressure coupling

In cases where the fluctuations in pressure or volume are important *per se* (*e.g.* to calculate thermodynamic properties), especially for small systems, it may be a problem that the exact ensemble is not well defined for the weak-coupling scheme, and that it does not simulate the true NPT ensemble.

GROMACS also supports constant-pressure simulations using the Parrinello-Rahman approach [37, 38], which is similar to the Nosé-Hoover temperature coupling, and in theory gives the true NPT ensemble. With the Parrinello-Rahman barostat, the box vectors as represented by the matrix $\boldsymbol{b}$ obey the matrix equation of motion[2]

$$\frac{\mathrm{d}\boldsymbol{b}^2}{\mathrm{d}t^2} = V\boldsymbol{W}^{-1}\boldsymbol{b}'^{-1}\left(\boldsymbol{P}-\boldsymbol{P}_{ref}\right). \tag{3.62}$$

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:

$$\frac{\mathrm{d}^2\boldsymbol{r}_i}{\mathrm{d}t^2} = \frac{\boldsymbol{F}_i}{m_i} - \boldsymbol{M}\frac{\mathrm{d}\boldsymbol{r}_i}{\mathrm{d}t}, \tag{3.63}$$

$$\boldsymbol{M} = \boldsymbol{b}^{-1}\left[\boldsymbol{b}\frac{\mathrm{d}\boldsymbol{b}'}{\mathrm{d}t} + \frac{\mathrm{d}\boldsymbol{b}}{\mathrm{d}t}\boldsymbol{b}'\right]\boldsymbol{b}'^{-1}. \tag{3.64}$$

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):

$$\left(\boldsymbol{W}^{-1}\right)_{ij} = \frac{4\pi^2\beta_{ij}}{3\tau_p^2 L}. \tag{3.65}$$

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.

[2]The box matrix representation $\boldsymbol{b}$ in GROMACS corresponds 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.

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. Additionally, using the leap-frog algorithm, the pressure at time t is not available until after the time step has completed, and so the pressure from the previous step must be used, which makes the algorithm not directly reversible, and may not be appropriate for high precision thermodynamic calculations.

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

$$\begin{aligned} \gamma(t) &= \frac{1}{n}\int_0^{L_z}\left\{P_{zz}(z,t) - \frac{P_{xx}(z,t)+P_{yy}(z,t)}{2}\right\}\mathrm{d}z & (3.66)\\ &= \frac{L_z}{n}\left\{P_{zz}(t) - \frac{P_{xx}(t)+P_{yy}(t)}{2}\right\}, & (3.67)\end{aligned}$$

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_{zz}

$$\Delta P_{zz} = \frac{\Delta t}{\tau_p}\{P_{0zz} - P_{zz}(t)\} \tag{3.68}$$

$$\mu_{zz} = 1 + \beta_{zz}\Delta P_{zz} \tag{3.69}$$

This is similar to normal pressure coupling, except that the factor of $1/3$ 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

$$\mu_{x/y} = 1 + \frac{\Delta t}{2\,\tau_p}\beta_{x/y}\left(\frac{n\gamma_0}{\mu_{zz}L_z} - \left\{P_{zz}(t) + \Delta P_{zz} - \frac{P_{xx}(t)+P_{yy}(t)}{2}\right\}\right) \tag{3.70}$$

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_{zz} is also set to zero, which is necessary for obtaining the correct surface tension.

MTTK pressure control algorithms

As mentioned in the previous section, one weakness of leap-frog integration is in constant pressure simulations, since the pressure requires a calculation of both the virial and the kinetic energy at the full time step; for leap-frog, this information is not available until *after* the full timestep. Velocity

Verlet does allow the calculation, at the cost of an extra round of global communication, and can compute, mod any integration errors, the true NPT ensemble.

The full equations, combining both pressure coupling and temperature coupling, are taken from Martyna *et al.* [34] and Tuckerman [39] and are referred to here as MTTK equations (Martyna-Tuckerman-Tobias-Klein). We introduce for convenience $\epsilon = (1/3)\ln(V/V_0)$, where V_0 is a reference volume. The momentum of ϵ is $v_\epsilon = p_\epsilon/W = \dot{\epsilon} = \dot{V}/3V$, and define $\alpha = 1 + 3/N_{dof}$ (see Ref [39])

The isobaric equations are

$$\begin{aligned}
\dot{\boldsymbol{r}}_i &= \frac{\boldsymbol{p}_i}{m_i} + \frac{p_\epsilon}{W}\boldsymbol{r}_i \\
\frac{\dot{\boldsymbol{p}}_i}{m_i} &= \frac{1}{m_i}\boldsymbol{F}_i - \alpha\frac{p_\epsilon}{W}\frac{\boldsymbol{p}_i}{m_i} \\
\dot{\epsilon} &= \frac{p_\epsilon}{W} \\
\frac{\dot{p}_\epsilon}{W} &= \frac{3V}{W}(P_{\text{int}} - P) + (\alpha - 1)\left(\sum_{n=1}^{N}\frac{\boldsymbol{p}_i^2}{m_i}\right), \qquad (3.71)
\end{aligned}$$

(3.72)

where

$$P_{\text{int}} = P_{\text{kin}} - P_{\text{vir}} = \frac{1}{3V}\left[\sum_{i=1}^{N}\left(\frac{\boldsymbol{p}_i^2}{2m_i} - \boldsymbol{r}_i\cdot\boldsymbol{F}_i\right)\right]. \qquad (3.73)$$

The terms including α are required to make phase space incompressible [39]. The ϵ acceleration term can be rewritten as

$$\frac{\dot{p}_\epsilon}{W} = \frac{3V}{W}\left(\alpha P_{\text{kin}} - P_{\text{vir}} - P\right) \qquad (3.74)$$

In terms of velocities, these equations become

$$\begin{aligned}
\dot{\boldsymbol{r}}_i &= \boldsymbol{v}_i + v_\epsilon\boldsymbol{r}_i \\
\dot{\boldsymbol{v}}_i &= \frac{1}{m_i}\boldsymbol{F}_i - \alpha v_\epsilon\boldsymbol{v}_i \\
\dot{\epsilon} &= v_\epsilon \\
\dot{v_\epsilon} &= \frac{3V}{W}(P_{\text{int}} - P) + (\alpha - 1)\left(\sum_{n=1}^{N}\frac{1}{2}m_i\boldsymbol{v}_i^2\right) \\
P_{\text{int}} &= P_{\text{kin}} - P_{\text{vir}} = \frac{1}{3V}\left[\sum_{i=1}^{N}\left(\frac{1}{2}m_i\boldsymbol{v}_i^2 - \boldsymbol{r}_i\cdot\boldsymbol{F}_i\right)\right] \qquad (3.75)
\end{aligned}$$

For these equations, the conserved quantity is

$$H = \sum_{i=1}^{N}\frac{\boldsymbol{p}_i^2}{2m_i} + U\left(\boldsymbol{r}_1, \boldsymbol{r}_2, \ldots, \boldsymbol{r}_N\right) + \frac{p_\epsilon}{2W} + PV \qquad (3.76)$$

The next step is to add temperature control. Adding Nosé-Hoover chains, including to the barostat degree of freedom, where we use η for the barostat Nosé-Hoover variables, and Q' for the coupling constants of the thermostats of the barostats, we get

$$\begin{aligned}
\dot{\boldsymbol{r}}_i &= \frac{\boldsymbol{p}_i}{m_i} + \frac{p_\epsilon}{W}\boldsymbol{r}_i \\
\frac{\dot{\boldsymbol{p}}_i}{m_i} &= \frac{1}{m_i}\boldsymbol{F}_i - \alpha\frac{p_\epsilon}{W}\frac{\boldsymbol{p}_i}{m_i} - \frac{p_{\xi_1}}{Q_1}\frac{\boldsymbol{p}_i}{m_i} \\
\dot{\epsilon} &= \frac{p_\epsilon}{W} \\
\frac{\dot{p}_\epsilon}{W} &= \frac{3V}{W}(\alpha P_{\text{kin}} - P_{\text{vir}} - P) - \frac{p_{\eta_1}}{Q'_1}p_\epsilon \\
\dot{\xi}_k &= \frac{p_{\xi_k}}{Q_k} \\
\dot{\eta}_k &= \frac{p_{\eta_k}}{Q'_k} \\
\dot{p}_{\xi_k} &= G_k - \frac{p_{\xi_{k+1}}}{Q_{k+1}} \quad k = 1, \ldots, M-1 \\
\dot{p}_{\eta_k} &= G'_k - \frac{p_{\eta_{k+1}}}{Q'_{k+1}} \quad k = 1, \ldots, M-1 \\
\dot{p}_{\xi_M} &= G_M \\
\dot{p}_{\eta_M} &= G'_M,
\end{aligned} \tag{3.77}$$

where

$$\begin{aligned}
P_{\text{int}} &= P_{\text{kin}} - P_{\text{vir}} = \frac{1}{3V}\left[\sum_{i=1}^{N}\left(\frac{\boldsymbol{p}_i^2}{2m_i} - \boldsymbol{r}_i \cdot \boldsymbol{F}_i\right)\right] \\
G_1 &= \sum_{i=1}^{N}\frac{\boldsymbol{p}_i^2}{m_i} - N_f kT \\
G_k &= \frac{p_{\xi_{k-1}}^2}{2Q_{k-1}} - kT \;\; k = 2, \ldots, M \\
G'_1 &= \frac{p_\epsilon^2}{2W} - kT \\
G'_k &= \frac{p_{\eta_{k-1}}^2}{2Q'_{k-1}} - kT \;\; k = 2, \ldots, M
\end{aligned} \tag{3.78}$$

The conserved quantity is now

$$\begin{aligned}
H = \sum_{i=1}^{N}\frac{\boldsymbol{p}_i}{2m_i} + U\left(\boldsymbol{r}_1, \boldsymbol{r}_2, \ldots, \boldsymbol{r}_N\right) + \frac{p_\epsilon^2}{2W} + PV + \\
\sum_{k=1}^{M}\frac{p_{\xi_k}^2}{2Q_k} + \sum_{k=1}^{M}\frac{p_{\eta_k}^2}{2Q'_k} + N_f kT\xi_1 + kT\sum_{i=2}^{M}\xi_k + kT\sum_{k=1}^{M}\eta_k
\end{aligned} \tag{3.79}$$

Returning to the Trotter decomposition formalism, for pressure control and temperature control [34] we get:

$$iL = iL_1 + iL_2 + iL_{\epsilon,1} + iL_{\epsilon,2} + iL_{\mathrm{NHC-baro}} + iL_{\mathrm{NHC}} \tag{3.80}$$

where "NHC-baro" corresponds to the Nosè-Hoover chain of the barostat, and NHC corresponds to the NHC of the particles,

$$iL_1 = \sum_{i=1}^{N} \left[\frac{\boldsymbol{p}_i}{m_i} + \frac{p_\epsilon}{W} \boldsymbol{r}_i \right] \cdot \frac{\partial}{\partial \boldsymbol{r}_i} \tag{3.81}$$

$$iL_2 = \sum_{i=1}^{N} \boldsymbol{F}_i - \alpha \frac{p_\epsilon}{W} \boldsymbol{p}_i \cdot \frac{\partial}{\partial \boldsymbol{p}_i} \tag{3.82}$$

$$iL_{\epsilon,1} = \frac{p_\epsilon}{W} \frac{\partial}{\partial \epsilon} \tag{3.83}$$

$$iL_{\epsilon,2} = G_\epsilon \frac{\partial}{\partial p_\epsilon} \tag{3.84}$$

and where

$$G_\epsilon = 3V \left(\alpha P_{\mathrm{kin}} - P_{\mathrm{vir}} - P \right) \tag{3.85}$$

Using the Trotter decomposition, we get

$$\begin{aligned} \exp(iL\Delta t) = {} & \exp\left(iL_{\mathrm{NHC-baro}}\Delta t/2\right) \exp\left(iL_{\mathrm{NHC}}\Delta t/2\right) \\ & \exp\left(iL_{\epsilon,2}\Delta t/2\right) \exp\left(iL_2\Delta t/2\right) \\ & \exp\left(iL_{\epsilon,1}\Delta t\right) \exp\left(iL_1\Delta t\right) \\ & \exp\left(iL_2\Delta t/2\right) \exp\left(iL_{\epsilon,2}\Delta t/2\right) \\ & \exp\left(iL_{\mathrm{NHC}}\Delta t/2\right) \exp\left(iL_{\mathrm{NHC-baro}}\Delta t/2\right) + \mathcal{O}(\Delta t^3) \end{aligned} \tag{3.86}$$

The action of $\exp\left(iL_1\Delta t\right)$ comes from the solution of the the differential equation $\dot{\boldsymbol{r}}_i = \boldsymbol{v}_i + v_\epsilon \boldsymbol{r}_i$ with $\boldsymbol{v}_i = \boldsymbol{p}_i/m_i$ and v_ϵ constant with initial condition $\boldsymbol{r}_i(0)$, evaluate at $t = \Delta t$. This yields the evolution

$$\boldsymbol{r}_i(\Delta t) = \boldsymbol{r}_i(0) e^{v_\epsilon \Delta t} + \Delta t \boldsymbol{v}_i(0) e^{v_\epsilon \Delta t/2} \frac{\sinh\left(v_\epsilon \Delta t/2\right)}{v_\epsilon \Delta t/2}. \tag{3.87}$$

The action of $\exp\left(iL_2\Delta t/2\right)$ comes from the solution of the differential equation $\dot{\boldsymbol{v}}_i = \frac{\boldsymbol{F}_i}{m_i} - \alpha v_\epsilon \boldsymbol{v}_i$, yielding

$$\boldsymbol{v}_i(\Delta t/2) = \boldsymbol{v}_i(0) e^{-\alpha v_\epsilon \Delta t/2} + \frac{\Delta t}{2m_i} \boldsymbol{F}_i(0) e^{-\alpha v_\epsilon \Delta t/4} \frac{\sinh\left(\alpha v_\epsilon \Delta t/4\right)}{\alpha v_\epsilon \Delta t/4}. \tag{3.88}$$

md-vv-avek uses the full step kinetic energies for determining the pressure with the pressure control, but the half-step-averaged kinetic energy for the temperatures, which can be written as a Trotter decomposition as

$$\begin{aligned} \exp(iL\Delta t) = {} & \exp\left(iL_{\mathrm{NHC-baro}}\Delta t/2\right) \exp\left(iL_{\epsilon,2}\Delta t/2\right) \exp\left(iL_2\Delta t/2\right) \\ & \exp\left(iL_{\mathrm{NHC}}\Delta t/2\right) \exp\left(iL_{\epsilon,1}\Delta t\right) \exp\left(iL_1\Delta t\right) \exp\left(iL_{\mathrm{NHC}}\Delta t/2\right) \\ & \exp\left(iL_2\Delta t/2\right) \exp\left(iL_{\epsilon,2}\Delta t/2\right) \exp\left(iL_{\mathrm{NHC-baro}}\Delta t/2\right) + \mathcal{O}(\Delta t^3) \end{aligned} \tag{3.89}$$

With constraints, the equations become significantly more complicated, in that each of these equations need to be solved iteratively for the constraint forces. The discussion of the details of the iteration is beyond the scope of this manual; readers are encouraged to see the implementation described in [40].

Infrequent evaluation of temperature and pressure coupling

Temperature and pressure control require global communication to compute the kinetic energy and virial, which can become costly if performed every step for large systems. We can rearrange the Trotter decomposition to give alternate symplectic, reversible integrator with the coupling steps every n steps instead of every steps. These new integrators will diverge if the coupling time step is too large, as the auxiliary variable integrations will not converge. However, in most cases, long coupling times are more appropriate, as they disturb the dynamics less [34].

Standard velocity Verlet with Nosé-Hoover temperature control has a Trotter expansion

$$\begin{aligned} \exp(iL\Delta t) \quad \approx \quad & \exp\left(iL_{\mathrm{NHC}}\Delta t/2\right)\exp\left(iL_2\Delta t/2\right) \\ & \exp\left(iL_1\Delta t\right)\exp\left(iL_2\Delta t/2\right)\exp\left(iL_{\mathrm{NHC}}\Delta t/2\right). \end{aligned} \tag{3.90}$$

If the Nosé-Hoover chain is sufficiently slow with respect to the motions of the system, we can write an alternate integrator over n steps for velocity Verlet as

$$\begin{aligned} \exp(iL\Delta t) \quad \approx \quad & \left(\exp\left(iL_{\mathrm{NHC}}(n\Delta t/2)\right)\left[\exp\left(iL_2\Delta t/2\right)\right.\right. \\ & \left.\left.\exp\left(iL_1\Delta t\right)\exp\left(iL_2\Delta t/2\right)\right]^n\exp\left(iL_{\mathrm{NHC}}(n\Delta t/2)\right)\right). \end{aligned} \tag{3.91}$$

For pressure control, this becomes

$$\begin{aligned} \exp(iL\Delta t) \quad \approx \quad & \exp\left(iL_{\mathrm{NHC-baro}}(n\Delta t/2)\right)\exp\left(iL_{\mathrm{NHC}}(n\Delta t/2)\right) \\ & \exp\left(iL_{\epsilon,2}(n\Delta t/2)\right)\left[\exp\left(iL_2\Delta t/2\right)\right. \\ & \exp\left(iL_{\epsilon,1}\Delta t\right)\exp\left(iL_1\Delta t\right) \\ & \left.\exp\left(iL_2\Delta t/2\right)\right]^n\exp\left(iL_{\epsilon,2}(n\Delta t/2)\right) \\ & \exp\left(iL_{\mathrm{NHC}}(n\Delta t/2)\right)\exp\left(iL_{\mathrm{NHC-baro}}(n\Delta t/2)\right), \end{aligned} \tag{3.92}$$

where the box volume integration occurs every step, but the auxiliary variable integrations happen every n steps.

3.4.10 The complete update algorithm

The complete algorithm for the update of velocities and coordinates is given using leap-frog in Fig. 3.9. 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 freeze group (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

$$\boldsymbol{v}(t+\frac{\Delta t}{2}) \;=\; \boldsymbol{f}_g * \lambda * \left[\boldsymbol{v}(t-\frac{\Delta t}{2}) + \frac{\boldsymbol{F}(t)}{m}\Delta t + \boldsymbol{a}_h\Delta t\right], \tag{3.93}$$

THE UPDATE ALGORITHM

Given:
Positions $\boldsymbol{r}$ of all atoms at time t
Velocities $\boldsymbol{v}$ of all atoms at time $t - \frac{1}{2}\Delta t$
Accelerations $\boldsymbol{F}/m$ on all atoms at time t.
(Forces are computed disregarding any constraints)
Total kinetic energy and virial at $t - \Delta t$

$\Downarrow$

1. Compute the scaling factors λ and μ
according to eqns. 3.45 and 3.60

$\Downarrow$

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 3.9: The MD update algorithm with the leap-frog integrator

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

3.4.11 Output step

The most important output of the MD run is the *trajectory file*, which contains particle coordinates and (optionally) velocities at regular intervals. The trajectory file contains frames that could include positions, velocities and/or forces, as well as information about the dimensions of the simulation volume, integration step, integration time, etc. The interpretation of the time varies with the integrator chosen, as described above. For Velocity Verlet integrators, velocities labeled at time t are for that time. For other integrators (e.g. leap-frog, stochastic dynamics), the velocities labeled at time t are for time $t - \frac{1}{2}\Delta t$.

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.

GROMACS can also write reduced-precision coordinates for a subset of the simulation system to a special compressed trajectory file format. All the other tools can read and write this format. See sec. 7.3 for details on how to set up your `.mdp` file to have `mdrun` use this feature.

3.5 Shell molecular dynamics

GROMACS can simulate polarizability using the shell model of Dick and Overhauser [41]. 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). Successful applications of shell models in GROMACS have been published for N_2 [42] and water [43].

3.5.1 Optimization of the shell positions

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

$$\boldsymbol{F}_S \;=\; \boldsymbol{F}_{bond} + \boldsymbol{F}_{nb} \tag{3.94}$$

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

$$\boldsymbol{F}_{bond} \;=\; k_b \left(\boldsymbol{x}_S - \boldsymbol{x}_A\right). \tag{3.95}$$

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

$$\boldsymbol{F}_{nb} \;=\; \boldsymbol{F}_S - k_b\left(\boldsymbol{x}_S(n) - \boldsymbol{x}_A\right) \tag{3.96}$$

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

$$\boldsymbol{F}_S - k_b\left(\boldsymbol{x}_S(n) - \boldsymbol{x}_A\right) + k_b\left(\boldsymbol{x}_S(n+1) - \boldsymbol{x}_A\right) = 0 \tag{3.97}$$

if we write

$$\Delta\boldsymbol{x}_S = \boldsymbol{x}_S(n+1) - \boldsymbol{x}_S(n) \tag{3.98}$$

we finally obtain

$$\Delta\boldsymbol{x}_S = \boldsymbol{F}_S/k_b \tag{3.99}$$

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

$$\boldsymbol{x}_S(n+1) \;=\; \boldsymbol{x}_S(n) + \boldsymbol{F}_S/k_b. \tag{3.100}$$

3.6 Constraint algorithms

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

3.6.1 SHAKE

The SHAKE [44] 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 $\boldsymbol{r}$ reference, as

$$\mathrm{SHAKE}(\boldsymbol{r}^{'} \rightarrow \boldsymbol{r}''; \boldsymbol{r}) \tag{3.101}$$

This action is consistent with solving a set of Lagrange multipliers in the constrained equations of motion. SHAKE needs a *relative tolerance*; it will continue until all constraints are satisfied within that relative tolerance. 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

$$\sigma_k(\boldsymbol{r}_1 \ldots \boldsymbol{r}_N) = 0; \quad k = 1 \ldots K. \tag{3.102}$$

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

$$-\frac{\partial}{\partial \boldsymbol{r}_i}\left(V + \sum_{k=1}^{K} \lambda_k \sigma_k\right), \tag{3.103}$$

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

$$\boldsymbol{G}_i = -\sum_{k=1}^{K} \lambda_k \frac{\partial \sigma_k}{\partial \boldsymbol{r}_i} \tag{3.104}$$

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 [45] (sec. 5.5).

For velocity Verlet, an additional round of constraining must be done, to constrain the velocities of the second velocity half step, removing any component of the velocity parallel to the bond vector. This step is called RATTLE, and is covered in more detail in the original Andersen paper [46].

3.6.2 LINCS

The LINCS algorithm

LINCS is an algorithm that resets bonds to their correct lengths after an unconstrained update [47]. 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 parallel version of LINCS, P-LINCS, is described in subsection 3.17.3.

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

$$\frac{\mathrm{d}^2\boldsymbol{r}}{\mathrm{d}t^2} = \boldsymbol{M}^{-1}\boldsymbol{F}, \tag{3.105}$$

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

$$g_i(\boldsymbol{r}) = |\boldsymbol{r}_{i_1} - \boldsymbol{r}_{i_2}| - d_i = 0 \qquad i = 1, \ldots, K. \tag{3.106}$$

In a numerical integration scheme, LINCS is applied after an unconstrained update, just like SHAKE. The algorithm works in two steps (see figure Fig. 3.10). 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 [47]. 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 this equation:

$$B_{hi} = \frac{\partial g_h}{\partial r_i} \tag{3.107}$$

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

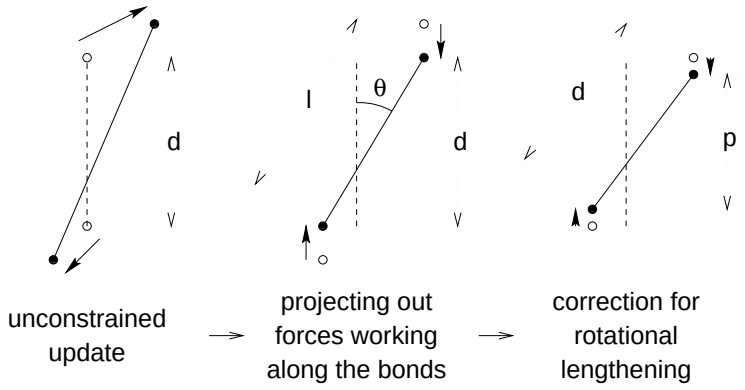


Figure 3.10: The three position updates needed for one time step. The dashed line is the old bond of length d, the solid lines are the new bonds. $l = d\cos\theta$ and $p = (2d^2 - l^2)^{\frac{1}{2}}$.

$\boldsymbol{r}_{n+1}^{unc}$ by

$$\begin{aligned} \boldsymbol{r}_{n+1} = (\boldsymbol{I} - \boldsymbol{T}_n\boldsymbol{B}_n)\boldsymbol{r}_{n+1}^{unc} + \boldsymbol{T}_n\boldsymbol{d} = \\ \boldsymbol{r}_{n+1}^{unc} - \boldsymbol{M}^{-1}\boldsymbol{B}_n(\boldsymbol{B}_n\boldsymbol{M}^{-1}\boldsymbol{B}_n^T)^{-1}(\boldsymbol{B}_n\boldsymbol{r}_{n+1}^{unc} - \boldsymbol{d}) \end{aligned} \tag{3.108}$$

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. 3.105 and 3.106 can be found in [47].

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, p_i, on the old direction is set to

$$p_i = \sqrt{2d_i^2 - l_i^2}, \tag{3.109}$$

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

$$\boldsymbol{r}_{n+1}^{*} = (\boldsymbol{I} - \boldsymbol{T}_n\boldsymbol{B}_n)\boldsymbol{r}_{n+1} + \boldsymbol{T}_n\boldsymbol{p}. \tag{3.110}$$

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:

$$\begin{aligned} (\boldsymbol{B}_n\boldsymbol{M}^{-1}\boldsymbol{B}_n^T)^{-1} &= \boldsymbol{S}\boldsymbol{S}^{-1}(\boldsymbol{B}_n\boldsymbol{M}^{-1}\boldsymbol{B}_n^T)^{-1}\boldsymbol{S}^{-1}\boldsymbol{S} \\ &= \boldsymbol{S}(\boldsymbol{S}\boldsymbol{B}_n\boldsymbol{M}^{-1}\boldsymbol{B}_n^T\boldsymbol{S})^{-1}\boldsymbol{S} = \boldsymbol{S}(\boldsymbol{I} - \boldsymbol{A}_n)^{-1}\boldsymbol{S} \end{aligned} \tag{3.111}$$

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:

$$(\boldsymbol{I} - \boldsymbol{A}_n)^{-1} = \boldsymbol{I} + \boldsymbol{A}_n + \boldsymbol{A}_n^2 + \boldsymbol{A}_n^3 + \ldots \tag{3.112}$$

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.

For molecules with all bonds constrained the eigenvalues of A are around 0.4. This means that with each additional order in the expansion eqn. 3.112 the deviations decrease by a factor 0.4. But for relatively isolated triangles of constraints the largest eigenvalue is around 0.7. Such triangles can occur when removing hydrogen angle vibrations with an additional angle constraint in alcohol groups or when constraining water molecules with LINCS, for instance with flexible constraints. The constraints in such triangles converge twice as slow as the other constraints. Therefore, starting with GROMACS 4, additional terms are added to the expansion for such triangles

$$(\boldsymbol{I} - \boldsymbol{A}_n)^{-1} \approx \boldsymbol{I} + \boldsymbol{A}_n + \ldots + \boldsymbol{A}_n^{N_i} + \left(\boldsymbol{A}_n^* + \ldots + \boldsymbol{A}_n^{*\,N_i}\right) \boldsymbol{A}_n^{N_i} \tag{3.113}$$

where N_i is the normal order of the expansion and $\boldsymbol{A}^*$ only contains the elements of $\boldsymbol{A}$ that couple constraints within rigid triangles, all other elements are zero. In this manner, the accuracy of angle constraints comes close to that of the other constraints, while the series of matrix vector multiplications required for determining the expansion only needs to be extended for a few constraint couplings. This procedure is described in the P-LINCS paper[48].

The LINCS Parameters

The accuracy of LINCS depends on the number of matrices used in the expansion eqn. 3.112. 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 `*.mdp` file. 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 `*.mdp` file.

3.7 Simulated Annealing

The well known simulated annealing (SA) protocol is supported in GROMACS, and you can even couple multiple groups of atoms separately with an arbitrary number of reference temperatures that change during the simulation. The annealing is implemented by simply changing the current reference temperature for each group in the temperature coupling, so the actual relaxation and coupling properties depends on the type of thermostat you use and how hard you are coupling it. Since we are changing the reference temperature it is important to remember that the system will

NOT instantaneously reach this value - you need to allow for the inherent relaxation time in the coupling algorithm too. If you are changing the annealing reference temperature faster than the temperature relaxation you will probably end up with a crash when the difference becomes too large.

The annealing protocol is specified as a series of corresponding times and reference temperatures for each group, and you can also choose whether you only want a single sequence (after which the temperature will be coupled to the last reference value), or if the annealing should be periodic and restart at the first reference point once the sequence is completed. You can mix and match both types of annealing and non-annealed groups in your simulation.

3.8 Stochastic Dynamics

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

$$m_i \frac{\mathrm{d}^2 \boldsymbol{r}_i}{\mathrm{d}t^2} = -m_i \gamma_i \frac{\mathrm{d}\boldsymbol{r}_i}{\mathrm{d}t} + \boldsymbol{F}_i(\boldsymbol{r}) + \mathring{\boldsymbol{r}}_i, \tag{3.114}$$

where γ_i is the friction constant $[1/\mathrm{ps}]$ and $\mathring{\boldsymbol{r}}_i(t)$ is a noise process with $\langle \mathring{r}_i(t)\ \mathring{r}_j(t+s) \rangle = 2 m_i \gamma_i k_B T \delta(s) \delta_{ij}$. When $1/\gamma_i$ 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 simulating a system in vacuum there is the additional advantage that there is no accumulation of errors for the overall translational and rotational degrees of freedom. When $1/\gamma_i$ 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.

In GROMACS there are two algorithms to integrate equation (3.114): a simple and efficient one and a more complex leap-frog algorithm [49], which is now deprecated. The accuracy of both integrators is equivalent to the normal MD leap-frog and Velocity Verlet integrator, except with constraints where the complex SD integrator samples at a temperature that is slightly too high (although that error is smaller than the one from the Velocity Verlet integrator that uses the kinetic energy from the full-step velocity). The simple integrator is nearly identical to the common way of discretizing the Langevin equation, but the friction and velocity term are applied in an impulse fashion [50]. The simple integrator is:

$$\boldsymbol{v}' = \boldsymbol{v}(t - \frac{1}{2}\Delta t) + \frac{1}{m}\boldsymbol{F}(t)\Delta t \tag{3.115}$$

$$\Delta \boldsymbol{v} = -\alpha\, \boldsymbol{v}'(t + \frac{1}{2}\Delta t) + \sqrt{\frac{k_B T}{m}(1-\alpha^2)}\, \boldsymbol{r}_i^G \tag{3.116}$$

$$\boldsymbol{r}(t + \Delta t) = \boldsymbol{r}(t) + \left(\boldsymbol{v}' + \frac{1}{2}\Delta \boldsymbol{v}\right)\Delta t \tag{3.117}$$

$$\boldsymbol{v}(t + \frac{1}{2}\Delta t) = \boldsymbol{v}' + \Delta \boldsymbol{v} \tag{3.118}$$

$$\alpha = 1 - e^{-\gamma \Delta t} \tag{3.119}$$

where $\boldsymbol{r}_i^G$ is Gaussian distributed noise with $\mu = 0, \sigma = 1$. The velocity is first updated a full time step without friction and noise to get $\boldsymbol{v}'$, identical to the normal update in leap-frog. The friction

and noise are then applied as an impulse at step $t + \Delta t$. The advantage of this scheme is that the velocity-dependent terms act at the full time step, which makes the correct integration of forces that depend on both coordinates and velocities, such as constraints and dissipative particle dynamics (DPD, not implented yet), straightforward. With constraints, the coordinate update eqn. 3.117 is split into a normal leap-frog update and a $\Delta \boldsymbol{v}$. After both of these updates the constraints are applied to coordinates and velocities.

In the deprecated complex algorithm, four Gaussian random numbers are required per integration step per degree of freedom, and with constraints the coordinates need to be constrained twice per integration step. Depending on the computational cost of the force calculation, this can take a significant part of the simulation time. Exact continuation of a stochastic dynamics simulation is not possible, because the state of the random number generator is not stored.

When using SD as a thermostat, an appropriate value for γ is e.g. 0.5 ps^{-1}, since this results in a friction that is lower than the internal friction of water, while it still provides efficient thermostatting.

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

$$\frac{\mathrm{d}\boldsymbol{r}_i}{\mathrm{d}t} = \frac{1}{\gamma_i}\boldsymbol{F}_i(\boldsymbol{r}) + \mathring{\boldsymbol{r}}_i \tag{3.120}$$

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

$$\boldsymbol{r}_i(t+\Delta t) = \boldsymbol{r}_i(t) + \frac{\Delta t}{\gamma_i}\boldsymbol{F}_i(\boldsymbol{r}(t)) + \sqrt{2k_BT\frac{\Delta t}{\gamma_i}}\,\boldsymbol{r}_i^G, \tag{3.121}$$

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 $\gamma_i = m_i\,\gamma_i$, where the friction constants γ_i can be different for different groups of atoms. Because the system is assumed to be over-damped, large timesteps 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 steepest descent, conjugate gradients, or l-bfgs (limited-memory Broyden-Fletcher-Goldfarb-Shanno quasi-Newtonian minimizer...we prefer the abbreviation). 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

$$\boldsymbol{r}_{n+1} = \boldsymbol{r}_n + \frac{\boldsymbol{F}_n}{\max(|\boldsymbol{F}_n|)} h_n, \tag{3.122}$$

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 $h_{n+1} = 1.2h_n$.
If ($V_{n+1} \geq V_n$) the new positions are rejected and $h_n = 0.2h_n$.

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

$$f = 2\pi\nu\sqrt{2mkT}, \tag{3.123}$$

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. In GROMACS conjugate gradient can not be used with constraints, including the SETTLE algorithm for water [45], as this has not been implemented. If water is present it must be of a flexible model, which can be specified in the `*.mdp` file by `define = -DFLEXIBLE`.

This is not really a restriction, since the accuracy of conjugate gradient is only required for minimization prior to a normal-mode analysis, which cannot be performed with constraints. For most other purposes steepest descent is efficient enough.

3.10.3 L-BFGS

The original BFGS algorithm works by successively creating better approximations of the inverse Hessian matrix, and moving the system to the currently estimated minimum. The memory requirements for this are proportional to the square of the number of particles, so it is not practical

for large systems like biomolecules. Instead, we use the L-BFGS algorithm of Nocedal [51, 52], which approximates the inverse Hessian by a fixed number of corrections from previous steps. This sliding-window technique is almost as efficient as the original method, but the memory requirements are much lower - proportional to the number of particles multiplied with the correction steps. In practice we have found it to converge faster than conjugate gradients, but due to the correction steps it is not yet parallelized. It is also noteworthy that switched or shifted interactions usually improve the convergence, since sharp cut-offs mean the potential function at the current coordinates is slightly different from the previous steps used to build the inverse Hessian approximation.

3.11 Normal-Mode Analysis

Normal-mode analysis [53, 54, 55] can be performed using GROMACS, by diagonalization of the mass-weighted Hessian H:

$$R^T M^{-1/2} H M^{-1/2} R = \text{diag}(\lambda_1, \ldots, \lambda_{3N}) \tag{3.124}$$

$$\lambda_i = (2\pi\omega_i)^2 \tag{3.125}$$

where M contains the atomic masses, R is a matrix that contains the eigenvectors as columns, λ_i are the eigenvalues and ω_i are the corresponding frequencies.

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

$$H_{ij} = \frac{\partial^2 V}{\partial x_i \partial x_j} \tag{3.126}$$

where x_i and x_j denote the atomic x, y or z coordinates. In practice, this equation is not used, but the Hessian is calculated numerically from the force as:

$$H_{ij} = -\frac{f_i(\mathbf{x} + h\mathbf{e}_j) - f_i(\mathbf{x} - h\mathbf{e}_j)}{2h} \tag{3.127}$$

$$f_i = -\frac{\partial V}{\partial x_i} \tag{3.128}$$

where $\mathbf{e}_j$ is the unit vector in direction j. 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. The tolerance required depends on the type of system, but a rough indication is 0.001 kJ mol^{-1}. Minimization should be done with conjugate gradients or L-BFGS in double precision.

A number of GROMACS programs are involved in these calculations. First, the energy should be minimized using `mdrun`. Then, `mdrun` computes the Hessian. **Note** that for generating the run input file, one should use the minimized conformation from the full precision trajectory file, as the structure file is not accurate enough. `g_nmeig` does the diagonalization and the sorting of the normal modes according to their frequencies. Both `mdrun` and `g_nmeig` should be run in double precision. The normal modes can be analyzed with the program `g_anaeig`. Ensembles of structures at any temperature and for any subset of normal modes can be generated with `g_nmens`. An overview of normal-mode analysis and the related principal component analysis (see sec. 8.10) can be found in [56].

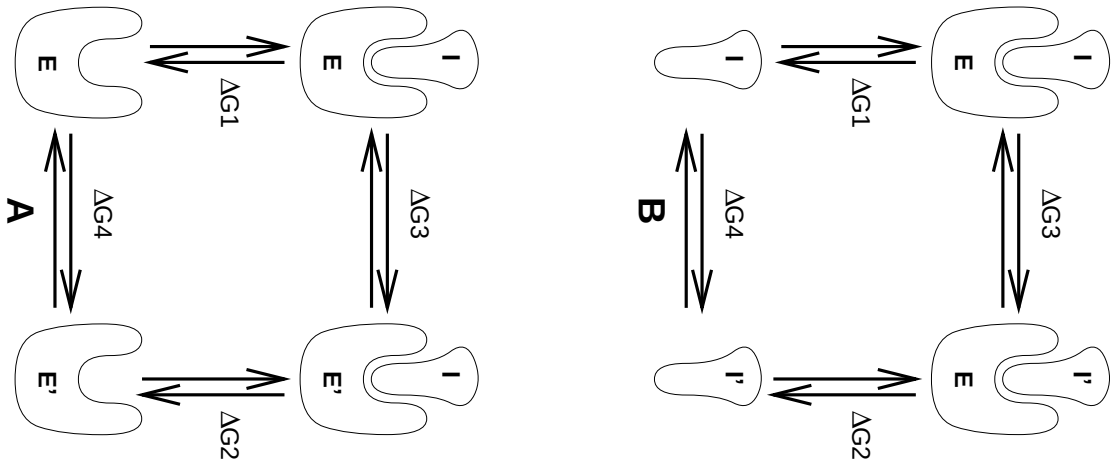


Figure 3.11: Free energy cycles. **A:** to calculate ΔG_{12}, the free energy difference between the binding of inhibitor **I** to enzymes **E** respectively **E**′. **B:** to calculate ΔG_{12}, the free energy difference for binding of inhibitors **I** respectively **I**′ to enzyme **E**.

3.12 Free energy calculations

3.12.1 Slow-growth methods

Free energy calculations can be performed in GROMACS using a number of methods, including "slow-growth." An example problem might be calculating 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. 3.11A we can write:

$$\Delta G_1 - \Delta G_2 = \Delta G_3 - \Delta G_4 \tag{3.129}$$

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. 3.11B) we can again use eqn. 3.129 to compute the desired property.

Free energy differences between two molecular species can be calculated in GROMACS using the "slow-growth" method. 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 (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:

$$H(p,q;0) = H^{\mathrm{A}}(p,q); \quad H(p,q;1) = H^{\mathrm{B}}(p,q). \tag{3.130}$$

In GROMACS, the functional form of the λ-dependence is different for the various force-field contributions and is described in section sec. 4.5.

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:

$$\begin{aligned} A(\lambda) &= -k_B T \ln Q & (3.131)\\ Q &= c \iint \exp[-\beta H(p,q;\lambda)]\, dp\, dq & (3.132)\\ G(\lambda) &= -k_B T \ln \Delta & (3.133)\\ \Delta &= c \iiint \exp[-\beta H(p,q;\lambda) - \beta pV]\, dp\, dq\, dV & (3.134)\\ G &= A + pV, & (3.135) \end{aligned}$$

where $\beta = 1/(k_B T)$ 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 with respect to λ as an ensemble average:

$$\frac{dA}{d\lambda} = \frac{\iint (\partial H/\partial\lambda) \exp[-\beta H(p,q;\lambda)]\, dp\, dq}{\iint \exp[-\beta H(p,q;\lambda)]\, dp\, dq} = \left\langle \frac{\partial H}{\partial \lambda} \right\rangle_{NVT;\lambda}, \tag{3.136}$$

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:

$$\begin{aligned} A^{\mathrm{B}}(V,T) - A^{\mathrm{A}}(V,T) &= \int_0^1 \left\langle \frac{\partial H}{\partial \lambda} \right\rangle_{NVT;\lambda} d\lambda & (3.137)\\ G^{\mathrm{B}}(p,T) - G^{\mathrm{A}}(p,T) &= \int_0^1 \left\langle \frac{\partial H}{\partial \lambda} \right\rangle_{NpT;\lambda} d\lambda. & (3.138) \end{aligned}$$

If one wishes to evaluate $G^{\mathrm{B}}(p,T) - G^{\mathrm{A}}(p,T)$, 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 (but, in principle, exact) correction:

$$G^{\mathrm{B}}(p) - G^{\mathrm{A}}(p) = A^{\mathrm{B}}(V) - A^{\mathrm{A}}(V) - \int_p^{p^{\mathrm{B}}} [V^{\mathrm{B}}(p') - V]\, dp' \tag{3.139}$$

Here we omitted the constant T from the notation. This correction is roughly equal to $-\frac{1}{2}(p^{\mathrm{B}} - p)\Delta V = (\Delta V)^2/(2\kappa V)$, where ΔV is the volume change at p and κ is the isothermal compressibility. This is usually small; 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 as much as 22 bar, but a correction to the Helmholtz free energy of just -1 kJ mol^{-1}.

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 $-\frac{3}{2} k_B T \ln(m^{\mathrm{B}}/m^{\mathrm{A}})$. **Note** that this is only true in the absence of constraints.

3.12.2 Thermodynamic integration

GROMACS offers the possibility to integrate eq. 3.137 or eq. 3.138 in one simulation over the full range from A to B. However, if the change is large and insufficient sampling 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 easily be 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 afterward by an appropriate numerical integration procedure.

GROMACS now also supports the use of Bennett's Acceptance Ratio [57] for calculating values of ΔG for transformations from state A to state B using the program `g_bar`. The same data can also be used to calculate free energies using MBAR [58], though the analysis currently requires external tools from the external `pymbar` package, at https://SimTK.org/home/pymbar.

The λ-dependence for the force-field contributions is described in detail in section sec. 4.5.

3.13 Replica exchange

Replica exchange molecular dynamics (REMD) is a method that can be used to speed up the sampling of any type of simulation, especially if conformations are separated by relatively high energy barriers. It involves simulating multiple replicas of the same system at different temperatures and randomly exchanging the complete state of two replicas at regular intervals with the probability:

$$P(1 \leftrightarrow 2) = \min\left(1, \exp\left[\left(\frac{1}{k_B T_1} - \frac{1}{k_B T_2}\right)(U_1 - U_2)\right]\right) \tag{3.140}$$

where T_1 and T_2 are the reference temperatures and U_1 and U_2 are the instantaneous potential energies of replicas 1 and 2 respectively. After exchange the velocities are scaled by $(T_1/T_2)^{\pm 0.5}$ and a neighbor search is performed the next step. This combines the fast sampling and frequent barrier-crossing of the highest temperature with correct Boltzmann sampling at all the different temperatures [59, 60]. We only attempt exchanges for neighboring temperatures as the probability decreases very rapidly with the temperature difference. One should not attempt exchanges for all possible pairs in one step. If, for instance, replicas 1 and 2 would exchange, the chance of exchange for replicas 2 and 3 not only depends on the energies of replicas 2 and 3, but also on the energy of replica 1. In GROMACS this is solved by attempting exchange for all "odd" pairs on "odd" attempts and for all "even" pairs on "even" attempts. If we have four replicas: 0, 1, 2 and 3, ordered in temperature and we attempt exchange every 1000 steps, pairs 0-1 and 2-3 will be tried at steps 1000, 3000 etc. and pair 1-2 at steps 2000, 4000 etc.

How should one choose the temperatures? The energy difference can be written as:

$$U_1 - U_2 = N_{df}\frac{c}{2}k_B(T_1 - T_2) \tag{3.141}$$

where N_{df} is the total number of degrees of freedom of one replica and c is 1 for harmonic potentials and around 2 for protein/water systems. If $T_2 = (1+\epsilon)T_1$ the probability becomes:

$$P(1 \leftrightarrow 2) = \exp\left(-\frac{\epsilon^2 c\, N_{df}}{2(1+\epsilon)}\right) \approx \exp\left(-\epsilon^2 \frac{c}{2} N_{df}\right) \tag{3.142}$$

Thus for a probability of $e^{-2} \approx 0.135$ one obtains $\epsilon \approx 2/\sqrt{c\,N_{df}}$. With all bonds constrained one has $N_{df} \approx 2\,N_{atoms}$ and thus for $c = 2$ one should choose ϵ as $1/\sqrt{N_{atoms}}$. However there is one problem when using pressure coupling. The density at higher temperatures will decrease, leading to higher energy [61], which should be taken into account. The GROMACS website features a so-called "REMD calculator," that lets you type in the temperature range and the number of atoms, and based on that proposes a set of temperatures.

An extension to the REMD for the isobaric-isothermal ensemble was proposed by Okabe *et al.* [62]. In this work the exchange probability is modified to:

$$P(1 \leftrightarrow 2) = \min\left(1, \exp\left[\left(\frac{1}{k_B T_1} - \frac{1}{k_B T_2}\right)(U_1 - U_2) + \left(\frac{P_1}{k_B T_1} - \frac{P_2}{k_B T_2}\right)(V_1 - V_2)\right]\right) \tag{3.143}$$

where P_1 and P_2 are the respective reference pressures and V_1 and V_2 are the respective instantaneous volumes in the simulations. In most cases the differences in volume are so small that the second term is negligible. It only plays a role when the difference between P_1 and P_2 is large or in phase transitions.

Hamiltonian replica exchange is also supported in GROMACS. In Hamiltonian replica exchange, each replica has a different Hamiltonian, defined by the free energy pathway specified for the simulation. The exchange probability to maintain the correct ensemble probabilities is:

$$P(1 \leftrightarrow 2) = \min\left(1, \exp\left[\left(\frac{1}{k_B T} - \frac{1}{k_B T}\right)((U_1(x_2) - U_1(x_1)) + (U_2(x_1) - U_2(x_2)))\right]\right) \tag{3.144}$$

The separate Hamiltonians are defined by the free energy functionality of GROMACS, with swaps made between the different values of λ defined in the mdp file.

Hamiltonian and temperature replica exchange can also be performed simultaneously, using the acceptance criteria:

$$P(1 \leftrightarrow 2) = \min\left(1, \exp\left[\left(\frac{1}{k_B T} - \right)\left(\frac{U_1(x_2) - U_1(x_1)}{k_B T_1} + \frac{U_2(x_1) - U_2(x_2)}{k_B T_2}\right)\right]\right) \tag{3.145}$$

Gibbs sampling replica exchange has also been implemented in GROMACS [63]. In Gibbs sampling replica exchange, all possible pairs are tested for exchange, allowing swaps between replicas that are not neighbors.

Gibbs sampling replica exchange requires no additional potential energy calculations. However there is an additional communication cost in Gibbs sampling replica exchange, as for some permutations, more than one round of swaps must take place. In some cases, this extra communication cost might affect the efficiency.

All replica exchange variants are options of the `mdrun` program. It will only work when MPI is installed, due to the inherent parallelism in the algorithm. For efficiency each replica can run on a separate rank. See the manual page of `mdrun` on how to use these multinode features.

3.14 Essential Dynamics sampling

The results from Essential Dynamics (see sec. 8.10) 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 force (free-energy gradient) on that coordinate in that position. Another application is to enhance sampling efficiency with respect to usual MD [64, 65]. 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.

Another possibility to enhance sampling is flooding. Here a flooding potential is added to certain (collective) degrees of freedom to expel the system out of a region of phase space [66].

The procedure for essential dynamics sampling or flooding is as follows. First, the eigenvectors and eigenvalues need to be determined using covariance analysis (`g_covar`) or normal-mode analysis (`g_nmeig`). Then, this information is fed into `make_edi`, which has many options for selecting vectors and setting parameters, see `gmx make_edi -h`. The generated `edi` input file is then passed to `mdrun`.

3.15 Expanded Ensemble

In an expanded ensemble simulation [67], both the coordinates and the thermodynamic ensemble are treated as configuration variables that can be sampled over. The probability of any given state can be written as:

$$P(\vec{x}, k) \propto \exp\left(-\beta_k U_k + g_k\right), \tag{3.146}$$

where $\beta_k = \frac{1}{k_B T_k}$ is the β corresponding to the kth thermodynamic state, and g_k is a user-specified weight factor corresponding to the kth state. This space is therefore a *mixed*, *generalized*, or *expanded* ensemble which samples from multiple thermodynamic ensembles simultaneously. g_k is chosen to give a specific weighting of each subensemble in the expanded ensemble, and can either be fixed, or determined by an iterative procedure. The set of g_k is frequently chosen to give each thermodynamic ensemble equal probability, in which case g_k is equal to the free energy in non-dimensional units, but they can be set to arbitrary values as desired. Several different algorithms can be used to equilibrate these weights, described in the mdp option listings.

In GROMACS, this space is sampled by alternating sampling in the k and $\vec{x}$ directions. Sampling in the $\vec{x}$ direction is done by standard molecular dynamics sampling; sampling between the different thermodynamics states is done by Monte Carlo, with several different Monte Carlo moves supported. The k states can be defined by different temperatures, or choices of the free energy λ variable, or both. Expanded ensemble simulations thus represent a serialization of the replica exchange formalism, allowing a single simulation to explore many thermodynamic states.

3.16 Parallelization

The CPU time required for a simulation can be reduced by running the simulation in parallel over more than one core. Ideally, one would want to have linear scaling: running on N cores makes the simulation N times faster. In practice this can only be achieved for a small number of cores.

The scaling will depend a lot on the algorithms used. Also, different algorithms can have different restrictions on the interaction ranges between atoms.

3.17 Domain decomposition

Since most interactions in molecular simulations are local, domain decomposition is a natural way to decompose the system. In domain decomposition, a spatial domain is assigned to each rank, which will then integrate the equations of motion for the particles that currently reside in its local domain. With domain decomposition, there are two choices that have to be made: the division of the unit cell into domains and the assignment of the forces to domains. Most molecular simulation packages use the half-shell method for assigning the forces. But there are two methods that always require less communication: the eighth shell [68] and the midpoint [69] method. GROMACS currently uses the eighth shell method, but for certain systems or hardware architectures it might be advantageous to use the midpoint method. Therefore, we might implement the midpoint method in the future. Most of the details of the domain decomposition can be found in the GROMACS 4 paper [5].

3.17.1 Coordinate and force communication

In the most general case of a triclinic unit cell, the space in divided with a 1-, 2-, or 3-D grid in parallelepipeds that we call domain decomposition cells. Each cell is assigned to a particle-particle rank. The system is partitioned over the ranks at the beginning of each MD step in which neighbor searching is performed. Since the neighbor searching is based on charge groups, charge groups are also the units for the domain decomposition. Charge groups are assigned to the cell where their center of geometry resides. Before the forces can be calculated, the coordinates from some neighboring cells need to be communicated, and after the forces are calculated, the forces need to be communicated in the other direction. The communication and force assignment is based on zones that can cover one or multiple cells. An example of a zone setup is shown in Fig. 3.12.

The coordinates are communicated by moving data along the "negative" direction in x, y or z to the next neighbor. This can be done in one or multiple pulses. In Fig. 3.12 two pulses in x are required, then one in y and then one in z. The forces are communicated by reversing this procedure. See the GROMACS 4 paper [5] for details on determining which non-bonded and bonded forces should be calculated on which rank.

3.17.2 Dynamic load balancing

When different ranks have a different computational load (load imbalance), all ranks will have to wait for the one that takes the most time. One would like to avoid such a situation. Load imbalance can occur due to three reasons:

- inhomogeneous particle distribution
- inhomogeneous interaction cost distribution (charged/uncharged, water/non-water due to GROMACS water innerloops)

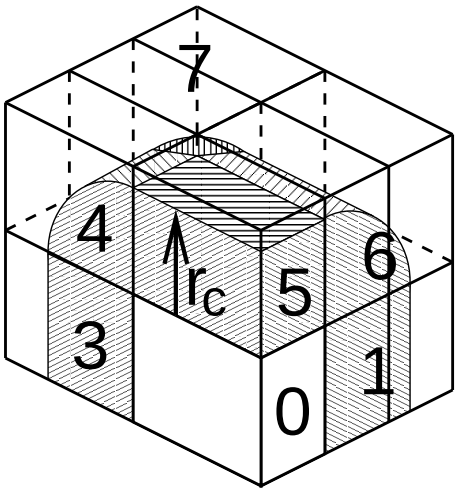


Figure 3.12: A non-staggered domain decomposition grid of 3×2×2 cells. Coordinates in zones 1 to 7 are communicated to the corner cell that has its home particles in zone 0. r_c is the cut-off radius.

- statistical fluctuation (only with small particle numbers)

So we need a dynamic load balancing algorithm where the volume of each domain decomposition cell can be adjusted *independently*. To achieve this, the 2- or 3-D domain decomposition grids need to be staggered. Fig. 3.13 shows the most general case in 2-D. Due to the staggering, one might require two distance checks for deciding if a charge group needs to be communicated: a non-bonded distance and a bonded distance check.

By default, `mdrun` automatically turns on the dynamic load balancing during a simulation when the total performance loss due to the force calculation imbalance is 5% or more. **Note** that the reported force load imbalance numbers might be higher, since the force calculation is only part of work that needs to be done during an integration step. The load imbalance is reported in the log file at log output steps and when the `-v` option is used also on screen. The average load imbalance and the total performance loss due to load imbalance are reported at the end of the log file.

There is one important parameter for the dynamic load balancing, which is the minimum allowed scaling. By default, each dimension of the domain decomposition cell can scale down by at least a factor of 0.8. For 3-D domain decomposition this allows cells to change their volume by about a factor of 0.5, which should allow for compensation of a load imbalance of 100%. The minimum allowed scaling can be changed with the `-dds` option of `mdrun`.

3.17.3 Constraints in parallel

Since with domain decomposition parts of molecules can reside on different ranks, bond constraints can cross cell boundaries. Therefore a parallel constraint algorithm is required. GROMACS uses the P-LINCS algorithm [48], which is the parallel version of the LINCS algorithm [47] . (see 3.6.2). The P-LINCS procedure is illustrated in Fig. 3.14. When molecules cross the cell boundaries, atoms in such molecules up to (`lincs_order + 1`) bonds away are communicated

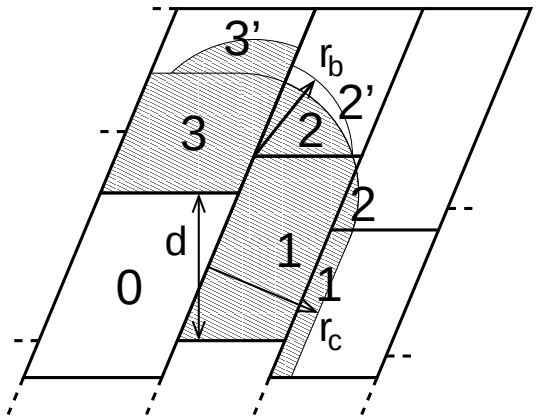


Figure 3.13: The zones to communicate to the rank of zone 0, see the text for details. r_c and r_b are the non-bonded and bonded cut-off radii respectively, d is an example of a distance between following, staggered boundaries of cells.

interaction	range	option	default
non-bonded	r_c = max(r_{list},r_{VdW},r_{Coul})	`mdp` file	
two-body bonded	max(r_{mb},r_c)	`mdrun -rdd`	starting conf. + 10%
multi-body bonded	r_{mb}	`mdrun -rdd`	starting conf. + 10%
constraints	r_{con}	`mdrun -rcon`	est. from bond lengths
virtual sites	r_{con}	`mdrun -rcon`	0

Table 3.3: The interaction ranges with domain decomposition.

over the cell boundaries. Then, the normal LINCS algorithm can be applied to the local bonds plus the communicated ones. After this procedure, the local bonds are correctly constrained, even though the extra communicated ones are not. One coordinate communication step is required for the initial LINCS step and one for each iteration. Forces do not need to be communicated.

3.17.4 Interaction ranges

Domain decomposition takes advantage of the locality of interactions. This means that there will be limitations on the range of interactions. By default, `mdrun` tries to find the optimal balance between interaction range and efficiency. But it can happen that a simulation stops with an error message about missing interactions, or that a simulation might run slightly faster with shorter interaction ranges. A list of interaction ranges and their default values is given in Table 3.3.

In most cases the defaults of `mdrun` should not cause the simulation to stop with an error message of missing interactions. The range for the bonded interactions is determined from the distance between bonded charge-groups in the starting configuration, with 10% added for headroom. For the constraints, the value of r_{con} is determined by taking the maximum distance that (`lincs_order + 1`) bonds can cover when they all connect at angles of 120 degrees. The actual constraint communication is not limited by r_{con}, but by the minimum cell size L_C, which has the following lower

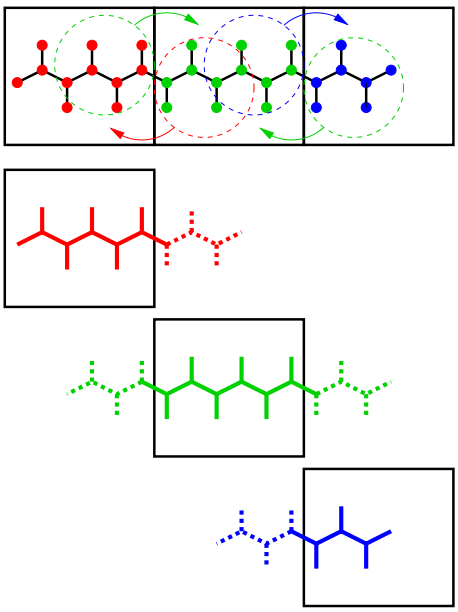


Figure 3.14: Example of the parallel setup of P-LINCS with one molecule split over three domain decomposition cells, using a matrix expansion order of 3. The top part shows which atom coordinates need to be communicated to which cells. The bottom parts show the local constraints (solid) and the non-local constraints (dashed) for each of the three cells.

limit:

$$L_C \geq \max(r_{\mathrm{mb}}, r_{\mathrm{con}}) \tag{3.147}$$

Without dynamic load balancing the system is actually allowed to scale beyond this limit when pressure scaling is used. **Note** that for triclinic boxes, L_C is not simply the box diagonal component divided by the number of cells in that direction, rather it is the shortest distance between the triclinic cells borders. For rhombic dodecahedra this is a factor of $\sqrt{3/2}$ shorter along x and y.

When $r_{\mathrm{mb}} > r_c$, `mdrun` employs a smart algorithm to reduce the communication. Simply communicating all charge groups within r_{mb} would increase the amount of communication enormously. Therefore only charge-groups that are connected by bonded interactions to charge groups which are not locally present are communicated. This leads to little extra communication, but also to a slightly increased cost for the domain decomposition setup. In some cases, *e.g.* coarse-grained simulations with a very short cut-off, one might want to set r_{mb} by hand to reduce this cost.

3.17.5 Multiple-Program, Multiple-Data PME parallelization

Electrostatics interactions are long-range, therefore special algorithms are used to avoid summation over many atom pairs. In GROMACS this is usually . PME (sec. 4.8.2). Since with PME all particles interact with each other, global communication is required. This will usually be the limiting factor for scaling with domain decomposition. To reduce the effect of this problem, we have come up with a Multiple-Program, Multiple-Data approach [5]. Here, some ranks are selected to do only the PME mesh calculation, while the other ranks, called particle-particle (PP) ranks, do all the rest of the work. For rectangular boxes the optimal PP to PME rank ratio is usually 3:1,

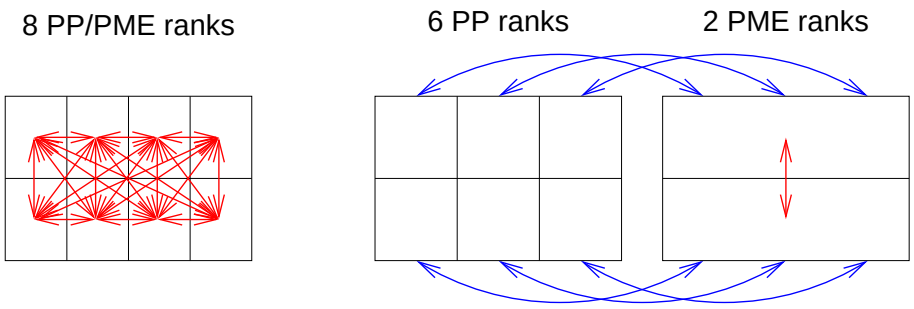


Figure 3.15: Example of 8 ranks without (left) and with (right) MPMD. The PME communication (red arrows) is much higher on the left than on the right. For MPMD additional PP - PME coordinate and force communication (blue arrows) is required, but the total communication complexity is lower.

for rhombic dodecahedra usually 2:1. When the number of PME ranks is reduced by a factor of 4, the number of communication calls is reduced by about a factor of 16. Or put differently, we can now scale to 4 times more ranks. In addition, for modern 4 or 8 core machines in a network, the effective network bandwidth for PME is quadrupled, since only a quarter of the cores will be using the network connection on each machine during the PME calculations.

`mdrun` will by default interleave the PP and PME ranks. If the ranks are not number consecutively inside the machines, one might want to use `mdrun -ddorder pp_pme`. For machines with a real 3-D torus and proper communication software that assigns the ranks accordingly one should use `mdrun -ddorder cartesian`.

To optimize the performance one should usually set up the cut-offs and the PME grid such that the PME load is 25 to 33% of the total calculation load. `grompp` will print an estimate for this load at the end and also `mdrun` calculates the same estimate to determine the optimal number of PME ranks to use. For high parallelization it might be worthwhile to optimize the PME load with the `mdp` settings and/or the number of PME ranks with the `-npme` option of `mdrun`. For changing the electrostatics settings it is useful to know the accuracy of the electrostatics remains nearly constant when the Coulomb cut-off and the PME grid spacing are scaled by the same factor. **Note** that it is usually better to overestimate than to underestimate the number of PME ranks, since the number of PME ranks is smaller than the number of PP ranks, which leads to less total waiting time.

The PME domain decomposition can be 1-D or 2-D along the x and/or y axis. 2-D decomposition is also known as pencil decomposition because of the shape of the domains at high parallelization. 1-D decomposition along the y axis can only be used when the PP decomposition has only 1 domain along x. 2-D PME decomposition has to have the number of domains along x equal to the number of the PP decomposition. `mdrun` automatically chooses 1-D or 2-D PME decomposition (when possible with the total given number of ranks), based on the minimum amount of communication for the coordinate redistribution in PME plus the communication for the grid overlap and transposes. To avoid superfluous communication of coordinates and forces between the PP and PME ranks, the number of DD cells in the x direction should ideally be the same or a multiple of the number of PME ranks. By default, `mdrun` takes care of this issue.

3.17.6 Domain decomposition flow chart

In Fig. 3.16 a flow chart is shown for domain decomposition with all possible communication for different algorithms. For simpler simulations, the same flow chart applies, without the algorithms and communication for the algorithms that are not used.

3.18 Implicit solvation

Implicit solvent models provide an efficient way of representing the electrostatic effects of solvent molecules, while saving a large piece of the computations involved in an accurate, aqueous description of the surrounding water in molecular dynamics simulations. Implicit solvation models offer several advantages compared with explicit solvation, including eliminating the need for the equilibration of water around the solute, and the absence of viscosity, which allows the protein to more quickly explore conformational space.

Implicit solvent calculations in GROMACS can be done using the generalized Born-formalism, and the Still [70], HCT [71], and OBC [72] models are available for calculating the Born radii.

Here, the free energy G_{solv} of solvation is the sum of three terms, a solvent-solvent cavity term (G_{cav}), a solute-solvent van der Waals term (G_{vdw}), and finally a solvent-solute electrostatics polarization term (G_{pol}).

The sum of G_{cav} and G_{vdw} corresponds to the (non-polar) free energy of solvation for a molecule from which all charges have been removed, and is commonly called G_{np}, calculated from the total solvent accessible surface area multiplied with a surface tension. The total expression for the solvation free energy then becomes:

$$G_{\mathrm{solv}} = G_{\mathrm{np}} + G_{\mathrm{pol}} \tag{3.148}$$

Under the generalized Born model, G_{pol} is calculated from the generalized Born equation [70]:

$$G_{\mathrm{pol}} = \left(1 - \frac{1}{\epsilon}\right) \sum_{i=1}^{n} \sum_{j>i}^{n} \frac{q_i q_j}{\sqrt{r_{ij}^2 + b_i b_j \exp\left(\frac{-r_{ij}^2}{4 b_i b_j}\right)}} \tag{3.149}$$

In GROMACS, we have introduced the substitution [73]:

$$c_i = \frac{1}{\sqrt{b_i}} \tag{3.150}$$

which makes it possible to introduce a cheap transformation to a new variable x when evaluating each interaction, such that:

$$x = \frac{r_{ij}}{\sqrt{b_i b_j}} = r_{ij} c_i c_j \tag{3.151}$$

In the end, the full re-formulation of 3.149 becomes:

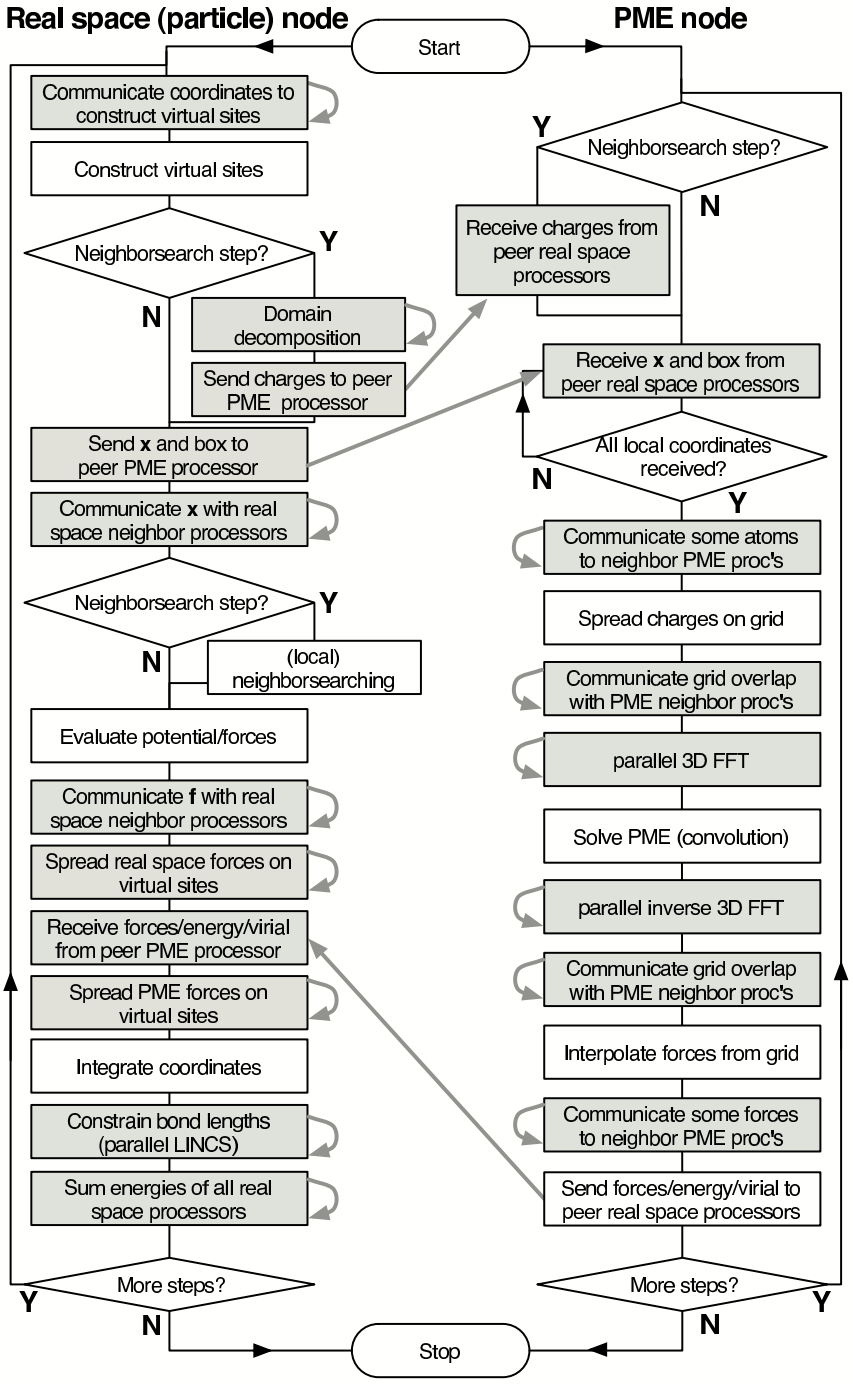


Figure 3.16: Flow chart showing the algorithms and communication (arrows) for a standard MD simulation with virtual sites, constraints and separate PME-mesh ranks.

$$G_{\mathrm{pol}} = \left(1 - \frac{1}{\epsilon}\right) \sum_{i=1}^{n} \sum_{j>i}^{n} \frac{q_i q_j}{\sqrt{b_i b_j}} \, \xi(x) = \left(1 - \frac{1}{\epsilon}\right) \sum_{i=1}^{n} q_i c_i \sum_{j>i}^{n} q_j c_j \, \xi(x) \tag{3.152}$$

The non-polar part (G_{np}) of Equation 3.148 is calculated directly from the Born radius of each atom using a simple ACE type approximation by Schaefer *et al.* [74], including a simple loop over all atoms. This requires only one extra solvation parameter, independent of atom type, but differing slightly between the three Born radii models.

Chapter 4

Interaction function and force fields

To accommodate the potential functions used in some popular force fields (see 4.10), 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. *Restraints*: position restraints, angle restraints, distance restraints, orientation restraints and dihedral restraints, all based on fixed lists.

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.

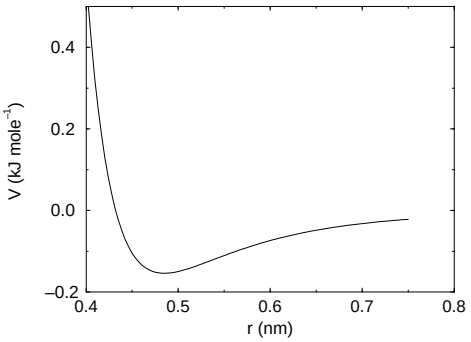


Figure 4.1: The Lennard-Jones interaction.

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. In the Verlet cut-off scheme, the potential is shifted by a constant such that it is zero at the cut-off distance.

The force derived from this potential is:

$$\boldsymbol{F}_i(\boldsymbol{r}_{ij}) = \left(12\, \frac{C_{ij}^{(12)}}{r_{ij}^{13}} - 6\, \frac{C_{ij}^{(6)}}{r_{ij}^{7}} \right) \frac{\boldsymbol{r}_{ij}}{r_{ij}} \tag{4.4}$$

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, only geometric averages (type 1 in the input section of the force-field file):

$$\begin{aligned} C_{ij}^{(6)} &= \left(C_{ii}^{(6)}\, C_{jj}^{(6)} \right)^{1/2} \\ C_{ij}^{(12)} &= \left(C_{ii}^{(12)}\, C_{jj}^{(12)} \right)^{1/2} \end{aligned} \tag{4.6}$$

or, alternatively the Lorentz-Berthelot rules can be used. An arithmetic average is used to calculate σ_{ij}, while a geometric average is used to calculate ϵ_{ij} (type 2):

$$\begin{aligned} \sigma_{ij} &= \tfrac{1}{2}(\sigma_{ii} + \sigma_{jj}) \\ \epsilon_{ij} &= (\epsilon_{ii}\, \epsilon_{jj})^{1/2} \end{aligned} \tag{4.7}$$

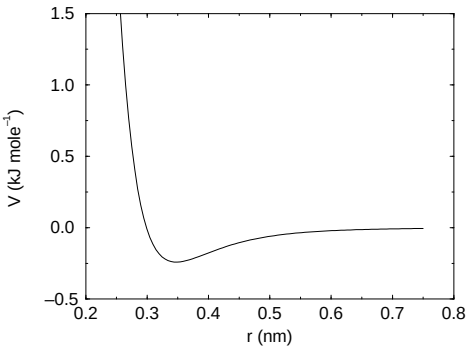


Figure 4.2: The Buckingham interaction.

finally an geometric average for both parameters can be used (type 3):

$$\begin{aligned} \sigma_{ij} &= (\sigma_{ii}\,\sigma_{jj})^{1/2} \\ \epsilon_{ij} &= (\epsilon_{ii}\,\epsilon_{jj})^{1/2} \end{aligned} \tag{4.8}$$

This last rule is used by the OPLS force field.

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:

$$V_{bh}(r_{ij}) = A_{ij}\exp(-B_{ij}r_{ij}) - \frac{C_{ij}}{r_{ij}^6} \tag{4.9}$$

See also Fig. 4.2. The force derived from this is:

$$\boldsymbol{F}_i(r_{ij}) = \left[A_{ij}B_{ij}\exp(-B_{ij}r_{ij}) - 6\frac{C_{ij}}{r_{ij}^7}\right]\frac{\boldsymbol{r}_{ij}}{r_{ij}} \tag{4.10}$$

4.1.3 Coulomb interaction

The Coulomb interaction between two charge particles is given by:

$$V_c(r_{ij}) = f\frac{q_i q_j}{\varepsilon_r r_{ij}} \tag{4.11}$$

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:

$$\boldsymbol{F}_i(\boldsymbol{r}_{ij}) = f\frac{q_i q_j}{\varepsilon_r r_{ij}^2}\frac{\boldsymbol{r}_{ij}}{r_{ij}} \tag{4.12}$$

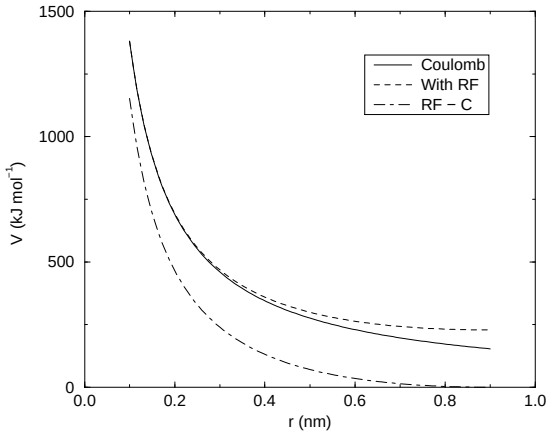


Figure 4.3: The Coulomb interaction (for particles with equal signed charge) with and without reaction field. In the latter case ε_r was 1, ε_{rf} was 78, and r_c was 0.9 nm. The dot-dashed line is the same as the dashed line, except for a constant.

A plain Coulomb interaction should only be used without cut-off or when all pairs fall within the cut-off, since there is an abrupt, large change in the force at the cut-off. In case you do want to use a cut-off, the potential can be shifted by a constant to make the potential the integral of the force. With the group cut-off scheme, this shift is only applied to non-excluded pairs. With the Verlet cut-off scheme, the shift is also applied to excluded pairs and self interactions, which makes the potential equivalent to a reaction field with $\varepsilon_{rf} = 1$ (see below).

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:

$$V_{crf} = f\frac{q_i q_j}{\varepsilon_r r_{ij}}\left[1 + \frac{\varepsilon_{rf}-\varepsilon_r}{2\varepsilon_{rf}+\varepsilon_r}\frac{r_{ij}^3}{r_c^3}\right] - f\frac{q_i q_j}{\varepsilon_r r_c}\frac{3\varepsilon_{rf}}{2\varepsilon_{rf}+\varepsilon_r} \tag{4.13}$$

in which the constant expression on the right makes the potential zero at the cut-off r_c. For charged cut-off spheres this corresponds to neutralization with a homogeneous background charge. We can rewrite eqn. 4.13 for simplicity as

$$V_{crf} = f\frac{q_i q_j}{\varepsilon_r}\left[\frac{1}{r_{ij}} + k_{rf}\, r_{ij}^2 - c_{rf}\right] \tag{4.14}$$

with

$$k_{rf} = \frac{1}{r_c^3}\frac{\varepsilon_{rf}-\varepsilon_r}{(2\varepsilon_{rf}+\varepsilon_r)} \tag{4.15}$$

$$c_{rf} = \frac{1}{r_c} + k_{rf}\, r_c^2 = \frac{1}{r_c}\frac{3\varepsilon_{rf}}{(2\varepsilon_{rf}+\varepsilon_r)} \tag{4.16}$$

For large ε_{rf} the k_{rf} goes to $r_c^{-3}/2$, while for ε_{rf} = ε_r the correction vanishes. 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:

$$\boldsymbol{F}_i(\boldsymbol{r}_{ij}) = f\frac{q_i q_j}{\varepsilon_r}\left[\frac{1}{r_{ij}^2} - 2k_{rf}r_{ij}\right]\frac{\boldsymbol{r}_{ij}}{r_{ij}} \tag{4.17}$$

The reaction-field correction should also be applied to all excluded atoms pairs, including self pairs, in which case the normal Coulomb term in eqns. 4.13 and 4.17 is absent.

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 [75]. In this case we can rewrite the constants k_{rf} and c_{rf} using the inverse Debye screening length κ:

$$\kappa^2 = \frac{2I\,F^2}{\varepsilon_0\varepsilon_{rf}RT} = \frac{F^2}{\varepsilon_0\varepsilon_{rf}RT}\sum_{i=1}^{K} c_i z_i^2 \tag{4.18}$$

$$k_{rf} = \frac{1}{r_c^3}\frac{(\varepsilon_{rf}-\varepsilon_r)(1+\kappa r_c) + \frac{1}{2}\varepsilon_{rf}(\kappa r_c)^2}{(2\varepsilon_{rf}+\varepsilon_r)(1+\kappa r_c) + \varepsilon_{rf}(\kappa r_c)^2} \tag{4.19}$$

$$c_{rf} = \frac{1}{r_c}\frac{3\varepsilon_{rf}(1+\kappa r_c + \frac{1}{2}(\kappa r_c)^2)}{(2\varepsilon_{rf}+\varepsilon_r)(1+\kappa r_c) + \varepsilon_{rf}(\kappa r_c)^2} \tag{4.20}$$

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 ($\kappa = 0$) eqns. 4.19 and 4.20 reduce to the simple forms of eqns. 4.15 and 4.16 respectively.

4.1.5 Modified non-bonded interactions

In GROMACS, 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 timestep 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) [76]. 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:

$$\boldsymbol{F}_i = cF(r_{ij})\frac{\boldsymbol{r}_{ij}}{r_{ij}} \tag{4.21}$$

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:

$$\begin{aligned} F_s(r) &= F_\alpha(r) & r < r_1 \\ F_s(r) &= F_\alpha(r) + S(r) & r_1 \le r < r_c \\ F_s(r) &= 0 & r_c \le r \end{aligned} \tag{4.22}$$

When $r_1 = 0$ this is a traditional shift function, otherwise it acts as a switch function. The corresponding shifted coulomb potential then reads:

$$V_s(r_{ij}) = f\Phi_s(r_{ij})q_iq_j \tag{4.23}$$

where $\Phi(r)$ is the potential function

$$\Phi_s(r) = \int_r^\infty F_s(x)\,dx \tag{4.24}$$

The GROMACS shift function should be smooth at the boundaries, therefore the following boundary conditions are imposed on the shift function:

$$\begin{aligned} S(r_1) &= 0 \\ S'(r_1) &= 0 \\ S(r_c) &= -F_\alpha(r_c) \\ S'(r_c) &= -F'_\alpha(r_c) \end{aligned} \tag{4.25}$$

A 3^{rd} degree polynomial of the form

$$S(r) = A(r - r_1)^2 + B(r - r_1)^3 \tag{4.26}$$

fulfills these requirements. The constants A and B are given by the boundary condition at r_c:

$$\begin{aligned} A &= -\frac{(\alpha+4)r_c - (\alpha+1)r_1}{r_c^{\alpha+2}\,(r_c - r_1)^2} \\ B &= \frac{(\alpha+3)r_c - (\alpha+1)r_1}{r_c^{\alpha+2}\,(r_c - r_1)^3} \end{aligned} \tag{4.27}$$

Thus the total force function is:

$$F_s(r) = \frac{\alpha}{r^{\alpha+1}} + A(r - r_1)^2 + B(r - r_1)^3 \tag{4.28}$$

and the potential function reads:

$$\Phi(r) = \frac{1}{r^\alpha} - \frac{A}{3}(r - r_1)^3 - \frac{B}{4}(r - r_1)^4 - C \tag{4.29}$$

where

$$C = \frac{1}{r_c^\alpha} - \frac{A}{3}(r_c - r_1)^3 - \frac{B}{4}(r_c - r_1)^4 \tag{4.30}$$

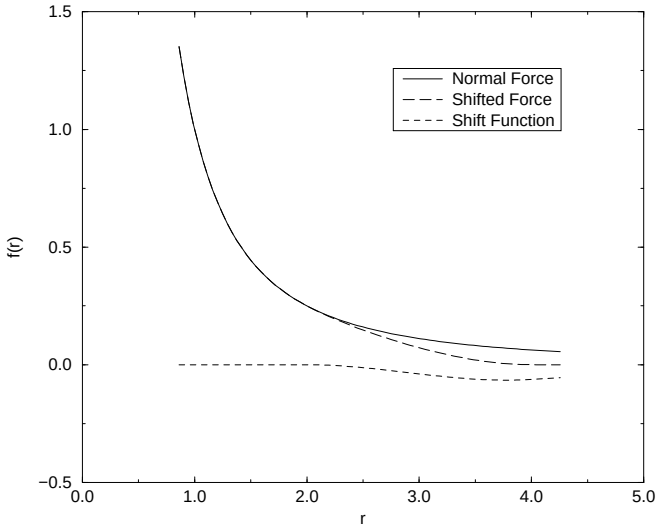


Figure 4.4: The Coulomb Force, Shifted Force and Shift Function $S(r)$, using r_1 = 2 and r_c = 4.

When $r_1 = 0$, the modified Coulomb force function is

$$F_s(r) = \frac{1}{r^2} - \frac{5r^2}{r_c^4} + \frac{4r^3}{r_c^5} \tag{4.31}$$

which is identical to the function recommended to be used as a short-range function in conjunction with a Poisson solver for the long-range part [77]. The modified Coulomb potential function is:

$$\Phi(r) = \frac{1}{r} - \frac{5}{3r_c} + \frac{5r^3}{3r_c^4} - \frac{r^4}{r_c^5} \tag{4.32}$$

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:

$$V(r) = f\frac{\text{erfc}(\beta r_{ij})}{r_{ij}} q_i q_j, \tag{4.33}$$

where β is a parameter that determines the relative weight between the direct space sum and the reciprocal space sum and $\text{erfc}(x)$ is the complementary error function. For further details on long-range electrostatics, see sec. 4.8.

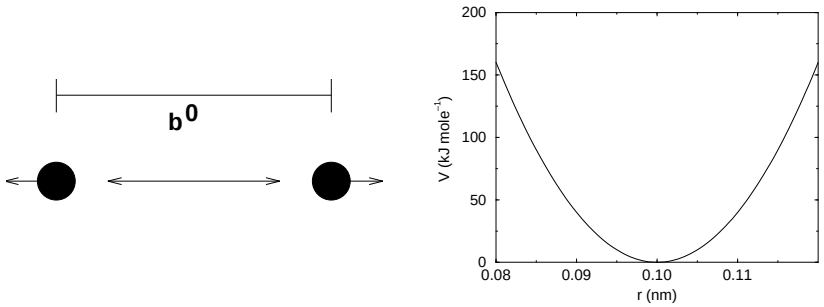


Figure 4.5: Principle of bond stretching (left), and the bond stretching potential (right).

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:

$$V_b\left(r_{ij}\right) = \frac{1}{2} k_{ij}^b (r_{ij} - b_{ij})^2 \tag{4.34}$$

See also Fig. 4.5, with the force given by:

$$\boldsymbol{F}_i(\boldsymbol{r}_{ij}) = k_{ij}^b (r_{ij} - b_{ij}) \frac{\boldsymbol{r}_{ij}}{r_{ij}} \tag{4.35}$$

Fourth power potential

In the GROMOS-96 force field [78], the covalent bond potential is, for reasons of computational efficiency, written as:

$$V_b\left(r_{ij}\right) = \frac{1}{4} k_{ij}^b \left(r_{ij}^2 - b_{ij}^2\right)^2 \tag{4.36}$$

The corresponding force is:

$$\boldsymbol{F}_i(\boldsymbol{r}_{ij}) = k_{ij}^b (r_{ij}^2 - b_{ij}^2)\, \boldsymbol{r}_{ij} \tag{4.37}$$

The force constants for this form of the potential are related to the usual harmonic force constant $k^{b,\mathrm{harm}}$ (sec. 4.2.1) as

$$2k^b b_{ij}^2 = k^{b,\mathrm{harm}} \tag{4.38}$$

The force constants are mostly derived from the harmonic ones used in GROMOS-87 [79]. 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 [80] between two atoms *i* and *j* is available in GROMACS. This potential differs from the harmonic potential in that it has an asymmetric potential well and a zero force at infinite distance. The functional form is:

$$V_{morse}(r_{ij}) = D_{ij}[1 - \exp(-\beta_{ij}(r_{ij} - b_{ij}))]^2, \tag{4.39}$$

See also Fig. 4.6, and the corresponding force is:

$$\begin{aligned} \mathbf{F}_{morse}(\mathbf{r}_{ij}) &= 2D_{ij}\beta_{ij}r_{ij}\exp(-\beta_{ij}(r_{ij} - b_{ij}))* \\ &\quad [1 - \exp(-\beta_{ij}(r_{ij} - b_{ij}))]\frac{\mathbf{r}_{ij}}{r_{ij}}, \end{aligned} \tag{4.40}$$

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}:

$$\beta_{ij} = \omega_{ij}\sqrt{\frac{\mu_{ij}}{2D_{ij}}} \tag{4.41}$$

and because $\omega = \sqrt{k/\mu}$, one can rewrite β_{ij} in terms of the harmonic force constant k_{ij}:

$$\beta_{ij} = \sqrt{\frac{k_{ij}}{2D_{ij}}} \tag{4.42}$$

For small deviations $(r_{ij} - b_{ij})$, one can approximate the exp-term to first-order using a Taylor expansion:

$$\exp(-x) \approx 1 - x \tag{4.43}$$

and substituting eqn. 4.42 and eqn. 4.43 in the functional form:

$$\begin{aligned} V_{morse}(r_{ij}) &= D_{ij}[1 - \exp(-\beta_{ij}(r_{ij} - b_{ij}))]^2 \\ &= D_{ij}[1 - (1 - \sqrt{\tfrac{k_{ij}}{2D_{ij}}}(r_{ij} - b_{ij}))]^2 \\ &= \tfrac{1}{2}k_{ij}(r_{ij} - b_{ij}))^2 \end{aligned} \tag{4.44}$$

we recover the harmonic bond stretching potential.

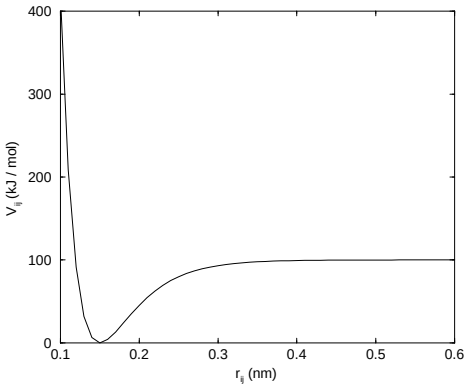


Figure 4.6: The Morse potential well, with bond length 0.15 nm.

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:

$$V_b\left(r_{ij}\right) = k_{ij}^b(r_{ij} - b_{ij})^2 + k_{ij}^b k_{ij}^{cub}(r_{ij} - b_{ij})^3 \tag{4.45}$$

A flexible water model (based on the SPC water model [81]) including a cubic bond stretching potential for the O-H bond was developed by Ferguson [82]. This model was found to yield a reasonable infrared spectrum. The Ferguson water model is available in the GROMACS library (`flexwat-ferguson.itp`). It should be noted that the potential is asymmetric: overstretching leads to infinitely low energies. The integration timestep is therefore limited to 1 fs.

The force corresponding to this potential is:

$$\boldsymbol{F}_i(\boldsymbol{r}_{ij}) = 2k_{ij}^b(r_{ij} - b_{ij})\,\frac{\boldsymbol{r}_{ij}}{r_{ij}} + 3k_{ij}^b k_{ij}^{cub}(r_{ij} - b_{ij})^2\,\frac{\boldsymbol{r}_{ij}}{r_{ij}} \tag{4.46}$$

4.2.4 FENE bond stretching potential

In coarse-grained polymer simulations the beads are often connected by a FENE (finitely extensible nonlinear elastic) potential [83]:

$$V_{\text{FENE}}(r_{ij}) = -\frac{1}{2}k_{ij}^b b_{ij}^2 \log\left(1 - \frac{r_{ij}^2}{b_{ij}^2}\right) \tag{4.47}$$

The potential looks complicated, but the expression for the force is simpler:

$$F_{\text{FENE}}(\boldsymbol{r}_{ij}) = -k_{ij}^b\left(1 - \frac{r_{ij}^2}{b_{ij}^2}\right)^{-1}\boldsymbol{r}_{ij} \tag{4.48}$$

At short distances the potential asymptotically goes to a harmonic potential with force constant k^b, while it diverges at distance b.

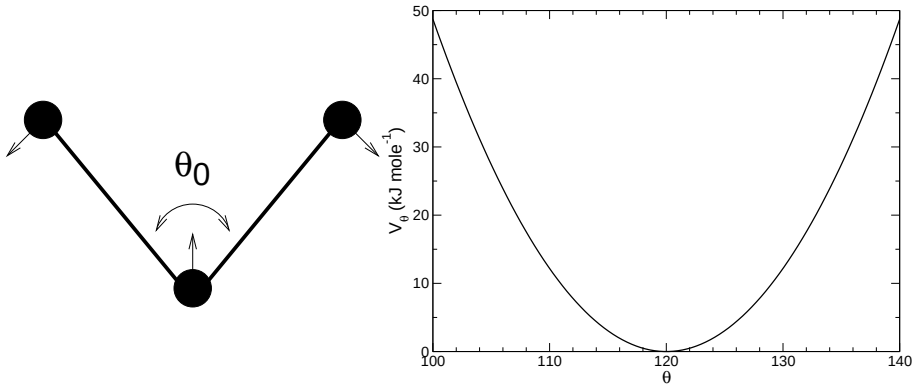


Figure 4.7: Principle of angle vibration (left) and the bond angle potential (right).

4.2.5 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}

$$V_a(\theta_{ijk}) = \frac{1}{2} k^{\theta}_{ijk} (\theta_{ijk} - \theta^0_{ijk})^2 \tag{4.49}$$

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:

$$\begin{aligned} \boldsymbol{F}_i &= -\frac{dV_a(\theta_{ijk})}{d\boldsymbol{r}_i} \\ \boldsymbol{F}_k &= -\frac{dV_a(\theta_{ijk})}{d\boldsymbol{r}_k} \qquad \text{where} \quad \theta_{ijk} = \arccos \frac{(\boldsymbol{r}_{ij} \cdot \boldsymbol{r}_{kj})}{r_{ij} r_{kj}} \\ \boldsymbol{F}_j &= -\boldsymbol{F}_i - \boldsymbol{F}_k \end{aligned} \tag{4.50}$$

The numbering i, j, k is in sequence of covalently bonded atoms. Atom j is in the middle; atoms i and k are at the ends (see Fig. 4.7). **Note** that in the input in topology files, angles are given in degrees and force constants in kJ/mol/rad^2.

4.2.6 Cosine based angle potential

In the GROMOS-96 force field a simplified function is used to represent angle vibrations:

$$V_a(\theta_{ijk}) = \frac{1}{2} k^{\theta}_{ijk} \left(\cos(\theta_{ijk}) - \cos(\theta^0_{ijk}) \right)^2 \tag{4.51}$$

where

$$\cos(\theta_{ijk}) = \frac{\boldsymbol{r}_{ij} \cdot \boldsymbol{r}_{kj}}{r_{ij} r_{kj}} \tag{4.52}$$

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 $k^{\theta,\mathrm{harm}}$ (4.2.5) by:

$$k^{\theta} \sin^2(\theta^0_{ijk}) = k^{\theta,\mathrm{harm}} \tag{4.53}$$

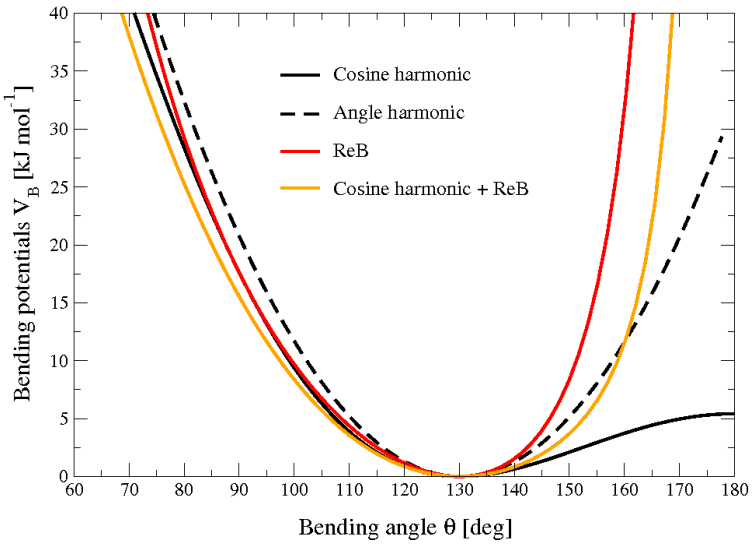


Figure 4.8: Bending angle potentials: cosine harmonic (solid black line), angle harmonic (dashed black line) and restricted bending (red) with the same bending constant $k_\theta = 85$ kJ mol^{-1} and equilibrium angle $\theta_0 = 130°$. The orange line represents the sum of a cosine harmonic ($k = 50$ kJ mol^{-1}) with a restricted bending ($k = 25$ kJ mol^{-1}) potential, both with $\theta_0 = 130°$.

In the GROMOS-96 manual there is a much more complicated conversion formula which is temperature dependent. The formulas are equivalent at 0 K and the differences at 300 K are on the order of 0.1 to 0.2%. **Note** that in the input in topology files, angles are given in degrees and force constants in kJ/mol.

4.2.7 Restricted bending potential

The restricted bending (ReB) potential [84] prevents the bending angle θ from reaching the $180°$ value. In this way, the numerical instabilities due to the calculation of the torsion angle and potential are eliminated when performing coarse-grained molecular dynamics simulations.

To systematically hinder the bending angles from reaching the $180°$ value, the bending potential 4.51 is divided by a $\sin^2\theta$ factor:

$$V_{\mathrm{ReB}}(\theta_i) = \frac{1}{2} k_\theta \frac{(\cos\theta_i - \cos\theta_0)^2}{\sin^2\theta_i}. \tag{4.54}$$

Figure Fig. 4.8 shows the comparison between the ReB potential, 4.54, and the standard one 4.51. The wall of the ReB potential is very repulsive in the region close to $180°$ and, as a result, the bending angles are kept within a safe interval, far from instabilities. The power 2 of $\sin\theta_i$ in the denominator has been chosen to guarantee this behavior and allows an elegant differentiation:

$$F_{\mathrm{ReB}}(\theta_i) = \frac{2k_\theta}{\sin^4\theta_i}(\cos\theta_i - \cos\theta_0)(1 - \cos\theta_i\cos\theta_0)\frac{\partial\cos\theta_i}{\partial\vec{r}_k}. \tag{4.55}$$

Due to its construction, the restricted bending potential cannot be used for equilibrium θ_0 values too close to 0° or 180° (from experience, at least 10° difference is recommended). It is very important that, in the starting configuration, all the bending angles have to be in the safe interval to avoid initial instabilities. This bending potential can be used in combination with any form of torsion potential. It will always prevent three consecutive particles from becoming collinear and, as a result, any torsion potential will remain free of singularities. It can be also added to a standard bending potential to affect the angle around 180°, but to keep its original form around the minimum (see the orange curve in Fig. 4.8).

4.2.8 Urey-Bradley potential

The Urey-Bradley bond-angle vibration between a triplet of atoms i - j - k is represented by a harmonic potential on the angle θ_{ijk} and a harmonic correction term on the distance between the atoms i and k. Although this can be easily written as a simple sum of two terms, it is convenient to have it as a single entry in the topology file and in the output as a separate energy term. It is used mainly in the CHARMm force field [85]. The energy is given by:

$$V_a(\theta_{ijk}) = \frac{1}{2}k^{\theta}_{ijk}(\theta_{ijk} - \theta^0_{ijk})^2 + \frac{1}{2}k^{UB}_{ijk}(r_{ik} - r^0_{ik})^2 \tag{4.56}$$

The force equations can be deduced from sections 4.2.1 and 4.2.5.

4.2.9 Bond-Bond cross term

The bond-bond cross term for three particles i, j, k forming bonds $i-j$ and $k-j$ is given by [86]:

$$V_{rr'} \;=\; k_{rr'}\left(|\boldsymbol{r}_i - \boldsymbol{r}_j| - r_{1e}\right)\left(|\boldsymbol{r}_k - \boldsymbol{r}_j| - r_{2e}\right) \tag{4.57}$$

where $k_{rr'}$ is the force constant, and r_{1e} and r_{2e} are the equilibrium bond lengths of the $i-j$ and $k-j$ bonds respectively. The force associated with this potential on particle i is:

$$\boldsymbol{F}_i = -k_{rr'}\left(|\boldsymbol{r}_k - \boldsymbol{r}_j| - r_{2e}\right)\frac{\boldsymbol{r}_i - \boldsymbol{r}_j}{|\boldsymbol{r}_i - \boldsymbol{r}_j|} \tag{4.58}$$

The force on atom k can be obtained by swapping i and k in the above equation. Finally, the force on atom j follows from the fact that the sum of internal forces should be zero: $\boldsymbol{F}_j = -\boldsymbol{F}_i - \boldsymbol{F}_k$.

4.2.10 Bond-Angle cross term

The bond-angle cross term for three particles i, j, k forming bonds $i-j$ and $k-j$ is given by [86]:

$$V_{r\theta} \;=\; k_{r\theta}\left(|\boldsymbol{r}_i - \boldsymbol{r}_k| - r_{3e}\right)\left(|\boldsymbol{r}_i - \boldsymbol{r}_j| - r_{1e} + |\boldsymbol{r}_k - \boldsymbol{r}_j| - r_{2e}\right) \tag{4.59}$$

where $k_{r\theta}$ is the force constant, r_{3e} is the $i-k$ distance, and the other constants are the same as in Equation 4.57. The force associated with the potential on atom i is:

$$\boldsymbol{F}_i \;=\; -k_{r\theta}\left[\left(|\boldsymbol{r}_i - \boldsymbol{r}_k| - r_{3e}\right)\frac{\boldsymbol{r}_i - \boldsymbol{r}_j}{|\boldsymbol{r}_i - \boldsymbol{r}_j|} + \left(|\boldsymbol{r}_i - \boldsymbol{r}_j| - r_{1e} + |\boldsymbol{r}_k - \boldsymbol{r}_j| - r_{2e}\right)\frac{\boldsymbol{r}_i - \boldsymbol{r}_k}{|\boldsymbol{r}_i - \boldsymbol{r}_k|}\right] \tag{4.60}$$

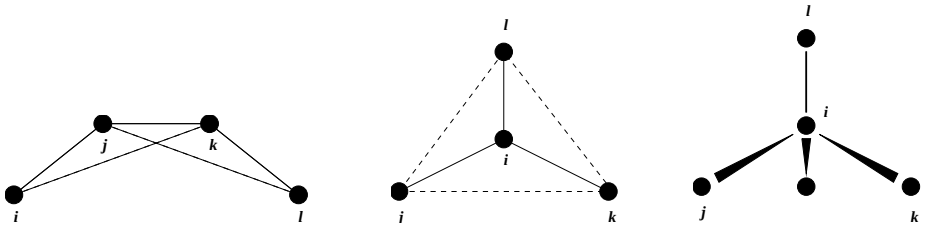


Figure 4.9: Principle of improper dihedral angles. Out of plane bending for rings (left), substituents of rings (middle), out of tetrahedral (right). The improper dihedral angle ξ is defined as the angle between planes (i,j,k) and (j,k,l) in all cases.

4.2.11 Quartic angle potential

For special purposes there is an angle potential that uses a fourth order polynomial:

$$V_q(\theta_{ijk}) \;=\; \sum_{n=0}^{5} C_n(\theta_{ijk} - \theta^0_{ijk})^n \tag{4.61}$$

4.2.12 Improper dihedrals

Improper dihedrals are meant to keep planar groups (*e.g.* aromatic rings) planar, or to prevent molecules from flipping over to their mirror images, see Fig. 4.9.

Improper dihedrals: harmonic type

The simplest improper dihedral potential is a harmonic potential; it is plotted in Fig. 4.10.

$$V_{id}(\xi_{ijkl}) = \frac{1}{2} k_\xi (\xi_{ijkl} - \xi_0)^2 \tag{4.62}$$

Since the potential is harmonic it is discontinuous, but since the discontinuity is chosen at 180° distance from ξ_0 this will never cause problems. **Note** that in the input in topology files, angles are given in degrees and force constants in kJ/mol/rad^2.

Improper dihedrals: periodic type

This potential is identical to the periodic proper dihedral (see below). There is a separate dihedral type for this (type 4) only to be able to distinguish improper from proper dihedrals in the parameter section and the output.

4.2.13 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 choice has consequences for the inclusion of special interactions between the first and the fourth

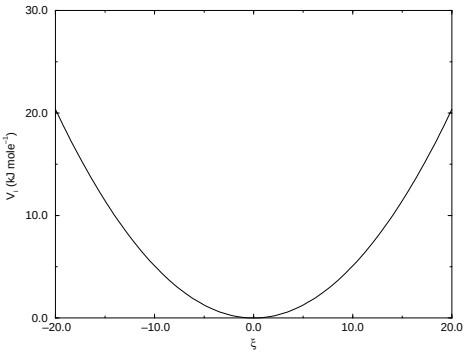


Figure 4.10: Improper dihedral potential.

atom of the dihedral quadruple. With the periodic GROMOS potential a special 1-4 LJ-interaction must be included; with the Ryckaert-Bellemans potential *for alkanes* the 1-4 interactions must be excluded from the non-bonded list. **Note:** Ryckaert-Bellemans potentials are also used in *e.g.* the OPLS force field in combination with 1-4 interactions. You should therefore not modify topologies generated by `pdb2gmx` in this case.

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). There are two dihedral function types in GROMACS topology files. There is the standard type 1 which behaves like any other bonded interactions. For certain force fields, type 9 is useful. Type 9 allows multiple potential functions to be applied automatically to a single dihedral in the `[ dihedral ]` section when multiple parameters are defined for the same atomtypes in the `[ dihedraltypes ]` section.

$$V_d(\phi_{ijkl}) = k_\phi(1 + \cos(n\phi - \phi_s)) \tag{4.63}$$

Proper dihedrals: Ryckaert-Bellemans function

For alkanes, the following proper dihedral potential is often used (see Fig. 4.12):

$$V_{rb}(\phi_{ijkl}) = \sum_{n=0}^{5} C_n(\cos(\psi))^n, \tag{4.64}$$

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.

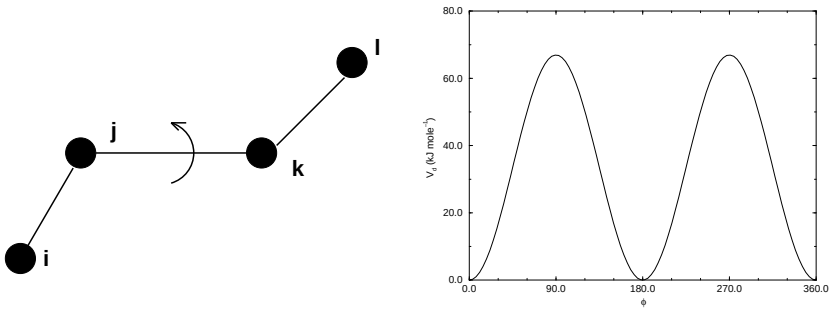


Figure 4.11: Principle of proper dihedral angle (left, in *trans* form) and the dihedral angle potential (right).

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}).

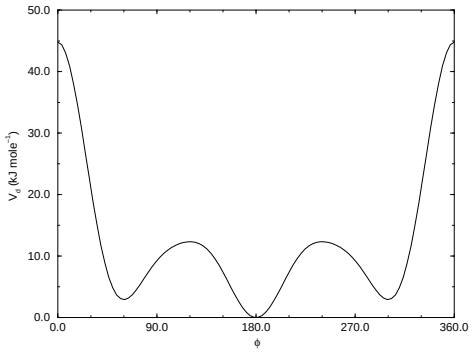


Figure 4.12: Ryckaert-Bellemans dihedral potential.

(**Note:** The use of this potential implies exclusion 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 Fourier dihedrals (see below):

$$V_{rb}(\phi_{ijkl}) \;=\; \frac{1}{2}\left[F_1(1+\cos(\phi)) + F_2(1-\cos(2\phi)) + F_3(1+\cos(3\phi)) + F_4(1-\cos(4\phi))\right] \tag{4.65}$$

Because of the equalities $\cos(2\phi) = 2\cos^2(\phi) - 1$, $\cos(3\phi) = 4\cos^3(\phi) - 3\cos(\phi)$ and $\cos(4\phi) = 8\cos^4(\phi) - 8\cos^2(\phi) + 1$ one can translate the OPLS parameters to Ryckaert-Bellemans parameters as follows:

$$\begin{aligned} C_0 &= F_2 + \tfrac{1}{2}(F_1 + F_3) \\ C_1 &= \tfrac{1}{2}(-F_1 + 3\,F_3) \\ C_2 &= -F_2 + 4\,F_4 \\ C_3 &= -2\,F_3 \\ C_4 &= -4\,F_4 \\ C_5 &= 0 \end{aligned} \tag{4.66}$$

with OPLS parameters in protein convention and RB parameters in polymer convention (this yields a minus sign for the odd powers of $\cos(\phi)$).
Note: Mind the conversion from **kcal mol**$^{-1}$ for literature OPLS and RB parameters to **kJ mol**$^{-1}$ in GROMACS.

Proper dihedrals: Fourier function

The OPLS potential function is given as the first three or four [87] cosine terms of a Fourier series. In GROMACS the four term function is implemented:

$$V_F(\phi_{ijkl}) \;=\; \frac{1}{2}\left[C_1(1+\cos(\phi)) + C_2(1-\cos(2\phi)) + C_3(1+\cos(3\phi)) + C_4(1+\cos(4\phi))\right], \tag{4.67}$$

Internally, GROMACS uses the Ryckaert-Bellemans code to compute Fourier dihedrals (see above), because this is more efficient.
Note: Mind the conversion from *k*cal mol^{-1} for literature OPLS parameters to **kJ mol**$^{-1}$ in GROMACS.

Proper dihedrals: Restricted torsion potential

In a manner very similar to the restricted bending potential (see 4.2.7), a restricted torsion/dihedral potential is introduced:

$$V_{\mathrm{ReT}}(\phi_i) = \frac{1}{2}k_\phi \frac{(\cos\phi_i - \cos\phi_0)^2}{\sin^2\phi_i} \tag{4.68}$$

with the advantages of being a function of $\cos\phi$ (no problems taking the derivative of $\sin\phi$) and of keeping the torsion angle at only one minimum value. In this case, the factor $\sin^2\phi$ does not allow the dihedral angle to move from the $[-180°{:}0]$ to $[0{:}180°]$ interval, i.e. it cannot have maxima both at $-\phi_0$ and $+\phi_0$ maxima, but only one of them. For this reason, all the dihedral

angles of the starting configuration should have their values in the desired angles interval and the the equilibrium ϕ_0 value should not be too close to the interval limits (as for the restricted bending potential, described in 4.2.7, at least 10° difference is recommended).

Proper dihedrals: Combined bending-torsion potential

When the four particles forming the dihedral angle become collinear (this situation will never happen in atomistic simulations, but it can occur in coarse-grained simulations) the calculation of the torsion angle and potential leads to numerical instabilities. One way to avoid this is to use the restricted bending potential (see 4.2.7) that prevents the dihedral from reaching the 180° value.

Another way is to disregard any effects of the dihedral becoming ill-defined, keeping the dihedral force and potential calculation continuous in entire angle range by coupling the torsion potential (in a cosine form) with the bending potentials of the adjacent bending angles in a unique expression:

$$V_{\mathrm{CBT}}(\theta_{i-1},\theta_i,\phi_i) = k_\phi \sin^3\theta_{i-1} \sin^3\theta_i \sum_{n=0}^{4} a_n \cos^n\phi_i. \tag{4.69}$$

This combined bending-torsion (CBT) potential has been proposed by [88] for polymer melt simulations and is extensively described in [84].

This potential has two main advantages:

- it does not only depend on the dihedral angle ϕ_i (between the $i-2$, $i-1$, i and $i+1$ beads) but also on the bending angles θ_{i-1} and θ_i defined from three adjacent beads ($i-2$, $i-1$ and i, and $i-1$, i and $i+1$, respectively). The two $\sin^3\theta$ pre-factors, tentatively suggested by [89] and theoretically discussed by [90], cancel the torsion potential and force when either of the two bending angles approaches the value of 180°.
- its dependence on ϕ_i is expressed through a polynomial in $\cos\phi_i$ that avoids the singularities in $\phi = 0^\circ$ or 180° in calculating the torsional force.

These two properties make the CBT potential well-behaved for MD simulations with weak constraints on the bending angles or even for steered / non-equilibrium MD in which the bending and torsion angles suffer major modifications. When using the CBT potential, the bending potentials for the adjacent θ_{i-1} and θ_i may have any form. It is also possible to leave out the two angle bending terms (θ_{i-1} and θ_i) completely. Fig. 4.13 illustrates the difference between a torsion potential with and without the $\sin^3\theta$ factors (blue and gray curves, respectively). Additionally, the derivative of V_{CBT} with respect to the Cartesian variables is straightforward:

$$\frac{\partial V_{\mathrm{CBT}}(\theta_{i-1},\theta_i,\phi_i)}{\partial \vec{r}_l} = \frac{\partial V_{\mathrm{CBT}}}{\partial \theta_{i-1}}\frac{\partial \theta_{i-1}}{\partial \vec{r}_l} + \frac{\partial V_{\mathrm{CBT}}}{\partial \theta_i}\frac{\partial \theta_i}{\partial \vec{r}_l} + \frac{\partial V_{\mathrm{CBT}}}{\partial \phi_i}\frac{\partial \phi_i}{\partial \vec{r}_l} \tag{4.70}$$

The CBT is based on a cosine form without multiplicity, so it can only be symmetrical around 0°. To obtain an asymmetrical dihedral angle distribution (e.g. only one maximum in $[-180^\circ{:}180^\circ]$ interval), a standard torsion potential such as harmonic angle or periodic cosine potentials should be used instead of a CBT potential. However, these two forms have the inconveniences of the force derivation ($1/\sin\phi$) and of the alignment of beads (θ_i or $\theta_{i-1} = 0^\circ, 180^\circ$). Coupling such non-$\cos\phi$ potentials with $\sin^3\theta$ factors does not improve simulation stability since there are cases

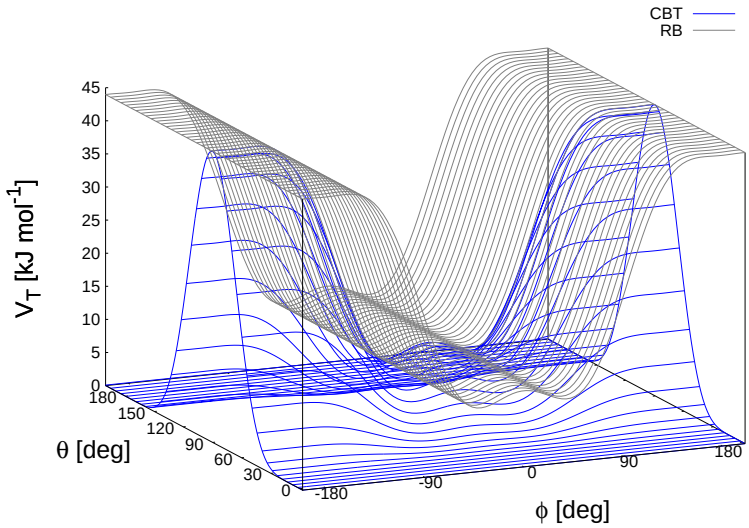


Figure 4.13: Blue: surface plot of the combined bending-torsion potential (4.69 with $k = 10$ kJ mol^{-1}, $a_0 = 2.41$, $a_1 = -2.95$, $a_2 = 0.36$, $a_3 = 1.33$) when, for simplicity, the bending angles behave the same ($\theta_1 = \theta_2 = \theta$). Gray: the same torsion potential without the $\sin^3\theta$ terms (Ryckaert-Bellemans type). ϕ is the dihedral angle.

in which θ and ϕ are simultaneously $180°$. The integration at this step would be possible (due to the cancelling of the torsion potential) but the next step would be singular (θ is not $180°$ and ϕ is very close to $180°$).

4.2.14 Tabulated bonded interaction functions

For full flexibility, any functional shape can be used for bonds, angles and dihedrals through user-supplied tabulated functions. The functional shapes are:

$$V_b(r_{ij}) = k\, f_n^b(r_{ij}) \quad (4.71)$$
$$V_a(\theta_{ijk}) = k\, f_n^a(\theta_{ijk}) \quad (4.72)$$
$$V_d(\phi_{ijkl}) = k\, f_n^d(\phi_{ijkl}) \quad (4.73)$$

where k is a force constant in units of energy and f is a cubic spline function; for details see 6.10.1. For each interaction, the force constant k and the table number n are specified in the topology. There are two different types of bonds, one that generates exclusions (type 8) and one that does not (type 9). For details see Table 5.5. The table files are supplied to the `mdrun` program. After the table file name an underscore, the letter "b" for bonds, "a" for angles or "d" for dihedrals and the table number are appended. For example, for a bond with $n = 0$ (and using the default table file name) the table is read from the file `table_b0.xvg`. Multiple tables can be supplied simply by using different values of n, and are applied to the appropriate bonds, as specified in the topology (Table 5.5). The format for the table files is three columns with x, $f(x)$, $-f'(x)$, where x should be uniformly-spaced. Requirements for entries in the topology are given in Table 5.5. The setup of the tables is as follows:

bonds: x is the distance in nm. For distances beyond the table length, `mdrun` will quit with an error message.
angles: x is the angle in degrees. The table should go from 0 up to and including 180 degrees; the derivative is taken in degrees.
dihedrals: x is the dihedral angle in degrees. The table should go from -180 up to and including 180 degrees; the IUPAC/IUB convention is used, *i.e.* zero is cis, the derivative is taken in degrees.

4.3 Restraints

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. Restraints and constraints refer to quite different algorithms in GROMACS.

4.3.1 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 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 feature has not, however, been implemented in GROMACS.

The following form is used:

$$V_{pr}(\boldsymbol{r}_i) = \frac{1}{2} k_{pr} |\boldsymbol{r}_i - \boldsymbol{R}_i|^2 \tag{4.74}$$

The potential is plotted in Fig. 4.14.

The potential form can be rewritten without loss of generality as:

$$V_{pr}(\boldsymbol{r}_i) = \frac{1}{2} \left[k^x_{pr} (x_i - X_i)^2 \, \hat{\mathbf{x}} + k^y_{pr} (y_i - Y_i)^2 \, \hat{\mathbf{y}} + k^z_{pr} (z_i - Z_i)^2 \, \hat{\mathbf{z}} \right] \tag{4.75}$$

Now the forces are:

$$\begin{array}{rcl} F^x_i & = & -k^x_{pr} \, (x_i - X_i) \\ F^y_i & = & -k^y_{pr} \, (y_i - Y_i) \\ F^z_i & = & -k^z_{pr} \, (z_i - Z_i) \end{array} \tag{4.76}$$

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.

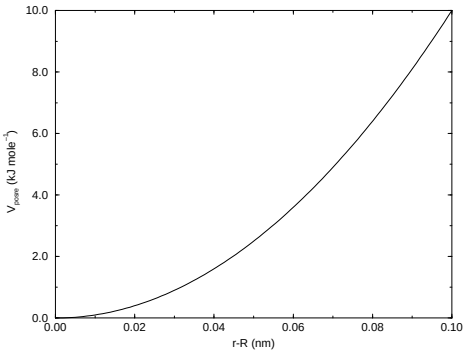


Figure 4.14: Position restraint potential.

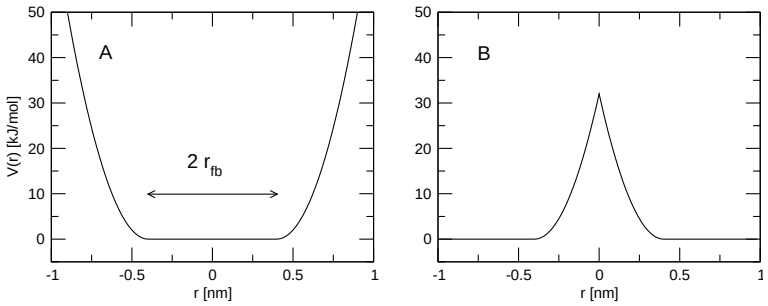


Figure 4.15: Flat-bottomed position restraint potential. (A) Not inverted, (B) inverted.

4.3.2 Flat-bottomed position restraints

Flat-bottomed position restraints can be used to restrain particles to part of the simulation volume. No force acts on the restrained particle within the flat-bottomed region of the potential, however a harmonic force acts to move the particle to the flat-bottomed region if it is outside it. It is possible to apply normal and flat-bottomed position restraints on the same particle (however, only with the same reference position $\boldsymbol{R}_i$). The following general potential is used (Figure 4.15A):

$$V_{\mathrm{fb}}(\boldsymbol{r}_i) = \frac{1}{2} k_{\mathrm{fb}} [d_g(\boldsymbol{r}_i; \boldsymbol{R}_i) - r_{\mathrm{fb}}]^2 \, H[d_g(\boldsymbol{r}_i; \boldsymbol{R}_i) - r_{\mathrm{fb}}], \tag{4.77}$$

where $\boldsymbol{R}_i$ is the reference position, r_{fb} is the distance from the center with a flat potential, k_{fb} the force constant, and H is the Heaviside step function. The distance $d_g(\boldsymbol{r}_i; \boldsymbol{R}_i)$ from the reference position depends on the geometry g of the flat-bottomed potential.

The following geometries for the flat-bottomed potential are supported:
Sphere ($g = 1$): The particle is kept in a sphere of given radius. The force acts towards the center of the sphere. The following distance calculation is used:

$$d_g(\boldsymbol{r}_i; \boldsymbol{R}_i) = |\boldsymbol{r}_i - \boldsymbol{R}_i| \tag{4.78}$$

Cylinder ($g = 2$): The particle is kept in a cylinder of given radius parallel to the z-axis. The force from the flat-bottomed potential acts towards the axis of the cylinder. The z-component of the force is zero.

$$d_g(\boldsymbol{r}_i; \boldsymbol{R}_i) = \sqrt{(x_i - X_i)^2 + (y_i - Y_i)^2} \tag{4.79}$$

Layer ($g = 3, 4, 5$): The particle is kept in a layer defined by the thickness and the normal of the layer. The layer normal can be parallel to the x, y, or z-axis. The force acts parallel to the layer normal.

$$d_g(\boldsymbol{r}_i; \boldsymbol{R}_i) = |x_i - X_i|, \quad \text{or} \quad d_g(\boldsymbol{r}_i; \boldsymbol{R}_i) = |y_i - Y_i|, \quad \text{or} \quad d_g(\boldsymbol{r}_i; \boldsymbol{R}_i) = |z_i - Z_i|. \tag{4.80}$$

It is possible to apply multiple independent flat-bottomed position restraints of different geometry on one particle. For example, applying a cylinder and a layer in z keeps a particle within a disk. Applying three layers in x, y, and z keeps the particle within a cuboid.

In addition, it is possible to invert the restrained region with the unrestrained region, leading to a potential that acts to keep the particle *outside* of the volume defined by $\boldsymbol{R}_i$, g, and r_{fb}. That feature is switched on by defining a negative r_{fb} in the topology. The following potential is used (Figure 4.15B):

$$V_{\mathrm{fb}}^{\mathrm{inv}}(\boldsymbol{r}_i) = \frac{1}{2} k_{\mathrm{fb}} [d_g(\boldsymbol{r}_i; \boldsymbol{R}_i) - |r_{\mathrm{fb}}|]^2 \, H[-(d_g(\boldsymbol{r}_i; \boldsymbol{R}_i) - |r_{\mathrm{fb}}|)]. \tag{4.81}$$

4.3.3 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:

$$V_{ar}(\boldsymbol{r}_i, \boldsymbol{r}_j, \boldsymbol{r}_k, \boldsymbol{r}_l) = k_{ar}(1 - \cos(n(\theta - \theta_0))), \quad \text{where } \theta = \arccos\left(\frac{\boldsymbol{r}_j - \boldsymbol{r}_i}{\|\boldsymbol{r}_j - \boldsymbol{r}_i\|} \cdot \frac{\boldsymbol{r}_l - \boldsymbol{r}_k}{\|\boldsymbol{r}_l - \boldsymbol{r}_k\|} \right) \tag{4.82}$$

For one pair of atoms and the z-axis:

$$V_{ar}(\boldsymbol{r}_i, \boldsymbol{r}_j) = k_{ar}(1 - \cos(n(\theta - \theta_0))), \quad \text{where } \theta = \arccos\left(\frac{\boldsymbol{r}_j - \boldsymbol{r}_i}{\|\boldsymbol{r}_j - \boldsymbol{r}_i\|} \cdot \begin{pmatrix} 0 \\ 0 \\ 1 \end{pmatrix} \right) \tag{4.83}$$

A multiplicity (n) of 2 is useful when you do not want to distinguish between parallel and anti-parallel vectors. The equilibrium angle θ should be between 0 and 180 degrees for multiplicity 1 and between 0 and 90 degrees for multiplicity 2.

4.3.4 Dihedral restraints

These are used to restrain the dihedral angle ϕ defined by four particles as in an improper dihedral (sec. 4.2.12) but with a slightly modified potential. Using:

$$\phi' = (\phi - \phi_0) \ \mathrm{MOD}\ 2\pi \tag{4.84}$$

where ϕ_0 is the reference angle, the potential is defined as:

$$V_{dihr}(\phi') = \begin{cases} \frac{1}{2}k_{dihr}(\phi' - \phi_0 - \Delta\phi)^2 & \text{for } \phi' > \Delta\phi \\ 0 & \text{for } \phi' \leq \Delta\phi \end{cases} \tag{4.85}$$

where $\Delta\phi$ is a user defined angle and k_{dihr} is the force constant. **Note** that in the input in topology files, angles are given in degrees and force constants in kJ/mol/rad^2.

4.3.5 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 from, for instance, experiments in nuclear magnetic resonance (NMR), on the motion of the system. Thus, MD can be used for structure refinement using NMR data. In GROMACS there are three ways to impose restraints on pairs of atoms:

- Simple harmonic restraints: use `[ bonds ]` type 6 . (see sec. 5.4).
- Piecewise linear/harmonic restraints: `[ bonds ]` type 10.
- Complex NMR distance restraints, optionally with pair, time and/or ensemble averaging.

The last two options will be detailed now.

The potential form for distance restraints is quadratic below a specified lower bound and between two specified upper bounds, and linear beyond the largest bound (see Fig. 4.16).

$$V_{dr}(r_{ij}) = \begin{cases} \frac{1}{2}k_{dr}(r_{ij} - r_0)^2 & \text{for } r_{ij} < r_0 \\ 0 & \text{for } r_0 \leq r_{ij} < r_1 \\ \frac{1}{2}k_{dr}(r_{ij} - r_1)^2 & \text{for } r_1 \leq r_{ij} < r_2 \\ \frac{1}{2}k_{dr}(r_2 - r_1)(2r_{ij} - r_2 - r_1) & \text{for } r_2 \leq r_{ij} \end{cases} \tag{4.86}$$

The forces are

$$\boldsymbol{F}_i = \begin{cases} -k_{dr}(r_{ij} - r_0)\frac{\boldsymbol{r}_{ij}}{r_{ij}} & \text{for } r_{ij} < r_0 \\ 0 & \text{for } r_0 \leq r_{ij} < r_1 \\ -k_{dr}(r_{ij} - r_1)\frac{\boldsymbol{r}_{ij}}{r_{ij}} & \text{for } r_1 \leq r_{ij} < r_2 \\ -k_{dr}(r_2 - r_1)\frac{\boldsymbol{r}_{ij}}{r_{ij}} & \text{for } r_2 \leq r_{ij} \end{cases} \tag{4.87}$$

For restraints not derived from NMR data, this functionality will usually suffice and a section of `[ bonds ]` type 10 can be used to apply individual restraints between pairs of atoms, see 5.7.1. For applying restraints derived from NMR measurements, more complex functionality might be required, which is provided through the `[ distance_restraints ]` section and is described below.

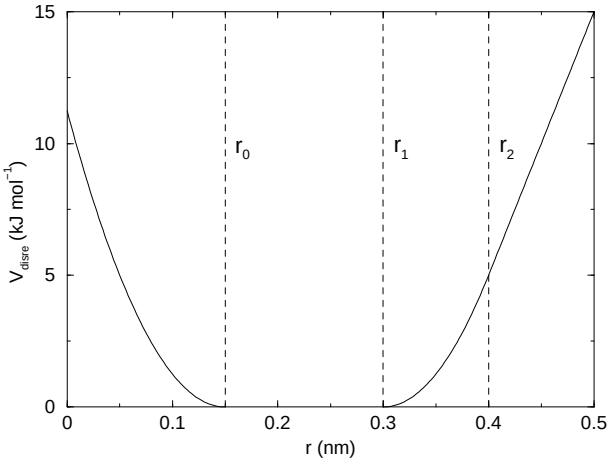


Figure 4.16: Distance Restraint potential.

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* distance [91]. The forces with time averaging are:

$$\boldsymbol{F}_i = \begin{cases} -k^a_{dr}(\bar{r}_{ij} - r_0)\frac{\boldsymbol{r}_{ij}}{r_{ij}} & \text{for} \quad \bar{r}_{ij} < r_0 \\ 0 & \text{for} \quad r_0 \leq \bar{r}_{ij} < r_1 \\ -k^a_{dr}(\bar{r}_{ij} - r_1)\frac{\boldsymbol{r}_{ij}}{r_{ij}} & \text{for} \quad r_1 \leq \bar{r}_{ij} < r_2 \\ -k^a_{dr}(r_2 - r_1)\frac{\boldsymbol{r}_{ij}}{r_{ij}} & \text{for} \quad r_2 \leq \bar{r}_{ij} \end{cases} \tag{4.88}$$

where $\bar{r}_{ij}$ is given by an exponential running average with decay time τ:

$$\bar{r}_{ij} = < r_{ij}^{-3} >^{-1/3} \tag{4.89}$$

The force constant k^a_{dr} is switched on slowly to compensate for the lack of history at the beginning of the simulation:

$$k^a_{dr} = k_{dr}\left(1 - \exp\left(-\frac{t}{\tau}\right)\right) \tag{4.90}$$

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 that is rotating around its χ dihedral angle, thereby coming close to various other groups. Such a mobile side chain can give rise to multiple NOEs that can not be fulfilled by a single structure.

The computation of the time averaged distance in the `mdrun` program is done in the following fashion:

$$\begin{aligned} \overline{r^{-3}}_{ij}(0) &= r_{ij}(0)^{-3} \\ \overline{r^{-3}}_{ij}(t) &= \overline{r^{-3}}_{ij}(t - \Delta t)\ \exp\left(-\tfrac{\Delta t}{\tau}\right) + r_{ij}(t)^{-3}\left[1 - \exp\left(-\tfrac{\Delta t}{\tau}\right)\right] \end{aligned} \tag{4.91}$$

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:

$$\boldsymbol{F}_i = \begin{cases} k^a_{dr}\sqrt{(r_{ij}-r_0)(\bar{r}_{ij}-r_0)}\frac{\boldsymbol{r}_{ij}}{r_{ij}} & \text{for} \quad r_{ij} < r_0 \quad \text{and} \quad \bar{r}_{ij} < r_0 \\ -k^a_{dr}\min\left(\sqrt{(r_{ij}-r_1)(\bar{r}_{ij}-r_1)}, r_2-r_1\right)\frac{\boldsymbol{r}_{ij}}{r_{ij}} & \text{for} \quad r_{ij} > r_1 \quad \text{and} \quad \bar{r}_{ij} > r_1 \\ 0 & \text{otherwise} \end{cases} \tag{4.92}$$

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 inversely proportional to the sixth power of the inter-atomic distance. Thus, when combining atom pairs, a fixed list of N restraints may be taken together, where the apparent "distance" is given by:

$$r_N(t) = \left[\sum_{n=1}^{N} \bar{r}_n(t)^{-6}\right]^{-1/6} \tag{4.93}$$

where we use r_{ij} or eqn. 4.89 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 derivative of the restraint potential with respect to the coordinates. This results in a conservative potential when time averaging is not 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 molecule that is "too rigid." 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:

$$\begin{aligned} r_1 &= r_1 * M^{-1/6} \\ r_2 &= r_2 * M^{-1/6} \end{aligned} \tag{4.94}$$

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:

$$\boldsymbol{F}_i = \begin{cases} 0 & r_N < r_1 \\ k_{dr}(r_N - r_1)\frac{\boldsymbol{r}_{ij}}{r_{ij}} & r_1 \leq r_N < r_2 \\ k_{dr}(r_2 - r_1)\frac{\boldsymbol{r}_{ij}}{r_{ij}} & r_N \geq r_2 \end{cases} \tag{4.95}$$

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:

```
[ 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 4.3.5, 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 must 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 that 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. 4.86. 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.

4.3.6 Orientation restraints

This section describes how orientations between vectors, as measured in certain NMR experiments, can be calculated and restrained in MD simulations. The presented refinement methodology and a comparison of results with and without time and ensemble averaging have been published [92].

Theory

In an NMR experiment, orientations of vectors can be measured when a molecule does not tumble completely isotropically in the solvent. Two examples of such orientation measurements are

residual dipolar couplings (between two nuclei) or chemical shift anisotropies. An observable for a vector $\boldsymbol{r}_i$ can be written as follows:

$$\delta_i = \frac{2}{3}\mathrm{tr}(\mathbf{S}\mathbf{D}_i) \tag{4.96}$$

where $\mathbf{S}$ is the dimensionless order tensor of the molecule. The tensor $\mathbf{D}_i$ is given by:

$$\mathbf{D}_i = \frac{c_i}{\|\boldsymbol{r}_i\|^\alpha}\begin{pmatrix} 3xx-1 & 3xy & 3xz \\ 3xy & 3yy-1 & 3yz \\ 3xz & 3yz & 3zz-1 \end{pmatrix} \tag{4.97}$$

$$\text{with:}\quad x = \frac{r_{i,x}}{\|\boldsymbol{r}_i\|},\quad y = \frac{r_{i,y}}{\|\boldsymbol{r}_i\|},\quad z = \frac{r_{i,z}}{\|\boldsymbol{r}_i\|} \tag{4.98}$$

For a dipolar coupling $\boldsymbol{r}_i$ is the vector connecting the two nuclei, $\alpha = 3$ and the constant c_i is given by:

$$c_i = \frac{\mu_0}{4\pi}\gamma_1^i\gamma_2^i\frac{\hbar}{4\pi} \tag{4.99}$$

where γ_1^i and γ_2^i are the gyromagnetic ratios of the two nuclei.

The order tensor is symmetric and has trace zero. Using a rotation matrix $\mathbf{T}$ it can be transformed into the following form:

$$\mathbf{T}^T\mathbf{S}\mathbf{T} = s\begin{pmatrix} -\frac{1}{2}(1-\eta) & 0 & 0 \\ 0 & -\frac{1}{2}(1+\eta) & 0 \\ 0 & 0 & 1 \end{pmatrix} \tag{4.100}$$

where $-1 \leq s \leq 1$ and $0 \leq \eta \leq 1$. s is called the order parameter and η the asymmetry of the order tensor $\mathbf{S}$. When the molecule tumbles isotropically in the solvent, s is zero, and no orientational effects can be observed because all δ_i are zero.

Calculating orientations in a simulation

For reasons which are explained below, the $\mathbf{D}$ matrices are calculated which respect to a reference orientation of the molecule. The orientation is defined by a rotation matrix $\mathbf{R}$, which is needed to least-squares fit the current coordinates of a selected set of atoms onto a reference conformation. The reference conformation is the starting conformation of the simulation. In case of ensemble averaging, which will be treated later, the structure is taken from the first subsystem. The calculated $\mathbf{D}_i^c$ matrix is given by:

$$\mathbf{D}_i^c(t) = \mathbf{R}(t)\mathbf{D}_i(t)\mathbf{R}^T(t) \tag{4.101}$$

The calculated orientation for vector i is given by:

$$\delta_i^c(t) = \frac{2}{3}\mathrm{tr}(\mathbf{S}(t)\mathbf{D}_i^c(t)) \tag{4.102}$$

The order tensor $\mathbf{S}(t)$ is usually unknown. A reasonable choice for the order tensor is the tensor which minimizes the (weighted) mean square difference between the calculated and the observed orientations:

$$MSD(t) = \left(\sum_{i=1}^N w_i\right)^{-1}\sum_{i=1}^N w_i(\delta_i^c(t) - \delta_i^{exp})^2 \tag{4.103}$$

To properly combine different types of measurements, the unit of w_i should be such that all terms are dimensionless. This means the unit of w_i is the unit of δ_i to the power -2. **Note** that scaling all w_i with a constant factor does not influence the order tensor.

Time averaging

Since the tensors $\mathbf{D}_i$ fluctuate rapidly in time, much faster than can be observed in an experiment, they should be averaged over time in the simulation. However, in a simulation the time and the number of copies of a molecule are limited. Usually one can not obtain a converged average of the $\mathbf{D}_i$ tensors over all orientations of the molecule. If one assumes that the average orientations of the $\boldsymbol{r}_i$ vectors within the molecule converge much faster than the tumbling time of the molecule, the tensor can be averaged in an axis system that rotates with the molecule, as expressed by equation (4.101). The time-averaged tensors are calculated using an exponentially decaying memory function:

$$\mathbf{D}_i^a(t) = \frac{\int_{u=t_0}^{t} \mathbf{D}_i^c(u) \exp\left(-\frac{t-u}{\tau}\right) \mathrm{d}u}{\int_{u=t_0}^{t} \exp\left(-\frac{t-u}{\tau}\right) \mathrm{d}u} \tag{4.104}$$

Assuming that the order tensor $\mathbf{S}$ fluctuates slower than the $\mathbf{D}_i$, the time-averaged orientation can be calculated as:

$$\delta_i^a(t) = \frac{2}{3}\mathrm{tr}(\mathbf{S}(t)\mathbf{D}_i^a(t)) \tag{4.105}$$

where the order tensor $\mathbf{S}(t)$ is calculated using expression (4.103) with $\delta_i^c(t)$ replaced by $\delta_i^a(t)$.

Restraining

The simulated structure can be restrained by applying a force proportional to the difference between the calculated and the experimental orientations. When no time averaging is applied, a proper potential can be defined as:

$$V = \frac{1}{2}k\sum_{i=1}^{N} w_i(\delta_i^c(t) - \delta_i^{exp})^2 \tag{4.106}$$

where the unit of k is the unit of energy. Thus the effective force constant for restraint i is kw_i. The forces are given by minus the gradient of V. The force $\boldsymbol{F}_i$ working on vector $\boldsymbol{r}_i$ is:

$$\begin{aligned}
\boldsymbol{F}_i(t) &= -\frac{\mathrm{d}V}{\mathrm{d}\boldsymbol{r}_i} \\
&= -kw_i(\delta_i^c(t) - \delta_i^{exp})\frac{\mathrm{d}\delta_i(t)}{\mathrm{d}\boldsymbol{r}_i} \\
&= -kw_i(\delta_i^c(t) - \delta_i^{exp})\frac{2c_i}{\|\boldsymbol{r}\|^{2+\alpha}}\left(2\mathbf{R}^T\mathbf{S}\mathbf{R}\boldsymbol{r}_i - \frac{2+\alpha}{\|\boldsymbol{r}\|^2}\mathrm{tr}(\mathbf{R}^T\mathbf{S}\mathbf{R}\boldsymbol{r}_i\boldsymbol{r}_i^T)\boldsymbol{r}_i\right)
\end{aligned}$$

Ensemble averaging

Ensemble averaging can be applied by simulating a system of M subsystems that each contain an identical set of orientation restraints. The systems only interact via the orientation restraint potential which is defined as:

$$V = M\frac{1}{2}k\sum_{i=1}^{N} w_i\langle\delta_i^c(t) - \delta_i^{exp}\rangle^2 \tag{4.107}$$

The force on vector $\boldsymbol{r}_{i,m}$ in subsystem m is given by:

$$\boldsymbol{F}_{i,m}(t) = -\frac{\mathrm{d}V}{\mathrm{d}\boldsymbol{r}_{i,m}} = -kw_i\langle\delta_i^c(t) - \delta_i^{exp}\rangle\frac{\mathrm{d}\delta_{i,m}^c(t)}{\mathrm{d}\boldsymbol{r}_{i,m}} \tag{4.108}$$

Time averaging

When using time averaging it is not possible to define a potential. We can still define a quantity that gives a rough idea of the energy stored in the restraints:

$$V = M\frac{1}{2}k^a\sum_{i=1}^{N} w_i\langle\delta_i^a(t) - \delta_i^{exp}\rangle^2 \tag{4.109}$$

The force constant k_a is switched on slowly to compensate for the lack of history at times close to t_0. It is exactly proportional to the amount of average that has been accumulated:

$$k^a = k\,\frac{1}{\tau}\int_{u=t_0}^{t}\exp\left(-\frac{t-u}{\tau}\right)\mathrm{d}u \tag{4.110}$$

What really matters is the definition of the force. It is chosen to be proportional to the square root of the product of the time-averaged and the instantaneous deviation. Using only the time-averaged deviation induces large oscillations. The force is given by:

$$\boldsymbol{F}_{i,m}(t) = \begin{cases} 0 & \text{for} \quad a\,b \le 0 \\ k^a w_i \dfrac{a}{|a|}\sqrt{a\,b}\,\dfrac{\mathrm{d}\delta_{i,m}^c(t)}{\mathrm{d}\boldsymbol{r}_{i,m}} & \text{for} \quad a\,b > 0 \end{cases} \tag{4.111}$$

$$\begin{aligned} a &= \langle\delta_i^a(t) - \delta_i^{exp}\rangle \\ b &= \langle\delta_i^c(t) - \delta_i^{exp}\rangle \end{aligned}$$

Using orientation restraints

Orientation restraints can be added to a molecule definition in the topology file in the section `[ orientation_restraints ]`. Here we give an example section containing five N-H residual dipolar coupling restraints:

```
[ orientation_restraints ]
; ai    aj  type  exp.  label  alpha     const.      obs.    weight
;                              Hz       nm^3        Hz     Hz^-2
  31    32     1     1      3      3     6.083    -6.73       1.0
  43    44     1     1      4      3     6.083    -7.87       1.0
  55    56     1     1      5      3     6.083    -7.13       1.0
  65    66     1     1      6      3     6.083    -2.57       1.0
  73    74     1     1      7      3     6.083    -2.10       1.0
```

The unit of the observable is Hz, but one can choose any other unit. In columns `ai` and `aj` you find the atom numbers of the particles to be restrained. The `type` column should always be 1. The `exp.` column denotes the experiment number, starting at 1. For each experiment a separate order tensor $\mathbf{S}$ is optimized. The label should be a unique number larger than zero for each restraint. The `alpha` column contains the power α that is used in equation (4.97) to calculate the orientation. The `const.` column contains the constant c_i used in the same equation. The constant should have the unit of the observable times nm$^\alpha$. The column `obs.` contains the observable, in any unit you like. The last column contains the weights w_i; the unit should be the inverse of the square of the unit of the observable.

Some parameters for orientation restraints can be specified in the `grompp.mdp` file, for a study of the effect of different force constants and averaging times and ensemble averaging see [92].

4.4 Polarization

Polarization can be treated by GROMACS by attaching shell (Drude) particles to atoms and/or virtual sites. The energy of the shell particle is then minimized at each time step in order to remain on the Born-Oppenheimer surface.

4.4.1 Simple polarization

This is merely a harmonic potential with equilibrium distance 0.

4.4.2 Water polarization

A special potential for water that allows anisotropic polarization of a single shell particle [43].

4.4.3 Thole polarization

Based on early work by Thole [93], Roux and coworkers have implemented potentials for molecules like ethanol [94, 95, 96]. Within such molecules, there are intra-molecular interactions between shell particles, however these must be screened because full Coulomb would be too strong. The potential between two shell particles i and j is:

$$V_{thole} = \frac{q_i q_j}{r_{ij}} \left[1 - \left(1 + \frac{\bar{r}_{ij}}{2} \right) \exp^{-\bar{r}_{ij}} \right] \tag{4.112}$$

Note that there is a sign error in Equation 1 of Noskov *et al.* [96]:

$$\bar{r}_{ij} = a\frac{r_{ij}}{(\alpha_i\alpha_j)^{1/6}} \tag{4.113}$$

where a is a magic (dimensionless) constant, usually chosen to be 2.6 [96]; α_i and α_j are the polarizabilities of the respective shell particles.

4.5 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.

Starting in GROMACS 4.6, λ is a vector, allowing different components of the free energy transformation to be carried out at different rates. Coulomb, Lennard-Jones, bonded, and restraint terms can all be controlled independently, as described in the `.mdp` options.

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.

$$V_b = \frac{1}{2}\left[(1-\lambda)k_b^A + \lambda k_b^B\right]\left[b - (1-\lambda)b_0^A - \lambda b_0^B\right]^2 \tag{4.114}$$

$$\frac{\partial V_b}{\partial \lambda} = \frac{1}{2}(k_b^B - k_b^A)\left[b - (1-\lambda)b_0^A + \lambda b_0^B\right]^2 + (b_0^A - b_0^B)\left[b - (1-\lambda)b_0^A - \lambda b_0^B\right]\left[(1-\lambda)k_b^A + \lambda k_b^B\right] \tag{4.115}$$

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:

$$V_d = \left[(1-\lambda)k_d^A + \lambda k_d^B\right]\left(1 + \cos\left[n_\phi\phi - (1-\lambda)\phi_s^A - \lambda\phi_s^B\right]\right) \tag{4.116}$$

$$\frac{\partial V_d}{\partial \lambda} = (k_d^B - k_d^A)\left(1 + \cos\left[n_\phi\phi - (1-\lambda)\phi_s^A - \lambda\phi_s^B\right]\right) + (\phi_s^B - \phi_s^A)\left[(1-\lambda)k_d^A - \lambda k_d^B\right]\sin\left[n_\phi\phi - (1-\lambda)\phi_s^A - \lambda\phi_s^B\right] \tag{4.117}$$

Note: that the multiplicity n_ϕ can not be parameterized because the function should remain periodic on the interval $[0, 2\pi]$.

Tabulated bonded interactions

For tabulated bonded interactions only the force constant can interpolated:

$$V = ((1-\lambda)k^A + \lambda k^B)\, f \tag{4.118}$$

$$\frac{\partial V}{\partial \lambda} = (k^B - k^A)\, f \tag{4.119}$$

Coulomb interaction

The Coulomb interaction between two particles of which the charge varies with λ is:

$$V_c = \frac{f}{\varepsilon_{rf} r_{ij}} \left[(1-\lambda) q_i^A q_j^A + \lambda\, q_i^B q_j^B\right] \tag{4.120}$$

$$\frac{\partial V_c}{\partial \lambda} = \frac{f}{\varepsilon_{rf} r_{ij}} \left[-q_i^A q_j^A + q_i^B q_j^B\right] \tag{4.121}$$

where $f = \frac{1}{4\pi\varepsilon_0} = 138.935\,485$ (see chapter 2).

Coulomb interaction with reaction field

The Coulomb interaction including a reaction field, between two particles of which the charge varies with λ is:

$$V_c = f\left[\frac{1}{r_{ij}} + k_{rf}\, r_{ij}^2 - c_{rf}\right] \left[(1-\lambda) q_i^A q_j^A + \lambda\, q_i^B q_j^B\right] \tag{4.122}$$

$$\frac{\partial V_c}{\partial \lambda} = f\left[\frac{1}{r_{ij}} + k_{rf}\, r_{ij}^2 - c_{rf}\right] \left[-q_i^A q_j^A + q_i^B q_j^B\right] \tag{4.123}$$

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:

$$V_{LJ} = \frac{(1-\lambda) C_{12}^A + \lambda\, C_{12}^B}{r_{ij}^{12}} - \frac{(1-\lambda) C_6^A + \lambda\, C_6^B}{r_{ij}^6} \tag{4.124}$$

$$\frac{\partial V_{LJ}}{\partial \lambda} = \frac{C_{12}^B - C_{12}^A}{r_{ij}^{12}} - \frac{C_6^B - C_6^A}{r_{ij}^6} \tag{4.125}$$

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 force field 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, so 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 $\frac{\partial E_k}{\partial \lambda}$ being incorrect [97]):

$$E_k = \frac{1}{2}\frac{\boldsymbol{p}^2}{(1-\lambda)m^A + \lambda m^B} \tag{4.126}$$

$$\frac{\partial E_k}{\partial \lambda} = -\frac{1}{2}\frac{\boldsymbol{p}^2(m^B - m^A)}{((1-\lambda)m^A + \lambda m^B)^2} \tag{4.127}$$

after taking the derivative, we *can* insert $\boldsymbol{p}$ = m$\boldsymbol{v}$, such that:

$$\frac{\partial E_k}{\partial \lambda} = -\frac{1}{2}\boldsymbol{v}^2(m^B - m^A) \tag{4.128}$$

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:

$$g_k = \boldsymbol{r}_k - d_k \tag{4.129}$$

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

$$g_k = \boldsymbol{r}_k - \left((1-\lambda)d_k^A + \lambda d_k^B\right) \tag{4.130}$$

the contribution C_λ to the Hamiltonian using Lagrange multipliers λ:

$$C_\lambda = \sum_k \lambda_k g_k \tag{4.131}$$

$$\frac{\partial C_\lambda}{\partial \lambda} = \sum_k \lambda_k \left(d_k^B - d_k^A\right) \tag{4.132}$$

4.5.1 Soft-core interactions

In a free-energy calculation where particles grow out of nothing, or particles disappear, using the the simple linear interpolation of the Lennard-Jones and Coulomb potentials as described in Equations 4.125 and 4.123 may lead to poor convergence. When the particles have nearly disappeared, or are close to appearing (at λ close to 0 or 1), the interaction energy will be weak enough for particles to get very close to each other, leading to large fluctuations in the measured values of $\partial V/\partial\lambda$ (which, because of the simple linear interpolation, depends on the potentials at both the endpoints of λ).

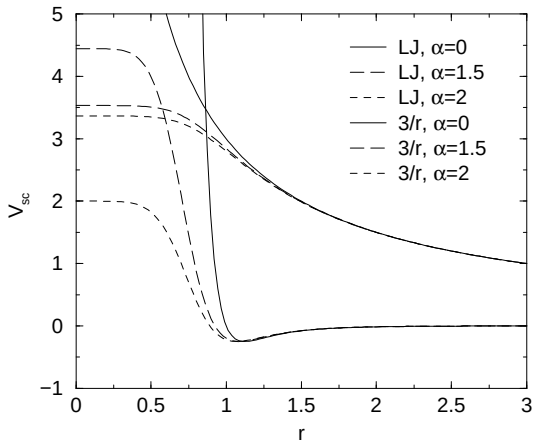


Figure 4.17: Soft-core interactions at $\lambda = 0.5$, with $p = 2$ and $C_6^A = C_{12}^A = C_6^B = C_{12}^B = 1$.

To circumvent these problems, the singularities in the potentials need to be removed. This can be done by modifying the regular Lennard-Jones and Coulomb potentials with "soft-core" potentials that limit the energies and forces involved at λ values between 0 and 1, but not *at* $\lambda = 0$ or 1.

In GROMACS the soft-core potentials V_{sc} are shifted versions of the regular potentials, so that the singularity in the potential and its derivatives at $r = 0$ is never reached:

$$V_{sc}(r) = (1-\lambda)V^A(r_A) + \lambda V^B(r_B) \tag{4.133}$$

$$r_A = \left(\alpha\sigma_A^6\lambda^p + r^6\right)^{\frac{1}{6}} \tag{4.134}$$

$$r_B = \left(\alpha\sigma_B^6(1-\lambda)^p + r^6\right)^{\frac{1}{6}} \tag{4.135}$$

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 (set with `sc_alpha` in the `.mdp` file), p is the soft-core λ power (set with `sc_power`), σ is the radius of the interaction, which is $(C_{12}/C_6)^{1/6}$ or an input parameter (`sc_sigma`) when C_6 or C_{12} is zero.

For intermediate λ, r_A and r_B alter the interactions very little for $r > \alpha^{1/6}\sigma$ and quickly switch the soft-core interaction to an almost constant value for smaller r (Fig. 4.17). The force is:

$$F_{sc}(r) = -\frac{\partial V_{sc}(r)}{\partial r} = (1-\lambda)F^A(r_A)\left(\frac{r}{r_A}\right)^5 + \lambda F^B(r_B)\left(\frac{r}{r_B}\right)^5 \tag{4.136}$$

where F^A and F^B are the "hard core" forces. The contribution to the derivative of the free energy is:

$$\begin{aligned}\frac{\partial V_{sc}(r)}{\partial\lambda} &= V^B(r_B) - V^A(r_A) + (1-\lambda)\frac{\partial V^A(r_A)}{\partial r_A}\frac{\partial r_A}{\partial\lambda} + \lambda\frac{\partial V^B(r_B)}{\partial r_B}\frac{\partial r_B}{\partial\lambda} \\ &= V^B(r_B) - V^A(r_A) + \\ &\quad \frac{p\alpha}{6}\left[\lambda F^B(r_B)r_B^{-5}\sigma_B^6(1-\lambda)^{p-1} - (1-\lambda)F^A(r_A)r_A^{-5}\sigma_A^6\lambda^{p-1}\right]\end{aligned} \tag{4.137}$$

The original GROMOS Lennard-Jones soft-core function [98] uses $p = 2$, but $p = 1$ gives a smoother $\partial H/\partial\lambda$ curve.

Another issue that should be considered is the soft-core effect of hydrogens without Lennard-Jones interaction. Their soft-core σ is set with `sc-sigma` in the `.mdp` file. These hydrogens produce peaks in $\partial H/\partial\lambda$ at λ is 0 and/or 1 for $p = 1$ and close to 0 and/or 1 with $p = 2$. Lowering `sc-sigma` will decrease this effect, but it will also increase the interactions with hydrogens relative to the other interactions in the soft-core state.

When soft core potentials are selected (by setting `sc-alpha` >0), and the Coulomb and Lennard-Jones potentials are turned on or off sequentially, then the Coulombic interaction is turned off linearly, rather than using soft core interactions, which should be less statistically noisy in most cases. This behavior can be overwritten by using the mdp option `sc-coul` to `yes`. Additionally, the soft-core interaction potential is only applied when either the A or B state has zero interaction potential. If both A and B states have nonzero interaction potential, default linear scaling described above is used. When both Coulombic and Lennard-Jones interactions are turned off simultaneously, a soft-core potential is used, and a hydrogen is being introduced or deleted, the sigma is set to `sc-sigma-min`, which itself defaults to `sc-sigma-default`.

Recently, a new formulation of the soft-core approach has been derived that in most cases gives lower and more even statistical variance than the standard soft-core path described above. [99, 100] Specifically, we have:

$$V_{sc}(r) = (1-\lambda)V^A(r_A) + \lambda V^B(r_B) \tag{4.138}$$

$$r_A = \left(\alpha\sigma_A^{48}\lambda^p + r^{48}\right)^{\frac{1}{48}} \tag{4.139}$$

$$r_B = \left(\alpha\sigma_B^{48}(1-\lambda)^p + r^{48}\right)^{\frac{1}{48}} \tag{4.140}$$

This "1-1-48" path is also implemented in GROMACS. Note that for this path the soft core α should satisfy $0.001 < \alpha < 0.003$, rather than $\alpha \approx 0.5$.

4.6 Methods

4.6.1 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 or two atoms are called *first neighbors, second neighbors* and *third neighbors*, respectively (see Fig. 4.18). Since the interactions of atom **i** with atoms **i+1** and **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 [101], the standard Lennard-

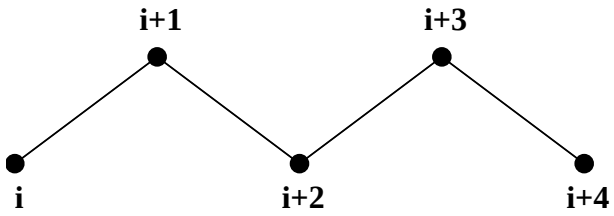


Figure 4.18: Atoms along an alkane chain.

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.

4.6.2 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. If we would apply a plain 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 protons, and as a result, induce a large dipole moment over the system. Therefore, we have to keep groups of atoms with total charge 0 together. These groups are called *charge groups*. Note that with a proper treatment of long-range electrostatics (e.g. particle-mesh Ewald (sec. 4.8.2), keeping charge groups together is not required.

4.6.3 Treatment of Cut-offs in the group scheme

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 neighbor searching.

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 non-bonded interaction parameters. See sec. 7.3 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 max(`rcoulomb`,`rvdw`) are computed during NS. 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. When mdrun is compiled to use mixed 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 the 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 be 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).

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.

4.7 Virtual interaction sites

Virtual interaction sites (called dummy atoms in GROMACS versions before 3.3) can be used in GROMACS in a number of ways. We write the position of the virtual site $\boldsymbol{r}_s$ as a function of the positions of other particles $\boldsymbol{r}_i$: $\boldsymbol{r}_s = f(\boldsymbol{r}_1..\boldsymbol{r}_n)$. The virtual site, which may carry charge or be involved in other interactions, can now be used in the force calculation. The force acting on the virtual site must be redistributed over the particles with mass in a consistent way. A good way to do this can be found in ref. [102]. We can write the potential energy as:

$$V = V(\boldsymbol{r}_s, \boldsymbol{r}_1, \ldots, \boldsymbol{r}_n) = V^*(\boldsymbol{r}_1, \ldots, \boldsymbol{r}_n) \tag{4.141}$$

The force on the particle i is then:

$$\boldsymbol{F}_i = -\frac{\partial V^*}{\partial \boldsymbol{r}_i} = -\frac{\partial V}{\partial \boldsymbol{r}_i} - \frac{\partial V}{\partial \boldsymbol{r}_s}\frac{\partial \boldsymbol{r}_s}{\partial \boldsymbol{r}_i} = \boldsymbol{F}_i^{direct} + \boldsymbol{F}'_i \tag{4.142}$$

The first term is the normal force. The second term is the force on particle i due to the virtual site, which can be written in tensor notation:

$$\boldsymbol{F}'_i = \begin{bmatrix} \frac{\partial x_s}{\partial x_i} & \frac{\partial y_s}{\partial x_i} & \frac{\partial z_s}{\partial x_i} \\ \frac{\partial x_s}{\partial y_i} & \frac{\partial y_s}{\partial y_i} & \frac{\partial z_s}{\partial y_i} \\ \frac{\partial x_s}{\partial z_i} & \frac{\partial y_s}{\partial z_i} & \frac{\partial z_s}{\partial z_i} \end{bmatrix} \boldsymbol{F}_s \tag{4.143}$$

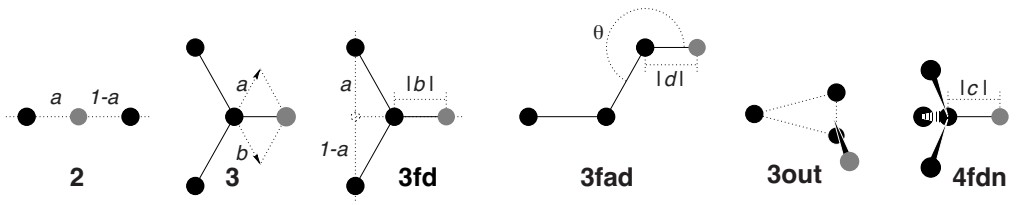


Figure 4.19: The six different types of virtual site construction in GROMACS. The constructing atoms are shown as black circles, the virtual sites in gray.

where $\boldsymbol{F}_s$ is the force on the virtual site and x_s, y_s and z_s are the coordinates of the virtual site. In this way, the total force and the total torque are conserved [102].

The computation of the virial (eqn. 3.24) is non-trivial when virtual sites are used. Since the virial involves a summation over all the atoms (rather than virtual sites), the forces must be redistributed from the virtual sites to the atoms (using eqn. 4.143) *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 virtual sites (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 virtual sites from surrounding atoms in GROMACS, which we classify by the number of constructing atoms. **Note** that all site types mentioned can 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 we strongly recommended using the first adequate virtual site type that will be sufficient for a certain purpose. Fig. 4.19 depicts 6 of the available virtual site constructions. The conceptually simplest construction types are linear combinations:

$$\boldsymbol{r}_s = \sum_{i=1}^{N} w_i \, \boldsymbol{r}_i \tag{4.144}$$

The force is then redistributed using the same weights:

$$\boldsymbol{F}'_i = w_i \, \boldsymbol{F}_s \tag{4.145}$$

The types of virtual sites supported in GROMACS are given in the list below. Constructing atoms in virtual sites can be virtual sites themselves, but only if they are higher in the list, i.e. virtual sites can be constructed from "particles" that are simpler virtual sites.

2. As a linear combination of two atoms (Fig. 4.19 2):

$$w_i = 1 - a \, , \quad w_j = a \tag{4.146}$$

   In this case the virtual site is on the line through atoms i and j.

3. As a linear combination of three atoms (Fig. 4.19 3):

$$w_i = 1 - a - b \, , \quad w_j = a \, , \quad w_k = b \tag{4.147}$$

   In this case the virtual site is in the plane of the other three particles.

3fd. In the plane of three atoms, with a fixed distance (Fig. 4.19 3fd):

$$\boldsymbol{r}_s = \boldsymbol{r}_i + b\frac{\boldsymbol{r}_{ij} + a\boldsymbol{r}_{jk}}{|\boldsymbol{r}_{ij} + a\boldsymbol{r}_{jk}|} \tag{4.148}$$

In this case the virtual site 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 virtual site can be computed as:

$$\begin{aligned} \boldsymbol{F}'_i &= \boldsymbol{F}_s - \gamma(\boldsymbol{F}_s - \boldsymbol{p}) \\ \boldsymbol{F}'_j &= (1-a)\gamma(\boldsymbol{F}_s - \boldsymbol{p}) \\ \boldsymbol{F}'_k &= a\gamma(\boldsymbol{F}_s - \boldsymbol{p}) \end{aligned} \quad \text{where} \quad \begin{aligned} \gamma &= \frac{b}{|\boldsymbol{r}_{ij} + a\boldsymbol{r}_{jk}|} \\ \boldsymbol{p} &= \frac{\boldsymbol{r}_{is}\cdot\boldsymbol{F}_s}{\boldsymbol{r}_{is}\cdot\boldsymbol{r}_{is}}\boldsymbol{r}_{is} \end{aligned} \tag{4.149}$$

3fad. In the plane of three atoms, with a fixed angle and distance (Fig. 4.19 3fad):

$$\boldsymbol{r}_s = \boldsymbol{r}_i + d\cos\theta\frac{\boldsymbol{r}_{ij}}{|\boldsymbol{r}_{ij}|} + d\sin\theta\frac{\boldsymbol{r}_\perp}{|\boldsymbol{r}_\perp|} \quad \text{where} \quad \boldsymbol{r}_\perp = \boldsymbol{r}_{jk} - \frac{\boldsymbol{r}_{ij}\cdot\boldsymbol{r}_{jk}}{\boldsymbol{r}_{ij}\cdot\boldsymbol{r}_{ij}}\boldsymbol{r}_{ij} \tag{4.150}$$

In this case the virtual site 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, instead of a and b (see also sec. 5.2.2). The force on particles i, j and k due to the force on the virtual site can be computed as (with $\boldsymbol{r}_\perp$ as defined in eqn. 4.150):

$$\begin{aligned} \boldsymbol{F}'_i &= \boldsymbol{F}_s - \frac{d\cos\theta}{|\boldsymbol{r}_{ij}|}\boldsymbol{F}_1 + \frac{d\sin\theta}{|\boldsymbol{r}_\perp|}\left(\frac{\boldsymbol{r}_{ij}\cdot\boldsymbol{r}_{jk}}{\boldsymbol{r}_{ij}\cdot\boldsymbol{r}_{ij}}\boldsymbol{F}_2 + \boldsymbol{F}_3\right) \\ \boldsymbol{F}'_j &= \frac{d\cos\theta}{|\boldsymbol{r}_{ij}|}\boldsymbol{F}_1 - \frac{d\sin\theta}{|\boldsymbol{r}_\perp|}\left(\boldsymbol{F}_2 + \frac{\boldsymbol{r}_{ij}\cdot\boldsymbol{r}_{jk}}{\boldsymbol{r}_{ij}\cdot\boldsymbol{r}_{ij}}\boldsymbol{F}_2 + \boldsymbol{F}_3\right) \\ \boldsymbol{F}'_k &= \frac{d\sin\theta}{|\boldsymbol{r}_\perp|}\boldsymbol{F}_2 \end{aligned}$$

$$\text{where } \boldsymbol{F}_1 = \boldsymbol{F}_s - \frac{\boldsymbol{r}_{ij}\cdot\boldsymbol{F}_s}{\boldsymbol{r}_{ij}\cdot\boldsymbol{r}_{ij}}\boldsymbol{r}_{ij}\,,\ \boldsymbol{F}_2 = \boldsymbol{F}_1 - \frac{\boldsymbol{r}_\perp\cdot\boldsymbol{F}_s}{\boldsymbol{r}_\perp\cdot\boldsymbol{r}_\perp}\boldsymbol{r}_\perp \text{ and } \boldsymbol{F}_3 = \frac{\boldsymbol{r}_{ij}\cdot\boldsymbol{F}_s}{\boldsymbol{r}_{ij}\cdot\boldsymbol{r}_{ij}}\boldsymbol{r}_\perp \tag{4.151}$$

3out. As a non-linear combination of three atoms, out of plane (Fig. 4.19 3out):

$$\boldsymbol{r}_s = \boldsymbol{r}_i + a\boldsymbol{r}_{ij} + b\boldsymbol{r}_{ik} + c(\boldsymbol{r}_{ij}\times\boldsymbol{r}_{ik}) \tag{4.152}$$

This enables the construction of virtual sites out of the plane of the other atoms. The force on particles i, j and k due to the force on the virtual site can be computed as:

$$\begin{aligned} \boldsymbol{F}'_j &= \begin{bmatrix} a & -c\,z_{ik} & c\,y_{ik} \\ c\,z_{ik} & a & -c\,x_{ik} \\ -c\,y_{ik} & c\,x_{ik} & a \end{bmatrix}\boldsymbol{F}_s \\ \boldsymbol{F}'_k &= \begin{bmatrix} b & c\,z_{ij} & -c\,y_{ij} \\ -c\,z_{ij} & b & c\,x_{ij} \\ c\,y_{ij} & -c\,x_{ij} & b \end{bmatrix}\boldsymbol{F}_s \\ \boldsymbol{F}'_i &= \boldsymbol{F}_s - \boldsymbol{F}'_j - \boldsymbol{F}'_k \end{aligned} \tag{4.153}$$

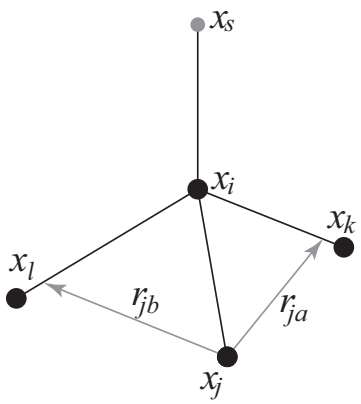


Figure 4.20: The new 4fdn virtual site construction, which is stable even when all constructing atoms are in the same plane.

4fdn. From four atoms, with a fixed distance, see separate Fig. 4.20. This construction is a bit complex, in particular since the previous type (4fd) could be unstable which forced us to introduce a more elaborate construction:

$$\begin{aligned}
\mathbf{r}_{ja} &= a\,\mathbf{r}_{ik} - \mathbf{r}_{ij} = a\,(\mathbf{x}_k - \mathbf{x}_i) - (\mathbf{x}_j - \mathbf{x}_i) \\
\mathbf{r}_{jb} &= b\,\mathbf{r}_{il} - \mathbf{r}_{ij} = b\,(\mathbf{x}_l - \mathbf{x}_i) - (\mathbf{x}_j - \mathbf{x}_i) \\
\mathbf{r}_m &= \mathbf{r}_{ja} \times \mathbf{r}_{jb} \\
\mathbf{x}_s &= \mathbf{x}_i + c\frac{\mathbf{r}_m}{|\mathbf{r}_m|}
\end{aligned} \tag{4.154}$$

In this case the virtual site is at a distance of $|c|$ from i, while a and b are parameters. **Note** that the vectors $\mathbf{r}_{ik}$ and $\mathbf{r}_{ij}$ are not normalized to save floating-point operations. The force on particles i, j, k and l due to the force on the virtual site are computed through chain rule derivatives of the construction expression. This is exact and conserves energy, but it does lead to relatively lengthy expressions that we do not include here (over 200 floating-point operations). The interested reader can look at the source code in `vsite.c`. Fortunately, this vsite type is normally only used for chiral centers such as C_α atoms in proteins.

The new 4fdn construct is identified with a 'type' value of 2 in the topology. The earlier 4fd type is still supported internally ('type' value 1), but it should not be used for new simulations. All current GROMACS tools will automatically generate type 4fdn instead.

N. A linear combination of N atoms with relative weights a_i. The weight for atom i is:

$$w_i = a_i \left(\sum_{j=1}^{N} a_j \right)^{-1} \tag{4.155}$$

There are three options for setting the weights:

COG center of geometry: equal weights

COM center of mass: a_i is the mass of atom i; when in free-energy simulations the mass of the atom is changed, only the mass of the A-state is used for the weight

COW center of weights: a_i is defined by the user

4.8 Long Range Electrostatics

4.8.1 Ewald summation

The total electrostatic energy of N particles and their periodic images is given by

$$V = \frac{f}{2} \sum_{n_x} \sum_{n_y} \sum_{n_z*} \sum_{i}^{N} \sum_{j}^{N} \frac{q_i q_j}{\mathbf{r}_{ij,\mathbf{n}}}. \tag{4.156}$$

$(n_x, n_y, n_z) = \mathbf{n}$ 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 [103]. The idea is to convert the single slowly-converging sum eqn. 4.156 into two quickly-converging terms and a constant term:

$$V = V_{\text{dir}} + V_{\text{rec}} + V_0 \tag{4.157}$$

$$V_{\text{dir}} = \frac{f}{2} \sum_{i,j}^{N} \sum_{n_x} \sum_{n_y} \sum_{n_z*} q_i q_j \frac{\text{erfc}(\beta r_{ij,\mathbf{n}})}{r_{ij,\mathbf{n}}} \tag{4.158}$$

$$V_{\text{rec}} = \frac{f}{2\pi V} \sum_{i,j}^{N} q_i q_j \sum_{m_x} \sum_{m_y} \sum_{m_z*} \frac{\exp\left(-(\pi \mathbf{m}/\beta)^2 + 2\pi i \mathbf{m} \cdot (\mathbf{r}_i - \mathbf{r}_j)\right)}{\mathbf{m}^2} \tag{4.159}$$

$$V_0 = -\frac{f\beta}{\sqrt{\pi}} \sum_{i}^{N} q_i^2, \tag{4.160}$$

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 for use in 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
```

The ratio of the box dimensions and the `fourierspacing` parameter 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.

4.8.2 PME

Particle-mesh Ewald is a method proposed by Tom Darden [12] to improve the performance of the reciprocal sum. Instead of directly summing wave vectors, the charges are assigned to a grid using interpolation. The implementation in GROMACS uses cardinal B-spline interpolation [13], which is referred to as smooth PME (SPME). The 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. For the parallelization of PME see the section on MPMD PME (3.17.5).

With the Verlet cut-off scheme, the PME direct space potential is shifted by a constant such that the potential is zero at the cut-off. This shift is small and since the net system charge is close to zero, the total shift is very small, unlike in the case of the Lennard-Jones potential where all shifts add up. We apply the shift anyhow, such that the potential is the exact integral of the force.

Using PME

As an example for using 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 (i.e. minimum number of grid points), and `pme-order` controls the interpolation order. Using

fourth-order (cubic) interpolation and this spacing should give electrostatic energies accurate to about $5 \cdot 10^{-3}$. 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.

4.8.3 P3M-AD

The Particle-Particle Particle-Mesh methods of Hockney & Eastwood can also be applied in GROMACS for the treatment of long range electrostatic interactions [104]. Although the P3M method was the first efficient long-range electrostatics method for molecular simulation, the smooth PME (SPME) method has largely replaced P3M as the method of choice in atomistic simulations. One performance disadvantage of the original P3M method was that it required 3 3D-FFT back transforms to obtain the forces on the particles. But this is not required for P3M and the forces can be derived through analytical differentiation of the potential, as done in PME. The resulting method is termed P3M-AD. The only remaining difference between P3M-AD and PME is the optimization of the lattice Green influence function for error minimization that P3M uses. However, in 2012 it has been shown that the SPME influence function can be modified to obtain P3M [105]. This means that the advantage of error minimization in P3M-AD can be used at the same computational cost and with the same code as PME, just by adding a few lines to modify the influence function. However, at optimal parameter setting the effect of error minimization in P3M-AD is less than 10%. P3M-AD does show large accuracy gains with interlaced (also known as staggered) grids, but that is not supported in GROMACS (yet).

P3M is used in GROMACS with exactly the same options as used with PME by selecting the electrostatics type:

```
coulombtype      = P3M-AD
```

4.8.4 Optimizing Fourier transforms and PME calculations

It is recommended to optimize the parameters for calculation of electrostatic interaction such as PME grid dimensions and cut-off radii. This is particularly relevant to do before launching long production runs.

GROMACS includes a special tool, `g_tune_pme`, which automates the process of selecting the optimal size of the grid and number of PME-only notes.

4.9 Long Range Van der Waals interactions

4.9.1 Dispersion correction

In this section, we derive long-range corrections due to the use of a cut-off for Lennard-Jones or Buckingham 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 (we are truncating a potential proportional to $-r^{-6}$), energy and pressure corrections are both negative. While the energy correction is usually small, it may be important for free energy calculations where differences between two different Hamiltonians are considered. In contrast, the pressure correction is very large and can not be neglected under any circumstances where a correct pressure is required, especially for any NPT simulations. Although it is, in principle, possible to parameterize a force field such that the pressure is close to the desired experimental value without correction, such a method makes the parameterization dependent on the cut-off and is therefore undesirable.

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:

$$V(r_{ij}) \;=\; -C_6\, r_{ij}^{-6} \tag{4.161}$$

and the corresponding force is:

$$\boldsymbol{F}_{ij} \;=\; -6\, C_6\, r_{ij}^{-8} \boldsymbol{r}_{ij} \tag{4.162}$$

In a periodic system it is not easy to calculate the full potentials, so usually a cut-off is applied, which can be abrupt or smooth. We will call the potential and force with cut-off V_c and $\boldsymbol{F}_c$. The long-range contribution to the dispersion energy in a system with N particles and particle density $\rho = N/V$ is:

$$V_{lr} \;=\; \frac{1}{2} N \rho \int_0^\infty 4\pi r^2 g(r)\, (V(r) - V_c(r))\, \mathrm{d}r \tag{4.163}$$

We will integrate this for the shift function, which is the most general form of van der Waals interaction available in GROMACS. The shift function has a constant difference S from 0 to r_1 and is 0 beyond the cut-off distance r_c. We can integrate eqn. 4.163, assuming that the density in the sphere within r_1 is equal to the global density and the radial distribution function $g(r)$ is 1 beyond r_1:

$$\begin{aligned} V_{lr} \;&=\; \frac{1}{2} N \left(\rho \int_0^{r_1} 4\pi r^2 g(r)\, C_6\, S\, \mathrm{d}r + \rho \int_{r_1}^{r_c} 4\pi r^2\, (V(r) - V_c(r))\, \mathrm{d}r + \rho \int_{r_c}^{\infty} 4\pi r^2 V(r)\, \mathrm{d}r \right) \\ &=\; \frac{1}{2} N \left(\left(\frac{4}{3}\pi\rho r_1^3 - 1 \right) C_6\, S + \rho \int_{r_1}^{r_c} 4\pi r^2\, (V(r) - V_c(r))\, \mathrm{d}r - \frac{4}{3}\pi N \rho\, C_6\, r_c^{-3} \right) \end{aligned} \tag{4.164}$$

where the term -1 corrects for the self-interaction. For a plain cut-off we only need to assume that $g(r)$ is 1 beyond r_c and the correction reduces to [106]:

$$V_{lr} \;=\; -\frac{2}{3}\pi N \rho\, C_6\, r_c^{-3} \tag{4.165}$$

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 -0.75 kJ mol^{-1} per molecule.

For a homogeneous mixture we need to define an *average dispersion constant*:

$$\langle C_6 \rangle = \frac{2}{N(N-1)} \sum_i^N \sum_{j>i}^N C_6(i,j) \tag{4.166}$$

In GROMACS, excluded pairs of atoms do not contribute to the average.

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.

Virial and pressure

The scalar virial of the system due to the dispersion interaction between two particles i and j is given by:

$$\Xi \;=\; -\frac{1}{2}\boldsymbol{r}_{ij} \cdot \boldsymbol{F}_{ij} \;=\; 3\,C_6\,r_{ij}^{-6} \tag{4.167}$$

The pressure is given by:

$$P \;=\; \frac{2}{3\,V}\,(E_{kin} - \Xi) \tag{4.168}$$

The long-range correction to the virial is given by:

$$\Xi_{lr} \;=\; \frac{1}{2}N\rho \int_0^\infty 4\pi r^2 g(r)(\Xi - \Xi_c)\,\mathrm{d}r \tag{4.169}$$

We can again integrate the long-range contribution to the virial assuming $g(r)$ is 1 beyond r_1:

$$\begin{aligned}\Xi_{lr} \;&=\; \frac{1}{2}N\rho \left(\int_{r_1}^{r_c} 4\pi r^2 (\Xi - \Xi_c)\,\mathrm{d}r + \int_{r_c}^{\infty} 4\pi r^2 3\,C_6\,r_{ij}^{-6}\,\mathrm{d}r \right) \\ &=\; \frac{1}{2}N\rho \left(\int_{r_1}^{r_c} 4\pi r^2 (\Xi - \Xi_c)\,\mathrm{d}r + 4\pi C_6\,r_c^{-3} \right)\end{aligned} \tag{4.170}$$

For a plain cut-off the correction to the pressure is [106]:

$$P_{lr} \;=\; -\frac{4}{3}\pi C_6\,\rho^2 r_c^{-3} \tag{4.171}$$

Using the same example of a water box, the correction to the virial is 0.75 kJ mol^{-1} per molecule, the corresponding correction to the pressure for SPC water is approximately -280 bar.

For homogeneous mixtures, we can again use the average dispersion constant $\langle C_6 \rangle$ (eqn. 4.166):

$$P_{lr} \;=\; -\frac{4}{3}\pi\,\langle C_6 \rangle\,\rho^2 r_c^{-3} \tag{4.172}$$

For inhomogeneous systems, eqn. 4.172 can be applied under the same restriction as holds for the energy (see sec. 4.9.1).

4.9.2 Lennard-Jones PME

In order to treat systems, using Lennard-Jones potentials, that are non-homogeneous outside of the cut-off distance, we can instead use the Particle-mesh Ewald method as discussed for electrostatics above. In this case the modified Ewald equations become

$$V \;=\; V_{\mathrm{dir}} + V_{\mathrm{rec}} + V_0 \tag{4.173}$$

$$V_{\text{dir}} = -\frac{1}{2}\sum_{i,j}^{N}\sum_{n_x}\sum_{n_y}\sum_{n_z*}\frac{C_6^{ij}\,g(\beta r_{ij,\mathbf{n}})}{{r_{ij,\mathbf{n}}}^6} \tag{4.174}$$

$$V_{\text{rec}} = \frac{\pi^{\frac{3}{2}}\beta^3}{2V}\sum_{m_x}\sum_{m_y}\sum_{m_z*} f(\pi|\mathbf{m}|/\beta)\times\sum_{i,j}^{N} C_6^{ij}\exp\left[-2\pi i\mathbf{m}\cdot(\mathbf{r_i}-\mathbf{r_j})\right] \tag{4.175}$$

$$V_0 = -\frac{\beta^6}{12}\sum_{i}^{N} C_6^{ii} \tag{4.176}$$

where $\mathbf{m} = (m_x, m_y, m_z)$, β is the parameter determining the weight between direct and reciprocal space, and C_6^{ij} is the combined dispersion parameter for particle i and j. The star indicates that terms with $i = j$ should be omitted when $((n_x, n_y, n_z) = (0, 0, 0))$, and $\mathbf{r}_{ij,\mathbf{n}}$ is the real distance between the particles. Following the derivation by Essmann [13], the functions f and g introduced above are defined as

$$f(x) = 1/3\left[(1-2x^2)\exp(-x^2) + 2x^3\sqrt{\pi}\,\text{erfc}(x)\right] \tag{4.177}$$

$$g(x) = \exp(-x^2)(1 + x^2 + \frac{x^4}{2}). \tag{4.178}$$

The above methodology works fine as long as the dispersion parameters can be combined geometrically (eqn. 4.6) in the same way as the charges for electrostatics

$$C_{6,\text{geom}}^{ij} = \left(C_6^{ii}\,C_6^{jj}\right)^{1/2} \tag{4.179}$$

For Lorentz-Berthelot combination rules (eqn. 4.7), the reciprocal part of this sum has to be calculated seven times due to the splitting of the dispersion parameter according to

$$C_{6,\text{L-B}}^{ij} = (\sigma_i + \sigma_j)^6 = \sum_{n=0}^{6} P_n \sigma_i^n \sigma_j^{(6-n)}, \tag{4.180}$$

for P_n the Pascal triangle coefficients. This introduces a non-negligible cost to the reciprocal part, requiring seven separate FFTs, and therefore this has been the limiting factor in previous attempts to implement LJ-PME. A solution to this problem is to use geometrical combination rules in order to calculate an approximate interaction parameter for the reciprocal part of the potential, yielding a total interaction of

$$V(r < r_c) = \underbrace{C_6^{\text{dir}} g(\beta r) r^{-6}}_{\text{Direct space}} + \underbrace{C_{6,\text{geom}}^{\text{recip}}[1 - g(\beta r)] r^{-6}}_{\text{Reciprocal space}}$$

$$= C_{6,\text{geom}}^{\text{recip}} r^{-6} + \left(C_6^{\text{dir}} - C_{6,\text{geom}}^{\text{recip}}\right) g(\beta r) r^{-6} \tag{4.181}$$

$$V(r > r_c) = \underbrace{C_{6,\text{geom}}^{\text{recip}}[1 - g(\beta r)] r^{-6}}_{\text{Reciprocal space}}. \tag{4.182}$$

This will preserve a well-defined Hamiltonian and significantly increase the performance of the simulations. The approximation does introduce some errors, but since the difference is located in the interactions calculated in reciprocal space, the effect will be very small compared to the total

interaction energy. In a simulation of a lipid bilayer, using a cut-off of 1.0 nm, the relative error in total dispersion energy was below 0.5%. A more thorough discussion of this can be found in [107].

In GROMACS we now perform the proper calculation of this interaction by subtracting, from the direct-space interactions, the contribution made by the approximate potential that is used in the reciprocal part

$$V_{\text{dir}} = C_6^{\text{dir}} r^{-6} - C_6^{\text{recip}}[1 - g(\beta r)]r^{-6}. \tag{4.183}$$

This potential will reduce to the expression in eqn. 4.174 when $C_6^{\text{dir}} = C_6^{\text{recip}}$, and the total interaction is given by

$$\begin{aligned} V(r < r_c) &= \underbrace{C_6^{\text{dir}} r^{-6} - C_6^{\text{recip}}[1 - g(\beta r)]r^{-6}}_{\text{Direct space}} + \underbrace{C_6^{\text{recip}}[1 - g(\beta r)]r^{-6}}_{\text{Reciprocal space}} \\ &= C_6^{\text{dir}} r^{-6} \qquad (4.184) \\ V(r > r_c) &= C_6^{\text{recip}}[1 - g(\beta r)]r^{-6}. \qquad (4.185) \end{aligned}$$

For the case when $C_6^{\text{dir}} \neq C_6^{\text{recip}}$ this will retain an unmodified LJ force up to the cut-off, and the error is an order of magnitude smaller than in simulations where the direct-space interactions do not account for the approximation used in reciprocal space. When using a VdW interaction modifier of potential-shift, the constant

$$\left(-C_6^{\text{dir}} + C_6^{\text{recip}}[1 - g(\beta r_c)]\right) r_c^{-6} \tag{4.186}$$

is added to eqn. 4.184 in order to ensure that the potential is continuous at the cutoff. Note that, in the same way as eqn. 4.183, this degenerates into the expected $-C_6 g(\beta r_c) r_c^{-6}$ when $C_6^{\text{dir}} = C_6^{\text{recip}}$. In addition to this, a long-range dispersion correction can be applied to correct for the approximation using a combination rule in reciprocal space. This correction assumes, as for the cut-off LJ potential, a uniform particle distribution. But since the error of the combination rule approximation is very small this long-range correction is not necessary in most cases. Also note that this homogenous correction does not correct the surface tension, which is an inhomogeneous property.

Using LJ-PME

As an example for using Particle-mesh Ewald summation for Lennard-Jones interactions in GROMACS, specify the following lines in your `.mdp` file:

```
vdwtype           = PME
rvdw              = 0.9
vdw-modifier      = Potential-Shift
rlist             = 0.9
rcoulomb          = 0.9
fourierspacing    = 0.12
pme-order         = 4
ewald-rtol-lj     = 0.001
lj-pme-comb-rule = geometric
```

The same Fourier grid and interpolation order are used if both LJ-PME and electrostatic PME are active, so the settings for `fourierspacing` and `pme-order` are common to both. `ewald-rtol-lj` controls the splitting between direct and reciprocal space in the same way as `ewald-rtol`. In addition to this, the combination rule to be used in reciprocal space is determined by `lj-pme-comb-rule`. If the current force field uses Lorentz-Berthelot combination rules, it is possible to set `lj-pme-comb-rule = geometric` in order to gain a significant increase in performance for a small loss in accuracy. The details of this approximation can be found in the section above.

Note that the use of a complete long-range dispersion correction means that as with Coulomb PME, `rvdw` is now a free parameter in the method, rather than being necessarily restricted by the force-field parameterization scheme. Thus it is now possible to optimize the cutoff, spacing, order and tolerance terms for accuracy and best performance.

Naturally, the use of LJ-PME rather than LJ cut-off adds computation and communication done for the reciprocal-space part, so for best performance in balancing the load of parallel simulations using PME-only ranks, more such ranks should be used. It may be possible to improve upon the automatic load-balancing used by `mdrun`.

4.10 Force field

A force field is built up from two distinct components:

- The set of equations (called the *s*) used to generate the potential energies and their derivatives, the forces. These are described in detail in the previous chapter.
- The parameters used in this set of equations. These are not given in this manual, but in the data files corresponding to your GROMACS distribution.

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. This means in principle that every change should be documented, verified by comparison to experimental data and published in a peer-reviewed journal before it can be used.

GROMACS 5.0 includes several force fields, and additional ones are available on the website. If you do not know which one to select we recommend GROMOS-96 for united-atom setups and OPLS-AA/L for all-atom parameters. That said, we describe the available options in some detail.

All-hydrogen force field

The GROMOS-87-based all-hydrogen force field is almost identical to the normal GROMOS-87 force field, 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 force field is only useful when you need the exact hydrogen positions, for instance for distance restraints derived from NMR measurements. When citing this force field please read the previous paragraph.

4.10.1 GROMOS-96

GROMACS supports the GROMOS-96 force fields [78]. All parameters for the 43A1, 43A2 (development, improved alkane dihedrals), 45A3, 53A5, and 53A6 parameter sets are included. All standard building blocks are included and topologies can be built automatically by `pdb2gmx`.

The GROMOS-96 force field is a further development of the GROMOS-87 force field. It has improvements over the GROMOS-87 force field for proteins and small molecules. **Note** that the sugar parameters present in 53A6 do correspond to those published in 2004[108], which are different from those present in 45A4, which is not included in GROMACS at this time. The 45A4 parameter set corresponds to a later revision of these parameters. The GROMOS-96 force field is not, however, recommended for use with long alkanes and lipids. The GROMOS-96 force field differs from the GROMOS-87 force field in a few respects:

- the force field parameters
- the parameters for the bonded interactions are not linked to atom types
- a fourth power bond stretching potential (4.2.1)
- an angle potential based on the cosine of the angle (4.2.6)

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 by 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 (`rvdw`) 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.

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, a coordinate trajectory file, or a combination of both. The file is fixed format; all floats are written as 15.9, and as such, 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 [78] for a complete description of the blocks. **Note** that all GROMACS programs can read compressed (.Z) or gzipped (.gz) files.

4.10.2 OPLS/AA

4.10.3 AMBER

GROMACS provides native support for the following AMBER force fields:

- AMBER94 [109]
- AMBER96 [110]
- AMBER99 [111]
- AMBER99SB [112]
- AMBER99SB-ILDN [113]
- AMBER03 [114]
- AMBERGS [115]

4.10.4 CHARMM

GROMACS supports the CHARMM force field for proteins [116, 117], lipids [118] and nucleic acids [119, 120]. The protein parameters (and to some extent the lipid and nucleic acid parameters) were thoroughly tested – both by comparing potential energies between the port and the standard parameter set in the CHARMM molecular simulation package, as well by how the protein force field behaves together with GROMACS-specific techniques such as virtual sites (enabling long time steps) and a fast implicit solvent recently implemented [73] – and the details and results are presented in the paper by Bjelkmar et al. [121]. The nucleic acid parameters, as well as the ones for HEME, were converted and tested by Michel Cuendet.

When selecting the CHARMM force field in `pdb2gmx` the default option is to use CMAP (for torsional correction map). To exclude CMAP, use `-nocmap`. The basic form of the CMAP term implemented in GROMACS is a function of the ϕ and ψ backbone torsion angles. This term is defined in the `.rtp` file by a `[ cmap ]` statement at the end of each residue supporting CMAP. The following five atom names define the two torsional angles. Atoms 1-4 define ϕ, and atoms 2-5 define ψ. The corresponding atom types are then matched to the correct CMAP type in the `cmap.itp` file that contains the correction maps.

A port of the CHARMM36 force field for use with GROMACS is also available at http://mackerell.umaryland.edu/charmm_ff.shtml#gromacs.

4.10.5 Coarse-grained force fields

Coarse-graining is a systematic way of reducing the number of degrees of freedom representing a system of interest. To achieve this, typically whole groups of atoms are represented by single beads and the coarse-grained force fields describes their effective interactions. Depending on the choice of parameterization, the functional form of such an interaction can be complicated and often tabulated potentials are used.

Coarse-grained models are designed to reproduce certain properties of a reference system. This can be either a full atomistic model or even experimental data. Depending on the properties to reproduce there are different methods to derive such force fields. An incomplete list of methods is given below:

- Conserving free energies
  - Simplex method
  - MARTINI force field (see next section)
- Conserving distributions (like the radial distribution function), so-called structure-based coarse-graining
  - (iterative) Boltzmann inversion
  - Inverse Monte Carlo
- Conversing forces
  - Force matching

Note that coarse-grained potentials are state dependent (e.g. temperature, density,...) and should be re-parametrized depending on the system of interest and the simulation conditions. This can for example be done using the Versatile Object-oriented Toolkit for Coarse-Graining Applications (VOTCA) [122]. The package was designed to assists in systematic coarse-graining, provides implementations for most of the algorithms mentioned above and has a well tested interface to GROMACS. It is available as open source and further information can be found at www.votca.org.

4.10.6 MARTINI

The MARTINI force field is a coarse-grain parameter set that allows for the construction of many systems, including proteins and membranes.

4.10.7 PLUM

The PLUM force field [123] is an example of a solvent-free protein-membrane model for which the membrane was derived from structure-based coarse-graining [124]. A GROMACS implementation can be found at code.google.com/p/plumx.

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. These do not strictly belong to the molecular topology, and are stored in the coordinate file `*.gro` (positions and velocities), or trajectory file `*.trr` (positions, velocities, forces).

This chapter describes the setup of the topology file, the `*.top` file and the database files: what the parameters stand for and how/where to change them if needed. First, all file formats are explained. Section 5.8.1 describes the organization of the files in each force field.

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 fields or modified versions of the standard force fields - contribute them to the force field archive!

5.2 Particle type

In GROMACS, there are three types of particles, see Table 5.1. Only regular atoms and virtual interaction sites are used in GROMACS; shells are necessary for polarizable models like the Shell-Water models [43].

Particle	Symbol
atoms	A
shells	S
virtual interaction sites	V (or D)

Table 5.1: Particle types in GROMACS

5.2.1 Atom types

Each force field defines a set of atom types, which have a characteristic name or number, and mass (in a.m.u.). These listings are found in the `atomtypes.atp` file (.atp = **a**tom **t**ype **p**arameter file). Therefore, it is in this file that you can begin to change and/or add an atom type. A sample from the `gromos43a1.ff` force field is listed below.

```
  O  15.99940 ;     carbonyl oxygen (C=O)
 OM  15.99940 ;     carboxyl oxygen (CO-)
 OA  15.99940 ;     hydroxyl, sugar or ester oxygen
 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)
 NR  14.00670 ;     aromatic nitrogen
 NZ  14.00670 ;     Arg NH (NH2)
 NE  14.00670 ;     Arg NE (NH)
  C  12.01100 ;     bare carbon
CH1  13.01900 ;     aliphatic or sugar CH-group
CH2  14.02700 ;     aliphatic or sugar CH2-group
CH3  15.03500 ;     aliphatic CH3-group
```

Note: GROMACS makes use of the atom types as a name, *not* as a number (as *e.g.* in GROMOS).

5.2.2 Virtual sites

Some force fields use virtual interaction sites (interaction sites that are constructed from other particle positions) on which certain interactions are located (*e.g.* on benzene rings, to reproduce the correct quadrupole). This is described in sec. 4.7.

To make virtual sites in your system, you should include a section `[ virtual_sites? ]` (for backward compatibility the old name `[ dummies? ]` can also be used) in your topology file, where the '`?`' stands for the number constructing particles for the virtual site. This will be '`2`' for type 2, '`3`' for types 3, 3fd, 3fad and 3out and '`4`' for type 4fdn. The last of these replace an older 4fd type (with the 'type' value 1) that could occasionally be unstable; while it is still supported internally in the code, the old 4fd type should not be used in new input files. The different types are explained in sec. 4.7.

Parameters for type 2 should look like this:

```
[ virtual_sites2 ]
```

```
; Site  from        funct  a
5       1     2     1      0.7439756
```

for type 3 like this:

```
[ virtual_sites3 ]
; Site  from              funct   a          b
5       1     2     3     1       0.7439756  0.128012
```

for type 3fd like this:

```
[ virtual_sites3 ]
; Site  from              funct   a          d
5       1     2     3     2       0.5        -0.105
```

for type 3fad like this:

```
[ virtual_sites3 ]
; Site  from              funct   theta      d
5       1     2     3     3       120        0.5
```

for type 3out like this:

```
[ virtual_sites3 ]
; Site  from              funct   a          b          c
5       1     2     3     4       -0.4       -0.4       6.9281
```

for type 4fdn like this:

```
[ virtual_sites4 ]
; Site  from                    funct   a          b          c
5       1     2     3     4     2       1.0        0.9       0.105
```

This will result in the construction of a virtual site, number 5 (first column ‘Site’), based on the positions of the atoms whose indices are 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 . .’). Obviously, the atom numbers (including virtual site number) depend on the molecule. It may be instructive to study the topologies for TIP4P or TIP5P water models that are included with the GROMACS distribution.

Note that if any constant bonded interactions are defined between virtual sites and/or normal atoms, they will be removed by grompp (unless the option tt -normvsbds is used). This removal of bonded interactions is done after generating exclusions, as the generation of exclusions is based on “chemically” bonded interactions.

Virtual sites can be constructed in a more generic way using basic geometric parameters. The directive that can be used is [virtual_sitesn]. Required parameters are listed in Table 5.5. An example entry for defining a virtual site at the center of geometry of a given set of atoms might be:

```
[ virtual_sitesn ]
; Site   funct    from
5        1        1     2     3     4
```

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

5.3 Parameter files

5.3.1 Atoms

The *static* properties (see Table 5.2 assigned to the atom types are assigned based on data in several places. The mass is listed in `atomtypes.atp` (see 5.2.1), whereas the charge is listed in `*.rtp` (.rtp = **r**esidue **t**opology **p**arameter file, see 5.6.1). This implies that the charges are only defined in the building blocks of amino acids, nucleic acids or otherwise, as defined by the user. When generating a topology (`*.top`) using the `pdb2gmx` program, the information from these files is combined.

5.3.2 Non-bonded parameters

The non-bonded parameters consist of the van der Waals parameters V (`c6` or σ, depending on the combination rule) and W (`c12` or ϵ), as listed in the file `ffnonbonded.itp`, where `ptype` is the particle type (see Table 5.1). As with the bonded parameters, entries in `[ *type ]` directives are applied to their counterparts in the topology file. Missing parameters generate warnings, except as noted below in section 5.3.4.

```
[ atomtypes ]
;name   at.num      mass      charge   ptype          V(c6)         W(c12)
    O        8  15.99940       0.000       A   0.22617E-02   0.74158E-06
   OM        8  15.99940       0.000       A   0.22617E-02   0.74158E-06
   .....

[ nonbond_params ]
  ; i    j func       V(c6)         W(c12)
    O    O    1 0.22617E-02   0.74158E-06
    O   OA    1 0.22617E-02   0.13807E-05
    .....
```

Note that most of the included force fields also include the `at.num.` column, but this same information is implied in the OPLS-AA `bond_type` column. The interpretation of the parameters V and W depends on the combination rule that was chosen in the `[ defaults ]` section of the topology file (see 5.7.1):

$$\text{for combination rule 1 :} \qquad \begin{array}{rcccll} \mathrm{V}_{ii} & = & C_i^{(6)} & = & 4\,\epsilon_i\sigma_i^6 & [\ \mathrm{kJ\ mol^{-1}\ nm^6}\ ] \\ \mathrm{W}_{ii} & = & C_i^{(12)} & = & 4\,\epsilon_i\sigma_i^{12} & [\ \mathrm{kJ\ mol^{-1}\ nm^{12}}\ ] \end{array} \tag{5.1}$$

$$\text{for combination rules 2 and 3 :} \quad \begin{array}{rcll} \mathrm{V}_{ii} & = & \sigma_i & [\text{ nm }] \\ \mathrm{W}_{ii} & = & \epsilon_i & [\text{ kJ mol}^{-1}\text{ }] \end{array} \tag{5.2}$$

Some or all combinations for different atom types can be given in the `[ nonbond_params ]` section, again with parameters V and W as defined above. Any combination that is not given will be computed from the parameters for the corresponding atom types, according to the combination rule:

$$\text{for combination rules 1 and 3 :} \quad \begin{array}{rcl} C_{ij}^{(6)} & = & \left(C_i^{(6)}\, C_j^{(6)}\right)^{\frac{1}{2}} \\ C_{ij}^{(12)} & = & \left(C_i^{(12)}\, C_j^{(12)}\right)^{\frac{1}{2}} \end{array} \tag{5.3}$$

$$\text{for combination rule 2 :} \quad \begin{array}{rcl} \sigma_{ij} & = & \frac{1}{2}(\sigma_i + \sigma_j) \\ \epsilon_{ij} & = & \sqrt{\epsilon_i\, \epsilon_j} \end{array} \tag{5.4}$$

When σ and ϵ need to be supplied (rules 2 and 3), it would seem it is impossible to have a non-zero C^{12} combined with a zero C^6 parameter. However, providing a negative σ will do exactly that, such that C^6 is set to zero and C^{12} is calculated normally. This situation represents a special case in reading the value of σ, and nothing more.

There is only one set of combination rules for Buckingham potentials:

$$\begin{array}{rcl} A_{ij} & = & (A_{ii}\, A_{jj})^{1/2} \\ B_{ij} & = & 2/\left(\frac{1}{B_{ii}} + \frac{1}{B_{jj}}\right) \\ C_{ij} & = & (C_{ii}\, C_{jj})^{1/2} \end{array} \tag{5.5}$$

5.3.3 Bonded parameters

The bonded parameters (*i.e.* bonds, bond angles, improper and proper dihedrals) are listed in `ffbonded.itp`. The entries in this database describe, respectively, the atom types in the interactions, the type of the interaction, and the parameters associated with that interaction. These parameters are then read by `grompp` when processing a topology and applied to the relevant bonded parameters, *i.e.* `bondtypes` are applied to entries in the `[ bonds ]` directive, etc. Any bonded parameter that is missing from the relevant `[ *type ]` directive generates a fatal error. The types of interactions are listed in Table 5.5. Example excerpts from such files follow:

```
[ 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
```

In the `ffbonded.itp` file, you can add bonded parameters. If you want to include parameters for new atom types, make sure you define them in `atomtypes.atp` as well.

5.3.4 Intramolecular pair interactions

Extra Lennard-Jones and electrostatic interactions between pairs of atoms in a molecule can be added in the `[ pairs ]` section of a molecule definition. The parameters for these interactions can be set independently from the non-bonded interaction parameters. In the GROMOS force fields, pairs are only used to modify the 1-4 interactions (interactions of atoms separated by three bonds). In these force fields the 1-4 interactions are excluded from the non-bonded interactions (see sec. 5.4).

```
[ 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
    .....
```

The pair interaction parameters for the atom types in `ffnonbonded.itp` are listed in the `[ pairtypes ]` section. The GROMOS force fields list all these interaction parameters explicitly, but this section might be empty for force fields like OPLS that calculate the 1-4 interactions by uniformly scaling the parameters. Pair parameters that are not present in the `[ pairtypes ]` section are only generated when `gen-pairs` is set to "yes" in the `[ defaults ]` directive of `forcefield.itp` (see 5.7.1). When `gen-pairs` is set to "no," `grompp` will give a warning for each pair type for which no parameters are given.

The normal pair interactions, intended for 1-4 interactions, have function type 1. Function type 2 and the `[ pairs_nb ]` are intended for free-energy simulations. When determining hydration free energies, the solute needs to be decoupled from the solvent. This can be done by adding a

B-state topology (see sec. 3.12) that uses zero for all solute non-bonded parameters, *i.e.* charges and LJ parameters. However, the free energy difference between the A and B states is not the total hydration free energy. One has to add the free energy for reintroducing the internal Coulomb and LJ interactions in the solute when in vacuum. This second step can be combined with the first step when the Coulomb and LJ interactions within the solute are not modified. For this purpose, there is a pairs function type 2, which is identical to function type 1, except that the B-state parameters are always identical to the A-state parameters. For searching the parameters in the `[ pairtypes ]` section, no distinction is made between function type 1 and 2. The pairs section `[ pairs_nb ]` is intended to replace the non-bonded interaction. It uses the unscaled charges and the non-bonded LJ parameters; it also only uses the A-state parameters. **Note** that one should add exclusions for all atom pairs listed in `[ pairs_nb ]`, otherwise such pairs will also end up in the normal neighbor lists.

Alternatively, this same behavior can be achieved without ever touching the topology, by using the `couple-moltype`, `couple-lambda0`, `couple-lambda1`, and `couple-intramol` keywords. See sections sec. 3.12 and sec. 6.1 for more information.

All three pair types always use plain Coulomb interactions, even when Reaction-field, PME, Ewald or shifted Coulomb interactions are selected for the non-bonded interactions. Energies for types 1 and 2 are written to the energy and log file in separate "LJ-14" and "Coulomb-14" entries per energy group pair. Energies for `[ pairs_nb ]` are added to the "LJ-(SR)" and "Coulomb-(SR)" terms.

5.3.5 Implicit solvation parameters

Starting with GROMACS 4.5, implicit solvent is supported. A section in the topology has been introduced to list those parameters:

```
[ implicit_genborn_params ]
; Atomtype  sar     st   pi      gbr      hct
NH1         0.155   1    1.028   0.17063  0.79 ; N
N           0.155   1    1       0.155    0.79 ; Proline backbone N
H           0.1     1    1       0.115    0.85 ; H
CT1         0.180   1    1.276   0.190    0.72 ; C
```

In this example the atom type is listed first, followed by five numbers, and a comment (following a semicolon).

Values in columns 1-3 are not currently used. They pertain to more elaborate surface area algorithms, the one from Qiu *et al.* [70] in particular. Column 4 contains the atomic van der Waals radii, which are used in computing the Born radii. The dielectric offset is specified in the `*.mdp` file, and gets subtracted from the input van der Waals radii for the different Born radii methods, as described by Onufriev *et al.* [72]. Column 5 is the scale factor for the HCT and OBC models. The values are taken from the Tinker implementation of the HCT pairwise scaling method [71]. This method has been modified such that the scaling factors have been adjusted to minimize differences between analytical surface areas and GB using the HCT algorithm. The scaling is further modified in that it is not applied pairwise as proposed by Hawkins *et al.* [71], but on a per-atom (rather than a per-pair) basis.

5.4 Exclusions

The exclusions for non-bonded interactions 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 5.7.1). Particles are considered bonded when they are connected by "chemical" bonds (`[ bonds ]` types 1 to 5, 7 or 8) or constraints (`[ constraints ]` type 1). Type 5 `[ bonds ]` can be used to create a connection between two atoms without creating an interaction. There is a harmonic interaction (`[ bonds ]` type 6) that does not connect the atoms by a chemical bond. There is also a second constraint type (`[ constraints ]` type 2) that fixes the distance, but does not connect the atoms by a chemical bond. For a complete list of all these interactions, see Table 5.5.

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).

5.5 Constraint algorithms

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 type 2 is not (see sec. 5.4). 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 5.7.5). Several types of bonds and angles (see Table 5.5) 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 [45], which is an analytical solution of SHAKE, specifically for water. SETTLE can be selected in the topology file. See, 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 `[ settles ]` directive defines the first atom of the water molecule. The settle funct is always 1, and the distance between O-H and H-H distances must be given. **Note** that the algorithm can also be used for TIP3P and TIP4P [125]. TIP3P just has another geometry. TIP4P has a virtual site, but since that is generated it does not need to be shaken (nor stirred).

5.6 pdb2gmx input files

The GROMACS program `pdb2gmx` generates a topology for the input coordinate file. Several formats are supported for that coordinate file, but `*.pdb` is the most commonly-used format (hence the name `pdb2gmx`). `pdb2gmx` searches for force fields in sub-directories of the GROMACS `share/top` directory and your working directory. Force fields are recognized from the file `forcefield.itp` in a directory with the extension `.ff`. The file `forcefield.doc` may be present, and if so, its first line will be used by `pdb2gmx` to present a short description to the user to help in choosing a force field. Otherwise, the user can choose a force field with the `-ff xxx` command-line argument to `pdb2gmx`, which indicates that a force field in a `xxx.ff` directory is desired. `pdb2gmx` will search first in the working directory, then in the GROMACS `share/top` directory, and use the first matching `xxx.ff` directory found.

Two general files are read by `pdb2gmx`: an atom type file (extension `.atp`, see 5.2.1) from the force-field directory, and a file called `residuetypes.dat` from either the working directory, or the GROMACS `share/top` directory. `residuetypes.dat` determines which residue names are considered protein, DNA, RNA, water, and ions.

`pdb2gmx` can read one or multiple databases with topological information for different types of molecules. A set of files belonging to one database should have the same basename, preferably telling something about the type of molecules (*e.g.* aminoacids, rna, dna). The possible files are:

- `<basename>.rtp`
- `<basename>.r2b` (optional)
- `<basename>.arn` (optional)
- `<basename>.hdb` (optional)
- `<basename>.n.tdb` (optional)
- `<basename>.c.tdb` (optional)

Only the `.rtp` file, which contains the topologies of the building blocks, is mandatory. Information from other files will only be used for building blocks that come from an `.rtp` file with the same base name. The user can add building blocks to a force field by having additional files with the same base name in their working directory. By default, only extra building blocks can be defined, but calling `pdb2gmx` with the `-rtpo` option will allow building blocks in a local file to replace the default ones in the force field.

5.6.1 Residue database

The files holding the residue databases have the extension `.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 database file contains information (bonds, charges, 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 5.7.1), or in a `.rtp` file in the working directory as explained in sec. 5.6. Defining topologies of new small molecules is probably easier by writing an include topology file `*.itp` directly. This will be discussed in section 5.7.2. When adding a new protein residue to the database, don't forget to add the residue name to the `residuetypes.dat` file, so that `grompp`, `make_ndx` and analysis tools can recognize the residue as a protein residue (see 8.1.1).

The `.rtp` files are only used by `pdb2gmx`. As mentioned before, the only extra information this program needs from the `.rtp` database is bonds, charges of atoms, charge groups, and improper dihedrals, because the rest is read from the coordinate input 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 such as ligands, polyatomic ions, crystallization co-solvents, etc. 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, indicating 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 ordered consecutively. When using the hydrogen database with `pdb2gmx` for adding missing hydrogens (see 5.6.4), the atom names defined in the `.rtp` entry should correspond exactly to the naming convention used in the hydrogen database. 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. Explicit parameters added to bonds, angles, dihedrals, and impropers 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 GROMOS-96 `.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 most force fields the `[ angles ]` field is only useful for overriding `.itp` parameters. For the GROMOS-96 force field the interaction number of all angles needs 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 function on a rotatable bond. In the case of CHARMM27 FF `pdb2gmx` can add correction maps to the dihedrals using the default `-cmap` option. Please refer to 4.10.4 for more information.

`pdb2gmx` sets the number of 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 that are separated 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 separate lines. All non-bonded and pair interactions between these atoms will be excluded.

5.6.2 Residue to building block database

Each force field has its own naming convention for residues. Most residues have consistent naming, but some, especially those with different protonation states, can have many different names. The `.r2b` files are used to convert standard residue names to the force-field build block names. If

ARG	protonated arginine
ARGN	neutral arginine
ASP	negatively charged aspartic acid
ASPH	neutral aspartic acid
CYS	neutral cysteine
CYS2	cysteine with sulfur bound to another cysteine or a heme
GLU	negatively charged glutamic acid
GLUH	neutral glutamic acid
HISD	neutral histidine with N_δ protonated
HISE	neutral histidine with N_ϵ protonated
HISH	positive histidine with both N_δ and N_ϵ protonated
HIS1	histidine bound to a heme
LYSN	neutral lysine
LYS	protonated lysine
HEME	heme

Table 5.3: Internal GROMACS residue naming convention.

no `.r2b` is present in the force-field directory or a residue is not listed, the building block name is assumed to be identical to the residue name. The `.r2b` can contain 2 or 5 columns. The 2-column format has the residue name in the first column and the building block name in the second. The 5-column format has 3 additional columns with the building block for the residue occurring in the N-terminus, C-terminus and both termini at the same time (single residue molecule). This is useful for, for instance, the AMBER force fields. If one or more of the terminal versions are not present, a dash should be entered in the corresponding column.

There is a GROMACS naming convention for residues which is only apparent (except for the `pdb2gmx` code) through the `.r2b` file and `specbond.dat` files. This convention is only of importance when you are adding residue types to an `.rtp` file. The convention is listed in Table 5.3. For special bonds with, for instance, a heme group, the GROMACS naming convention is introduced through `specbond.dat` (see 5.6.7), which can subsequently be translated by the `.r2b` file, if required.

5.6.3 Atom renaming database

Force fields often use atom names that do not follow IUPAC or PDB convention. The `.arn` database is used to translate the atom names in the coordinate file to the force-field names. Atoms that are not listed keep their names. The file has three columns: the building block name, the old atom name, and the new atom name, respectively. The residue name supports question-mark wildcards that match a single character.

An additional general atom renaming file called `xlateat.dat` is present in the `share/top` directory, which translates common non-standard atom names in the coordinate file to IUPAC/PDB convention. Thus, when writing force-field files, you can assume standard atom names and no further atom name translation is required, except for translating from standard atom names to the force-field ones.

5.6.4 Hydrogen database

The hydrogen database is stored in `.hdb` files. It contains information for the `pdb2gmx` program on how to connect hydrogen atoms to existing atoms. In versions of the database before GROMACS 3.3, hydrogen atoms were named after the atom they are connected to: the first letter of the atom name was replaced by an 'H.' In the versions from 3.3 onwards, the H atom has to be listed explicitly, because the old behavior was protein-specific and hence could not be generalized to other molecules. If more than 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 5.6.1) must be the same. The format of the hydrogen database is as follows:

```
; res    # additions
         # H add type    H        i        j        k
ALA      1
         1       1       H        N        -C       CA
ARG      4
         1       2       H        N        CA       C
         1       1       HE       NE       CD       CZ
         2       3       HH1      NH1      CZ       NE
         2       3       HH2      NH2      CZ       NE
```

On the first line we see the residue name (ALA or ARG) and the number of kinds of hydrogen atoms that may be added to this residue by the hydrogen database. After that follows one line for each addition, on which we see:

- The number of H atoms added
- The method for adding H atoms, which 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 plane 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 $> 90^o$.

  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. *ethylene -C=CH_2, or amide -C(=O)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 [126].

  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)=109.47^o, dihedral (n1-i-j-k)=trans, (n2-i-j-k)=trans+120 and (n3-i-j-k)=trans+240^o.

5 *one tetrahedral hydrogen,* e.g. C_3CH
One hydrogen atom (n′) is generated at a distance of 0.1 nm from atom i in tetrahedral conformation such that angle (n′-i-j)=(n′-i-k)=(n′-i-l)=109.47°.

6 *two tetrahedral hydrogens,* e.g. $C\text{-}CH_2\text{-}C$
Two hydrogen atoms (n1,n2) are generated at a distance of 0.1 nm from atom i in tetrahedral conformation on the plane bisecting angle j-i-k with angle (n1-i-n2)=(n1-i-j)=(n1-i-k)=109.47°.

7 *two water hydrogens*
Two hydrogens are generated around atom i according to SPC [81] water geometry. The symmetry axis will alternate between three coordinate axes in both directions.

10 *three water "hydrogens"*
Two hydrogens are generated around atom i according to SPC [81] water geometry. The symmetry axis will alternate between three coordinate axes in both directions. In addition, an extra particle is generated on the position of the oxygen with the first letter of the name replaced by 'M'. This is for use with four-atom water models such as TIP4P [125].

11 *four water "hydrogens"*
Same as above, except that two additional particles are generated on the position of the oxygen, with names 'LP1' and 'LP2.' This is for use with five-atom water models such as TIP5P [127].

- The name of the new H atom (or its prefix, *e.g.* `HD2` for the asparagine example given earlier).
- 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. For water, there is only one control atom.

Some more exotic cases can be approximately constructed from the above tools, and with suitable use of energy minimization are good enough for beginning MD simulations. For example secondary amine hydrogen, nitrenyl hydrogen (C=NH) and even ethynyl hydrogen could be approximately constructed using method 2 above for hydroxyl hydrogen.

5.6.5 Termini database

The termini databases are stored in `aminoacids.n.tdb` and `aminoacids.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. Their format is as follows (from `gromos43a1.ff/aminoacids.c.tdb`):

```
[ None ]

[ COO- ]
[ replace ]
C  C  C  12.011  0.27
O   O1  OM  15.9994  -0.635
```

```
OXT O2 OM 15.9994 -0.635
[ add ]
2 8 O C CA N
OM 15.9994 -0.635
[ bonds ]
C O1 gb_5
C O2 gb_5
[ angles ]
O1 C O2 ga_37
CA C O1 ga_21
CA C O2 ga_21
[ dihedrals ]
N CA C O2 gd_20
[ impropers ]
C CA O2 O1 gi_1
```

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 `[ COO- ]` is the first block, corresponding to changing the terminal carbon atom into a deprotonated carboxyl group. `[ None ]` is the second terminus type, corresponding to a terminus that leaves the molecule as it is. Block names cannot be any of the following: `replace`, `add`, `delete`, `bonds`, `angles`, `dihedrals`, `impropers`. Doing so would interfere with the parameters of the block, and would probably also be very confusing to human readers.

For each block the following options are present:

- `[ replace ]`
  Replace an existing atom by one with a different atom type, atom name, charge, and/or mass. This entry can be used to replace an atom that is present both in the input coordinates and in the `.rtp` database, but also to only rename an atom in the input coordinates such that it matches the name in the force field. In the latter case, there should also be a corresponding `[ add ]` section present that gives instructions to add the same atom, such that the position in the sequence and the bonding is known. Such an atom can be present in the input coordinates and kept, or not present and constructed by `pdb2gmx`. 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 (optional)
  - 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 (name of the new atom, number of atoms, type of addition, control atoms, see 5.6.4), 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, respectively, 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, respectively.

After this line, another line follows that specifies the details of the added atom(s), in the same way as for replacing atoms, *i.e.*:

- atom type
- mass
- charge
- charge group (optional)

Like in the hydrogen database (see 5.6.1), when more than one atom is connected to an existing one, a number will be appended to the end of the atom name. **Note** that, like in the hydrogen database, the atom name is now on the same line as the control atoms, whereas it was at the beginning of the second line prior to GROMACS version 3.3. When the charge group field is left out, the added atom will have the same charge group number as the atom that it is bonded to.

- `[ 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 `*.rtp` file, see 5.6.1.

5.6.6 Virtual site database

Since we cannot rely on the positions of hydrogens in input files, we need a special input file to decide the geometries and parameters with which to add virtual site hydrogens. For more complex virtual site constructs (*e.g.* when entire aromatic side chains are made rigid) we also need information about the equilibrium bond lengths and angles for all atoms in the side chain. This information is specified in the `.vsd` file for each force field. Just as for the termini, there is one such file for each class of residues in the `.rtp` file.

The virtual site database is not really a very simple list of information. The first couple of sections specify which mass centers (typically called MCH_3/MNH_3) to use for CH_3, NH_3, and NH_2 groups. Depending on the equilibrium bond lengths and angles between the hydrogens and heavy atoms we need to apply slightly different constraint distances between these mass centers. **Note** that we do *not* have to specify the actual parameters (that is automatic), just the type of mass center to use. To accomplish this, there are three sections names `[ CH3 ]`, `[ NH3 ]`, and `[ NH2 ]`. For each of these we expect three columns. The first column is the atom type bound to the 2/3 hydrogens,

the second column is the next heavy atom type which this is bound, and the third column the type of mass center to use. As a special case, in the `[ NH2 ]` section it is also possible to specify `planar` in the second column, which will use a different construction without mass center. There are currently different opinions in some force fields whether an NH_2 group should be planar or not, but we try hard to stick to the default equilibrium parameters of the force field.

The second part of the virtual site database contains explicit equilibrium bond lengths and angles for pairs/triplets of atoms in aromatic side chains. These entries are currently read by specific routines in the virtual site generation code, so if you would like to extend it *e.g.* to nucleic acids you would also need to write new code there. These sections are named after the short amino acid names (`[ PHE ]`, `[ TYR ]`, `[ TRP ]`, `[ HID ]`, `[ HIE ]`, `[ HIP ]`), and simply contain 2 or 3 columns with atom names, followed by a number specifying the bond length (in nm) or angle (in degrees). **Note** that these are approximations of the equilibrated geometry for the entire molecule, which might not be identical to the equilibrium value for a single bond/angle if the molecule is strained.

5.6.7 Special bonds

The primary mechanism used by `pdb2gmx` to generate inter-residue bonds relies on head-to-tail linking of backbone atoms in different residues to build a macromolecule. In some cases (*e.g.* disulfide bonds, a heme group, branched polymers), it is necessary to create inter-residue bonds that do not lie on the backbone. The file `specbond.dat` takes care of this function. It is necessary that the residues belong to the same `[ moleculetype ]`. The `-merge` and `-chainsep` functions of `pdb2gmx` can be useful when managing special inter-residue bonds between different chains.

The first line of `specbond.dat` indicates the number of entries that are in the file. If you add a new entry, be sure to increment this number. The remaining lines in the file provide the specifications for creating bonds. The format of the lines is as follows:

```
resA atomA nbondsA resB atomB nbondsB length newresA newresB
```

The columns indicate:

1. `resA` The name of residue A that participates in the bond.
2. `atomA` The name of the atom in residue A that forms the bond.
3. `nbondsA` The total number of bonds `atomA` can form.
4. `resB` The name of residue B that participates in the bond.
5. `atomB` The name of the atom in residue B that forms the bond.
6. `nbondsB` The total number of bonds `atomB` can form.
7. `length` The reference length for the bond. If `atomA` and `atomB` are not within `length` $\pm$ 10% in the coordinate file supplied to `pdb2gmx`, no bond will be formed.
8. `newresA` The new name of residue A, if necessary. Some force fields use *e.g.* CYS2 for a cysteine in a disulfide or heme linkage.

9. `newresB` The new name of residue B, likewise.

5.7 File formats

5.7.1 Topology file

The topology file is built following the GROMACS specification for a molecular topology. A `*.top` file can be generated by `pdb2gmx`. All possible entries in the topology file are listed in Tables 5.4 and 5.5. Also tabulated are: all the units of the parameters, which interactions can be perturbed for free energy calculations, which bonded interactions are used by `grompp` for generating exclusions, and which bonded interactions can be converted to constraints by `grompp`.

Parameters

interaction type	directive	# at.	f. tp	parameters	F. E.
mandatory	`defaults`			non-bonded function type; combination rule$^{(cr)}$; generate pairs (no/yes); fudge LJ (); fudge QQ ()	
mandatory	`atomtypes`			atom type; m (u); q (e); particle type; V$^{(cr)}$; W$^{(cr)}$	
	`bondtypes`	(see Table 5.5, directive `bonds`)			
	`pairtypes`	(see Table 5.5, directive `pairs`)			
	`angletypes`	(see Table 5.5, directive `angles`)			
	`dihedraltypes`$^{(*)}$	(see Table 5.5, directive `dihedrals`)			
	`constrainttypes`	(see Table 5.5, directive `constraints`)			
LJ	`nonbond_params`	2	1	$V^{(cr)}$; $W^{(cr)}$	
Buckingham	`nonbond_params`	2	2	a (kJ mol^{-1}); b (nm^{-1}); c_6 (kJ mol^{-1} nm^6)	

Molecule definition(s)

mandatory	`moleculetype`			molecule name; $n_{ex}^{(nrexcl)}$	
mandatory	`atoms`	1		atom type; residue number; residue name; atom name; charge group number; q (e); m (u)	type q, m
intra-molecular interaction and geometry definitions as described in Table 5.5					

System

mandatory	`system`	system name
mandatory	`molecules`	molecule name; number of molecules

'# at' is the required number of atom type indices for this directive
'f. tp' is the value used to select this function type
'F. E.' indicates which of the parameters for this interaction can be interpolated during free energy calculations
$^{(cr)}$ the combination rule determines the type of LJ parameters, see 5.3.2
$^{(*)}$ for `dihedraltypes` one can specify 4 atoms or the inner (outer for improper) 2 atoms
$^{(nrexcl)}$ exclude neighbors n_{ex} bonds away for non-bonded interactions
For free energy calculations, type, q and m or no parameters should be added for topology 'B' ($\lambda = 1$) on the same line, after the normal parameters.

Table 5.4: The topology (`*.top`) file.

Table 5.5: Details of `[ moleculetype ]` directives

Name of interaction	Topology file directive	num. atoms*	func. type†	Order of parameters and their units	use in F.E.?‡	Cross-references
bond	`bonds`§,¶	2	1	b_0 (nm); k_b (kJ mol^{-1} nm^{-2})	all	4.2.1
G96 bond	`bonds`§,¶	2	2	b_0 (nm); k_b (kJ mol^{-1} nm^{-4})	all	4.2.1
Morse	`bonds`§,¶	2	3	b_0 (nm); D (kJ mol^{-1}); β (nm^{-1})	all	4.2.2
cubic bond	`bonds`§,¶	2	4	b_0 (nm); $C_{i=2,3}$ (kJ mol^{-1} nm^{-i})		4.2.3
connection	`bonds`§	2	5			5.4
harmonic potential	`bonds`	2	6	b_0 (nm); k_b (kJ mol^{-1} nm^{-2})	all	4.2.1,5.4
FENE bond	`bonds`§	2	7	b_m (nm); k_b (kJ mol^{-1} nm^{-2})		4.2.4
tabulated bond	`bonds`§	2	8	table number (≥ 0); k (kJ mol^{-1})	k	4.2.14
tabulated bond‖	`bonds`	2	9	table number (≥ 0); k (kJ mol^{-1})	k	4.2.14,5.4
restraint potential	`bonds`	2	10	low, up$_1$, up$_2$ (nm); k_{dr} (kJ mol^{-1} nm^{-2})	all	4.3.5
extra LJ or Coulomb	`pairs`	2	1	V^{**}; W^{**}	all	5.3.4
extra LJ or Coulomb	`pairs`	2	2	fudge QQ (); q_i, q_j (e), V^{**}; W^{**}		5.3.4
extra LJ or Coulomb	`pairs_nb`	2	1	q_i, q_j (e); V^{**}; W^{**}		5.3.4
angle	`angles`¶	3	1	θ_0 (deg); k_θ (kJ mol^{-1} rad^{-2})	all	4.2.5
G96 angle	`angles`¶	3	2	θ_0 (deg); k_θ (kJ mol^{-1})	all	4.2.6
cross bond-bond	`angles`	3	3	r_{1e}, r_{2e} (nm); $k_{rr'}$ (kJ mol^{-1} nm^{-2})		4.2.9
cross bond-angle	`angles`	3	4	r_{1e}, r_{2e} r_{3e} (nm); $k_{r\theta}$ (kJ mol^{-1} nm^{-2})		4.2.10
Urey-Bradley	`angles`¶	3	5	θ_0 (deg); k_θ (kJ mol^{-1} rad^{-2}); r_{13} (nm); k_{UB} (kJ mol^{-1} nm^{-2})	all	4.2.8
quartic angle	`angles`¶	3	6	θ_0 (deg); $C_{i=0,1,2,3,4}$ (kJ mol^{-1} rad^{-i})		4.2.11

*The required number of atom indices for this directive
†The index to use to select this function type
‡Indicates which of the parameters for this interaction can be interpolated during free energy calculations
§This interaction type will be used by `grompp` for generating exclusions
¶This interaction type can be converted to constraints by `grompp`
**The combination rule determines the type of LJ parameters, see 5.3.2
‖No connection, and so no exclusions, are generated for this interaction

Table 5.5: Details of `[ moleculetype ]` directives

Name of interaction	Topology file directive	num. atoms*	func. type†	Order of parameters and their units	use in F.E.?‡	Cross-references
tabulated angle	`angles`	3	8	table number (≥ 0); k (kJ mol^{-1})	k	4.2.14
restricted bending potential	`angles`	3	10	θ_0 (deg); k_θ (kJ mol^{-1})		4.2.7
proper dihedral	`dihedrals`	4	1	ϕ_s (deg); k_ϕ (kJ mol^{-1}); multiplicity	ϕ, k	4.2.13
improper dihedral	`dihedrals`	4	2	ξ_0 (deg); k_ξ (kJ mol^{-1} rad^{-2})	all	4.2.12
Ryckaert-Bellemans dihedral	`dihedrals`	4	3	C_0, C_1, C_2, C_3, C_4, C_5 (kJ mol^{-1})	all	4.2.13
periodic improper dihedral	`dihedrals`	4	4	ϕ_s (deg); k_ϕ (kJ mol^{-1}); multiplicity	ϕ, k	4.2.12
Fourier dihedral	`dihedrals`	4	5	C_1, C_2, C_3, C_4 (kJ mol^{-1})	all	4.2.13
tabulated dihedral	`dihedrals`	4	8	table number (≥ 0); k (kJ mol^{-1})	k	4.2.14
proper dihedral (multiple)	`dihedrals`	4	9	ϕ_s (deg); k_ϕ (kJ mol^{-1}); multiplicity	ϕ, k	4.2.13
restricted dihedral	`dihedrals`	4	11	ϕ_0 (deg); k_ϕ (kJ mol^{-1})		4.2.13
combined bending-torsion potential	`dihedrals`	4	10	a_0, a_1, a_2, a_3, a_4 (kJ mol^{-1})		4.2.13
exclusions	`exclusions`	1		one or more atom indices		5.4
constraint	`constraints`§	2	1	b_0 (nm)	all	4.5,5.5
constraint‖	`constraints`	2	2	b_0 (nm)	all	4.5,5.5,5.4
SETTLE	`settles`	1	1	d_{OH}, d_{HH} (nm)		3.6.1,5.5
2-body virtual site	`virtual_sites2`	3	1	a ()		4.7
3-body virtual site	`virtual_sites3`	4	1	a, b ()		4.7
3-body virtual site (fd)	`virtual_sites3`	4	2	a (); d (nm)		4.7
3-body virtual site (fad)	`virtual_sites3`	4	3	θ (deg); d (nm)		4.7
3-body virtual site (out)	`virtual_sites3`	4	4	a, b (); c (nm^{-1})		4.7
4-body virtual site (fdn)	`virtual_sites4`	5	2	a, b (); c (nm)		4.7
N-body virtual site (COG)	`virtual_sitesn`	1	1	one or more constructing atom indices		4.7
N-body virtual site (COM)	`virtual_sitesn`	1	2	one or more constructing atom indices		4.7
N-body virtual site (COW)	`virtual_sitesn`	1	3	one or more pairs consisting of constructing atom index and weight		4.7
position restraint	`position_restraints`	1	1	k_x, k_y, k_z (kJ mol^{-1} nm^{-2})	all	4.3.1

Table 5.5: Details of `[ moleculetype ]` directives

Name of interaction	Topology file directive	num. atoms*	func. type†	Order of parameters and their units	use in F.E.?‡	Cross-references
flat-bottomed position restraint	`position_restraints`	1	2	g, r (nm), k (kJ mol^{-1} nm^{-2})		4.3.2
distance restraint	`distance_restraints`	2	1	type; label; low, up$_1$, up$_2$ (nm); weight ()		4.3.5
dihedral restraint	`dihedral_restraints`	4	1	ϕ_0 (deg); $\Delta\phi$ (deg);	all	4.3.4
orientation restraint	`orientation_restraints`	2	1	exp.; label; α; c (U nm$^{\alpha}$); obs. (U); weight (U^{-1})		4.3.6
angle restraint	`angle_restraints`	4	1	θ_0 (deg); k_c (kJ mol^{-1}); multiplicity	θ, k	4.3.3
angle restraint (z)	`angle_restraints_z`	2	1	θ_0 (deg); k_c (kJ mol^{-1}); multiplicity	θ, k	4.3.3

Description of the file layout:

- Semicolon (;) and newline characters surround comments
- On a line ending with \ the newline character is ignored.
- Directives are surrounded by `[` and `]`
- The topology hierarchy (which must be followed) consists of three levels:
  - the parameter level, which defines certain force-field specifications (see Table 5.4)
  - the molecule level, which should contain one or more molecule definitions (see Table 5.5)
  - the system level, containing only system-specific information (`[ system ]` and `[ molecules ]`)
- Items should be separated by spaces or tabs, not commas
- Atoms in molecules should be numbered consecutively starting at 1
- Atoms in the same charge group must be listed consecutively
- The file is parsed only once, 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
- It is possible to apply multiple bonded interactions of the same type on the same atoms
- Descriptive comment lines and empty lines are highly recommended
- Starting with GROMACS version 3.1.3, all directives at the parameter level can be used multiple times and there are no restrictions on the order, except that an atom type needs to be defined before it can be used in other parameter definitions
- If parameters for a certain interaction are defined multiple times for the same combination of atom types the last definition is used; starting with GROMACS version 3.1.3 `grompp` generates a warning for parameter redefinitions with different values
- 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.
- 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 "amber99.ff/forcefield.itp"

[ moleculetype ]
; name  nrexcl
Urea         3

[ atoms ]
   1  C  1  URE      C      1     0.880229  12.01000   ; amber C  type
   2  O  1  URE      O      2    -0.613359  16.00000   ; amber O  type
   3  N  1  URE     N1      3    -0.923545  14.01000   ; amber N  type
   4  H  1  URE    H11      4     0.395055   1.00800   ; amber H  type
   5  H  1  URE    H12      5     0.395055   1.00800   ; amber H  type
   6  N  1  URE     N2      6    -0.923545  14.01000   ; amber N  type
   7  H  1  URE    H21      7     0.395055   1.00800   ; amber H  type
   8  H  1  URE    H22      8     0.395055   1.00800   ; amber H  type

[ bonds ]
    1 2
    1 3
    1   6
    3 4
    3 5
    6 7
    6 8

[ dihedrals ]
;   ai    aj    ak    al funct  definition
     2     1     3     4   9
     2     1     3     5   9
     2     1     6     7   9
     2     1     6     8   9
     3     1     6     7   9
     3     1     6     8   9
     6     1     3     4   9
     6     1     3     5   9

[ dihedrals ]
     3     6     1     2   4
     1     4     3     5   4
     1     7     6     8   4

[ position_restraints ]
; you wouldn't normally use this for a molecule like Urea,
; but we include it here for didactic 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)

[ dihedral_restraints ]
; ai   aj    ak    al  type  label  phi  dphi  kfac  power
    3    6     1    2     1      1  180     0     1      2
    1    4     3    5     1      1  180     0     1      2

; Include TIP3P water topology
#include "amber99/tip3p.itp"

[ system ]
Urea in Water

[ molecules ]
;molecule name   nr.
Urea             1
SOL              1000
```

Here follows the explanatory text.

`#include "amber99.ff/forcefield.itp"`**:** this includes the information for the force field you are using, including bonded and non-bonded parameters. This example uses the AMBER99 force field, but your simulation may use a different force field. `grompp` will automatically go and find this file and copy-and-paste its content. That content can be seen in `share/top/amber99.ff/forcefield.itp`, and it is

```
#define _FF_AMBER
#define _FF_AMBER99

[ defaults ]
; nbfunc        comb-rule       gen-pairs       fudgeLJ fudgeQQ
1               2               yes             0.5     0.8333

#include "ffnonbonded.itp"
#include "ffbonded.itp"
#include "gbsa.itp"
```

The two `#define` statements set up the conditions so that future parts of the topology can know that the AMBER 99 force field is in use.

`[ defaults ]`**:**

- `nbfunc` is the non-bonded function type. Use 1 (Lennard-Jones) or 2 (Buckingham)
- `comb-rule` is the number of the combination rule (see 5.3.2).
- `gen-pairs` is for pair generation. The default is 'no', *i.e.* get 1-4 parameters from the pairtypes list. When parameters are not present in the list, stop with a fatal error. Setting 'yes' generates 1-4 parameters that are not present in the pair list from normal Lennard-Jones parameters using `fudgeLJ`

- `fudgeLJ` is the factor by which to multiply Lennard-Jones 1-4 interactions, default 1
- `fudgeQQ` is the factor by which to multiply electrostatic 1-4 interactions, default 1
- N is the power for the repulsion term in a 6-N potential (with nonbonded-type Lennard-Jones only), starting with GROMACS version 4.5, `mdrun` also reads and applies N, for values not equal to 12 tabulated interaction functions are used (in older version you would have to use user tabulated interactions).

Note that `gen-pairs`, `fudgeLJ`, `fudgeQQ`, and N are optional. `fudgeLJ` is only used when generate pairs is set to 'yes', and `fudgeQQ` is always used. However, if you want to specify N you need to give a value for the other parameters as well.

Then some other `#include` statements add in the large amount of data needed to describe the rest of the force field. We will skip these and return to `urea.top`. There we will see

`[ 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 ]` **:** LJ and Coulomb 1-4 interactions

`[ angles ]` **:** no comment

`[ dihedrals ]` **:** in this case there are 9 proper dihedrals (funct = 1), 3 improper (funct = 4) 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
```

In the original implementation of the potential for alkanes [128] no 1-4 interactions were used, which means that in order to implement that particular force field you need to remove the 1-4 interactions from the `[ pairs ]` section of your topology. In most modern force fields, like OPLS/AA or Amber the rules are different, and the Ryckaert-Bellemans potential is used as a cosine series in combination with 1-4 interactions.

`[ position_restraints ]` **:** harmonically restrain the selected particles to reference positions (4.3.1). The reference positions are read from a separate coordinate file by `grompp`.

`[ dihedral_restraints ]` **:** restrain selected dihedrals to a reference value. The implementation of dihedral restraints is described in section 4.3.4 of the manual. The parameters specified in the [dihedral_restraints] directive are as follows:

- `type` has only one possible value which is 1
- `label` is unused and has been removed from the code.

- `phi` is the value of ϕ_0 in eqn. 4.84 and eqn. 4.85 of the manual.
- `dphi` is the value of $\Delta\phi$ in eqn. 4.85 of the manual.
- `kfac` is analogous to `fac` in the implementation of distance restraints. It is the factor by which the force constant is multiplied. By doing so, different restraints can be maintained with different force constants.
- `power` is unused and has been removed from the code.

`#include "tip3p.itp"` **:** includes a topology file that was already constructed (see section 5.7.2).

`[ 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 molecule dissolved in 1000 water molecules. The molecule type SOL is defined in the `tip3p.itp` file. Each name here must correspond to a name given with `[ moleculetype ]` earlier in the topology. The order of the blocks of molecule types and the numbers of such molecules must match the coordinate file that accompanies the topology when supplied to `grompp`. The blocks of molecules do not need to be contiguous, but some tools (e.g. `genion`) may act only on the first or last such block of a particular molecule type. Also, these blocks have nothing to do with the definition of groups (see sec. 3.3 and sec. 8.1).

5.7.2 Molecule.itp file

If you construct a topology file you will use frequently (like the water molecule, `tip3p.itp`, which is already constructed for you) it is good to make a `molecule.itp` file. This only lists the information of one particular molecule and allows you to re-use the `[ moleculetype ]` in multiple systems without re-invoking `pdb2gmx` or manually copying and pasting. An example `urea.itp` follows:

```
[ moleculetype ]
; molname nrexcl
URE 3

[ atoms ]
   1  C  1  URE      C      1     0.880229  12.01000   ; amber C  type
...
   8  H  1  URE    H22      8     0.395055   1.00800   ; amber H  type

[ bonds ]
    1 2
...
    6 8
[ dihedrals ]
;   ai    aj    ak    al funct  definition
     2     1     3     4   9
...
```

```
     6     1     3     5   9
[ dihedrals ]
     3     6     1     2   4
     1     4     3     5   4
     1     7     6     8   4
```

Using `*.itp` files results in a very short `*.top` file:

```
;
;       Example topology file
;
; The force field files to be included
#include "amber99.ff/forcefield.itp"

#include "urea.itp"

; Include TIP3P water topology
#include "amber99/tip3p.itp"

[ system ]
Urea in Water

[ molecules ]
;molecule name   nr.
Urea             1
SOL              1000
```

5.7.3 Ifdef statements

A very powerful feature in GROMACS is the use of `#ifdef` statements in your `*.top` file. By making use of this statement, and associated `#define` statements like were seen in `amber99.ff/forcefield.itp` earlier, 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.* [129] or by Van Buuren and Berendsen [130]. In fact, you can use much of the functionality of the C preprocessor, `cpp`, because `grompp` contains similar pre-processing functions to scan the file. The way to make use of the `#ifdef` option is as follows:

- either use the option `define = -DDeLoof` in the `*.mdp` file (containing `grompp` input parameters), or use the line `#define DeLoof` early in your `*.top` or `*.itp` file; and
- 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
   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
...
```

This mechanism is used by `pdb2gmx` to implement optional position restraints (4.3.1) by `#include`-ing an `.itp` file whose contents will be meaningful only if a particular `#define` is set (and spelled correctly!)

5.7.4 Topologies for free energy calculations

Free energy differences between two systems, A and B, can be calculated as described in sec. 3.12. 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 under the `[ atoms ]` directive. 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 Tables 5.4 and 5.5. The λ-dependence of the interactions is described in section sec. 4.5. The bonded parameters that are used (on the line of the bonded interaction definition, or the ones looked up on atom types in the bonded type lists) is explained in Table 5.6. In most cases, things should work intuitively. When the A and B atom types in a bonded interaction are not all identical and parameters are not present for the B-state, either on the line or in the bonded types, `grompp` uses the A-state parameters and issues a warning. For free energy calculations, all or no 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. From GROMACS 4.6 onward, if λ is treated as a

B-state atom types all identical to A-state atom types	parameters on line		parameters in bonded types				message
			A atom types		B atom types		
	A	B	A	B	A	B	
yes	+AB	–	x	x			error
	+A	+B	x	x			
	–	–	–	–			
	–	–	+AB	–			
	–	–	+A	+B			
no	+AB	–	x	x	x	x	warning
	+A	+B	x	x	x	x	
	–	–	–	–	x	x	error
	–	–	+AB	–	–	–	warning
	–	–	+A	+B	–	–	warning
	–	–	+A	x	+B	–	
	–	–	+A	x	+	+B	

Table 5.6: The bonded parameters that are used for free energy topologies, on the line of the bonded interaction definition or looked up in the bond types section based on atom types. A and B indicate the parameters used for state A and B respectively, + and – indicate the (non-)presence of parameters in the topology, x indicates that the presence has no influence.

vector, then the `bonded-lambdas` component controls all bonded terms that are not explicitly labeled as restraints. Restrain terms are controlled by the `restraint-lambdas` component.

Below is an example of a topology which changes from 200 propanols to 200 pentanes using the GROMOS-96 force field.

```
; Include force field parameters
#include "gromos43a1.ff/forcefield.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-state parameter specifications, since the B parameters will be copied from the A parameters. Bonded interactions between atoms that are not perturbed do not need B parameter specifications, as is the case for the last bond in the example topology. Topologies using the OPLS/AA 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 interactions 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
```

5.7.5 Constraint forces

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 nm 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. 6.4. Below is an example for calculating the constraint force at 0.7 nm between two methanes in water, by combining the two methanes into one "molecule." **Note** that the definition of a "molecule" in GROMACS does not necessarily correspond to the chemical definition of a molecule. In GROMACS, a "molecule" can be defined as any group of atoms that one wishes to consider simultaneously. The added constraint is of function type 2, which means that it is not used for generating exclusions (see sec. 5.4). Note that the constraint free energy term is included in the derivative term, and is specifically included in the `bonded-lambdas` component. However, the free energy for changing constraints is *not* included in the potential energy differences used for BAR and MBAR, as this requires reevaluating the energy at each of the constraint components. This functionality is planned for later versions.

```
; Include force-field parameters
#include "gromos43a1.ff/forcefield.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 "gromos43a1.ff/spc.itp"

[ system ]
; Name
Methanes in Water

[ molecules ]
; Compound        #mols
Methanes             1
SOL                2002
```

5.7.6 Coordinate file

Files with the `.gro` file extension contain a molecular structure in GROMOS-87 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.

5.8 Force field organization

5.8.1 Force-field files

Many force fields are available by default. Force fields are detected by the presence of `<name>.ff` directories in the `$GMXLIB/share/gromacs/top` sub-directory and/or the working directory. The information regarding the location of the force field files is printed by `pdb2gmx` so you can easily keep track of which version of a force field is being called, in case you have made modifications in one location or another. The force fields included with GROMACS are:

- AMBER03 protein, nucleic AMBER94 (Duan et al., J. Comp. Chem. 24, 1999-2012, 2003)
- AMBER94 force field (Cornell et al., JACS 117, 5179-5197, 1995)
- AMBER96 protein, nucleic AMBER94 (Kollman et al., Acc. Chem. Res. 29, 461-469, 1996)
- AMBER99 protein, nucleic AMBER94 (Wang et al., J. Comp. Chem. 21, 1049-1074, 2000)
- AMBER99SB protein, nucleic AMBER94 (Hornak et al., Proteins 65, 712-725, 2006)
- AMBER99SB-ILDN protein, nucleic AMBER94 (Lindorff-Larsen et al., Proteins 78, 1950-58, 2010)

- AMBERGS force field (Garcia & Sanbonmatsu, PNAS 99, 2782-2787, 2002)
- CHARMM27 all-atom force field (CHARM22 plus CMAP for proteins)
- GROMOS96 43a1 force field
- GROMOS96 43a2 force field (improved alkane dihedrals)
- GROMOS96 45a3 force field (Schuler JCC 2001 22 1205)
- GROMOS96 53a5 force field (JCC 2004 vol 25 pag 1656)
- GROMOS96 53a6 force field (JCC 2004 vol 25 pag 1656)
- GROMOS96 54a7 force field (Eur. Biophys. J. (2011), 40,, 843-856, DOI: 10.1007/s00249-011-0700-9)
- OPLS-AA/L all-atom force field (2001 aminoacid dihedrals)

A force field is included at the beginning of a topology file with an `#include` statement followed by `<name>.ff/forcefield.itp`. This statement includes the force-field file, which, in turn, may include other force-field files. All the force fields are organized in the same way. An example of the `amber99.ff/forcefield.itp` was shown in 5.7.1.

For each force field, there several files which are only used by `pdb2gmx`. These are: residue databases (`.rtp`, see 5.6.1) the hydrogen database (`.hdb`, see 5.6.4), two termini databases (`.n.tdb` and `.c.tdb`, see 5.6.5) and the atom type database (`.atp`, see 5.2.1), which contains only the masses. Other optional files are described in sec. 5.6.

5.8.2 Changing force-field parameters

If one wants to change the parameters of few bonded interactions in a molecule, this is most easily accomplished by typing the parameters behind the definition of the bonded interaction directly in the `*.top` file under the `[ moleculetype ]` section (see 5.7.1 for the format and units). If one wants to change the parameters for all instances of a certain interaction one can change them in the force-field file or add a new `[ ???types ]` section after including the force field. When parameters for a certain interaction are defined multiple times, the last definition is used. As of GROMACS version 3.1.3, a warning is generated when parameters are redefined with a different value. Changing the Lennard-Jones parameters of an atom type is not recommended, because in the GROMOS force fields the Lennard-Jones parameters for several combinations of atom types are not generated according to the standard combination rules. Such combinations (and possibly others that do follow the combination rules) are defined in the `[ nonbond_params ]` section, and changing the Lennard-Jones parameters of an atom type has no effect on these combinations.

5.8.3 Adding atom types

As of GROMACS version 3.1.3, atom types can be added in an extra `[ atomtypes ]` section after the the inclusion of the normal force field. After the definition of the new atom type(s), additional non-bonded and pair parameters can be defined. In pre-3.1.3 versions of GROMACS, the new atom types needed to be added in the `[ atomtypes ]` section of the force-field files, because all non-bonded parameters above the last `[ atomtypes ]` section would be overwritten using the standard combination rules.

Chapter 6

Special Topics

6.1 Free energy implementation

For free energy calculations, there are two things that must be specified; the end states, and the pathway connecting the end states. The end states can be specified in two ways. The most straightforward is through the specification of end states in the topology file. Most potential forms support both an A state and a B state. Whenever both states are specified, then the A state corresponds to the initial free energy state, and the B state corresponds to the final state.

In some cases, the end state can also be defined in some cases without altering the topology, solely through the `.mdp` file, through the use of the `couple-moltype`,`couple-lambda0`, `couple-lambda1`, and `couple-intramol` mdp keywords. Any molecule type selected in `couple-moltype` will automatically have a B state implicitly constructed (and the A state redefined) according to the `couple-lambda` keywords. `couple-lambda0` and `couple-lambda1` define the non-bonded parameters that are present in the A state (`couple-lambda0`) and the B state (`couple-lambda1`). The choices are 'q','vdw', and 'vdw-q'; these indicate the Coulombic, van der Waals, or both parameters that are turned on in the respective state.

Once the end states are defined, then the path between the end states has to be defined. This path is defined solely in the .mdp file. Starting in 4.6, λ is a vector of components, with Coulombic, van der Waals, bonded, restraint, and mass components all able to be adjusted independently. This makes it possible to turn off the Coulombic term linearly, and then the van der Waals using soft core, all in the same simulation. This is especially useful for replica exchange or expanded ensemble simulations, where it is important to sample all the way from interacting to non-interacting states in the same simulation to improve sampling.

`fep-lambdas` is the default array of λ values ranging from 0 to 1. All of the other lambda arrays use the values in this array if they are not specified. The previous behavior, where the pathway is controlled by a single λ variable, can be preserved by using only `fep-lambdas` to define the pathway.

For example, if you wanted to first to change the Coulombic terms, then the van der Waals terms, changing bonded at the same time rate as the van der Waals, but changing the restraints throughout

the first two-thirds of the simulation, then you could use this λ vector:

```
coul-lambdas           = 0.0 0.2 0.5 1.0 1.0 1.0 1.0 1.0 1.0 1.0
vdw-lambdas            = 0.0 0.0 0.0 0.0 0.4 0.5 0.6 0.7 0.8 1.0
bonded-lambdas         = 0.0 0.0 0.0 0.0 0.4 0.5 0.6 0.7 0.8 1.0
restraint-lambdas      = 0.0 0.0 0.1 0.2 0.3 0.5 0.7 1.0 1.0 1.0
```

This is also equivalent to:

```
fep-lambdas            = 0.0 0.0 0.0 0.0 0.4 0.5 0.6 0.7 0.8 1.0
coul-lambdas           = 0.0 0.2 0.5 1.0 1.0 1.0 1.0 1.0 1.0 1.0
restraint-lambdas      = 0.0 0.0 0.1 0.2 0.3 0.5 0.7 1.0 1.0 1.0
```

The `fep-lambda array`, in this case, is being used as the default to fill in the bonded and van der Waals λ arrays. Usually, it's best to fill in all arrays explicitly, just to make sure things are properly assigned.

If you want to turn on only restraints going from A to B, then it would be:

```
restraint-lambdas      = 0.0 0.1 0.2 0.4 0.6 1.0
```

and all of the other components of the λ vector would be left in the A state.

To compute free energies with a vector λ using thermodynamic integration, then the TI equation becomes vector equation:

$$\Delta F = \int \langle \nabla H \rangle \cdot d\vec{\lambda} \tag{6.1}$$

or for finite differences:

$$\Delta F \approx \int \sum \langle \nabla H \rangle \cdot \Delta\lambda \tag{6.2}$$

The external `pymbar` script downloaded from https://SimTK.org/home/pymbar can compute this integral automatically from the GROMACS dhdl.xvg output.

6.2 Potential of mean force

A potential of mean force (PMF) is a potential that is obtained by integrating the mean force from an ensemble of configurations. In GROMACS, there are several different methods to calculate the mean force. Each method has its limitations, which are listed below.

- **pull code:** between the centers of mass of molecules or groups of molecules.
- **free-energy code with harmonic bonds or constraints:** between single atoms.
- **free-energy code with position restraints:** changing the conformation of a relatively immobile group of atoms.

- **pull code in limited cases:** between groups of atoms that are part of a larger molecule for which the bonds are constrained with SHAKE or LINCS. If the pull group if relatively large, the pull code can be used.

The pull and free-energy code a described in more detail in the following two sections.

Entropic effects

When a distance between two atoms or the centers of mass of two groups is constrained or restrained, there will be a purely entropic contribution to the PMF due to the rotation of the two groups [131]. For a system of two non-interacting masses the potential of mean force is:

$$V_{pmf}(r) = -(n_c - 1)k_B T \log(r) \tag{6.3}$$

where n_c is the number of dimensions in which the constraint works (i.e. $n_c = 3$ for a normal constraint and $n_c = 1$ when only the z-direction is constrained). Whether one needs to correct for this contribution depends on what the PMF should represent. When one wants to pull a substrate into a protein, this entropic term indeed contributes to the work to get the substrate into the protein. But when calculating a PMF between two solutes in a solvent, for the purpose of simulating without solvent, the entropic contribution should be removed. **Note** that this term can be significant; when at 300K the distance is halved, the contribution is 3.5 kJ mol^{-1}.

6.3 Non-equilibrium pulling

When the distance between two groups is changed continuously, work is applied to the system, which means that the system is no longer in equilibrium. Although in the limit of very slow pulling the system is again in equilibrium, for many systems this limit is not reachable within reasonable computational time. However, one can use the Jarzynski relation [132] to obtain the equilibrium free-energy difference ΔG between two distances from many non-equilibrium simulations:

$$\Delta G_{AB} = -k_B T \log \left\langle e^{-\beta W_{AB}} \right\rangle_A \tag{6.4}$$

where W_{AB} is the work performed to force the system along one path from state A to B, the angular bracket denotes averaging over a canonical ensemble of the initial state A and $\beta = 1/k_B T$.

6.4 The pull code

The pull code applies forces or constraints between the centers of mass of one or more pairs of groups of atoms. Each pull reaction coordinate is called a “coordinate” and it operates on two pull groups. A pull group can be part of one or more pull coordinates. Furthermore, a coordinate can also operate on a single group and an absolute reference position in space. The distance between a pair of groups can be determined in 1, 2 or 3 dimensions, or can be along a user-defined vector. The reference distance can be constant or can change linearly with time. Normally all atoms are weighted by their mass, but an additional weighting factor can also be used.

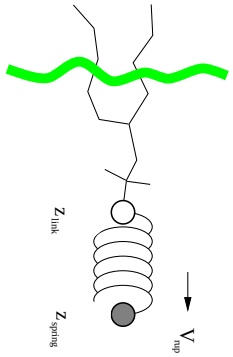


Figure 6.1: Schematic picture of pulling a lipid out of a lipid bilayer with umbrella pulling. V_{rup} is the velocity at which the spring is retracted, Z_{link} is the atom to which the spring is attached and Z_{spring} is the location of the spring.

Three different types of calculation are supported, and in all cases the reference distance can be constant or linearly changing with time.

1. **Umbrella pulling** A harmonic potential is applied between the centers of mass of two groups. Thus, the force is proportional to the displacement.

2. **Constraint pulling** The distance between the centers of mass of two groups is constrained. The constraint force can be written to a file. This method uses the SHAKE algorithm but only needs 1 iteration to be exact if only two groups are constrained.

3. **Constant force pulling** A constant force is applied between the centers of mass of two groups. Thus, the potential is linear. In this case there is no reference distance of pull rate.

Definition of the center of mass

In GROMACS, there are three ways to define the center of mass of a group. The standard way is a "plain" center of mass, possibly with additional weighting factors. With periodic boundary conditions it is no longer possible to uniquely define the center of mass of a group of atoms. Therefore, a reference atom is used. For determining the center of mass, for all other atoms in the group, the closest periodic image to the reference atom is used. This uniquely defines the center of mass. By default, the middle (determined by the order in the topology) atom is used as a reference atom, but the user can also select any other atom if it would be closer to center of the group.

For a layered system, for instance a lipid bilayer, 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. The `.mdp` option `pull_geometry = cylinder` does not use all the atoms of the reference group, but instead dynamically only those within a cylinder with radius `r_1` around the pull vector going through the pull group. This only works for distances defined in

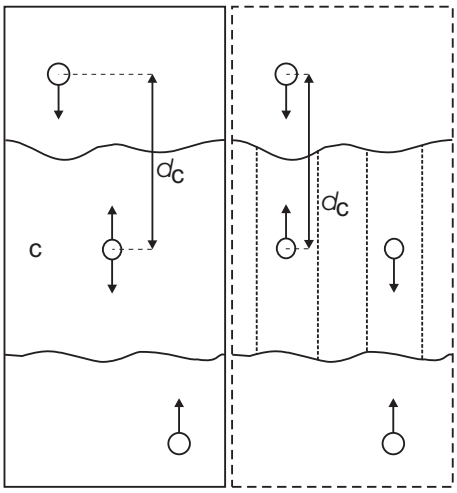


Figure 6.2: Comparison of a plain center of mass reference group versus a cylinder reference group applied to interface systems. C is the reference group. The circles represent the center of mass of two groups plus the reference group, d_c is the reference distance.

one dimension, and the cylinder is oriented with its long axis along this one dimension. A second cylinder can be defined with `r_0`, with a linear switch function that weighs the contribution of atoms between `r_0` and `r_1` with distance. This smooths the effects of atoms moving in and out of the cylinder (which causes jumps in the pull forces).

For a group of molecules in a periodic system, a plain reference group might not be well-defined. An example is a water slab that is connected periodically in x and y, but has two liquid-vapor interfaces along z. In such a setup, water molecules can evaporate from the liquid and they will move through the vapor, through the periodic boundary, to the other interface. Such a system is inherently periodic and there is no proper way of defining a "plain" center of mass along z. A proper solution is to using a cosine shaped weighting profile for all atoms in the reference group. The profile is a cosine with a single period in the unit cell. Its phase is optimized to give the maximum sum of weights, including mass weighting. This provides a unique and continuous reference position that is nearly identical to the plain center of mass position in case all atoms are all within a half of the unit-cell length. See ref [133] for details.

When relative weights w_i are used during the calculations, either by supplying weights in the input or due to cylinder geometry or due to cosine weighting, the weights need to be scaled to conserve momentum:

$$w_i' = w_i \sum_{j=1}^{N} w_j \, m_j \Big/ \sum_{j=1}^{N} w_j^2 \, m_j \tag{6.5}$$

where m_j is the mass of atom j of the group. The mass of the group, required for calculating the constraint force, is:

$$M = \sum_{i=1}^{N} w_i' \, m_i \tag{6.6}$$

The definition of the weighted center of mass is:

$$\boldsymbol{r}_{com} = \sum_{i=1}^{N} w_i' \, m_i \, \boldsymbol{r}_i \Big/ M \tag{6.7}$$

From the centers of mass the AFM, constraint, or umbrella force $\boldsymbol{F}_{com}$ on each group can be calculated. The force on the center of mass of a group is redistributed to the atoms as follows:

$$\boldsymbol{F}_i = \frac{w_i' \, m_i}{M} \, \boldsymbol{F}_{com} \tag{6.8}$$

Limitations

There is one theoretical limitation: strictly speaking, constraint forces can only be calculated between groups that are not connected by constraints to the rest of the system. If a group contains part of a molecule of which the bond lengths are constrained, the pull constraint and LINCS or SHAKE bond constraint algorithms should be iterated simultaneously. This is not done in GROMACS. This means that for simulations with `constraints = all-bonds` in the `.mdp` file pulling is, strictly speaking, limited to whole molecules or groups of molecules. In some cases this limitation can be avoided by using the free energy code, see sec. 6.7. In practice, the errors caused by not iterating the two constraint algorithms can be negligible when the pull group consists of a large amount of atoms and/or the pull force is small. In such cases, the constraint correction displacement of the pull group is small compared to the bond lengths.

6.5 Enforced Rotation

This module can be used to enforce the rotation of a group of atoms, as *e.g.* a protein subunit. There are a variety of rotation potentials, among them complex ones that allow flexible adaptations of both the rotated subunit as well as the local rotation axis during the simulation. An example application can be found in ref. [134].

6.5.1 Fixed Axis Rotation

Stationary Axis with an Isotropic Potential

In the fixed axis approach (see Fig. 6.3B), torque on a group of N atoms with positions $\boldsymbol{x}_i$ (denoted "rotation group") is applied by rotating a reference set of atomic positions – usually their initial positions $\boldsymbol{y}_i^0$ – at a constant angular velocity ω around an axis defined by a direction vector $\hat{\boldsymbol{v}}$ and a pivot point $\boldsymbol{u}$. To that aim, each atom with position $\boldsymbol{x}_i$ is attracted by a "virtual spring" potential to its moving reference position $\boldsymbol{y}_i = \boldsymbol{\Omega}(t)(\boldsymbol{y}_i^0 - \boldsymbol{u})$, where $\boldsymbol{\Omega}(t)$ is a matrix that describes the rotation around the axis. In the simplest case, the "springs" are described by a harmonic potential,

$$V^{\text{iso}} = \frac{k}{2} \sum_{i=1}^{N} w_i \left[\boldsymbol{\Omega}(t)(\boldsymbol{y}_i^0 - \boldsymbol{u}) - (\boldsymbol{x}_i - \boldsymbol{u}) \right]^2, \tag{6.9}$$

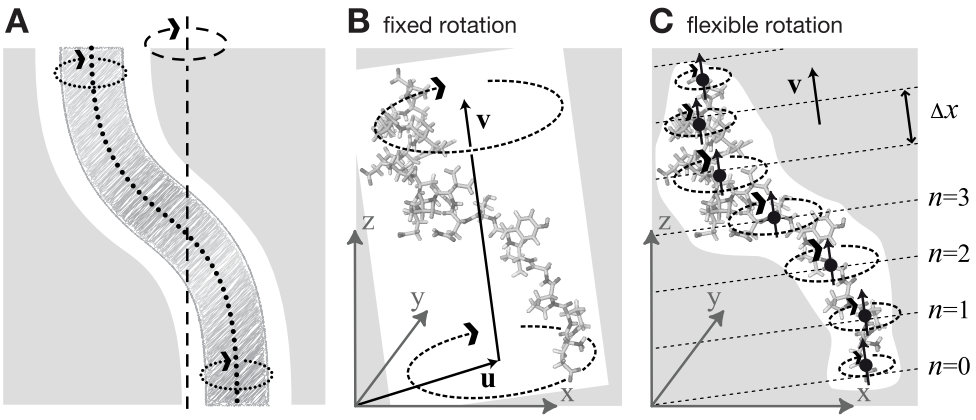


Figure 6.3: Comparison of fixed and flexible axis rotation. A: Rotating the sketched shape inside the white tubular cavity can create artifacts when a fixed rotation axis (dashed) is used. More realistically, the shape would revolve like a flexible pipe-cleaner (dotted) inside the bearing (gray). B: Fixed rotation around an axis $\boldsymbol{v}$ with a pivot point specified by the vector $\boldsymbol{u}$. C: Subdividing the rotating fragment into slabs with separate rotation axes (↑) and pivot points (•) for each slab allows for flexibility. The distance between two slabs with indices n and $n+1$ is Δx.

with optional mass-weighted prefactors $w_i = N\,m_i/M$ with total mass $M = \sum_{i=1}^{N} m_i$. The rotation matrix $\boldsymbol{\Omega}(t)$ is

$$\boldsymbol{\Omega}(t) = \begin{pmatrix} \cos\omega t + v_x^2\,\xi & v_x v_y\,\xi - v_z \sin\omega t & v_x v_z\,\xi + v_y \sin\omega t \\ v_x v_y\,\xi + v_z \sin\omega t & \cos\omega t + v_y^2\,\xi & v_y v_z\,\xi - v_x \sin\omega t \\ v_x v_z\,\xi - v_y \sin\omega t & v_y v_z\,\xi + v_x \sin\omega t & \cos\omega t + v_z^2\,\xi \end{pmatrix},$$

where v_x, v_y, and v_z are the components of the normalized rotation vector $\hat{\boldsymbol{v}}$, and $\xi := 1-\cos(\omega t)$. As illustrated in Fig. 6.4A for a single atom j, the rotation matrix $\boldsymbol{\Omega}(t)$ operates on the initial reference positions $\boldsymbol{y}_j^0 = \boldsymbol{x}_j(t_0)$ of atom j at $t = t_0$. At a later time t, the reference position has rotated away from its initial place (along the blue dashed line), resulting in the force

$$\boldsymbol{F}_j^{\text{iso}} = -\nabla_j\, V^{\text{iso}} = k\, w_j \left[\boldsymbol{\Omega}(t)(\boldsymbol{y}_j^0 - \boldsymbol{u}) - (\boldsymbol{x}_j - \boldsymbol{u})\right], \tag{6.10}$$

which is directed towards the reference position.

Pivot-Free Isotropic Potential

Instead of a fixed pivot vector $\boldsymbol{u}$ this potential uses the center of mass $\boldsymbol{x}_c$ of the rotation group as pivot for the rotation axis,

$$\boldsymbol{x}_c = \frac{1}{M}\sum_{i=1}^{N} m_i \boldsymbol{x}_i \qquad \text{and} \qquad \boldsymbol{y}_c^0 = \frac{1}{M}\sum_{i=1}^{N} m_i \boldsymbol{y}_i^0\,, \tag{6.11}$$

which yields the "pivot-free" isotropic potential

$$V^{\text{iso-pf}} = \frac{k}{2}\sum_{i=1}^{N} w_i \left[\boldsymbol{\Omega}(t)(\boldsymbol{y}_i^0 - \boldsymbol{y}_c^0) - (\boldsymbol{x}_i - \boldsymbol{x}_c)\right]^2, \tag{6.12}$$

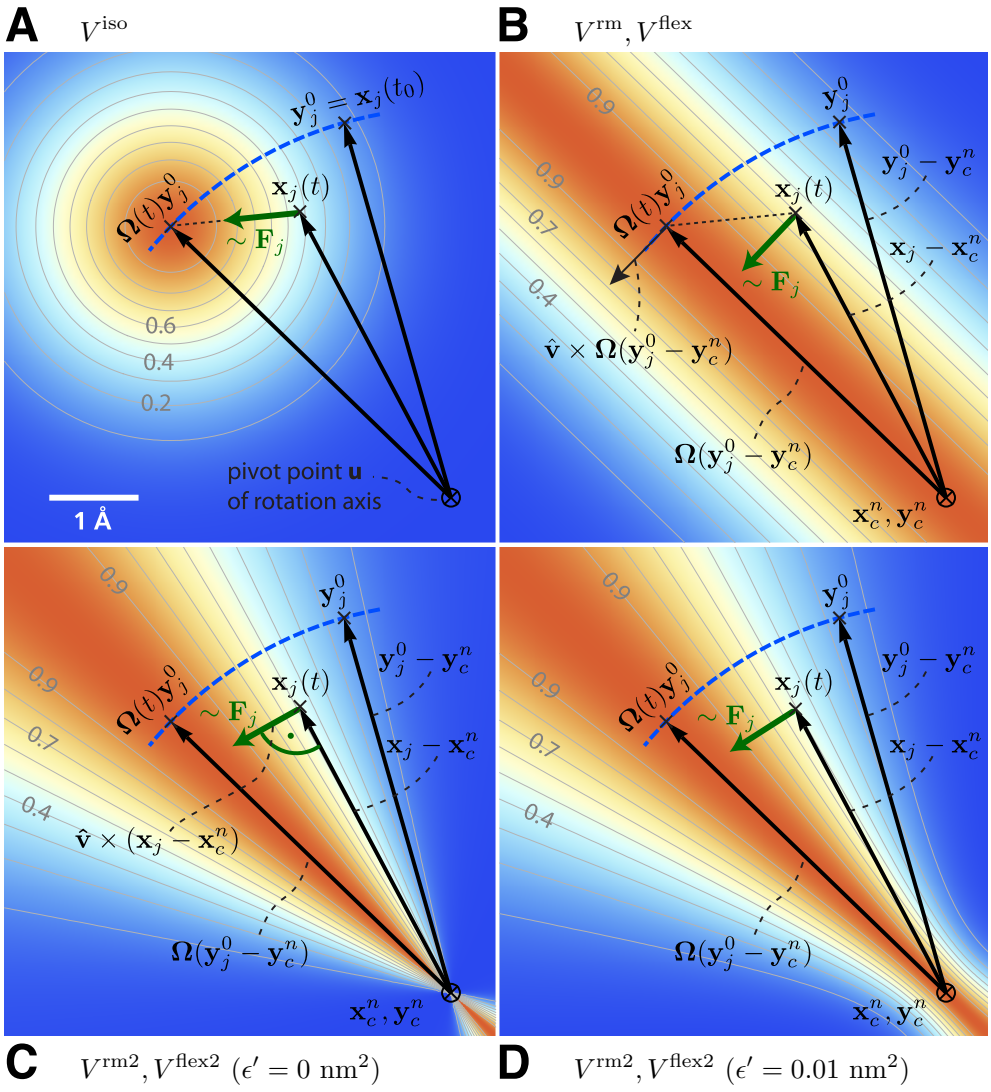


Figure 6.4: Selection of different rotation potentials and definition of notation. All four potentials V (color coded) are shown for a single atom at position $\boldsymbol{x}_j(t)$. A: Isotropic potential V^{iso}, B: radial motion potential V^{rm} and flexible potential V^{flex}, C–D: radial motion 2 potential V^{rm2} and flexible 2 potential V^{flex2} for $\epsilon' = 0\,\mathrm{nm}^2$ (C) and $\epsilon' = 0.01\,\mathrm{nm}^2$ (D). The rotation axis is perpendicular to the plane and marked by $\otimes$. The light gray contours indicate Boltzmann factors $e^{-V/(k_B T)}$ in the $\boldsymbol{x}_j$-plane for $T = 300\,\mathrm{K}$ and $k = 200\,\mathrm{kJ/(mol{\cdot}nm^2)}$. The green arrow shows the direction of the force $\boldsymbol{F}_j$ acting on atom j; the blue dashed line indicates the motion of the reference position.

with forces

$$\mathbf{F}_j^{\text{iso-pf}} = k\, w_j \left[\boldsymbol{\Omega}(t)(\boldsymbol{y}_j^0 - \boldsymbol{y}_c^0) - (\boldsymbol{x}_j - \boldsymbol{x}_c)\right]. \tag{6.13}$$

Without mass-weighting, the pivot $\boldsymbol{x}_c$ is the geometrical center of the group.

Parallel Motion Potential Variant

The forces generated by the isotropic potentials (eqns. 6.9 and 6.12) also contain components parallel to the rotation axis and thereby restrain motions along the axis of either the whole rotation group (in case of V^{iso}) or within the rotation group (in case of $V^{\text{iso-pf}}$). For cases where unrestrained motion along the axis is preferred, we have implemented a "parallel motion" variant by eliminating all components parallel to the rotation axis for the potential. This is achieved by projecting the distance vectors between reference and actual positions

$$\boldsymbol{r}_i = \boldsymbol{\Omega}(t)(\boldsymbol{y}_i^0 - \boldsymbol{u}) - (\boldsymbol{x}_i - \boldsymbol{u}) \tag{6.14}$$

onto the plane perpendicular to the rotation vector,

$$\boldsymbol{r}_i^{\perp} := \boldsymbol{r}_i - (\boldsymbol{r}_i \cdot \hat{\boldsymbol{v}})\hat{\boldsymbol{v}}\,, \tag{6.15}$$

yielding

$$\begin{aligned} V^{\text{pm}} &= \frac{k}{2}\sum_{i=1}^{N} w_i (\boldsymbol{r}_i^{\perp})^2 \\ &= \frac{k}{2}\sum_{i=1}^{N} w_i \Big\{\boldsymbol{\Omega}(t)(\boldsymbol{y}_i^0 - \boldsymbol{u}) - (\boldsymbol{x}_i - \boldsymbol{u}) \\ &\quad - \Big\{\Big[\boldsymbol{\Omega}(t)(\boldsymbol{y}_i^0 - \boldsymbol{u}) - (\boldsymbol{x}_i - \boldsymbol{u})\Big] \cdot \hat{\boldsymbol{v}}\Big\}\,\hat{\boldsymbol{v}}\Big\}^2, \end{aligned} \tag{6.16}$$

and similarly

$$\boldsymbol{F}_j^{\text{pm}} = k\, w_j\, \boldsymbol{r}_j^{\perp}. \tag{6.17}$$

Pivot-Free Parallel Motion Potential

Replacing in eqn. 6.16 the fixed pivot $\boldsymbol{u}$ by the center of mass $\boldsymbol{x_c}$ yields the pivot-free variant of the parallel motion potential. With

$$\boldsymbol{s}_i = \boldsymbol{\Omega}(t)(\boldsymbol{y}_i^0 - \boldsymbol{y}_c^0) - (\boldsymbol{x}_i - \boldsymbol{x}_c) \tag{6.18}$$

the respective potential and forces are

$$V^{\text{pm-pf}} = \frac{k}{2}\sum_{i=1}^{N} w_i (\boldsymbol{s}_i^{\perp})^2\,, \tag{6.19}$$

$$\boldsymbol{F}_j^{\text{pm-pf}} = k\, w_j\, \boldsymbol{s}_j^{\perp}. \tag{6.20}$$

Radial Motion Potential

In the above variants, the minimum of the rotation potential is either a single point at the reference position $\boldsymbol{y}_i$ (for the isotropic potentials) or a single line through $\boldsymbol{y}_i$ parallel to the rotation axis (for the parallel motion potentials). As a result, radial forces restrict radial motions of the atoms. The two subsequent types of rotation potentials, V^{rm} and V^{rm2}, drastically reduce or even eliminate this effect. The first variant, V^{rm} (Fig. 6.4B), eliminates all force components parallel to the vector connecting the reference atom and the rotation axis,

$$V^{\mathrm{rm}} = \frac{k}{2} \sum_{i=1}^{N} w_i \left[\boldsymbol{p}_i \cdot (\boldsymbol{x}_i - \boldsymbol{u})\right]^2 , \tag{6.21}$$

with

$$\boldsymbol{p}_i := \frac{\hat{\boldsymbol{v}} \times \boldsymbol{\Omega}(t)(\boldsymbol{y}_i^0 - \boldsymbol{u})}{\|\hat{\boldsymbol{v}} \times \boldsymbol{\Omega}(t)(\boldsymbol{y}_i^0 - \boldsymbol{u})\|} . \tag{6.22}$$

This variant depends only on the distance $\boldsymbol{p}_i \cdot (\boldsymbol{x}_i - \boldsymbol{u})$ of atom i from the plane spanned by $\hat{\boldsymbol{v}}$ and $\boldsymbol{\Omega}(t)(\boldsymbol{y}_i^0 - \boldsymbol{u})$. The resulting force is

$$\mathbf{F}_j^{\mathrm{rm}} = -k\, w_j \left[\boldsymbol{p}_j \cdot (\boldsymbol{x}_j - \boldsymbol{u})\right] \boldsymbol{p}_j . \tag{6.23}$$

Pivot-Free Radial Motion Potential

Proceeding similar to the pivot-free isotropic potential yields a pivot-free version of the above potential. With

$$\boldsymbol{q}_i := \frac{\hat{\boldsymbol{v}} \times \boldsymbol{\Omega}(t)(\boldsymbol{y}_i^0 - \boldsymbol{y}_c^0)}{\|\hat{\boldsymbol{v}} \times \boldsymbol{\Omega}(t)(\boldsymbol{y}_i^0 - \boldsymbol{y}_c^0)\|} , \tag{6.24}$$

the potential and force for the pivot-free variant of the radial motion potential read

$$V^{\mathrm{rm\text{-}pf}} = \frac{k}{2} \sum_{i=1}^{N} w_i \left[\boldsymbol{q}_i \cdot (\boldsymbol{x}_i - \boldsymbol{x}_c)\right]^2 , \tag{6.25}$$

$$\mathbf{F}_j^{\mathrm{rm\text{-}pf}} = -k\, w_j \left[\boldsymbol{q}_j \cdot (\boldsymbol{x}_j - \boldsymbol{x}_c)\right] \boldsymbol{q}_j + k \frac{m_j}{M} \sum_{i=1}^{N} w_i \left[\boldsymbol{q}_i \cdot (\boldsymbol{x}_i - \boldsymbol{x}_c)\right] \boldsymbol{q}_i . \tag{6.26}$$

Radial Motion 2 Alternative Potential

As seen in Fig. 6.4B, the force resulting from V^{rm} still contains a small, second-order radial component. In most cases, this perturbation is tolerable; if not, the following alternative, V^{rm2}, fully eliminates the radial contribution to the force, as depicted in Fig. 6.4C,

$$V^{\mathrm{rm2}} = \frac{k}{2} \sum_{i=1}^{N} w_i \frac{\left[(\hat{\boldsymbol{v}} \times (\boldsymbol{x}_i - \boldsymbol{u})) \cdot \boldsymbol{\Omega}(t)(\boldsymbol{y}_i^0 - \boldsymbol{u})\right]^2}{\|\hat{\boldsymbol{v}} \times (\boldsymbol{x}_i - \boldsymbol{u})\|^2 + \epsilon'} , \tag{6.27}$$

where a small parameter ϵ' has been introduced to avoid singularities. For $\epsilon' = 0\,\mathrm{nm}^2$, the equipotential planes are spanned by $\boldsymbol{x}_i - \boldsymbol{u}$ and $\hat{\boldsymbol{v}}$, yielding a force perpendicular to $\boldsymbol{x}_i - \boldsymbol{u}$, thus not contracting or expanding structural parts that moved away from or toward the rotation axis.

Choosing a small positive ϵ' (*e.g.*, $\epsilon' = 0.01\,\mathrm{nm}^2$, Fig. 6.4D) in the denominator of eqn. 6.27 yields a well-defined potential and continuous forces also close to the rotation axis, which is not the case for $\epsilon' = 0\,\mathrm{nm}^2$ (Fig. 6.4C). With

$$\begin{aligned} \boldsymbol{r}_i &:= \boldsymbol{\Omega}(t)(\boldsymbol{y}_i^0 - \boldsymbol{u}) && (6.28)\\ \boldsymbol{s}_i &:= \frac{\hat{\boldsymbol{v}} \times (\boldsymbol{x}_i - \boldsymbol{u})}{\|\hat{\boldsymbol{v}} \times (\boldsymbol{x}_i - \boldsymbol{u})\|} \equiv \Psi_i \;\; \hat{\boldsymbol{v}} \times (\boldsymbol{x}_i - \boldsymbol{u}) && (6.29)\\ \Psi_i^* &:= \frac{1}{\|\hat{\boldsymbol{v}} \times (\boldsymbol{x}_i - \boldsymbol{u})\|^2 + \epsilon'} && (6.30) \end{aligned}$$

the force on atom j reads

$$\boldsymbol{F}_j^{\mathrm{rm2}} = -k \left\{ w_j \; (\boldsymbol{s}_j \cdot \boldsymbol{r}_j) \; \left[\frac{\Psi_j^*}{\Psi_j} \boldsymbol{r}_j - \frac{\Psi_j^{*2}}{\Psi_j^3} (\boldsymbol{s}_j \cdot \boldsymbol{r}_j) \boldsymbol{s}_j \right] \right\} \times \hat{\boldsymbol{v}}. \tag{6.31}$$

Pivot-Free Radial Motion 2 Potential

The pivot-free variant of the above potential is

$$V^{\mathrm{rm2\text{-}pf}} = \frac{k}{2} \sum_{i=1}^{N} w_i \, \frac{\left[(\hat{\boldsymbol{v}} \times (\boldsymbol{x}_i - \boldsymbol{x}_c)) \cdot \boldsymbol{\Omega}(t)(\boldsymbol{y}_i^0 - \boldsymbol{y}_c) \right]^2}{\|\hat{\boldsymbol{v}} \times (\boldsymbol{x}_i - \boldsymbol{x}_c)\|^2 + \epsilon'} \,. \tag{6.32}$$

With

$$\begin{aligned} \boldsymbol{r}_i &:= \boldsymbol{\Omega}(t)(\boldsymbol{y}_i^0 - \boldsymbol{y}_c) && (6.33)\\ \boldsymbol{s}_i &:= \frac{\hat{\boldsymbol{v}} \times (\boldsymbol{x}_i - \boldsymbol{x}_c)}{\|\hat{\boldsymbol{v}} \times (\boldsymbol{x}_i - \boldsymbol{x}_c)\|} \equiv \Psi_i \;\; \hat{\boldsymbol{v}} \times (\boldsymbol{x}_i - \boldsymbol{x}_c) && (6.34)\\ \Psi_i^* &:= \frac{1}{\|\hat{\boldsymbol{v}} \times (\boldsymbol{x}_i - \boldsymbol{x}_c)\|^2 + \epsilon'} && (6.35) \end{aligned}$$

the force on atom j reads

$$\begin{aligned} \boldsymbol{F}_j^{\mathrm{rm2\text{-}pf}} \;=\; & -k \left\{ w_j \; (\boldsymbol{s}_j \cdot \boldsymbol{r}_j) \; \left[\frac{\Psi_j^*}{\Psi_j} \boldsymbol{r}_j - \frac{\Psi_j^{*2}}{\Psi_j^3} (\boldsymbol{s}_j \cdot \boldsymbol{r}_j) \boldsymbol{s}_j \right] \right\} \times \hat{\boldsymbol{v}} \\ & + k \, \frac{m_j}{M} \left\{ \sum_{i=1}^{N} w_i \; (\boldsymbol{s}_i \cdot \boldsymbol{r}_i) \; \left[\frac{\Psi_i^*}{\Psi_i} \boldsymbol{r}_i - \frac{\Psi_i^{*2}}{\Psi_i^3} (\boldsymbol{s}_i \cdot \boldsymbol{r}_i) \; \boldsymbol{s}_i \right] \right\} \times \hat{\boldsymbol{v}} \,. \end{aligned} \tag{6.36}$$

6.5.2 Flexible Axis Rotation

As sketched in Fig. 6.3A–B, the rigid body behavior of the fixed axis rotation scheme is a drawback for many applications. In particular, deformations of the rotation group are suppressed when the equilibrium atom positions directly depend on the reference positions. To avoid this limitation, eqns. 6.26 and 6.32 will now be generalized towards a "flexible axis" as sketched in Fig. 6.3C. This will be achieved by subdividing the rotation group into a set of equidistant slabs perpendicular to the rotation vector, and by applying a separate rotation potential to each of these slabs. Fig. 6.3C shows the midplanes of the slabs as dotted straight lines and the centers as thick black dots.

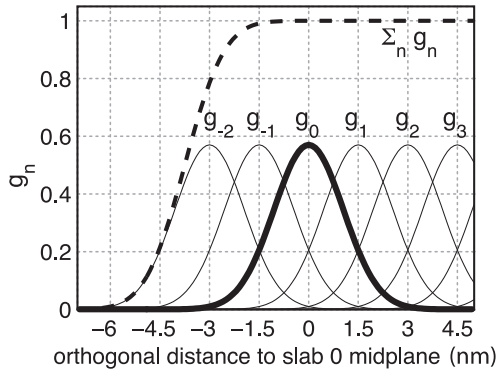


Figure 6.5: Gaussian functions g_n centered at $n\,\Delta x$ for a slab distance $\Delta x = 1.5$ nm and $n \geq -2$. Gaussian function g_0 is highlighted in bold; the dashed line depicts the sum of the shown Gaussian functions.

To avoid discontinuities in the potential and in the forces, we define "soft slabs" by weighing the contributions of each slab n to the total potential function V^{flex} by a Gaussian function

$$g_n(\boldsymbol{x}_i) = \Gamma \exp\left(-\frac{\beta_n^2(\boldsymbol{x}_i)}{2\sigma^2}\right), \tag{6.37}$$

centered at the midplane of the nth slab. Here σ is the width of the Gaussian function, Δx the distance between adjacent slabs, and

$$\beta_n(\boldsymbol{x}_i) := \boldsymbol{x}_i \cdot \hat{\boldsymbol{v}} - n\,\Delta x\,. \tag{6.38}$$

A most convenient choice is $\sigma = 0.7\Delta x$ and

$$1/\Gamma = \sum_{n\in Z} \exp\left(-\frac{(n-\frac{1}{4})^2}{2\cdot 0.7^2}\right) \approx 1.75464\,,$$

which yields a nearly constant sum, essentially independent of $\boldsymbol{x}_i$ (dashed line in Fig. 6.5), *i.e.*,

$$\sum_{n\in Z} g_n(\boldsymbol{x}_i) = 1 + \epsilon(\boldsymbol{x}_i)\,, \tag{6.39}$$

with $|\epsilon(\boldsymbol{x}_i)| < 1.3\cdot 10^{-4}$. This choice also implies that the individual contributions to the force from the slabs add up to unity such that no further normalization is required.

To each slab center $\boldsymbol{x}_c^n$, all atoms contribute by their Gaussian-weighted (optionally also mass-weighted) position vectors $g_n(\boldsymbol{x}_i)\,\boldsymbol{x}_i$. The instantaneous slab centers $\boldsymbol{x}_c^n$ are calculated from the current positions $\boldsymbol{x}_i$,

$$\boldsymbol{x}_c^n = \frac{\sum_{i=1}^N g_n(\boldsymbol{x}_i)\,m_i\,\boldsymbol{x}_i}{\sum_{i=1}^N g_n(\boldsymbol{x}_i)\,m_i}\,, \tag{6.40}$$

while the reference centers $\boldsymbol{y}_c^n$ are calculated from the reference positions $\boldsymbol{y}_i^0$,

$$\boldsymbol{y}_c^n = \frac{\sum_{i=1}^N g_n(\boldsymbol{y}_i^0)\,m_i\,\boldsymbol{y}_i^0}{\sum_{i=1}^N g_n(\boldsymbol{y}_i^0)\,m_i}\,. \tag{6.41}$$

Due to the rapid decay of g_n, each slab will essentially involve contributions from atoms located within $\approx 3\Delta x$ from the slab center only.

Flexible Axis Potential

We consider two flexible axis variants. For the first variant, the slab segmentation procedure with Gaussian weighting is applied to the radial motion potential (eqn. 6.26 / Fig. 6.4B), yielding as the contribution of slab n

$$V^n = \frac{k}{2} \sum_{i=1}^{N} w_i \, g_n(\boldsymbol{x}_i) \, [\boldsymbol{q}_i^n \cdot (\boldsymbol{x}_i - \boldsymbol{x}_c^n)]^2 \,,$$

and a total potential function

$$V^{\text{flex}} = \sum_n V^n \,. \tag{6.42}$$

Note that the global center of mass $\boldsymbol{x}_c$ used in eqn. 6.26 is now replaced by $\boldsymbol{x}_c^n$, the center of mass of the slab. With

$$\boldsymbol{q}_i^n := \frac{\hat{\boldsymbol{v}} \times \boldsymbol{\Omega}(t)(\boldsymbol{y}_i^0 - \boldsymbol{y}_c^n)}{\|\hat{\boldsymbol{v}} \times \boldsymbol{\Omega}(t)(\boldsymbol{y}_i^0 - \boldsymbol{y}_c^n)\|} \tag{6.43}$$

$$b_i^n := \boldsymbol{q}_i^n \cdot (\boldsymbol{x}_i - \boldsymbol{x}_c^n) \,, \tag{6.44}$$

the resulting force on atom j reads

$$\begin{aligned} \boldsymbol{F}_j^{\text{flex}} = & - k \, w_j \sum_n g_n(\boldsymbol{x}_j) \, b_j^n \left\{ \boldsymbol{q}_j^n - b_j^n \frac{\beta_n(\boldsymbol{x}_j)}{2\sigma^2} \hat{\boldsymbol{v}} \right\} \\ & + k \, m_j \sum_n \frac{g_n(\boldsymbol{x}_j)}{\sum_h g_n(\boldsymbol{x}_h)} \sum_{i=1}^{N} w_i \, g_n(\boldsymbol{x}_i) \, b_i^n \left\{ \boldsymbol{q}_i^n - \frac{\beta_n(\boldsymbol{x}_j)}{\sigma^2} \, [\boldsymbol{q}_i^n \cdot (\boldsymbol{x}_j - \boldsymbol{x}_c^n)] \, \hat{\boldsymbol{v}} \right\} . \end{aligned} \tag{6.45}$$

Note that for V^{flex}, as defined, the slabs are fixed in space and so are the reference centers $\boldsymbol{y}_c^n$. If during the simulation the rotation group moves too far in $\boldsymbol{v}$ direction, it may enter a region where – due to the lack of nearby reference positions – no reference slab centers are defined, rendering the potential evaluation impossible. We therefore have included a slightly modified version of this potential that avoids this problem by attaching the midplane of slab $n = 0$ to the center of mass of the rotation group, yielding slabs that move with the rotation group. This is achieved by subtracting the center of mass $\boldsymbol{x}_c$ of the group from the positions,

$$\tilde{\boldsymbol{x}}_i = \boldsymbol{x}_i - \boldsymbol{x}_c \,, \quad \text{and} \quad \tilde{\boldsymbol{y}}_i^0 = \boldsymbol{y}_i^0 - \boldsymbol{y}_c^0 \,, \tag{6.46}$$

such that

$$V^{\text{flex-t}} = \frac{k}{2} \sum_n \sum_{i=1}^{N} w_i \, g_n(\tilde{\boldsymbol{x}}_i) \left[\frac{\hat{\boldsymbol{v}} \times \boldsymbol{\Omega}(t)(\tilde{\boldsymbol{y}}_i^0 - \tilde{\boldsymbol{y}}_c^n)}{\|\hat{\boldsymbol{v}} \times \boldsymbol{\Omega}(t)(\tilde{\boldsymbol{y}}_i^0 - \tilde{\boldsymbol{y}}_c^n)\|} \cdot (\tilde{\boldsymbol{x}}_i - \tilde{\boldsymbol{x}}_c^n) \right]^2 . \tag{6.47}$$

To simplify the force derivation, and for efficiency reasons, we here assume $\boldsymbol{x}_c$ to be constant, and thus $\partial \boldsymbol{x}_c / \partial x = \partial \boldsymbol{x}_c / \partial y = \partial \boldsymbol{x}_c / \partial z = 0$. The resulting force error is small (of order $O(1/N)$ or $O(m_j/M)$ if mass-weighting is applied) and can therefore be tolerated. With this assumption, the forces $\boldsymbol{F}^{\text{flex-t}}$ have the same form as eqn. 6.45.

Flexible Axis 2 Alternative Potential

In this second variant, slab segmentation is applied to V^{rm2} (eqn. 6.32), resulting in a flexible axis potential without radial force contributions (Fig. 6.4C),

$$V^{\mathrm{flex2}} = \frac{k}{2}\sum_{i=1}^{N}\sum_{n} w_i\, g_n(\boldsymbol{x}_i)\frac{\left[(\hat{\boldsymbol{v}}\times(\boldsymbol{x}_i-\boldsymbol{x}_c^n))\cdot\boldsymbol{\Omega}(t)(\boldsymbol{y}_i^0-\boldsymbol{y}_c^n)\right]^2}{\|\hat{\boldsymbol{v}}\times(\boldsymbol{x}_i-\boldsymbol{x}_c^n)\|^2+\epsilon'}\,. \tag{6.48}$$

With

$$\boldsymbol{r}_i^n := \boldsymbol{\Omega}(t)(\boldsymbol{y}_i^0-\boldsymbol{y}_c^n) \tag{6.49}$$

$$\boldsymbol{s}_i^n := \frac{\hat{\boldsymbol{v}}\times(\boldsymbol{x}_i-\boldsymbol{x}_c^n)}{\|\hat{\boldsymbol{v}}\times(\boldsymbol{x}_i-\boldsymbol{x}_c^n)\|} \equiv \psi_i\ \ \hat{\boldsymbol{v}}\times(\boldsymbol{x}_i-\boldsymbol{x}_c^n) \tag{6.50}$$

$$\psi_i^* := \frac{1}{\|\hat{\boldsymbol{v}}\times(\boldsymbol{x}_i-\boldsymbol{x}_c^n)\|^2+\epsilon'} \tag{6.51}$$

$$W_j^n := \frac{g_n(\boldsymbol{x}_j)\,m_j}{\sum_h g_n(\boldsymbol{x}_h)\,m_h} \tag{6.52}$$

$$\boldsymbol{S}^n := \sum_{i=1}^{N} w_i\, g_n(\boldsymbol{x}_i)\,(\boldsymbol{s}_i^n\cdot\boldsymbol{r}_i^n)\left[\frac{\psi_i^*}{\psi_i}\boldsymbol{r}_i^n-\frac{\psi_i^{*2}}{\psi_i^3}(\boldsymbol{s}_i^n\cdot\boldsymbol{r}_i^n)\,\boldsymbol{s}_i^n\right] \tag{6.53}$$

the force on atom j reads

$$\begin{aligned}\boldsymbol{F}_j^{\mathrm{flex2}} = {} & -k\left\{\sum_n w_j\, g_n(\boldsymbol{x}_j)\,(\boldsymbol{s}_j^n\cdot\boldsymbol{r}_j^n)\left[\frac{\psi_j^*}{\psi_j}\boldsymbol{r}_j^n-\frac{\psi_j^{*2}}{\psi_j^3}(\boldsymbol{s}_j^n\cdot\boldsymbol{r}_j^n)\,\boldsymbol{s}_j^n\right]\right\}\times\hat{\boldsymbol{v}}\\ & +k\left\{\sum_n W_j^n\,\boldsymbol{S}^n\right\}\times\hat{\boldsymbol{v}}-k\left\{\sum_n W_j^n\,\frac{\beta_n(\boldsymbol{x}_j)}{\sigma^2}\frac{1}{\psi_j}\ \ \boldsymbol{s}_j^n\cdot\boldsymbol{S}^n\right\}\hat{\boldsymbol{v}}\\ & +\frac{k}{2}\left\{\sum_n w_j\, g_n(\boldsymbol{x}_j)\frac{\beta_n(\boldsymbol{x}_j)}{\sigma^2}\frac{\psi_j^*}{\psi_j^2}(\boldsymbol{s}_j^n\cdot\boldsymbol{r}_j^n)^2\right\}\hat{\boldsymbol{v}}.\end{aligned} \tag{6.54}$$

Applying transformation (6.46) yields a "translation-tolerant" version of the flexible 2 potential, $V^{\mathrm{flex2\text{-}t}}$. Again, assuming that $\partial\boldsymbol{x}_c/\partial x$, $\partial\boldsymbol{x}_c/\partial y$, $\partial\boldsymbol{x}_c/\partial z$ are small, the resulting equations for $V^{\mathrm{flex2\text{-}t}}$ and $\boldsymbol{F}^{\mathrm{flex2\text{-}t}}$ are similar to those of V^{flex2} and $\boldsymbol{F}^{\mathrm{flex2}}$.

6.5.3 Usage

To apply enforced rotation, the particles i that are to be subjected to one of the rotation potentials are defined via index groups `rot_group0`, `rot_group1`, etc., in the `.mdp` input file. The reference positions $\boldsymbol{y}_i^0$ are read from a special `.trr` file provided to `grompp`. If no such file is found, $\boldsymbol{x}_i(t=0)$ are used as reference positions and written to `.trr` such that they can be used for subsequent setups. All parameters of the potentials such as k, ϵ', etc. (Table 6.1) are provided as `.mdp` parameters; `rot_type` selects the type of the potential. The option `rot_massw` allows to choose whether or not to use mass-weighted averaging. For the flexible potentials, a cutoff value $g_n^{\min}$ (typically $g_n^{\min}=0.001$) makes shure that only significant contributions to V and $\boldsymbol{F}$ are evaluated, *i.e.* terms with $g_n(\boldsymbol{x})<g_n^{\min}$ are omitted. Table 6.2 summarizes observables that are written to additional output files and which are described below.

Table 6.1: Parameters used by the various rotation potentials. x's indicate which parameter is actually used for a given potential.

parameter			k	$\hat{\boldsymbol{v}}$	$\boldsymbol{u}$	ω	ϵ'	Δx	$g_n^{\min}$
.mdp input variable name			k	vec	pivot	rate	eps	slab_dist	min_gauss
unit			$\frac{\text{kJ}}{\text{mol}\cdot\text{nm}^2}$	-	nm	°/ps	nm^2	nm	-
fixed axis potentials:		eqn.							
isotropic	V^{iso}	(6.9)	x	x	x	x	-	-	-
— pivot-free	$V^{\text{iso-pf}}$	(6.12)	x	x	-	x	-	-	-
parallel motion	V^{pm}	(6.16)	x	x	x	x	-	-	-
— pivot-free	$V^{\text{pm-pf}}$	(6.20)	x	x	-	x	-	-	-
radial motion	V^{rm}	(6.21)	x	x	x	x	-	-	-
— pivot-free	$V^{\text{rm-pf}}$	(6.26)	x	x	-	x	-	-	-
radial motion 2	V^{rm2}	(6.27)	x	x	x	x	x	-	-
— pivot-free	$V^{\text{rm2-pf}}$	(6.32)	x	x	-	x	x	-	-
flexible axis potentials:		eqn.							
flexible	V^{flex}	(6.42)	x	x	-	x	-	x	x
— transl. tol.	$V^{\text{flex-t}}$	(6.47)	x	x	-	x	-	x	x
flexible 2	V^{flex2}	(6.48)	x	x	-	x	x	x	x
— transl. tol.	$V^{\text{flex2-t}}$	-	x	x	-	x	x	x	x

Table 6.2: Quantities recorded in output files during enforced rotation simulations. All slab-wise data is written every `nstsout` steps, other rotation data every `nstrout` steps.

quantity	unit	equation	output file	fixed	flexible
$V(t)$	kJ/mol	see 6.1	`rotation`	x	x
$\theta_{\text{ref}}(t)$	degrees	$\theta_{\text{ref}}(t) = \omega t$	`rotation`	x	x
$\theta_{\text{av}}(t)$	degrees	(6.55)	`rotation`	x	-
$\theta_{\text{fit}}(t)$, $\theta_{\text{fit}}(t,n)$	degrees	(6.57)	`rotangles`	-	x
$\boldsymbol{y}_0(n)$, $\boldsymbol{x}_0(t,n)$	nm	(6.40, 6.41)	`rotslabs`	-	x
$\tau(t)$	kJ/mol	(6.58)	`rotation`	x	-
$\tau(t,n)$	kJ/mol	(6.58)	`rottorque`	-	x

Angle of Rotation Groups: Fixed Axis

For fixed axis rotation, the average angle $\theta_{\mathrm{av}}(t)$ of the group relative to the reference group is determined via the distance-weighted angular deviation of all rotation group atoms from their reference positions,

$$\theta_{\mathrm{av}} = \sum_{i=1}^{N} r_i \, \theta_i \Big/ \sum_{i=1}^{N} r_i \, . \tag{6.55}$$

Here, r_i is the distance of the reference position to the rotation axis, and the difference angles θ_i are determined from the atomic positions, projected onto a plane perpendicular to the rotation axis through pivot point $\boldsymbol{u}$ (see eqn. 6.15 for the definition of $\perp$),

$$\cos\theta_i = \frac{(\boldsymbol{y}_i - \boldsymbol{u})^{\perp} \cdot (\boldsymbol{x}_i - \boldsymbol{u})^{\perp}}{\|(\boldsymbol{y}_i - \boldsymbol{u})^{\perp} \cdot (\boldsymbol{x}_i - \boldsymbol{u})^{\perp}\|} \, . \tag{6.56}$$

The sign of θ_{av} is chosen such that $\theta_{\mathrm{av}} > 0$ if the actual structure rotates ahead of the reference.

Angle of Rotation Groups: Flexible Axis

For flexible axis rotation, two outputs are provided, the angle of the entire rotation group, and separate angles for the segments in the slabs. The angle of the entire rotation group is determined by an RMSD fit of $\boldsymbol{x}_i$ to the reference positions $\boldsymbol{y}_i^0$ at $t = 0$, yielding θ_{fit} as the angle by which the reference has to be rotated around $\hat{\boldsymbol{v}}$ for the optimal fit,

$$\mathrm{RMSD}(\boldsymbol{x}_i, \; \boldsymbol{\Omega}(\theta_{\mathrm{fit}})\boldsymbol{y}_i^0) \stackrel{!}{=} \min \, . \tag{6.57}$$

To determine the local angle for each slab n, both reference and actual positions are weighted with the Gaussian function of slab n, and $\theta_{\mathrm{fit}}(t, n)$ is calculated as in eqn. 6.57) from the Gaussian-weighted positions.

For all angles, the `.mdp` input option `rot_fit_method` controls whether a normal RMSD fit is performed or whether for the fit each position $\boldsymbol{x}_i$ is put at the same distance to the rotation axis as its reference counterpart $\boldsymbol{y}_i^0$. In the latter case, the RMSD measures only angular differences, not radial ones.

Angle Determination by Searching the Energy Minimum

Alternatively, for `rot_fit_method = potential`, the angle of the rotation group is determined as the angle for which the rotation potential energy is minimal. Therefore, the used rotation potential is additionally evaluated for a set of angles around the current reference angle. In this case, the `rotangles.log` output file contains the values of the rotation potential at the chosen set of angles, while `rotation.xvg` lists the angle with minimal potential energy.

Torque

The torque $\boldsymbol{\tau}(t)$ exerted by the rotation potential is calculated for fixed axis rotation via

$$\boldsymbol{\tau}(t) = \sum_{i=1}^{N} \boldsymbol{r}_i(t) \times \boldsymbol{f}_i^{\perp}(t), \tag{6.58}$$

where $\boldsymbol{r}_i(t)$ is the distance vector from the rotation axis to $\boldsymbol{x}_i(t)$ and $\boldsymbol{f}_i^{\perp}(t)$ is the force component perpendicular to $\boldsymbol{r}_i(t)$ and $\hat{\boldsymbol{v}}$. For flexible axis rotation, torques $\boldsymbol{\tau}_n$ are calculated for each slab using the local rotation axis of the slab and the Gaussian-weighted positions.

6.6 Computational Electrophysiology

The Computational Electrophysiology (CompEL) protocol [135] allows the simulation of ion flux through membrane channels, driven by transmembrane potentials or ion concentration gradients. Just as in real cells, CompEL establishes transmembrane potentials by sustaining a small imbalance of charges Δq across the membrane, which gives rise to a potential difference ΔU according to the membrane capacitance:

$$\Delta U = \Delta q / C_{membrane} \tag{6.59}$$

The transmembrane electric field and concentration gradients are controlled by `.mdp` options, which allow the user to set reference counts for the ions on either side of the membrane. If a difference between the actual and the reference numbers persists over a certain time span, specified by the user, a number of ion/water pairs are exchanged between the compartments until the reference numbers are restored. Alongside the calculation of channel conductance and ion selectivity, CompEL simulations also enable determination of the channel reversal potential, an important characteristic obtained in electrophysiology experiments.

In a CompEL setup, the simulation system is divided into two compartments **A** and **B** with independent ion concentrations. This is best achieved by using double bilayer systems with a copy (or copies) of the channel/pore of interest in each bilayer (Fig. 6.6 A, B). If the channel axes point in the same direction, channel flux is observed simultaneously at positive and negative potentials in this way, which is for instance important for studying channel rectification.

The potential difference ΔU across the membrane is easily calculated with the `gmx potential` utility. By this, the potential drop along z or the pore axis is exactly known in each time interval of the simulation (Fig. 6.6 C). Type and number of ions n_i of charge q_i, traversing the channel in the simulation, are written to the `swapions.xvg` output file, from which the average channel conductance G in each interval Δt is determined by:

$$G = \frac{\sum_i n_i q_i}{\Delta t \, \Delta U} \, . \tag{6.60}$$

The ion selectivity is calculated as the number flux ratio of different species. Best results are obtained by averaging these values over several overlapping time intervals.

The calculation of reversal potentials is best achieved using a small set of simulations in which a given transmembrane concentration gradient is complemented with small ion imbalances of varying magnitude. For example, if one compartment contains 1 M salt and the other 0.1 M, and given charge neutrality otherwise, a set of simulations with $\Delta q = 0\,e$, $\Delta q = 2\,e$, $\Delta q = 4\,e$ could be used. Fitting a straight line through the current-voltage relationship of all obtained I-U pairs near zero current will then yield U_{rev}.

6.6.1 Usage

The following `.mdp` options control the CompEL protocol:

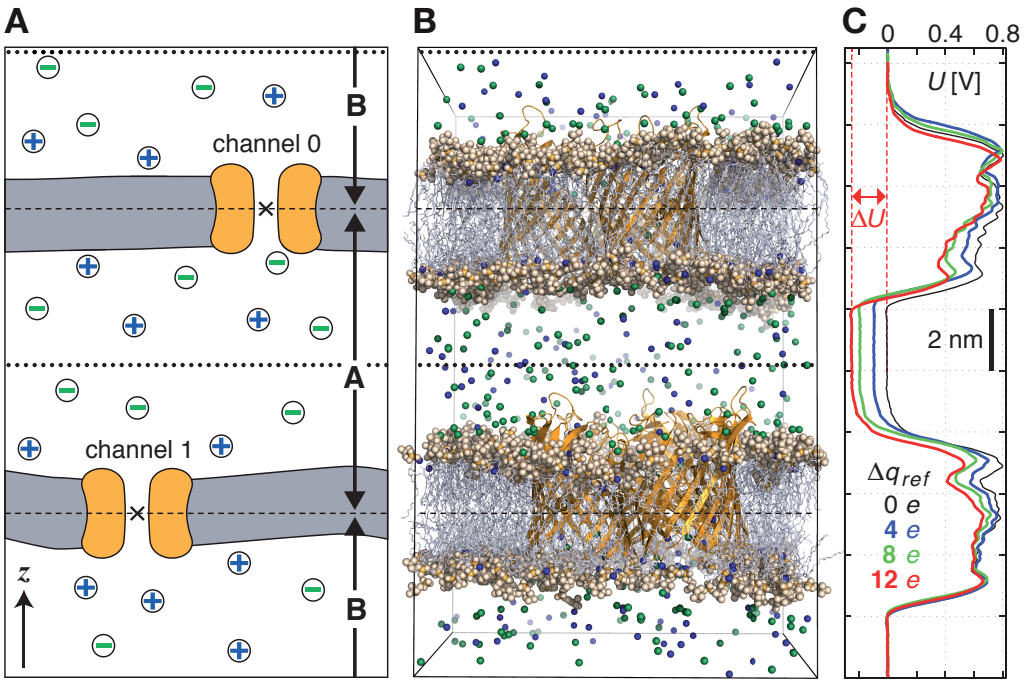


Figure 6.6: Typical double-membrane setup for CompEL simulations (A, B). Plot (C) shows the potential difference ΔU resulting from the selected charge imbalance Δq_{ref} between the compartments.

```
swapcoords      = Z             ; Swap positions: no, X, Y, Z
swap_frequency  = 100           ; Swap attempt frequency
```

Choose `Z` if your membrane is in the xy-plane (Fig. 6.6 A, B). Ions will be exchanged between compartments depending on their z-positions alone. `swap_frequency` determines how often a swap attempt will be made. This step requires that the positions of the ions, solvent, and swap groups are communicated between the parallel ranks, so if chosen too small it can decrease the simulation performance.

```
split_group0    = channel0      ; Defines compartment boundary
split_group1    = channel1      ; Defines other compartment boundary
massw_split0    = no            ; use mass-weighted center?
massw_split1    = no
```

`split_group0` and `split_group1` are two index groups that define the boundaries between the two compartments, which are usually the centers of the channels. If `massw_split0` or `massw_split1` are set to `yes`, the center of mass of each index group is used as boundary, here in z-direction. Otherwise, the geometrical centers will be used ($\times$ in Fig. 6.6 A). If, such as here, a membrane channel is selected as split group, the center of the channel will define the dividing plane between the compartments (dashed horizontal line in the figure). All index groups must be defined in the index file.

```
swap_group      = NA+_CL-       ; Ions to be included in exchange
solvent_group   = SOL           ; Group name of solvent molecules
```

```
cyl0_r          = 5.0          ; Split cylinder 0: pore radius (nm)
cyl0_up         = 0.75         ; Split cylinder 0 upper extension (nm)
cyl0_down       = 0.75         ; Split cylinder 0 lower extension (nm)
cyl1_r          = 5.0          ; same for other channel
cyl1_up         = 0.75
cyl1_down       = 0.75
coupl_steps     = 10           ; Average over these many swap steps
threshold       = 1            ; Do not swap if < threshold
```

`swap_group` identifies the index group of ions that should be involved in the flux and exchange cycles, `solvent_group` defines the solvent group with which they are swapped. The cylinder options only influence the counting of ions, i.e., ions will be counted as having traveled through either channel 0 or channel 1 according to the definition of (channel) cylinder radius, upper and lower extension, relative to the location of the respective split group. This will not affect the actual flux or exchange, but will provide you with the ion permeation numbers across each of the channels. Note that an ion can only be counted as passing through a particular channel if it is detected *within* the defined split cylinder in a swap step. If `swap_frequency` is chosen too high, a particular ion may be detected in compartment **A** in one swap step, and in compartment **B** in the following swap step, so it will be unclear through which of the channels it has passed.

`coupl_steps` sets the number of swap attempt steps. A discrepancy between actual and reference ion numbers in each compartment must persist over this many attempts before an actual exchange takes place. If `coupl_steps` is set to 1, then the momentary ion distribution determines whether ions are exchanged. `coupl_steps` > 1 will use the time-average of ion distributions over the selected number of attempt steps instead. This can be useful, for example, when ions diffuse near compartment boundaries, which would lead to numerous unproductive ion exchanges. A `threshold` of 1 means that a swap is performed if the average ion count in a compartment differs by at least 1 from the requested values. Higher thresholds will lead to toleration of larger differences. Ions are exchanged until the requested number $\pm$ the threshold is reached.

```
anionsA  = -1                  ; Reference count of anions in A
cationsA = -1                  ; ... of cations in A
anionsB  = -1                  ; ... of anions in B
cationsB = -1                  ; ... of cations in B
```

These options set the requested number of anions and cations for each of the two compartments. A number of `-1` means fix the numbers found in time step 0. Note that these numbers need to add up to the total number of ions in the swap group.

Note that a double-layered system for CompEL simulations can be easily prepared by duplicating an existing membrane/channel MD system in the direction of the membrane normal (typically z) with `gmx editconf -translate 0 0 <l_z>`, where `l_z` is the box length in that direction. If you have already defined index groups for the channel for the single-layered system, `gmx make_ndx -n index.ndx -twin` will provide you with the groups for the double-layered system.

To suppress large fluctuations of the membranes along the swap direction, it may be useful to apply a harmonic potential (acting only in the swap dimension) between each of the two channel and/or bilayer centers using umbrella pulling (see section 6.4).

Multimeric channels

If a split group consists of more than one molecule, the correct PBC image of all molecules with respect to each other has to be chosen such that the channel center can be correctly determined. GROMACS assumes that the starting structure in the `.tpr` file has the correct PBC representation. Set the following environment variable to check whether that is the case:

- `GMX_COMPELDUMP`: output the starting structure after it has been made whole to `.pdb` file.

6.7 Calculating a PMF using the free-energy code

The free-energy coupling-parameter approach (see sec. 3.12) provides several ways to calculate potentials of mean force. A potential of mean force between two atoms can be calculated by connecting them with a harmonic potential or a constraint. For this purpose there are special potentials that avoid the generation of extra exclusions, see sec. 5.4. When the position of the minimum or the constraint length is 1 nm more in state B than in state A, the restraint or constraint force is given by $\partial H/\partial\lambda$. The distance between the atoms can be changed as a function of λ and time by setting `delta-lambda` in the `.mdp` file. The results should be identical (although not numerically due to the different implementations) to the results of the pull code with umbrella sampling and constraint pulling. Unlike the pull code, the free energy code can also handle atoms that are connected by constraints.

Potentials of mean force can also be calculated using position restraints. With position restraints, atoms can be linked to a position in space with a harmonic potential (see 4.3.1). These positions can be made a function of the coupling parameter λ. The positions for the A and the B states are supplied to `grompp` with the `-r` and `-rb` options, respectively. One could use this approach to do targeted MD; note that we do not encourage the use of targeted MD for proteins. A protein can be forced from one conformation to another by using these conformations as position restraint coordinates for state A and B. One can then slowly change λ from 0 to 1. The main drawback of this approach is that the conformational freedom of the protein is severely limited by the position restraints, independent of the change from state A to B. Also, the protein is forced from state A to B in an almost straight line, whereas the real pathway might be very different. An example of a more fruitful application is a solid system or a liquid confined between walls where one wants to measure the force required to change the separation between the boundaries or walls. Because the boundaries (or walls) already need to be fixed, the position restraints do not limit the system in its sampling.

6.8 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 instead of a harmonic potential.

For the remaining degrees of freedom, the shortest oscillation period (as measured from a simu-

lation) 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 virtual interaction sites instead of normal atoms. Whereas a normal atom is connected to the molecule with bonds, angles and dihedrals, a virtual site's position is calculated from the position of three nearby heavy atoms in a predefined manner (see also sec. 4.7). 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 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. 6.8.1 and have previously been described in full detail [136].

Using both virtual sites 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 virtual site constructions.

All modifications described in this section can be performed using the GROMACS topology building tool `pdb2gmx`. Separate options exist to increase hydrogen masses, virtualize all hydrogen atoms, or also virtualize all aromatic residues. **Note** that when all hydrogen atoms are virtualized, those inside the aromatic residues will be virtualized as well, *i.e.* hydrogens in the aromatic residues are treated differently depending on the treatment of the aromatic residues.

Parameters for the virtual site constructions for the hydrogen atoms are inferred from the force-field 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 force fields, but these parameters are hard-coded into `pdb2gmx` due to the complex nature of the construction needed for a whole aromatic group.

6.8.1 Hydrogen bond-angle vibrations

Construction of virtual sites

The goal of defining hydrogen atoms as virtual sites 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 virtual sites, *e.g.* bond-angle vibration between the constructing atoms could translate into hydrogen bond-length vibration. Additionally, since virtual sites are by definition massless, in order to preserve total system mass, the mass of each hydrogen atom that is treated as

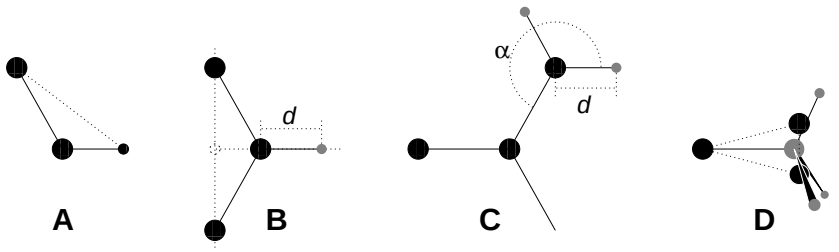


Figure 6.7: The different types of virtual site constructions used for hydrogen atoms. The atoms used in the construction of the virtual site(s) are depicted as black circles, virtual sites as gray ones. Hydrogens are smaller than heavy atoms. A: fixed bond angle, note that here the hydrogen is not a virtual site; 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.

virtual site 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. 6.7).

- *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. 6.7A. Doing so 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 virtual site.

- *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. Instead, 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. 6.7B.

- *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. Instead, 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. 6.7C.

- *amine group (umbrella -NH_2 or -NH_3^+) hydrogens:* Amine hydrogens with rotational freedom cannot be constructed as virtual sites 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

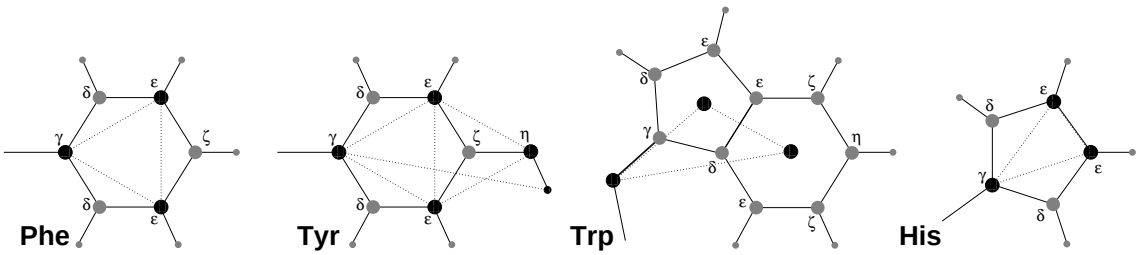


Figure 6.8: The different types of virtual site constructions used for aromatic residues. The atoms used in the construction of the virtual site(s) are depicted as black circles, virtual sites as gray ones. Hydrogens are smaller than heavy atoms. A: phenylalanine; B: tyrosine (note that the hydroxyl hydrogen is *not* a virtual site); C: tryptophan; D: histidine.

vectors and their outer product, resulting in an amine group with rotational freedom intact, but without other internal degrees of freedom. See Fig. 6.7D.

6.8.2 Out-of-plane vibrations in aromatic groups

The planar arrangements in the side chains of the aromatic residues lends itself perfectly to a virtual-site construction, giving a perfectly planar group without the inherently unstable 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 virtual sites type (see sec. 4.7) from these three. Each of the aromatic residues require a different approach:

- *Phenylalanine:* C_γ, $C_{\epsilon 1}$, and $C_{\epsilon 2}$ are kept as normal atoms, but with each a mass of one third the total mass of the phenyl group. See Fig. 6.7A.

- *Tyrosine:* The ring is treated identically to the phenylalanine ring. Additionally, constraints are defined between $C_{\epsilon 1}$, $C_{\epsilon 2}$, 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 virtual site. See Fig. 6.7B.

- *Tryptophan:* 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 ($C_{\delta 2}$ and $C_{\epsilon 2}$ 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. 6.7C.

- *Histidine:* C_γ, $C_{\epsilon 1}$ and $N_{\epsilon 2}$ are kept as normal atoms, but with masses redistributed such that the center of mass of the ring is preserved. See Fig. 6.7D.

6.9 Viscosity calculation

The shear viscosity is a property of liquids that can be determined easily by experiment. It is useful for parameterizing a force field 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 influences the rates of conformational changes of molecules solvated in the liquid.

The viscosity can be calculated from an equilibrium simulation using an Einstein relation:

$$\eta = \frac{1}{2}\frac{V}{k_B T}\lim_{t\to\infty}\frac{\mathrm{d}}{\mathrm{d}t}\left\langle\left(\int_{t_0}^{t_0+t} P_{xz}(t')\mathrm{d}t'\right)^2\right\rangle_{t_0} \tag{6.61}$$

This can be done with `g_energy`. This method converges very slowly [137], and as such a nanosecond simulation might not be long enough for an accurate determination of the viscosity. 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 [137]. 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:

$$a_x(z) + \frac{\eta}{\rho}\frac{\partial^2 v_x(z)}{\partial z^2} = 0 \tag{6.62}$$

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:

$$a_x(z) = A\cos\left(\frac{2\pi z}{l_z}\right) \tag{6.63}$$

where l_z is the height of the box. The generated velocity profile is:

$$v_x(z) = V\cos\left(\frac{2\pi z}{l_z}\right) \tag{6.64}$$

$$V = A\frac{\rho}{\eta}\left(\frac{l_z}{2\pi}\right)^2 \tag{6.65}$$

The viscosity can be calculated from A and V:

$$\eta = \frac{A}{V}\rho\left(\frac{l_z}{2\pi}\right)^2 \tag{6.66}$$

In the simulation V is defined as:

$$V = \frac{\sum_{i=1}^{N} m_i v_{i,x} 2\cos\left(\frac{2\pi z}{l_z}\right)}{\sum_{i=1}^{N} m_i} \tag{6.67}$$

The generated velocity profile is not coupled to the heat bath. Moreover, 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 being:

$$\mathrm{sh}_{\max} = \max_z \left| \frac{\partial v_x(z)}{\partial z} \right| = A \frac{\rho}{\eta} \frac{l_z}{2\pi} \tag{6.68}$$

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 ps. 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 box is elongated in the z-direction. In general, a simulation length of 100 ps 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 [30], which can be written in terms of the shear rate as:

$$T_s = \frac{\eta \tau}{2\rho C_v} \mathrm{sh}_{\max}^2 \tag{6.69}$$

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}] $\mathrm{sh}_{\max}^2$. 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 (6.66).

6.10 Tabulated interaction functions

6.10.1 Cubic splines for potentials

In some of the inner loops of GROMACS, look-up tables are used for computation of potential and forces. The tables are interpolated using a cubic spline algorithm. There are separate tables for electrostatic, dispersion, and repulsion interactions, but for the sake of caching performance these have been combined into a single array. The cubic spline interpolation for $x_i \leq x < x_{i+1}$ looks like this:

$$V_s(x) = A_0 + A_1\,\epsilon + A_2\,\epsilon^2 + A_3\,\epsilon^3 \tag{6.70}$$

where the table spacing h and fraction ϵ are given by:

$$h = x_{i+1} - x_i \tag{6.71}$$

$$\epsilon = (x - x_i)/h \tag{6.72}$$

so that $0 \leq \epsilon < 1$. From this, we can calculate the derivative in order to determine the forces:

$$-V_s'(x) \;=\; -\frac{\mathrm{d}V_s(x)}{\mathrm{d}\epsilon}\frac{\mathrm{d}\epsilon}{\mathrm{d}x} \;=\; -(A_1 + 2A_2\,\epsilon + 3A_3\,\epsilon^2)/h \tag{6.73}$$

The four coefficients are determined from the four conditions that V_s and $-V_s'$ at both ends of each interval should match the exact potential V and force $-V'$. This results in the following errors for each interval:

$$|V_s - V|_{max} \;=\; V'''' \frac{h^4}{384} + O(h^5) \tag{6.74}$$

$$|V_s' - V'|_{max} \;=\; V'''' \frac{h^3}{72\sqrt{3}} + O(h^4) \tag{6.75}$$

$$|V_s'' - V''|_{max} \;=\; V'''' \frac{h^2}{12} + O(h^3) \tag{6.76}$$

V and V' are continuous, while V" is the first discontinuous derivative. The number of points per nanometer is 500 and 2000 for mixed- and double-precision versions of GROMACS, respectively. This means that the errors in the potential and force will usually be smaller than the mixed precision accuracy.

GROMACS stores A_0, A_1, A_2 and A_3. The force routines get a table with these four parameters and a scaling factor s 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 = $s\mathrm{r}_{ij} - n_0$) and ϵ^2
4. Do the interpolation to calculate the potential V and the scalar force f
5. Calculate the vector force $\boldsymbol{F}$ by multiplying f with $\boldsymbol{r}_{ij}$

Note that table look-up 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.

6.10.2 User-specified potential functions

You can also use your own s without editing the GROMACS code. The potential function should be according to the following equation

$$V(r_{ij}) \;=\; \frac{q_i q_j}{4\pi\epsilon_0} f(r_{ij}) + C_6\, g(r_{ij}) + C_{12}\, h(r_{ij}) \tag{6.77}$$

where f, g, and h are 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
```

`mdrun` will read a single non-bonded table file, or multiple when `energygrp-table` is set (see below). The name of the file(s) can be set with the `mdrun` option `-table`. The table file should contain seven columns of table look-up data in the order: x, $f(x)$, $-f'(x)$, $g(x)$, $-g'(x)$, $h(x)$, $-h'(x)$. The x should run from 0 to $r_c + 1$ (the value of `table_extension` can be changed in the `.mdp` file). You can choose the spacing you like; for the standard tables GROMACS uses a spacing of 0.002 and 0.0005 nm when you run in mixed and double precision, respectively. 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. 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, and 6-12 Lennard-Jones potentials combined with a normal Coulomb.

If you want to have different functional forms between different groups of atoms, this can be set through energy groups. Different tables can be used for non-bonded interactions between different energy groups pairs through the `.mdp` option `energygrp-table` (see sec. 7.3). Atoms that should interact with a different potential should be put into different energy groups. Between group pairs which are not listed in `energygrp-table`, the normal user tables will be used. This makes it easy to use a different functional form between a few types of atoms.

6.11 Mixed Quantum-Classical simulation techniques

In a molecular mechanics (MM) force field, the influence of electrons is expressed by empirical parameters that are assigned on the basis of experimental data, or on the basis of results from high-level quantum chemistry calculations. These are valid for the ground state of a given covalent structure, and the MM approximation is usually sufficiently accurate for ground-state processes in which the overall connectivity between the atoms in the system remains unchanged. However, for processes in which the connectivity does change, such as chemical reactions, or processes that involve multiple electronic states, such as photochemical conversions, electrons can no longer be ignored, and a quantum mechanical description is required for at least those parts of the system in which the reaction takes place.

One approach to the simulation of chemical reactions in solution, or in enzymes, is to use a combination of quantum mechanics (QM) and molecular mechanics (MM). The reacting parts of the

system are treated quantum mechanically, with the remainder being modeled using the force field. The current version of GROMACS provides interfaces to several popular Quantum Chemistry packages (MOPAC [138], GAMESS-UK [139], Gaussian [140] and CPMD [141]).

GROMACS interactions between the two subsystems are either handled as described by Field *et al.* [142] or within the ONIOM approach by Morokuma and coworkers [143, 144].

6.11.1 Overview

Two approaches for describing the interactions between the QM and MM subsystems are supported in this version:

1. **Electronic Embedding** The electrostatic interactions between the electrons of the QM region and the MM atoms and between the QM nuclei and the MM atoms are included in the Hamiltonian for the QM subsystem:

$$H^{QM/MM} = H_e^{QM} - \sum_i^n \sum_J^M \frac{e^2 Q_J}{4\pi\epsilon_0 r_{iJ}} + \sum_A^N \sum_J^M \frac{e^2 Z_A Q_J}{e\pi\epsilon_0 R_{AJ}}, \tag{6.78}$$

   where n and N are the number of electrons and nuclei in the QM region, respectively, and M is the number of charged MM atoms. The first term on the right hand side is the original electronic Hamiltonian of an isolated QM system. The first of the double sums is the total electrostatic interaction between the QM electrons and the MM atoms. The total electrostatic interaction of the QM nuclei with the MM atoms is given by the second double sum. Bonded interactions between QM and MM atoms are described at the MM level by the appropriate force-field terms. Chemical bonds that connect the two subsystems are capped by a hydrogen atom to complete the valence of the QM region. The force on this atom, which is present in the QM region only, is distributed over the two atoms of the bond. The cap atom is usually referred to as a link atom.

2. **ONIOM** In the ONIOM approach, the energy and gradients are first evaluated for the isolated QM subsystem at the desired level of *ab initio* theory. Subsequently, the energy and gradients of the total system, including the QM region, are computed using the molecular mechanics force field and added to the energy and gradients calculated for the isolated QM subsystem. Finally, in order to correct for counting the interactions inside the QM region twice, a molecular mechanics calculation is performed on the isolated QM subsystem and the energy and gradients are subtracted. This leads to the following expression for the total QM/MM energy (and gradients likewise):

$$E_{tot} = E_I^{QM} + E_{I+II}^{MM} - E_I^{MM}, \tag{6.79}$$

   where the subscripts I and II refer to the QM and MM subsystems, respectively. The superscripts indicate at what level of theory the energies are computed. The ONIOM scheme has the advantage that it is not restricted to a two-layer QM/MM description, but can easily handle more than two layers, with each layer described at a different level of theory.

6.11.2 Usage

To make use of the QM/MM functionality in GROMACS, one needs to:

1. introduce link atoms at the QM/MM boundary, if needed;
2. specify which atoms are to be treated at a QM level;
3. specify the QM level, basis set, type of QM/MM interface and so on.

Adding link atoms

At the bond that connects the QM and MM subsystems, a link atoms is introduced. In GROMACS the link atom has special atomtype, called LA. This atomtype is treated as a hydrogen atom in the QM calculation, and as a virtual site in the force-field calculation. The link atoms, if any, are part of the system, but have no interaction with any other atom, except that the QM force working on it is distributed over the two atoms of the bond. In the topology, the link atom (LA), therefore, is defined as a virtual site atom:

```
[ virtual_sites2 ]
LA QMatom MMatom 1 0.65
```

See sec. 5.2.2 for more details on how virtual sites are treated. The link atom is replaced at every step of the simulation.

In addition, the bond itself is replaced by a constraint:

```
[ constraints ]
QMatom MMatom 2 0.153
```

Note that, because in our system the QM/MM bond is a carbon-carbon bond (0.153 nm), we use a constraint length of 0.153 nm, and dummy position of 0.65. The latter is the ratio between the ideal C-H bond length and the ideal C-C bond length. With this ratio, the link atom is always 0.1 nm away from the `QMatom`, consistent with the carbon-hydrogen bond length. If the QM and MM subsystems are connected by a different kind of bond, a different constraint and a different dummy position, appropriate for that bond type, are required.

Specifying the QM atoms

Atoms that should be treated at a QM level of theory, including the link atoms, are added to the index file. In addition, the chemical bonds between the atoms in the QM region are to be defined as connect bonds (bond type 5) in the topology file:

```
[ bonds ]
QMatom1 QMatom2 5
QMatom2 QMatom3 5
```

Specifying the QM/MM simulation parameters

In the `.mdp` file, the following parameters control a QM/MM simulation.

`QMMM = no`
: If this is set to `yes`, a QM/MM simulation is requested. Several groups of atoms can be described at different QM levels separately. These are specified in the QMMM-grps field separated by spaces. The level of *ab initio* theory at which the groups are described is specified by `QMmethod` and `QMbasis` Fields. Describing the groups at different levels of theory is only possible with the ONIOM QM/MM scheme, specified by `QMMMscheme`.

`QMMM-grps =`
: groups to be described at the QM level

`QMMMscheme = normal`
: Options are `normal` and `ONIOM`. This selects the QM/MM interface. `normal` implies that the QM subsystem is electronically embedded in the MM subsystem. There can only be one `QMMM-grps` that is modeled at the `QMmethod` and `QMbasis` level of *ab initio* theory. The rest of the system is described at the MM level. The QM and MM subsystems interact as follows: MM point charges are included in the QM one-electron Hamiltonian and all Lennard-Jones interactions are described at the MM level. If `ONIOM` is selected, the interaction between the subsystem is described using the ONIOM method by Morokuma and co-workers. There can be more than one QMMM-grps each modeled at a different level of QM theory (QMmethod and QMbasis).

`QMmethod =`
: Method used to compute the energy and gradients on the QM atoms. Available methods are AM1, PM3, RHF, UHF, DFT, B3LYP, MP2, CASSCF, MMVB and CPMD. For CASSCF, the number of electrons and orbitals included in the active space is specified by `CASelectrons` and `CASorbitals`. For CPMD, the plane-wave cut-off is specified by the `planewavecutoff` keyword.

`QMbasis =`
: Gaussian basis set used to expand the electronic wave-function. Only Gaussian basis sets are currently available, i.e. STO-3G, 3-21G, 3-21G*, 3-21+G*, 6-21G, 6-31G, 6-31G*, 6-31+G*, and 6-311G. For CPMD, which uses plane wave expansion rather than atom-centered basis functions, the `planewavecutoff` keyword controls the plane wave expansion.

`QMcharge =`
: The total charge in *e* of the `QMMM-grps`. In case there are more than one `QMMM-grps`, the total charge of each ONIOM layer needs to be specified separately.

`QMmult =`
: The multiplicity of the `QMMM-grps`. In case there are more than one `QMMM-grps`, the multiplicity of each ONIOM layer needs to be specified separately.

`CASorbitals =`
: The number of orbitals to be included in the active space when doing a CASSCF computation.

`CASelectrons =`
: The number of electrons to be included in the active space when doing a CASSCF computation.

`SH = no`
: If this is set to yes, a QM/MM MD simulation on the excited state-potential energy surface and enforce a diabatic hop to the ground-state when the system hits the conical intersection hyperline in the course the simulation. This option only works in combination with the CASSCF method.

6.11.3 Output

The energies and gradients computed in the QM calculation are added to those computed by GROMACS. In the `.edr` file there is a section for the total QM energy.

6.11.4 Future developments

Several features are currently under development to increase the accuracy of the QM/MM interface. One useful feature is the use of delocalized MM charges in the QM computations. The most important benefit of using such smeared-out charges is that the Coulombic potential has a finite value at interatomic distances. In the point charge representation, the partially-charged MM atoms close to the QM region tend to "over-polarize" the QM system, which leads to artifacts in the calculation.

What is needed as well is a transition state optimizer.

6.12 Adaptive Resolution Scheme

The adaptive resolution scheme [145, 146] (AdResS) couples two systems with different resolutions by a force interpolation scheme. In contrast to the mixed Quantum-Classical simulation techniques of the previous section, the number of high resolution particles is not fixed, but can vary over the simulation time.

Below we discuss AdResS for a double resolution (atomistic and coarse grained) representation of the same system. See Fig. 6.9 for illustration. The details of implementation described in this section were published in [147, 148].

Every molecule needs a well-defined mapping point (usually the center of mass) but any other linear combination of particle coordinates is also sufficient. In the topology the mapping point is defined by a virtual site. The forces in the coarse-grained region are functions of the mapping point positions only. In this implementation molecules are modeled by charge groups or sets of charge groups, which actually allows one to have multiple mapping points per molecule. This can be useful for bigger molecules like polymers. In that case one has to also extend the AdResS

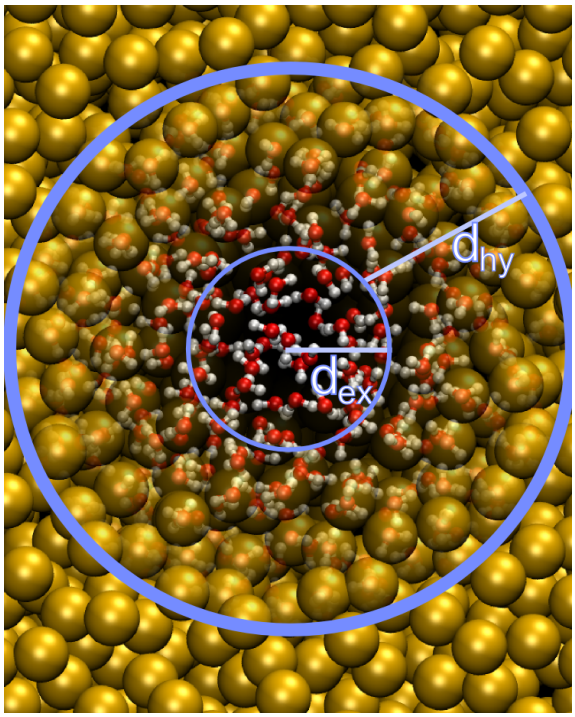


Figure 6.9: A schematic illustration of the AdResS method for water.

description to bonded interactions [149], which will be implemented into GROMACSin one of the future versions.

The force between two molecules is given by [145] [1]:

$$\vec{F}_{\alpha\beta} = w_\alpha w_\beta \vec{F}^{\mathrm{ex,mol}}_{\alpha\beta} + [1 - w_\alpha w_\beta]\, \vec{F}^{\mathrm{cg,mol}}_{\alpha\beta} \,, \tag{6.80}$$

where α and β label the two molecules and w_α, w_β are the adaptive weights of the two molecules. The first part, which represents the explicit interaction of the molecules, can be written as:

$$\vec{F}^{\mathrm{ex,mol}}_{\alpha\beta} = \sum_{i\in\alpha}\sum_{j\in\beta} \vec{F}^{\mathrm{ex}}_{ij} \,, \tag{6.81}$$

where $\vec{F}^{\mathrm{ex}}_{ij}$ is the force between the ith atom in αth molecule and the jth atom in the βth molecule, which is given by an explicit force field. The second part of eqn. 6.80 comes from the coarse-grained interaction of the molecules. In GROMACSa slightly extended case is implemented:

$$\vec{F}_{\alpha\beta} = \sum_{i\in\alpha}\sum_{j\in\beta} w_i w_j \vec{F}^{\mathrm{ex}}_{ij} + [1 - w_\alpha w_\beta]\, \vec{F}^{\mathrm{cg,mol}}_{\alpha\beta} \,, \tag{6.82}$$

where w_i and w_j are atom-wise weights, which are determined by the `adress-site` option. For `adress-site` being the center of mass, atom i has the weight of the center of mass of its *charge group*. The weight w_α of molecule α is determined by the position of coarse-grained particle,

[1] Note that the equation obeys Newton's third law, which is not the case for other interpolation schemes [150].

which is constructed as a virtual site from the atomistic particles as specified in the topology. This extension allows one to perform all kind of AdResS variations, but the common case can be recovered by using a center of mass virtual site in the topology, `adress-site=COM` and putting all atoms (except the virtual site representing the coarse-grained interaction) of a molecule into one charge group. For big molecules, it is sometimes useful to use an atom-based weight, which can be either be achieved by setting `adress-site=atomperatom` or putting every atom into a separate charge group (the center of mass of a charge group with one atom is the atom itself).

The coarse-grained force field $\vec{F}^{\mathrm{cg}}$ is usually derived from the atomistic system by structure-based coarse-graining (see sec. 4.10.5). To specify which atoms belong to a coarse-grained representation, energy groups are used. Each coarse-grained interaction has to be associated with a specific energy group, which is why the virtual sites representing the coarse-grained interactions also have to be in different charge groups. The energy groups which are treated as coarse-grained interactions are then listed in `adress_cg_grp_names`. The most important element of this interpolation (see eqn. 6.80 and eqn. 6.82) is the adaptive weighting function (for illustration see Fig. 6.9):

$$w(x) = \begin{cases} 1 & : \text{atomistic/explicit region} \\ 0 < w < 1 & : \text{hybrid region} \\ 0 & : \text{coarse} - \text{grained region} \end{cases} , \tag{6.83}$$

which has a value between 0 and 1. This definition of w gives a purely explicit force in the explicit region and a purely coarse-grained force in the coarse-grained region, so essentially eqn. 6.80 only the hybrid region has mixed interactions which would not appear in a standard simulation. In GROMACS, a $\cos^2$-like function is implemented as a weighting function:

$$w(x) = \begin{cases} 0 & : \quad x > d_{\mathrm{ex}} + d_{\mathrm{hy}} \\ \cos^2\left(\frac{\pi}{2d_{\mathrm{hy}}}(x - d_{\mathrm{ex}})\right) & : d_{\mathrm{ex}} + d_{\mathrm{hy}} > x > d_{\mathrm{ex}} \\ 1 & : \quad d_{\mathrm{ex}} > x \end{cases} , \tag{6.84}$$

where d_{ex} and d_{hy} are the sizes of the explicit and the hybrid region, respectively. Depending on the physical interest of the research, other functions could be implemented as long as the following boundary conditions are fulfilled: The function is 1) continuous, 2) monotonic and 3) has zero derivatives at the boundaries. Spherical and one-dimensional splitting of the simulation box has been implemented (`adress-type` option) and depending on this, the distance x to the center of the explicit region is calculated as follows:

$$x = \begin{cases} |(\vec{R}_\alpha - \vec{R}_{\mathrm{ct}}) \cdot \hat{e}| & : \text{splitting in } \hat{e} \text{ direction} \\ |\vec{R}_\alpha - \vec{R}_{\mathrm{ct}}| & : \text{spherical splitting} \end{cases} , \tag{6.85}$$

where $\vec{R}_{\mathrm{ct}}$ is the center of the explicit zone (defined by `adress-reference-coords` option). $\vec{R}_\alpha$ is the mapping point of the αth molecule. For the center of mass mapping, it is given by:

$$R_\alpha = \frac{\sum_{i \in \alpha} m_i r_i}{\sum_{i \in \alpha} m_i} \tag{6.86}$$

Note that the value of the weighting function depends exclusively on the mapping of the molecule.

The interpolation of forces (see eqn. 6.82) can produce inhomogeneities in the density and affect the structure of the system in the hybrid region.

One way of reducing the density inhomogeneities is by the application of the so-called thermodynamic force (TF) [151]. Such a force consists of a space-dependent external field applied in the hybrid region on the coarse-grained site of each molecule. It can be specified for each of the species of the system. The TF compensates the pressure profile [152] that emerges under a homogeneous density profile. Therefore, it can correct the local density inhomogeneities in the hybrid region and it also allows the coupling of atomistic and coarse-grained representations which by construction have different pressures at the target density. The field can be determined by an iterative procedure, which is described in detail in the manual of the VOTCA package [122]. Setting the `adress-interface-correction` to `thermoforce` enables the TF correction and `adress-tf-grp-names` defines the energy groups to act on.

6.12.1 Example: Adaptive resolution simulation of water

In this section the set up of an adaptive resolution simulation coupling atomistic SPC [81] water to its coarse-grained representation will be explained (as used in [152]). The following steps are required to setup the simulation:

- Perform a reference all-atom simulation
- Create a coarse-grained representation and save it as tabulated interaction function
- Create a hybrid topology for the SPC water
- Modify the atomistic coordinate file to include the coarse grained representation
- Define the geometry of the adaptive simulation in the grompp input file
- Create an index file

The coarse-grained representation of the interaction is stored as tabulated interaction function see 6.10.2. The convention is to use the $C^{(12)}$ columns with the $C^{(12)}$- coefficient set to 1. All other columns should be zero. The VOTCA manual has detailed instructions and a tutorial for SPC water on how to coarse-grain the interaction using various techniques. Here we named the coarse grained interaction CG, so the corresponding tabulated file is `table_CG_CG.xvg`. To create the topology one can start from the atomistic topology file (e.g. share/gromacs/top/oplsaa.ff/spc.itp), we are assuming rigid water here. In the VOTCA tutorial the file is named `hybrid_spc.itp`. The only difference to the atomistic topology is the addition of a coarse-grained virtual site:

```
[ moleculetype ]
; molname       nrexcl
SOL             2

[ atoms ]
;   nr   type  resnr residue  atom   cgnr     charge       mass
     1  opls_116   1    SOL     OW      1      -0.82
     2  opls_117   1    SOL    HW1      1       0.41
     3  opls_117   1    SOL    HW2      1       0.41
     4     CG      1    SOL     CG      2       0
```

```
[ settles ]
; OW    funct   doh     dhh
1       1       0.1     0.16330

[ exclusions ]
1       2       3
2       1       3
3       1       2

[ virtual_sites3 ]
; Site from funct a d
4 1 2 3 1 0.05595E+00 0.05595E+00
```

The virtual site type 3 with the specified coefficients places the virtual site in the center of mass of the molecule (for larger molecules virtual_sitesn has to be used). We now need to include our modified water model in the topology file and define the type `CG`. In `topol.top`:

```
#include "ffoplsaa.itp"

[ atomtypes ]
;name  mass        charge    ptype   sigma     epsilon
 CG    0.00000     0.0000    V       1    0.25

#include "hybrid_spc.itp"

[ system ]
Adaptive water
[ molecules ]
SOL    8507
```

The σ and ϵ values correspond to $C_6 = 1$ and $C_{12} = 1$ and thus the table file should contain the coarse-grained interaction in either the C_6 or C_{12} column. In the example the OPLS force field is used where σ and ϵ are specified. Note that for force fields which define atomtypes directly in terms of C_6 and C_{12}, one can simply set $C_6 = 0$ and $C_{12} = 1$. See section 6.10.2 for more details on tabulated interactions. Since now the water molecule has a virtual site the coordinate file also needs to include that.

```
adaptive water coordinates
34028
    1SOL     OW    1   0.283   0.886   0.647
    1SOL    HW1    2   0.359   0.884   0.711
    1SOL    HW2    3   0.308   0.938   0.566
    1SOL     CG    4   0.289   0.889   0.646
    1SOL     OW    5   1.848   0.918   0.082
    1SOL    HW1    6   1.760   0.930   0.129
    1SOL    HW2    7   1.921   0.912   0.150
    1SOL     CG    8   1.847   0.918   0.088
    (...)
```

This file can be created manually or using the VOTCA tool `csg_map` with the `--hybrid` option.
In the grompp input file the AdResS feature needs to be enabled and the geometry defined.

```
(...)
; AdResS relevant options
energygrps               = CG
energygrp_table          = CG CG

; Method for doing Van der Waals
vdw-type                 = user

adress                   = yes
adress_type              = xsplit
adress_ex_width          = 1.5
adress_hy_width          = 1.5
adress_interface_correction = off
adress_reference_coords = 8 0 0
adress_cg_grp_names = CG
```

Here we are defining an energy group `CG` which consists of the coarse-grained virtual site. As discussed above, the coarse-grained interaction is usually tabulated. This requires the `vdw-type` parameter to be set to `user`. In the case where multi-component systems are coarse-grained, an energy group has to be defined for each component. Note that all the energy groups defining coarse-grained representations have to be listed again in `adress_cg_grp_names` to distinguish them from regular energy groups.
The index file has to be updated to have a group CG which includes all the coarse-grained virtual sites. This can be done easily using the `make_ndx` tool of gromacs.

6.13 Using VMD plug-ins for trajectory file I/O

GROMACS tools are able to use the plug-ins found in an existing installation of VMD in order to read and write trajectory files in formats that are not native to GROMACS. You will be able to supply an AMBER DCD-format trajectory filename directly to GROMACS tools, for example.

This requires a VMD installation not older than version 1.8, that your system provides the dlopen function so that programs can determine at run time what plug-ins exist, and that you build shared libraries when building GROMACS. CMake will find the vmd executable in your path, and from it, or the environment variable `VMDDIR` at configuration or run time, locate the plug-ins. Alternatively, the `VMD_PLUGIN_PATH` can be used at run time to specify a path where these plug-ins can be found. Note that these plug-ins are in a binary format, and that format must match the architecture of the machine attempting to use them.

6.14 Interactive Molecular Dynamics

GROMACS supports the interactive molecular dynamics (IMD) protocol as implemented by VMD to control a running simulation in NAMD. IMD allows to monitor a running GROMACS simulation from a VMD client. In addition, the user can interact with the simulation by pulling on atoms, residues or fragments with a mouse or a force-feedback device. Additional information about the GROMACS implementation and an exemplary GROMACS IMD system can be found

on this homepage.

6.14.1 Simulation input preparation

The GROMACS implementation allows transmission and interaction with a part of the running simulation only, e.g. in cases where no water molecules should be transmitted or pulled. The group is specified via the `.mdp` option `IMD-group`. When `IMD-group` is empty, the IMD protocol is disabled and cannot be enabled via the switches in `mdrun`. To interact with the entire system, `IMD-group` can be set to `System`. When using `grompp`, a `.gro` file to be used as VMD input is written out (`-imd` switch of `grompp`).

6.14.2 Starting the simulation

Communication between VMD and GROMACS is achieved via TCP sockets and thus enables controlling an `mdrun` running locally or on a remote cluster. The port for the connection can be specified with the `-imdport` switch of `mdrun`, 8888 is the default. If a port number of 0 or smaller is provided, GROMACS automatically assigns a free port to use with IMD.

Every N steps, the `mdrun` client receives the applied forces from VMD and sends the new positions to the client. VMD permits increasing or decreasing the communication frequency interactively. By default, the simulation starts and runs even if no IMD client is connected. This behavior is changed by the `-imdwait` switch of `mdrun`. After startup and whenever the client has disconnected, the integration stops until reconnection of the client. When the `-imdterm` switch is used, the simulation can be terminated by pressing the stop button in VMD. This is disabled by default. Finally, to allow interacting with the simulation (i.e. pulling from VMD) the `-imdpull` switch has to be used. Therefore, a simulation can only be monitored but not influenced from the VMD client when none of `-imdwait`, `-imdterm` or `-imdpull` are set. However, since the IMD protocol requires no authentication, it is not advisable to run simulations on a host directly reachable from an insecure environment. Secure shell forwarding of TCP can be used to connect to running simulations not directly reachable from the interacting host. Note that the IMD command line switches of `mdrun` are hidden by default and show up in the help text only with `gmx mdrun -h -hidden`.

6.14.3 Connecting from VMD

In VMD, first the structure corresponding to the IMD group has to be loaded (*File* $\rightarrow$ *New Molecule*). Then the IMD connection window has to be used (*Extensions* $\rightarrow$ *Simulation* $\rightarrow$ *IMD Connect (NAMD)*). In the IMD connection window, hostname and port have to be specified and followed by pressing *Connect*. *Detach Sim* allows disconnecting without terminating the simulation, while *Stop Sim* ends the simulation on the next neighbor searching step (if allowed by `-imdterm`).

The timestep transfer rate allows adjusting the communication frequency between simulation and IMD client. Setting the keep rate loads every N^{th} frame into VMD instead of discarding them when a new one is received. The displayed energies are in SI units in contrast to energies displayed from NAMD simulations.

Chapter 7

Run parameters and Programs

7.1 On-line 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/gromacs/html/online.html`. If you installed from Linux packages it can typically be found as `/usr/share/gromacs/html/online.html`. You can also use the online manual from the GROMACS web site, `http://manual.gromacs.org/current`.

In addition, we install standard UNIX man pages 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` environment variable, and you should be able to type *e.g.* `man gmx-grompp`. You can also use the `-h` flag on the command line (e.g. `gmx grompp -h`) to see the same information, as well as `gmx help grompp`. The list of all programs are available from `gmx help`.

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 NFS support.

Default Name	Ext.	Type	Default Option	Description
atomtp	.atp	Asc		Atomtype file used by pdb2gmx
eiwit	.brk	Asc	-f	Brookhaven data bank file
state	.cpt	xdr		Checkpoint 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
conf	.esp	Asc	-c	Coordinate file in ESPResSo format
conf	.g96	Asc	-c	Coordinate file in Gromos-96 format
conf	.gro	Asc	-c	Coordinate file in Gromos-87 format
conf	.gro		-c	Structure: gro g96 pdb esp tpr tpb tpa
out	.gro		-o	Structure: gro g96 pdb esp
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
residue	.rtp	Asc		Residue Type file used by pdb2gmx
doc	.tex	Asc	-o	LaTeX file
topol	.top	Asc	-p	Topology 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 (architecture specific)
traj	.trr			Full precision trajectory: trr trj cpt
traj	.trr	xdr		Trajectory in portable xdr format
root	.xpm	Asc		X PixMap compatible matrix file
traj	.xtc		-f	Trajec., input: xtc trr trj cpt gro g96 pdb
traj	.xtc		-f	Trajectory, output: 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.

7.3 Run Parameters

7.3.1 General

Default values are given in parentheses. The first option in the list 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

`include:`
directories to include in your topology. Format:
`-I/home/john/mylib -I../otherlib`

`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:

`-DFLEXIBLE`
Will tell `grompp` to include flexible water in stead of rigid water into your topology, this can be useful for normal mode analysis.

`-DPOSRES`
Will tell `grompp` to include posre.itp into your topology, used for position restraints.

7.3.3 Run control

`integrator:` (Despite the name, this list includes algorithms that are not actually integrators. `steep` and all entries following it are in this category)

`md`
A leap-frog algorithm for integrating Newton's equations of motion.

`md-vv`
A velocity Verlet algorithm for integrating Newton's equations of motion. For constant NVE simulations started from corresponding points in the same trajectory, the trajectories are analytically, but not binary, identical to the `md` leap-frog integrator. The the kinetic energy, which is determined from the whole step velocities and is therefore slightly too high. The advantage of this integrator is more accurate, reversible Nose-Hoover and Parrinello-Rahman coupling integration based on Trotter expansion, as well as (slightly too small) full step velocity output. This all comes at the cost off extra computation, especially with constraints and extra communication in parallel. Note that for nearly all production simulations the `md` integrator is accurate enough.

md-vv-avek
A velocity Verlet algorithm identical to `md-vv`, except that the kinetic energy is determined as the average of the two half step kinetic energies as in the `md` integrator, and this thus more accurate. With Nose-Hoover and/or Parrinello-Rahman coupling this comes with a slight increase in computational cost.

sd
An accurate and efficient leap-frog stochastic dynamics integrator. With constraints, coordinates needs to be constrained twice per integration step. Depending on the computational cost of the force calculation, this can take a significant part of the simulation time. 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`. When used as a thermostat, an appropriate value for `tau-t` is 2 ps, since this results in a friction that is lower than the internal friction of water, while it is high enough to remove excess heat NOTE: temperature deviations decay twice as fast as with a Berendsen thermostat with the same `tau-t`.

sd2
This used to be the default sd integrator, but is now deprecated. Four Gaussian random numbers are required per coordinate per step. With constraints, the temperature will be slightly too high.

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 (`ref-t`). 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`.

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`. For a minimization prior to a normal mode analysis, which requires a very high accuracy, GROMACS should be compiled in double precision.

l-bfgs
A quasi-Newtonian algorithm for energy minimization according to the low-memory Broyden-Fletcher-Goldfarb-Shanno approach. In practice this seems to converge faster than Conjugate Gradients, but due to the correction steps necessary it is not (yet) parallelized.

nm
Normal mode analysis is performed on the structure in the `tpr` file. GROMACS should be compiled in double precision.

tpi

Test particle insertion. The last molecule in the topology is the test particle. A trajectory should be provided with the `-rerun` option of `mdrun`. This trajectory should not contain the molecule to be inserted. Insertions are performed `nsteps` times in each frame at random locations and with random orientiations of the molecule. When `nstlist` is larger than one, `nstlist` insertions are performed in a sphere with radius `rtpi` around a the same random location using the same neighborlist (and the same long-range energy when `rvdw` or `rcoulomb>rlist`, which is only allowed for single-atom molecules). Since neighborlist construction is expensive, one can perform several extra insertions with the same list almost for free. The random seed is set with `ld-seed`. The temperature for the Boltzmann weighting is set with `ref-t`, this should match the temperature of the simulation of the original trajectory. Dispersion correction is implemented correctly for tpi. All relevant quantities are written to the file specified with the `-tpi` option of `mdrun`. The distribution of insertion energies is written to the file specified with the `-tpid` option of `mdrun`. No trajectory or energy file is written. Parallel tpi gives identical results to single node tpi. For charged molecules, using PME with a fine grid is most accurate and also efficient, since the potential in the system only needs to be calculated once per frame.

tpic

Test particle insertion into a predefined cavity location. The procedure is the same as for `tpi`, except that one coordinate extra is read from the trajectory, which is used as the insertion location. The molecule to be inserted should be centered at 0,0,0. Gromacs does not do this for you, since for different situations a different way of centering might be optimal. Also `rtpi` sets the radius for the sphere around this location. Neighbor searching is done only once per frame, `nstlist` is not used. Parallel tpic gives identical results to single node tpic.

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: (0)

maximum number of steps to integrate or minimize, -1 is no maximum

init-step: (0)

The starting step. The time at an step i in a run is calculated as: t = `tinit` + `dt`*(`init-step` + i). The free-energy lambda is calculated as: lambda = `init-lambda` + `delta-lambda`*(`init-step` + i). Also non-equilibrium MD parameters can depend on the step number. Thus for exact restarts or redoing part of a run it might be necessary to set `init-step` to the step number of the restart frame. `gmx convert-tpr` does this automatically.

comm-mode:

Linear

Remove center of mass translation

Angular
Remove center of mass translation and rotation around the center of mass

None
No restriction on the center of mass motion

nstcomm: (100) [steps]
frequency for center of mass motion removal

comm-grps:
group(s) for center of mass motion removal, default is the whole system

7.3.4 Langevin dynamics

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: (-1) [integer]
used to initialize random generator for thermal noise for stochastic and Brownian dynamics. When `ld-seed` is set to -1, a pseudo random seed is used. When running BD or SD on multiple processors, each processor uses a seed equal to `ld-seed` plus the processor number.

7.3.5 Energy minimization

emtol: (10.0) [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.

nbfgscorr: (10)
Number of correction steps to use for L-BFGS minimization. A higher number is (at least theoretically) more accurate, but slower.

7.3.6 Shell Molecular Dynamics

When shells or flexible constraints are present in the system the positions of the shells and the lengths of the flexible constraints are optimized at every time step until either the RMS force on the shells and constraints is less than emtol, or a maximum number of iterations (niter) has been reached

emtol: (10.0) [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 10.0.

niter: (20)
maximum number of iterations for optimizing the shell positions and the flexible constraints.

fcstep: (0) [ps^2]
the step size for optimizing the flexible constraints. Should be chosen as mu/(d^{2}V/dq^2) where mu is the reduced mass of two particles in a flexible constraint and d^{2}V/dq^2 is the second derivative of the potential in the constraint direction. Hopefully this number does not differ too much between the flexible constraints, as the number of iterations and thus the runtime is very sensitive to `fcstep`. Try several values!

7.3.7 Test particle insertion

rtpi: (0.05) [nm]
the test particle insertion radius see integrators `tpi` and `tpic`

7.3.8 Output control

nstxout: (0) [steps]
number of steps that elapse between writing coordinates to output trajectory file, the last coordinates are always written

nstvout: (0) [steps]
number of steps that elapse between writing velocities to output trajectory, the last velocities are always written

nstfout: (0) [steps]
number of steps that elapse between writing forces to output trajectory.

nstlog: (1000) [steps]
number of steps that elapse between writing energies to the log file, the last energies are always written

nstcalcenergy: (100)
number of steps that elapse between calculating the energies, 0 is never. This option is only relevant with dynamics. With a twin-range cut-off setup `nstcalcenergy` should be equal to or a multiple of `nstlist`. This option affects the performance in parallel simulations, because calculating energies requires global communication between all processes which can become a bottleneck at high parallelization.

nstenergy: (1000) [steps]
number of steps that else between writing energies to energy file, the last energies are always written, should be a multiple of `nstcalcenergy`. Note that the exact sums and

fluctuations over all MD steps modulo `nstcalcenergy` are stored in the energy file, so `g_energy` can report exact energy averages and fluctuations also when `nstenergy`>1

nstxout-compressed: (0) [steps]
number of steps that elapse between writing position coordinates using lossy compression

compressed-x-precision: (1000) [real]
precision with which to write to the compressed trajectory file

compressed-x-grps:
group(s) to write to the compressed trajectory file, by default the whole system is written (if `nstxout-compressed` > 0)

energygrps:
group(s) to write to energy file

7.3.9 Neighbor searching

cutoff-scheme:

Verlet
Generate a pair list with buffering. The buffer size is automatically set based on `verlet-buffer-tolerance`, unless this is set to -1, in which case `rlist` will be used. This option has an explicit, exact cut-off at `rvdw=rcoulomb`. Currently only cut-off, reaction-field, PME electrostatics and plain LJ are supported. Some `mdrun` functionality is not yet supported with the `Verlet` scheme, but `grompp` checks for this. Native GPU acceleration is only supported with `Verlet`. With GPU-accelerated PME or with separate PME ranks, `mdrun` will automatically tune the CPU/GPU load balance by scaling `rcoulomb` and the grid spacing. This can be turned off with `-notunepme`. `Verlet` is faster than `group` when there is no water, or if `group` would use a pair-list buffer to conserve energy.

group
Generate a pair list for groups of atoms. These groups correspond to the charge groups in the topology. This was the only cut-off treatment scheme before version 4.6. There is no explicit buffering of the pair list. This enables efficient force calculations for water, but energy is only conserved when a buffer is explicitly added.

nstlist: (10) [steps]

>0
Frequency to update the neighbor list (and the long-range forces, when using twin-range cut-offs). When this is 0, the neighbor list is made only once. With energy minimization the neighborlist will be updated for every energy evaluation when `nstlist`>0. With `cutoff-scheme=Verlet` and `verlet-buffer-tolerance` set, `nstlist` is actually a minimum value and `mdrun` might increase it, unless it is set to 1. With parallel simulations and/or non-bonded force calculation on the GPU, a value of 20

or 40 often gives the best performance. With `cutoff-scheme=Group` and non-exact cut-off's, `nstlist` will affect the accuracy of your simulation and it can not be chosen freely.

0

The neighbor list is only constructed once and never updated. This is mainly useful for vacuum simulations in which all particles see each other.

-1

Automated update frequency, only supported with `cutoff-scheme=group`. This can only be used with switched, shifted or user potentials where the cut-off can be smaller than `rlist`. One then has a buffer of size `rlist` minus the longest cut-off. The neighbor list is only updated when one or more particles have moved further than half the buffer size from the center of geometry of their charge group as determined at the previous neighbor search. Coordinate scaling due to pressure coupling or the `deform` option is taken into account. This option guarantees that their are no cut-off artifacts, but for larger systems this can come at a high computational cost, since the neighbor list update frequency will be determined by just one or two particles moving slightly beyond the half buffer length (which does not necessarily imply that the neighbor list is invalid), while 99.99% of the particles are fine.

nstcalclr: (-1) [steps]

Controls the period between calculations of long-range forces when using the group cut-off scheme.

1

Calculate the long-range forces every single step. This is useful to have separate neighbor lists with buffers for electrostatics and Van der Waals interactions, and in particular it makes it possible to have the Van der Waals cutoff longer than electrostatics (useful *e.g.* with PME). However, there is no point in having identical long-range cutoffs for both interaction forms and update them every step - then it will be slightly faster to put everything in the short-range list.

>1

Calculate the long-range forces every `nstcalclr` steps and use a multiple-time-step integrator to combine forces. This can now be done more frequently than `nstlist` since the lists are stored, and it might be a good idea *e.g.* for Van der Waals interactions that vary slower than electrostatics.

-1

Calculate long-range forces on steps where neighbor searching is performed. While this is the default value, you might want to consider updating the long-range forces more frequently.

Note that twin-range force evaluation might be enabled automatically by PP-PME load balancing. This is done in order to maintain the chosen Van der Waals interaction radius even if the load balancing is changing the electrostatics cutoff. If the `.mdp` file already specifies twin-range interactions (*e.g.* to evaluate Lennard-Jones interactions with a longer cutoff than the PME electrostatics every 2-3 steps), the load balancing will have also a small effect on

Lennard-Jones, since the short-range cutoff (inside which forces are evaluated every step) is changed.

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 (only with `cutoff-scheme=group`).

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`. For best performance without cut-offs on a single MPI rank, use `nstlist=0`, `ns-type=simple`

xy
Use periodic boundary conditions in x and y directions only. This works only with `ns-type=grid` and can be used in combination with `walls`. Without walls or with only one wall the system size is infinite in the z direction. Therefore pressure coupling or Ewald summation methods can not be used. These disadvantages do not apply when two walls are used.

periodic-molecules:

no
molecules are finite, fast molecular PBC can be used

yes
for systems with molecules that couple to themselves through the periodic boundary conditions, this requires a slower PBC algorithm and molecules are not made whole in the output

verlet-buffer-tolerance: (0.005) [kJ/mol/ps]
Useful only with `cutoff-scheme=Verlet`. This sets the maximum allowed error for pair interactions per particle caused by the Verlet buffer, which indirectly sets `rlist`. As both `nstlist` and the Verlet buffer size are fixed (for performance reasons), particle pairs not in the pair list can occasionally get within the cut-off distance during `nstlist`-1 nsteps. This causes very small jumps in the energy. In a constant-temperature ensemble, these very small energy jumps can be estimated for a given cut-off and `rlist`. The estimate assumes a homogeneous particle distribution, hence the errors might be slightly underestimated for multi-phase systems. For longer pair-list life-time (`nstlist`-1)*dt the buffer is overestimated, because the interactions between particles are ignored. Combined with cancellation

of errors, the actual drift of the total energy is usually one to two orders of magnitude smaller. Note that the generated buffer size takes into account that the GROMACS pair-list setup leads to a reduction in the drift by a factor 10, compared to a simple particle-pair based list. Without dynamics (energy minimization etc.), the buffer is 5% of the cut-off. For NVE simulations the initial temperature is used, unless this is zero, in which case a buffer of 10% is used. For NVE simulations the tolerance usually needs to be lowered to achieve proper energy conservation on the nanosecond time scale. To override the automated buffer setting, use `verlet-buffer-tolerance`=-1 and set `rlist` manually.

`rlist: (1) [nm]`
Cut-off distance for the short-range neighbor list. With `cutoff-scheme=Verlet`, this is by default set by the `verlet-buffer-tolerance` option and the value of `rlist` is ignored.

`rlistlong: (-1) [nm]`
Cut-off distance for the long-range neighbor list. This parameter is only relevant for a twin-range cut-off setup with switched potentials. In that case a buffer region is required to account for the size of charge groups. In all other cases this parameter is automatically set to the longest cut-off distance.

7.3.10 Electrostatics

`coulombtype:`

`Cut-off`
Twin range cut-offs with neighborlist cut-off `rlist` and Coulomb cut-off `rcoulomb`, where `rcoulomb`≥`rlist`.

`Ewald`
Classical Ewald sum electrostatics. The real-space cut-off `rcoulomb` should be equal to `rlist`. Use *e.g.* `rlist=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.

`PME`
Fast smooth Particle-Mesh Ewald (SPME) 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 2-3*10^{-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.

`P3M-AD`
Particle-Particle Particle-Mesh algorithm with analytical derivative for for long range electrostatic interactions. The method and code is identical to SPME, except that the influence function is optimized for the grid. This gives a slight increase in accuracy.

Reaction-Field electrostatics
Reaction field with Coulomb cut-off `rcoulomb`, where `rcoulomb` $\geq$ `rlist`. The dielectric constant beyond the cut-off is `epsilon-rf`. The dielectric constant can be set to infinity by setting `epsilon-rf=0`.

Generalized-Reaction-Field
Generalized reaction field with Coulomb cut-off `rcoulomb`, where `rcoulomb` $\geq$ `rlist`. The dielectric constant beyond the cut-off is `epsilon-rf`. 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].

Reaction-Field-zero
In GROMACS, normal reaction-field electrostatics with `cutoff-scheme=group` leads to bad energy conservation. `Reaction-Field-zero` solves this by making the potential zero beyond the cut-off. It can only be used with an infinite dielectric constant (`epsilon-rf=0`), because only for that value the force vanishes at the 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. This, and the fact that table lookups are used instead of analytical functions make `Reaction-Field-zero` computationally more expensive than normal reaction-field.

Reaction-Field-nec
The same as `Reaction-Field`, but implemented as in GROMACS versions before 3.3. No reaction-field correction is applied to excluded atom pairs and self pairs. The 1-4 interactions are calculated using a reaction-field. The missing correction due to the excluded pairs that do not have a 1-4 interaction is up to a few percent of the total electrostatic energy and causes a minor difference in the forces and the pressure.

Shift
Analogous to `Shift` for `vdwtype`. You might want to use `Reaction-Field-zero` instead, which has a similar potential shape, but has a physical interpretation and has better energies due to the exclusion correction terms.

Encad-Shift
The Coulomb potential is decreased over the whole range, using the definition from the Encad simulation package.

Switch
Analogous to `Switch` for `vdwtype`. Switching the Coulomb potential can lead to serious artifacts, advice: use `Reaction-Field-zero` instead.

User
`mdrun` will now expect to find a file `table.xvg` with user-defined potential functions for repulsion, dispersion and Coulomb. When pair interactions are present, `mdrun` also expects to find a file `tablep.xvg` for the pair interactions. When the same interactions should be used for non-bonded and pair interactions the user can specify the same file name for both table files. These files should contain 7 columns: the `x` value, `f(x)`, `-f'(x)`, `g(x)`, `-g'(x)`, `h(x)`, `-h'(x)`, where `f(x)` is the Coulomb function, `g(x)` the dispersion function and `h(x)` the repulsion function. When `vdwtype` is not set to `User` the values for `g`, `-g'`, `h` and `-h'` are ignored. For the non-bonded interactions `x` values should run from 0 to the largest cut-off distance

+ `table-extension` and should be uniformly spaced. For the pair interactions the table length in the file will be used. The optimal spacing, which is used for non-user tables, is `0.002` [nm] when you run in mixed precision or `0.0005` [nm] when you run in double precision. The function value at `x=0` is not important. More information is in the printed manual.

`PME-Switch`
A combination of PME and a switch function for the direct-space part (see above). `rcoulomb` is allowed to be smaller than `rlist`. This is mainly useful constant energy simulations (note that using `PME` with `cutoff-scheme=Verlet` will be more efficient).

`PME-User`
A combination of PME and user tables (see above). `rcoulomb` is allowed to be smaller than `rlist`. The PME mesh contribution is subtracted from the user table by `mdrun`. Because of this subtraction the user tables should contain about 10 decimal places.

`PME-User-Switch`
A combination of PME-User and a switching function (see above). The switching function is applied to final particle-particle interaction, *i.e.* both to the user supplied function and the PME Mesh correction part.

`coulomb-modifier:`

`Potential-shift-Verlet`
Selects `Potential-shift` with the Verlet cutoff-scheme, as it is (nearly) free; selects `None` with the group cutoff-scheme.

`Potential-shift`
Shift the Coulomb potential by a constant such that it is zero at the cut-off. This makes the potential the integral of the force. Note that this does not affect the forces or the sampling.

`None`
Use an unmodified Coulomb potential. With the group scheme this means no exact cut-off is used, energies and forces are calculated for all pairs in the neighborlist.

`rcoulomb-switch: (0) [nm]`
where to start switching the Coulomb potential, only relevant when force or potential switching is used

`rcoulomb: (1) [nm]`
distance for the Coulomb cut-off

`epsilon-r: (1)`
The relative dielectric constant. A value of 0 means infinity.

`epsilon-rf: (0)`
The relative dielectric constant of the reaction field. This is only used with reaction-field electrostatics. A value of 0 means infinity.

7.3.11 VdW

vdwtype:

Cut-off
Twin range cut-offs with neighbor list cut-off `rlist` and VdW cut-off `rvdw`, where `rvdw` $\geq$ `rlist`.

PME
Fast smooth Particle-mesh Ewald (SPME) for VdW interactions. The grid dimensions are controlled with `fourierspacing` in the same way as for electrostatics, and the interpolation order is controlled with `pme-order`. The relative accuracy of direct/reciprocal space is controlled by `ewald-rtol-lj`, and the specific combination rules that are to be used by the reciprocal routine are set using `lj-pme-comb-rule`.

Shift
This functionality is deprecated and replaced by `vdw-modifier = Force-switch`. 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.

Switch
This functionality is deprecated and replaced by `vdw-modifier = Potential-switch`. The LJ (not Buckingham) potential is normal out to `rvdw-switch`, after which it is switched off to reach zero at `rvdw`. Both the potential and force functions are continuously smooth, but be aware that all switch functions will give rise to a bulge (increase) in the force (since we are switching the potential). 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.

Encad-Shift
The LJ (not Buckingham) potential is decreased over the whole range, using the definition from the Encad simulation package.

User
See `user` for `coulombtype`. 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. When `coulombtype` is not set to `User` the values for `f` and `-f'` are ignored.

vdw-modifier:

Potential-shift-Verlet
Selects `Potential-shift` with the Verlet cutoff-scheme, as it is (nearly) free; selects `None` with the group cutoff-scheme.

Potential-shift
Shift the Van der Waals potential by a constant such that it is zero at the cut-off. This makes the potential the integral of the force. Note that this does not affect the forces or the sampling.

None
Use an unmodified Van der Waals potential. With the group scheme this means no exact cut-off is used, energies and forces are calculated for all pairs in the neighborlist.

Force-switch
Smoothly switches the forces to zero between `rvdw-switch` and `rvdw`. This shifts the potential shift over the whole range and switches it to zero at the cut-off. Note that this is more expensive to calculate than a plain cut-off and it is not required for energy conservation, since `Potential-shift` conserves energy just as well.

Potential-switch
Smoothly switches the potential to zero between `rvdw-switch` and `rvdw`. Note that this introduces articifically large forces in the switching region and is much more expensive to calculate. This option should only be used if the force field you are using requires this.

rvdw-switch: (0) [nm]
where to start switching the LJ force and possibly the potential, only relevant when force or potential switching is used

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

7.3.12 Tables

table-extension: (1) [nm]
Extension of the non-bonded potential lookup tables beyond the largest cut-off distance. The value should be large enough to account for charge group sizes and the diffusion between neighbor-list updates. Without user defined potential the same table length is used for the lookup tables for the 1-4 interactions, which are always tabulated irrespective of the use of tables for the non-bonded interactions. The value of `table-extension` in no way affects the values of `rlist`, `rcoulomb`, or `rvdw`.

energygrp-table:
When user tables are used for electrostatics and/or VdW, here one can give pairs of energy groups for which seperate user tables should be used. The two energy groups will be appended to the table file name, in order of their definition in `energygrps`, seperated by underscores. For example, if `energygrps = Na Cl Sol` and `energygrp-table = Na Na Na Cl`, `mdrun` will read `table_Na_Na.xvg` and `table_Na_Cl.xvg` in addition to the normal `table.xvg` which will be used for all other energy group pairs.

7.3.13 Ewald

fourierspacing: (0.12) [nm]
For ordinary Ewald, the ratio of the box dimensions and the spacing determines a lower bound for the number of wave vectors to use in each (signed) direction. For PME and P3M, that ratio determines a lower bound for the number of Fourier-space grid points that will be used along that axis. In all cases, the number for each direction can be overridden by entering a non-zero value for `fourier_n[xyz]`. For optimizing the relative load of the particle-particle interactions and the mesh part of PME, it is useful to know that the accuracy of the electrostatics remains nearly constant when the Coulomb cut-off and the PME grid spacing are scaled by the same factor.

fourier-nx (0) ; fourier-ny (0) ; fourier-nz: (0)
Highest magnitude of wave vectors in reciprocal space when using Ewald. Grid size when using PME or P3M. These values override `fourierspacing` per direction. The best choice is powers of 2, 3, 5 and 7. Avoid large primes.

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 `rcoulomb` 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.

ewald-rtol-lj (1e-3)
When doing PME for VdW-interactions, `ewald-rtol-lj` is used to control the relative strength of the dispersion potential at `rvdw` in the same way as `ewald-rtol` controls the electrostatic potential.

lj-pme-comb-rule (Geometric)
The combination rules used to combine VdW-parameters in the reciprocal part of LJ-PME. Geometric rules are much faster than Lorentz-Berthelot and usually the recommended choice, even when the rest of the force field uses the Lorentz-Berthelot rules.

Geometric
Apply geometric combination rules

Lorentz-Berthelot
Apply Lorentz-Berthelot combination rules

ewald-geometry: (3d)

3d
The Ewald sum is performed in all three dimensions.

3dc
The reciprocal sum is still performed in 3D, but a force and potential correction applied in the `z` dimension to produce a pseudo-2D summation. If your system has a slab geometry in the `x-y` plane you can try to increase the `z`-dimension of the box (a box height of 3 times the slab height is usually ok) and use this option.

epsilon-surface: (0)
This controls the dipole correction to the Ewald summation in 3D. The default value of zero means it is turned off. Turn it on by setting it to the value of the relative permittivity of the imaginary surface around your infinite system. Be careful - you shouldn't use this if you have free mobile charges in your system. This value does not affect the slab 3DC variant of the long range corrections.

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.

7.3.14 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 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. For NVT simulations the conserved energy quantity is written to energy and log file.

andersen
Temperature coupling by randomizing a fraction of the particles at each timestep. Reference temperature and coupling groups are selected as above. `tau-t` is the average time between randomization of each molecule. Inhibits particle dynamics somewhat, but little or no ergodicity issues. Currently only implemented with velocity Verlet, and not implemented with constraints.

andersen-massive
Temperature coupling by randomizing all particles at infrequent timesteps. Reference temperature and coupling groups are selected as above. `tau-t` is the time between randomization of all molecules. Inhibits particle dynamics somewhat, but little or no ergodicity issues. Currently only implemented with velocity Verlet.

v-rescale
Temperature coupling using velocity rescaling with a stochastic term (JCP 126, 014101).

This thermostat is similar to Berendsen coupling, with the same scaling using `tau-t`, but the stochastic term ensures that a proper canonical ensemble is generated. The random seed is set with `ld-seed`. This thermostat works correctly even for `tau-t=0`. For NVT simulations the conserved energy quantity is written to the energy and log file.

nsttcouple: (-1)
The frequency for coupling the temperature. The default value of -1 sets `nsttcouple` equal to `nstlist`, unless `nstlist`$\leq$0, then a value of 10 is used. For velocity Verlet integrators `nsttcouple` is set to 1.

nh-chain-length (10)
the number of chained Nose-Hoover thermostats for velocity Verlet integrators, the leap-frog `md` integrator only supports 1. Data for the NH chain variables is not printed to the .edr, but can be using the `GMX_NOSEHOOVER_CHAINS` environment variable

tc-grps:
groups to couple separately to temperature bath

tau-t: [ps]
time constant for coupling (one for each group in `tc-grps`), -1 means no temperature coupling

ref-t: [K]
reference temperature for coupling (one for each group in `tc-grps`)

7.3.15 Pressure coupling

pcoupl:

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.

Parrinello-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. For simulations where the exact fluctation of the NPT ensemble are important, or if the pressure coupling time is very short it may not be appropriate, as the previous time step pressure is used in some steps of the GROMACS implementation for the current time step pressure.

MTTK
Martyna-Tuckerman-Tobias-Klein implementation, only useable with `md-vv` or `md-vv-avek`, very similar to Parrinello-Rahman. 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. Currently only supports isotropic scaling.

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 convergence of the surface-tension, it can also be set to zero to have a box with constant height.

nstpcouple: (-1)
The frequency for coupling the pressure. The default value of -1 sets `nstpcouple` equal to `nstlist`, unless `nstlist` $\leq$0, then a value of 10 is used. For velocity Verlet integrators `nstpcouple` is set to 1.

tau-p: (1) [ps]
time constant for coupling

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

refcoord-scaling:

no
The reference coordinates for position restraints are not modified. Note that with this option the virial and pressure will depend on the absolute positions of the reference coordinates.

all
The reference coordinates are scaled with the scaling matrix of the pressure coupling.

com
Scale the center of mass of the reference coordinates with the scaling matrix of the pressure coupling. The vectors of each reference coordinate to the center of mass are not scaled. Only one COM is used, even when there are multiple molecules with position restraints. For calculating the COM of the reference coordinates in the starting configuration, periodic boundary conditions are not taken into account.

7.3.16 Simulated annealing

Simulated annealing is controlled separately for each temperature group in GROMACS. The reference temperature is a piecewise linear function, but you can use an arbitrary number of points for each group, and choose either a single sequence or a periodic behaviour for each group. The actual annealing is performed by dynamically changing the reference temperature used in the thermostat algorithm selected, so remember that the system will usually not instantaneously reach the reference temperature!

annealing:
Type of annealing for each temperature group

no
No simulated annealing - just couple to reference temperature value.

single
A single sequence of annealing points. If your simulation is longer than the time of the last point, the temperature will be coupled to this constant value after the annealing sequence has reached the last time point.

periodic
The annealing will start over at the first reference point once the last reference time is reached. This is repeated until the simulation ends.

annealing-npoints:
A list with the number of annealing reference/control points used for each temperature group. Use 0 for groups that are not annealed. The number of entries should equal the number of temperature groups.

annealing-time:
List of times at the annealing reference/control points for each group. If you are using periodic annealing, the times will be used modulo the last value, *i.e.* if the values are 0, 5, 10, and 15, the coupling will restart at the 0ps value after 15ps, 30ps, 45ps, etc. The number of entries should equal the sum of the numbers given in `annealing-npoints`.

annealing-temp:
List of temperatures at the annealing reference/control points for each group. The number of entries should equal the sum of the numbers given in `annealing-npoints`.

Confused? OK, let's use an example. Assume you have two temperature groups, set the group selections to `annealing = single periodic`, the number of points of each group to `annealing-npoints = 3 4`, the times to `annealing-time = 0 3 6 0 2 4 6` and finally temperatures to `annealing-temp = 298 280 270 298 320 320 298`. The first group will be coupled to 298K at 0ps, but the reference temperature will drop linearly to reach 280K at 3ps, and then linearly between 280K and 270K from 3ps to 6ps. After this is stays constant, at 270K. The second group is coupled to 298K at 0ps, it increases linearly to 320K at 2ps, where it stays constant until 4ps. Between 4ps and 6ps it decreases to 298K, and then it starts over with the same pattern again, *i.e.* rising linearly from 298K to 320K between 6ps and 8ps. Check the summary printed by `grompp` if you are unsure!

7.3.17 Velocity generation

gen-vel:

no
Do not generate velocities. The velocities are set to zero when there are no velocities in the input structure file.

yes
Generate velocities in `grompp` 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: (-1) [integer]
used to initialize random generator for random velocities, when `gen-seed` is set to -1, a pseudo random seed is used.

7.3.18 Bonds

constraints:

none
No constraints except for those defined explicitly in the topology, *i.e.* bonds are represented by a harmonic (or other) potential or a Morse potential (depending on the setting of `morse`) and angles by a harmonic (or other) potential.

h-bonds
Convert the bonds with H-atoms to constraints.

all-bonds
Convert all bonds to constraints.

h-angles
Convert all bonds and additionally the angles that involve H-atoms to bond-constraints.

all-angles
Convert all bonds and angles to bond-constraints.

constraint-algorithm:

LINCS
LINear Constraint Solver. With domain decomposition the parallel version P-LINCS is used. The accuracy in set with `lincs-order`, which sets the number of matrices in the expansion for the matrix inversion. After the matrix inversion correction the algorithm does an iterative correction to compensate for lengthening due to rotation. The number of such iterations can be controlled with `lincs-iter`. The root mean square relative constraint deviation 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 slightly 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. SHAKE does not support constraints between atoms on different nodes, thus it can not be used with domain decompositon when inter charge-group constraints are present. SHAKE can not be used with energy minimization.

continuation:
This option was formerly known as `unconstrained-start`.

no
apply constraints to the start configuration and reset shells

yes
do not apply constraints to the start configuration and do not reset shells, useful for exact coninuation and reruns

shake-tol: (0.0001)
relative tolerance for SHAKE

lincs-order: (4)
Highest order in the expansion of the constraint coupling matrix. When constraints form triangles, an additional expansion of the same order is applied on top of the normal expansion only for the couplings within such triangles. For "normal" MD simulations an order of 4 usually suffices, 6 is needed for large time-steps with virtual sites or BD. For accurate energy minimization an order of 8 or more might be required. With domain decomposition, the cell size is limited by the distance spanned by `lincs-order`+1 constraints. When one wants to scale further than this limit, one can decrease `lincs-order` and increase `lincs-iter`, since the accuracy does not deteriorate when (1+`lincs-iter`)*`lincs-order` remains constant.

lincs-iter: (1)
Number of iterations to correct for rotational lengthening in LINCS. For normal runs a single step is sufficient, but for NVE runs where you want to conserve energy accurately or for accurate energy minimization you might want to increase it to 2.

lincs-warnangle: (30) [degrees]
maximum angle that a bond can rotate before LINCS will complain

morse:

no
bonds are represented by a harmonic potential

yes
bonds are represented by a Morse potential

7.3.19 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

```
energygrp-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.

7.3.20 Walls

nwall: 0
When set to `1` there is a wall at `z=0`, when set to `2` there is also a wall at `z=z-box`. Walls can only be used with `pbc=xy`. When set to `2` pressure coupling and Ewald summation can be used (it is usually best to use semiisotropic pressure coupling with the `x/y` compressibility set to 0, as otherwise the surface area will change). Walls interact wit the rest of the system through an optional `wall-atomtype`. Energy groups `wall0` and `wall1` (for `nwall=2`) are added automatically to monitor the interaction of energy groups with each wall. The center of mass motion removal will be turned off in the `z`-direction.

wall-atomtype:
the atom type name in the force field for each wall. By (for example) defining a special wall atom type in the topology with its own combination rules, this allows for independent tuning of the interaction of each atomtype with the walls.

wall-type:

9-3
LJ integrated over the volume behind the wall: 9-3 potential

10-4
LJ integrated over the wall surface: 10-4 potential

12-6
direct LJ potential with the z distance from the wall

table
user defined potentials indexed with the z distance from the wall, the tables are read analogously to the `energygrp-table` option, where the first name is for a "normal" energy group and the second name is `wall0` or `wall1`, only the dispersion and repulsion columns are used

wall-r-linpot: -1 (nm)
Below this distance from the wall the potential is continued linearly and thus the force is constant. Setting this option to a postive value is especially useful for equilibration when some atoms are beyond a wall. When the value is ≤ 0 (<0 for `wall-type=table`), a fatal error is generated when atoms are beyond a wall.

wall-density: [nm^{-3}/nm^{-2}]
the number density of the atoms for each wall for wall types `9-3` and `10-4`

wall-ewald-zfac: 3
The scaling factor for the third box vector for Ewald summation only, the minimum is 2. Ewald summation can only be used with `nwall=2`, where one should use `ewald-geometry=3dc`. The empty layer in the box serves to decrease the unphysical Coulomb interaction between periodic images.

7.3.21 COM pulling

pull:

no
No center of mass pulling. All the following pull options will be ignored (and if present in the `.mdp` file, they unfortunately generate warnings)

umbrella
Center of mass pulling using an umbrella potential between the reference group and one or more groups.

constraint
Center of mass pulling using a constraint between the reference group and one or more groups. The setup is identical to the option `umbrella`, except for the fact that a rigid constraint is applied instead of a harmonic potential.

constant-force
Center of mass pulling using a linear potential and therefore a constant force. For this option there is no reference position and therefore the parameters `pull-init` and `pull-rate` are not used.

pull-geometry:

distance
Pull along the vector connecting the two groups. Components can be selected with `pull-dim`.

direction
Pull in the direction of `pull-vec`.

direction-periodic
As `direction`, but allows the distance to be larger than half the box size. With this geometry the box should not be dynamic (*e.g.* no pressure scaling) in the pull dimensions and the pull force is not added to virial.

cylinder
Designed for pulling with respect to a layer where the reference COM is given by a local cylindrical part of the reference group. The pulling is in the direction of `pull-vec`. From the reference group a cylinder is selected around the axis going through the pull group with direction `pull-vec` using two radii. The radius `pull-r1` gives the radius within which all the relative weights are one, between `pull-r1` and `pull-r0` the weights are switched to zero. Mass weighting is also used. Note that the radii should be smaller than half the box size. For tilted cylinders they should be even smaller than half the box size since the distance of an atom in the reference group from the COM of the pull group has both a radial and an axial component.

pull-dim: (Y Y Y)
the distance components to be used with geometry `distance`, and also sets which components are printed to the output files

pull-r1: (1) [nm]
the inner radius of the cylinder for geometry `cylinder`

pull-r0: (1) [nm]
the outer radius of the cylinder for geometry `cylinder`

pull-constr-tol: (1e-6)
the relative constraint tolerance for constraint pulling

pull-start:

no
do not modify `pull-init`

yes
add the COM distance of the starting conformation to `pull-init`

pull-print-reference: (10)

no
do not print the COM of the first group in each pull coordinate

yes
print the COM of the first group in each pull coordinate

pull-nstxout: (10)
frequency for writing out the COMs of all the pull group

pull-nstfout: (1)
frequency for writing out the force of all the pulled group

pull-ngroups: (1)
The number of pull groups, not including the absolute reference group, when used. Pull groups can be reused in multiple pull coordinates. Below only the pull options for group 1 are given, further groups simply increase the group index number.

pull-ncoords: (1)
The number of pull coordinates. Below only the pull options for coordinate 1 are given, further coordinates simply increase the coordinate index number.

pull-group1-name:
The name of the pull group, is looked up in the index file or in the default groups to obtain the atoms involved.

pull-group1-weights:
Optional relative weights which are multiplied with the masses of the atoms to give the total weight for the COM. The number should be 0, meaning all 1, or the number of atoms in the pull group.

pull-group1-pbcatom: (0)
The reference atom for the treatment of periodic boundary conditions inside the group (this has no effect on the treatment of the pbc between groups). This option is only important when the diameter of the pull group is larger than half the shortest box vector. For determining the COM, all atoms in the group are put at their periodic image which is closest to `pull-group1-pbcatom`. A value of 0 means that the middle atom (number wise) is used. This parameter is not used with geometry `cylinder`. A value of -1 turns on cosine weighting, which is useful for a group of molecules in a periodic system, *e.g.* a water slab (see Engin et al. J. Chem. Phys. B 2010).

pull-coord1-groups:
The two groups indices should be given on which this pull coordinate will operate. The first index can be 0, in which case an absolute reference of `pull-coord1-origin` is used. With an absolute reference the system is no longer translation invariant and one should think about what to do with the center of mass motion.

pull-coord1-origin: (0.0 0.0 0.0)
The pull reference position for use with an absolute reference.

pull-coord1-vec: (0.0 0.0 0.0)
The pull direction. `grompp` normalizes the vector.

pull-coord1-init: (0.0) [nm]
The reference distance at t=0.

pull-coord1-rate: (0) [nm/ps]
The rate of change of the reference position.

pull-coord1-k: (0) [kJ mol^{-1} nm^{-2} / [kJ mol^{-1} nm^{-1}]]
The force constant. For umbrella pulling this is the harmonic force constant in [kJ mol^{-1} nm^{-2}]. For constant force pulling this is the force constant of the linear potential, and thus minus (!) the constant force in [kJ mol^{-1} nm^{-1}].

pull-coord1-kB: (pull-k1) [kJ mol^{-1} nm^{-2} / [kJ mol^{-1} nm^{-1}]]
As `pull-coord1-k`, but for state B. This is only used when `free-energy` is turned on. The force constant is then (1 - lambda)*`pull-coord1-k` + lambda*`pull-coord1-kB`.

7.3.22 NMR refinement

disre:

no
ignore distance restraint information in topology file

simple
simple (per-molecule) distance restraints.

ensemble
distance restraints over an ensemble of molecules in one simulation box. Normally, one would perform ensemble averaging over multiple subsystems, each in a separate box, using `mdrun -multi`;s upply `topol0.tpr`, `topol1.tpr`, ... with different coordinates and/or velocities. The environment variable `GMX_DISRE_ENSEMBLE_SIZE` sets the number of systems within each ensemble (usually equal to the `mdrun -multi` value).

disre-weighting:

equal (default)
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. The forces are conservative when `disre-tau` is zero.

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 product of the time-averaged violation and the instantaneous violation

disre-fc: (1000) [kJ mol^{-1} nm^{-2}]
force constant for distance restraints, which is multiplied by a (possibly) different factor for each restraint given in the `fac` column of the interaction in the topology file.

disre-tau: (0) [ps]
time constant for distance restraints running average. A value of zero turns off time averaging.

nstdisreout: (100) [steps]
period between steps when the running time-averaged and instantaneous distances of all atom pairs involved in restraints are written to the energy file (can make the energy file very large)

orire:

no
ignore orientation restraint information in topology file

yes
use orientation restraints, ensemble averaging can be performed with `mdrun -multi`

orire-fc: (0) [kJ mol]
force constant for orientation restraints, which is multiplied by a (possibly) different weight factor for each restraint, can be set to zero to obtain the orientations from a free simulation

orire-tau: (0) [ps]
time constant for orientation restraints running average. A value of zero turns off time averaging.

orire-fitgrp:
fit group for orientation restraining. This group of atoms is used to determine the rotation `R` of the system with respect to the reference orientation. The reference orientation is the starting conformation of the first subsystem. For a protein, backbone is a reasonable choice

nstorireout: (100) [steps]
period between steps when the running time-averaged and instantaneous orientations for all restraints, and the molecular order tensor are written to the energy file (can make the energy file very large)

7.3.23 Free energy calculations

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 (as specified with `dhdl-derivatives`), or the Hamiltonian differences with respect to other lambda values (as specified with `foreign-lambda`) to the energy file and/or to `dhdl.xvg`, where they can be processed by, for example `g_bar`. 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.

expanded
Turns on expanded ensemble simulation, where the alchemical state becomes a dynamic variable, allowing jumping between different Hamiltonians. See the expanded ensemble options for controlling how expanded ensemble simulations are performed. The different Hamiltonians used in expanded ensemble simulations are defined by the other free energy options.

init-lambda: (-1)
starting value for lambda (float). Generally, this should only be used with slow growth (*i.e.* nonzero `delta-lambda`). In other cases, `init-lambda-state` should be specified instead. Must be greater than or equal to 0.

delta-lambda: (0)
increment per time step for lambda

init-lambda-state: (-1)
starting value for the lambda state (integer). Specifies which columm of the lambda vector (`coul-lambdas`, `vdw-lambdas`, `bonded-lambdas`, `restraint-lambdas`, `mass-lambdas`, `temperature-lambdas`, `fep-lambdas`) should be used. This is a zero-based index: `init-lambda-state` 0 means the first column, and so on.

fep-lambdas: ()
Zero, one or more lambda values for which Delta H values will be determined and written to dhdl.xvg every `nstdhdl` steps. Values must be between 0 and 1. Free energy differences between different lambda values can then be determined with `g_bar`. `fep-lambdas` is different from the other -lambdas keywords because all components of the lambda vector that are not specified will use `fep-lambdas` (including restraint-lambdas and therefore the pull code restraints).

coul-lambdas: ()
Zero, one or more lambda values for which Delta H values will be determined and written to dhdl.xvg every `nstdhdl` steps. Values must be between 0 and 1. Only the electrostatic interactions are controlled with this component of the lambda vector (and only if the lambda=0 and lambda=1 states have differing electrostatic interactions).

vdw-lambdas: ()
Zero, one or more lambda values for which Delta H values will be determined and written to dhdl.xvg every `nstdhdl` steps. Values must be between 0 and 1. Only the van der Waals interactions are controlled with this component of the lambda vector.

bonded-lambdas: ()
Zero, one or more lambda values for which Delta H values will be determined and written to dhdl.xvg every `nstdhdl` steps. Values must be between 0 and 1. Only the bonded interactions are controlled with this component of the lambda vector.

restraint-lambdas: ()
Zero, one or more lambda values for which Delta H values will be determined and written to dhdl.xvg every `nstdhdl` steps. Values must be between 0 and 1. Only the restraint interactions: dihedral restraints, and the pull code restraints are controlled with this component of the lambda vector.

mass-lambdas: ()
Zero, one or more lambda values for which Delta H values will be determined and written to dhdl.xvg every `nstdhdl` steps. Values must be between 0 and 1. Only the particle masses are controlled with this component of the lambda vector.

temperature-lambdas: ()
Zero, one or more lambda values for which Delta H values will be determined and written to dhdl.xvg every `nstdhdl` steps. Values must be between 0 and 1. Only the temperatures controlled with this component of the lambda vector. Note that these lambdas should not be used for replica exchange, only for simulated tempering.

calc-lambda-neighbors (1)
Controls the number of lambda values for which Delta H values will be calculated and written out, if `init-lambda-state` has been set. A positive value will limit the number of lambda points calculated to only the nth neighbors of `init-lambda-state`: for example, if `init-lambda-state` is 5 and this parameter has a value of 2, energies for lambda points 3-7 will be calculated and writen out. A value of -1 means all lambda points will be written out. For normal BAR such as with g_bar, a value of 1 is sufficient, while for MBAR -1 should be used.

sc-alpha: (0)
the soft-core alpha parameter, a value of 0 results in linear interpolation of the LJ and Coulomb interactions

sc-r-power: (6)
the power of the radial term in the soft-core equation. Possible values are 6 and 48. 6 is more standard, and is the default. When 48 is used, then sc-alpha should generally be much lower (between 0.001 and 0.003).

sc-coul: (no)
Whether to apply the soft core free energy interaction transformation to the Columbic interaction of a molecule. Default is no, as it is generally more efficient to turn off the Coulomic interactions linearly before turning off the van der Waals interactions.

sc-power: (0)
the power for lambda in the soft-core function, only the values 1 and 2 are supported

sc-sigma: (0.3) [nm]
the soft-core sigma for particles which have a C6 or C12 parameter equal to zero or a sigma smaller than `sc-sigma`

couple-moltype:
Here one can supply a molecule type (as defined in the topology) for calculating solvation or coupling free energies. There is a special option `system` that couples all molecule types in the system. This can be useful for equilibrating a system starting from (nearly) random coordinates. `free-energy` has to be turned on. The Van der Waals interactions and/or charges in this molecule type can be turned on or off between lambda=0 and lambda=1, depending on the settings of `couple-lambda0` and `couple-lambda1`. If you want to

decouple one of several copies of a molecule, you need to copy and rename the molecule definition in the topology.

couple-lambda0:

vdw-q
all interactions are on at lambda=0

vdw
the charges are zero (no Coulomb interactions) at lambda=0

q
the Van der Waals interactions are turned at lambda=0; soft-core interactions will be required to avoid singularities

none
the Van der Waals interactions are turned off and the charges are zero at lambda=0; soft-core interactions will be required to avoid singularities.

couple-lambda1:
analogous to `couple-lambda1`, but for lambda=1

couple-intramol:

no
All intra-molecular non-bonded interactions for moleculetype `couple-moltype` are replaced by exclusions and explicit pair interactions. In this manner the decoupled state of the molecule corresponds to the proper vacuum state without periodicity effects.

yes
The intra-molecular Van der Waals and Coulomb interactions are also turned on/off. This can be useful for partitioning free-energies of relatively large molecules, where the intra-molecular non-bonded interactions might lead to kinetically trapped vacuum conformations. The 1-4 pair interactions are not turned off.

nstdhdl: (100)
the frequency for writing dH/dlambda and possibly Delta H to dhdl.xvg, 0 means no ouput, should be a multiple of `nstcalcenergy`.

dhdl-derivatives: (yes)
If yes (the default), the derivatives of the Hamiltonian with respect to lambda at each `nstdhdl` step are written out. These values are needed for interpolation of linear energy differences with `g_bar` (although the same can also be achieved with the right `foreign lambda` setting, that may not be as flexible), or with thermodynamic integration

dhdl-print-energy: (no)
Include either the total or the potential energy in the dhdl file. Options are 'no', 'potential', or 'total'. This information is needed for later free energy analysis if the states of interest are at different temperatures. If all states are at the same temperature, this information is not needed. 'potential' is useful in case one is using `mdrun -rerun` to generate the

dhdl.xvg file. When rerunning from an existing trajectory, the kinetic energy will often not be correct, and thus one must compute the residual free energy from the potential alone, with the kinetic energy component computed analytically.

separate-dhdl-file: (yes)

yes
the free energy values that are calculated (as specified with the foreign-lambda and dhdl-derivatives settings) are written out to a separate file, with the default name dhdl.xvg. This file can be used directly with g_bar.

no
The free energy values are written out to the energy output file (ener.edr, in accumulated blocks at every nstenergy steps), where they can be extracted with g_energy or used directly with g_bar.

dh-hist-size: (0)
If nonzero, specifies the size of the histogram into which the Delta H values (specified with foreign-lambda) and the derivative dH/dl values are binned, and written to ener.edr. This can be used to save disk space while calculating free energy differences. One histogram gets written for each foreign lambda and two for the dH/dl, at every nstenergy step. Be aware that incorrect histogram settings (too small size or too wide bins) can introduce errors. Do not use histograms unless you're certain you need it.

dh-hist-spacing (0.1)
Specifies the bin width of the histograms, in energy units. Used in conjunction with dh-hist-size. This size limits the accuracy with which free energies can be calculated. Do not use histograms unless you're certain you need it.

7.3.24 Expanded Ensemble calculations

nstexpanded
The number of integration steps beween attempted moves changing the system Hamiltonian in expanded ensemble simulations. Must be a multiple of nstcalcenergy, but can be greater or less than nstdhdl.

lmc-stats:

no
No Monte Carlo in state space is performed.

metropolis-transition
Uses the Metropolis weights to update the expanded ensemble weight of each state. Min1,exp(-(beta_new u_new - beta_old u_old)

barker-transition
Uses the Barker transition critera to update the expanded ensemble weight of each state i, defined by exp(-beta_new u_new)/[exp(-beta_new u_new)+exp(-beta_old u_old)

wang-landau
Uses the Wang-Landau algorithm (in state space, not energy space) to update the expanded ensemble weights.

min-variance
Uses the minimum variance updating method of Escobedo et al. to update the expanded ensemble weights. Weights will not be the free energies, but will rather emphasize states that need more sampling to give even uncertainty.

lmc-mc-move:

no
No Monte Carlo in state space is performed.

metropolis-transition
Randomly chooses a new state up or down, then uses the Metropolis critera to decide whether to accept or reject: Min1,exp(-(beta_new u_new - beta_old u_old)

barker-transition
Randomly chooses a new state up or down, then uses the Barker transition critera to decide whether to accept or reject: exp(-beta_new u_new)/[exp(-beta_new u_new)+exp(-beta_old u_old)]

gibbs
Uses the conditional weights of the state given the coordinate (exp(-beta_i u_i) / sum_k exp(beta_i u_i) to decide which state to move to.

metropolized-gibbs

Uses the conditional weights of the state given the coordinate (exp(-beta_i u_i) / sum_k exp(beta_i u_i) to decide which state to move to, EXCLUDING the current state, then uses a rejection step to ensure detailed balance. Always more efficient that Gibbs, though only marginally so in many situations, such as when only the nearest neighbors have decent phase space overlap.

lmc-seed: (-1)
random seed to use for Monte Carlo moves in state space. When `lmc-seed` is set to -1, a pseudo random seed is us

mc-temperature:
Temperature used for acceptance/rejection for Monte Carlo moves. If not specified, the temperature of the simulation specified in the first group of `ref_t` is used.

wl-ratio: (0.8)
The cutoff for the histogram of state occupancies to be reset, and the free energy incrementor to be reset as delta -¿ delta*wl-scale. If we define the Nratio = (number of samples at each histogram) / (average number of samples at each histogram). `wl-ratio` of 0.8 means that means that the histogram is only considered flat if all Nratio $>$ 0.8 AND simultaneously all 1/Nratio $>$ 0.8.

wl-scale: (0.8)
Each time the histogram is considered flat, then the current value of the Wang-Landau incrementor for the free energies is multiplied by `wl-scale`. Value must be between 0 and 1.

init-wl-delta: (1.0)
The initial value of the Wang-Landau incrementor in kT. Some value near 1 kT is usually most efficient, though sometimes a value of 2-3 in units of kT works better if the free energy differences are large.

wl-oneovert: (no)
Set Wang-Landau incrementor to scale with 1/(simulation time) in the large sample limit. There is significant evidence that the standard Wang-Landau algorithms in state space presented here result in free energies getting 'burned in' to incorrect values that depend on the initial state. when `wl-oneovert` is true, then when the incrementor becomes less than 1/N, where N is the mumber of samples collected (and thus proportional to the data collection time, hence '1 over t'), then the Wang-Lambda incrementor is set to 1/N, decreasing every step. Once this occurs, `wl-ratio` is ignored, but the weights will still stop updating when the equilibration criteria set in `lmc-weights-equil` is achieved.

lmc-repeats: (1)
Controls the number of times that each Monte Carlo swap type is performed each iteration. In the limit of large numbers of Monte Carlo repeats, then all methods converge to Gibbs sampling. The value will generally not need to be different from 1.

lmc-gibbsdelta: (-1)
Limit Gibbs sampling to selected numbers of neighboring states. For Gibbs sampling, it is sometimes inefficient to perform Gibbs sampling over all of the states that are defined. A positive value of `lmc-gibbsdelta` means that only states plus or minus `lmc-gibbsdelta` are considered in exchanges up and down. A value of -1 means that all states are considered. For less than 100 states, it is probably not that expensive to include all states.

lmc-forced-nstart: (0)
Force initial state space sampling to generate weights. In order to come up with reasonable initial weights, this setting allows the simulation to drive from the initial to the final lambda state, with `lmc-forced-nstart` steps at each state before moving on to the next lambda state. If `lmc-forced-nstart` is sufficiently long (thousands of steps, perhaps), then the weights will be close to correct. However, in most cases, it is probably better to simply run the standard weight equilibration algorithms.

nst-transition-matrix: (-1)
Frequency of outputting the expanded ensemble transition matrix. A negative number means it will only be printed at the end of the simulation.

symmetrized-transition-matrix: (no)
Whether to symmetrize the empirical transition matrix. In the infinite limit the matrix will be symmetric, but will diverge with statistical noise for short timescales. Forced symmetrization, by using the matrix T_sym = 1/2 (T + transpose(T)), removes problems like the existence of (small magnitude) negative eigenvalues.

mininum-var-min: (100)
The `min-variance` strategy (option of `lmc-stats` is only valid for larger number of samples, and can get stuck if too few samples are used at each state. `mininum-var-min` is the minimum number of samples that each state that are allowed before the `min-variance` strategy is activated if selected.

init-lambda-weights:
The initial weights (free energies) used for the expanded ensemble states. Default is a vector of zero weights. format is similar to the lambda vector settings in `fep-lambdas`, except the weights can be any floating point number. Units are kT. Its length must match the lambda vector lengths.

lmc-weights-equil: (no)

no
Expanded ensemble weights continue to be updated throughout the simulation.

yes
The input expanded ensemble weights are treated as equilibrated, and are not updated throughout the simulation.

wl-delta
Expanded ensemble weight updating is stopped when the Wang-Landau incrementor falls below the value specified by `weight-equil-wl-delta`.

number-all-lambda
Expanded ensemble weight updating is stopped when the number of samples at all of the lambda states is greater than the value specified by `weight-equil-number-all-lambda`.

number-steps
Expanded ensemble weight updating is stopped when the number of steps is greater than the level specified by `weight-equil-number-steps`.

number-samples
Expanded ensemble weight updating is stopped when the number of total samples across all lambda states is greater than the level specified by `weight-equil-number-samples`.

count-ratio
Expanded ensemble weight updating is stopped when the ratio of samples at the least sampled lambda state and most sampled lambda state greater than the value specified by `weight-equil-count-ratio`.

simulated-tempering: (no)
Turn simulated tempering on or off. Simulated tempering is implemented as expanded ensemble sampling with different temperatures instead of different Hamiltonians.

sim-temp-low: (300)
Low temperature for simulated tempering.

sim-temp-high: (300)
High temperature for simulated tempering.

simulated-tempering-scaling: (linear)
Controls the way that the temperatures at intermediate lambdas are calculated from the temperature-lambda part of the lambda vector.

linear
Linearly interpolates the temperatures using the values of temperature-lambda,*i.e.* if sim-temp-low=300, sim-temp-high=400, then lambda=0.5 correspond to a temperature of 350. A nonlinear set of temperatures can always be implemented with uneven spacing in lambda.

geometric
Interpolates temperatures geometrically between sim-temp-low and sim-temp-high. The i:th state has temperature sim-temp-low * (sim-temp-high/sim-temp-low) raised to the power of (i/(ntemps-1)). This should give roughly equal exchange for constant heat capacity, though of course things simulations that involve protein folding have very high heat capacity peaks.

exponential
Interpolates temperatures exponentially between sim-temp-low and sim-temp-high. The ith state has temperature sim-temp-low + (sim-temp-high-sim-temp-low)*((exp(temperat 1)/(exp(1.0)-1)).

7.3.25 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).

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. To avoid spurious contibrutions to the virial and pressure due to large forces between completely frozen atoms you need to use energy group exclusions, this also saves computing time. Note that coordinates of frozen atoms are not scaled by pressure-coupling algorithms.

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.

deform: (0 0 0 0 0 0) [nm ps^{-1}]
The velocities of deformation for the box elements: a(x) b(y) c(z) b(x) c(x) c(y). Each step the box elements for which `deform` is non-zero are calculated as: box(ts)+(t-ts)*deform, off-diagonal elements are corrected for periodicity. The coordinates are transformed accordingly. Frozen degrees of freedom are (purposely) also transformed. The time ts is set to t at the first step and at steps at which x and v are written to trajectory to ensure exact restarts. Deformation can be used together with semiisotropic or anisotropic pressure coupling when the appropriate compressibilities are set to zero. The diagonal elements can be used to strain a solid. The off-diagonal elements can be used to shear a solid or a liquid.

7.3.26 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-`$\star$, 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

¡h3¿Mixed quantum/classical molecular dynamics¡!–QuietIdx¿QM/MM¡!–EQuietIdx–¿¡/h3¿

QMMM:

no
No QM/MM.

yes
Do a QM/MM simulation. Several groups can be described at different QM levels separately. These are specified in the `QMMM-grps` field separated by spaces. The level of ¡i¿ab initio¡/i¿ theory at which the groups are described is specified by `QMmethod` and `QMbasis` Fields. Describing the groups at different levels of theory is only possible with the ONIOM QM/MM scheme, specified by `QMMMscheme`.

QMMM-grps:
groups to be descibed at the QM level

QMMMscheme:

normal
normal QM/MM. There can only be one `QMMM-grps` that is modelled at the `QMmethod` and `QMbasis` level of *ab initio* theory. The rest of the system is described at the MM level. The QM and MM subsystems interact as follows: MM point charges are included in the QM one-electron hamiltonian and all Lennard-Jones interactions are described at the MM level.

ONIOM
The interaction between the subsystem is described using the ONIOM method by Morokuma and co-workers. There can be more than one `QMMM-grps` each modeled at a different level of QM theory (`QMmethod` and `QMbasis`).

QMmethod: (RHF)
Method used to compute the energy and gradients on the QM atoms. Available methods are AM1, PM3, RHF, UHF, DFT, B3LYP, MP2, CASSCF, and MMVB. For CASSCF, the number of electrons and orbitals included in the active space is specified by `CASelectrons` and `CASorbitals`.

QMbasis: (STO-3G)
Basis set used to expand the electronic wavefuntion. Only Gaussian basis sets are currently available, i.e. STO-3G, 3-21G, 3-21G*, 3-21+G*, 6-21G, 6-31G, 6-31G*, 6-31+G*, and 6-311G.

QMcharge: (0) [integer]
The total charge in `e` of the `QMMM-grps`. In case there are more than one `QMMM-grps`, the total charge of each ONIOM layer needs to be specified separately.

QMmult: (1) [integer]
The multiplicity of the `QMMM-grps`. In case there are more than one `QMMM-grps`, the multiplicity of each ONIOM layer needs to be specified separately.

CASorbitals: (0) [integer]
The number of orbitals to be included in the active space when doing a CASSCF computation.

CASelectrons: (0) [integer]
The number of electrons to be included in the active space when doing a CASSCF computation.

SH:

no
No surface hopping. The system is always in the electronic ground-state.

yes
Do a QM/MM MD simulation on the excited state-potential energy surface and enforce a *diabatic* hop to the ground-state when the system hits the conical intersection hyperline in the course the simulation. This option only works in combination with the CASSCF method.

7.3.27 Implicit solvent

implicit-solvent:

no
No implicit solvent

GBSA
Do a simulation with implicit solvent using the Generalized Born formalism. Three different methods for calculating the Born radii are available, Still, HCT and OBC. These are specified with the `gb-algorithm` field. The non-polar solvation is specified with the `sa-algorithm` field.

gb-algorithm:

Still
Use the Still method to calculate the Born radii

HCT
Use the Hawkins-Cramer-Truhlar method to calculate the Born radii

OBC
Use the Onufriev-Bashford-Case method to calculate the Born radii

nstgbradii: (1) [steps]
Frequency to (re)-calculate the Born radii. For most practial purposes, setting a value larger than 1 violates energy conservation and leads to unstable trajectories.

rgbradii: (1.0) [nm]
Cut-off for the calculation of the Born radii. Currently must be equal to rlist

gb-epsilon-solvent: (80)
Dielectric constant for the implicit solvent

gb-saltconc: (0) [M]
Salt concentration for implicit solvent models, currently not used

gb-obc-alpha (1); gb-obc-beta (0.8); gb-obc-gamma (4.85);
Scale factors for the OBC model. Default values are OBC(II). Values for OBC(I) are 0.8, 0 and 2.91 respectively

gb-dielectric-offset: (0.009) [nm]
Distance for the di-electric offset when calculating the Born radii. This is the offset between the center of each atom the center of the polarization energy for the corresponding atom

sa-algorithm

Ace-approximation
Use an Ace-type approximation (default)

None
No non-polar solvation calculation done. For GBSA only the polar part gets calculated

sa-surface-tension: [kJ mol^{-1} nm^{-2}]
Default value for surface tension with SA algorithms. The default value is -1; Note that if this default value is not changed it will be overridden by `grompp` using values that are specific for the choice of radii algorithm (0.0049 kcal/mol/Angstrom^2 for Still, 0.0054 kcal/mol/Angstrom^2 for HCT/OBC) Setting it to 0 will while using an sa-algorithm other than None means no non-polar calculations are done.

7.3.28 Adaptive Resolution Simulation

adress: (no)
Decide whether the AdResS feature is turned on.

adress-type: (Off)

Off
Do an AdResS simulation with weight equal 1, which is equivalent to an explicit (normal) MD simulation. The difference to disabled AdResS is that the AdResS variables are still read-in and hence are defined.

Constant
Do an AdResS simulation with a constant weight, `adress-const-wf` defines the value of the weight

XSplit
Do an AdResS simulation with simulation box split in x-direction, so basically the weight is only a function of the x coordinate and all distances are measured using the x coordinate only.

Sphere
Do an AdResS simulation with spherical explicit zone.

adress-const-wf: (1)
Provides the weight for a constant weight simulation (`adress-type`=Constant)

adress-ex-width: (0)
Width of the explicit zone, measured from `adress-reference-coords`.

adress-hy-width: (0)
Width of the hybrid zone.

adress-reference-coords: (0,0,0)
Position of the center of the explicit zone. Periodic boundary conditions apply for measuring the distance from it.

adress-cg-grp-names
The names of the coarse-grained energy groups. All other energy groups are considered explicit and their interactions will be automatically excluded with the coarse-grained groups.

adress-site: (COM) The mapping point from which the weight is calculated.

COM
The weight is calculated from the center of mass of each charge group.

COG
The weight is calculated from the center of geometry of each charge group.

Atom
The weight is calculated from the position of 1st atom of each charge group.

AtomPerAtom
The weight is calculated from the position of each individual atom.

adress-interface-correction: (Off)

Off
Do not a apply any interface correction.

thermoforce
Apply thermodynamic force interface correction. The table can be specified using the `-tabletf` option of `mdrun`. The table should contain the potential and force (acting on molecules) as function of the distance from `adress-reference-coords`.

adress-tf-grp-names
The names of the energy groups to which the `thermoforce` is applied if enabled in `adress-interface-correction`. If no group is given the default table is applied.

adress-ex-forcecap: (0)
Cap the force in the hybrid region, useful for big molecules. 0 disables force capping.

7.3.29 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 modify code. You can pass integers and reals to your subroutine. Check the inputrec definition in `src/include/types/inputrec.h`

Chapter 8

Analysis

In this chapter different ways of analyzing your trajectory are described. The names of the corresponding analysis programs are given. Specific information on the in- and output of these programs can be found in the online 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. 8.1.2 explains a newer concept of dynamic selections, which is currently supported by a few tools. Then, the different analysis tools are presented.

8.1 Using Groups

`gmx make_ndx, gmx mk_angndx, gmx select`

In chapter 3, it was explained how *groups of atoms* can be used in `mdrun` (see sec. 3.3). In most analysis programs, groups of atoms must also be chosen. Most programs can generate several default index groups, but groups can always be read from an index file. Let's consider the example of 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 of finding a B atom at 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.

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 `gmx make_ndx` or `gmx select`. To generate an index file with angles or dihedrals, use `gmx 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.

Each tool that can use groups will offer the available alternatives for the user to choose. That choice can be made with the number of the group, or its name. In fact, the first few letters of the group name will suffice if that will distinguish the group from all others. There are ways to use Unix shell features to choose group names on the command line, rather than interactively. Consult www.gromacs.org for suggestions.

8.1.1 Default Groups

When no index file is supplied to analysis tools or `grompp`, a number of default groups are 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 hydrogens and hydrogens on the N-terminus

`SideChain`
: protein side chain atoms; that is all atoms except N, C_α, C, O, backbone amide hydrogens, 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 virtual site constructions of NH_3 groups and tryptophan side-chains), 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

`RNA`
: all RNA atoms

`Water`
: water molecules (names like `SOL`, `WAT`, `HOH`, etc.) See `residuetypes.dat` for a full listing

`non-Water`
: anything not covered by the `Water` group

`Ion`
: any name matching an Ion entry in `residuetypes.dat`

`Water_and_Ions`
: combination of the `Water` and `Ions` groups

`molecule_name`
: for all residues/molecules which are not recognized as protein, DNA, or RNA; one group per residue/molecule name is generated

`Other`
: all atoms which are neither protein, DNA, nor RNA.

Empty groups will not be generated. Most of the groups only contain protein atoms. An atom is considered a protein atom if its residue name is listed in the `residuetypes.dat` file and is listed as a "Protein" entry. The process for determinding DNA, RNA, etc. is analogous. If you need to modify these classifications, then you can copy the file from the library directory into your working directory and edit the local copy.

8.1.2 Selections

`gmx select`
Currently, a few analysis tools support an extended concept of *(dynamic) selections*. There are three main differences to traditional index groups:

- The selections are specified as text instead of reading fixed atom indices from a file, using a syntax similar to VMD. The text can be entered interactively, provided on the command line, or from a file.
- The selections are not restricted to atoms, but can also specify that the analysis is to be performed on, e.g., center-of-mass positions of a group of atoms. Some tools may not support selections that do not evaluate to single atoms, e.g., if they require information that is available only for single atoms, like atom names or types.
- The selections can be dynamic, i.e., evaluate to different atoms for different trajectory frames. This allows analyzing only a subset of the system that satisfies some geometric criteria.

As an example of a simple selection, `resname ABC and within 2 of resname DEF` selects all atoms in residues named ABC that are within 2 nm of any atom in a residue named DEF.

Tools that accept selections can also use traditional index files similarly to older tools: it is possible to give an `.ndx` file to the tool, and directly select a group from the index file as a selection, either by group number or by group name. The index groups can also be used as a part of a more complicated selection.

To get started, you can run `gmx select` with a single structure, and use the interactive prompt to try out different selections. The tool provides, among others, output options `-on` and `-ofpdb` to write out the selected atoms to an index file and to a `.pdb` file, respectively. This does not allow testing selections that evaluate to center-of-mass positions, but other selections can be tested and the result examined.

The detailed syntax and the individual keywords that can be used in selections can be accessed by typing `help` in the interactive prompt of any selection-enabled tool, as well as with `gmx help selections`. The help is divided into subtopics that can be accessed with, e.g., `help syntax` / `gmx help selections syntax`. Some individual selection keywords have extended help as well, which can be accessed with, e.g., `help keywords within`.

The interactive prompt does not currently provide much editing capabilities. If you need them, you can run the program under `rlwrap`.

For tools that do not yet support the selection syntax, you can use `gmx select -on` to generate static index groups to pass to the tool. However, this only allows for a small subset (only the first bullet from the above list) of the flexibility that fully selection-aware tools offer.

It is also possible to write your own analysis tools to take advantage of the flexibility of these selections: see the `template.cpp` file in the `share/gromacs/template` directory of your installation for an example.

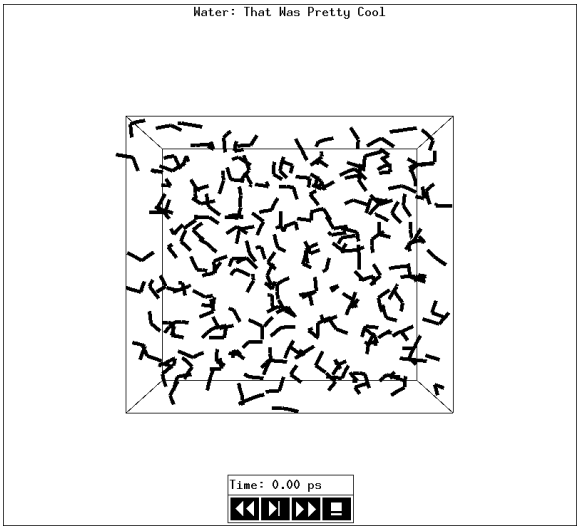


Figure 8.1: The window of `gmx view` showing a box of water.

8.2 Looking at your trajectory

`gmx view`
Before analyzing your trajectory it is often informative to look at your trajectory first. GROMACS comes with a simple trajectory viewer `gmx view`; the advantage with this one is that it does not require OpenGL, which usually isn't present on *e.g.* supercomputers. It is also possible to generate a hard-copy 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

`gmx energy, gmx traj`
To analyze some or all *energies* and other properties, such as *total pressure*, *pressure tensor*, *density*, *box-volume* and *box-sizes*, use the program `gmx energy`. A choice can be made from a list 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 `gmx traj -com -ov`. It is however recommended to remove the center-of-mass velocity every step (see chapter 3)!

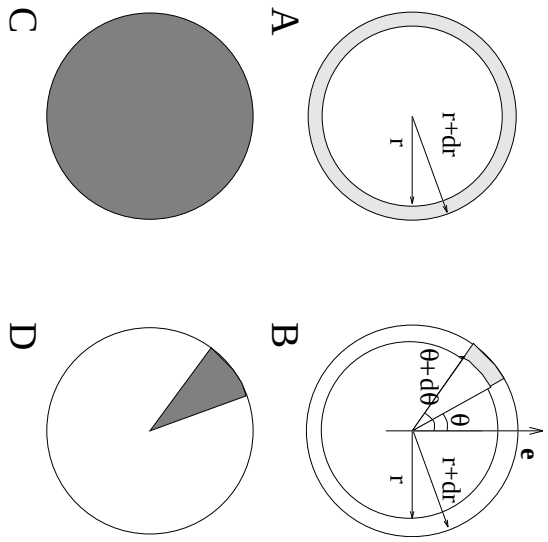


Figure 8.2: Definition of slices in `gmx rdf`: A. $g_{AB}(r)$. B. $g_{AB}(r,\theta)$. The slices are colored gray. C. Normalization $\langle\rho_B\rangle_{local}$. D. Normalization $\langle\rho_B\rangle_{local,\,\theta}$. Normalization volumes are colored gray.

8.4 Radial distribution functions

`gmx 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).

Usually the value of r_{max} is half of the box length. The averaging is also performed in time. In practice the analysis program `gmx 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 [81] is given in Fig. 8.3.

With `gmx 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.

$$g_{AB}(r,\theta) = \frac{1}{\langle\rho_B\rangle_{local,\,\theta}} \frac{1}{N_A} \sum_{i\in A}^{N_A} \sum_{j\in B}^{N_B} \frac{\delta(r_{ij}-r)\delta(\theta_{ij}-\theta)}{2\pi r^2 sin(\theta)} \tag{8.4}$$

$$cos(\theta_{ij}) = \frac{\mathbf{r}_{ij}\cdot\mathbf{e}}{\|r_{ij}\|\;\|e\|} \tag{8.5}$$

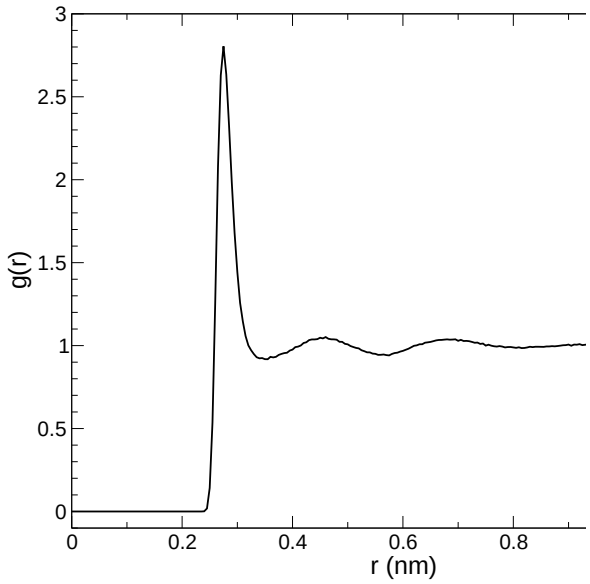


Figure 8.3: $g_{OO}(r)$ for Oxygen-Oxygen of SPC-water.

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 [106]. We describe here the implementation of the various correlation function flavors in the GROMACS code. The definition of the (ACF) $C_f(t)$ for a property $f(t)$ is:

$$C_f(t) \;=\; \langle f(\xi)f(\xi+t)\rangle_\xi \tag{8.6}$$

where the notation on the right hand side indicates 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 data with intervals much shorter than the time scale of interest. 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 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 [106].

8.5.3 Special forms of the ACF

There are some important varieties on the ACF, *e.g.* the ACF of a vector $\boldsymbol{p}$:

$$C_{\boldsymbol{p}}(t) \;=\; \int_0^\infty P_n(\cos\angle\left(\boldsymbol{p}(\xi), \boldsymbol{p}(\xi+t)\right)\mathrm{d}\xi \tag{8.12}$$

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 1^{st} and 2^{nd} order Legendre polynomial (eqn. 8.12). This can also be used for rotational autocorrelation (`gmx rotacf`) and dipole autocorrelation (`gmx dipoles`).

In order to study torsion angle dynamics, we define a dihedral autocorrelation function as [153]:

$$C(t) \;=\; \left\langle\cos(\theta(\tau)-\theta(\tau+t))\right\rangle_\tau \tag{8.13}$$

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 `gmx velacc` calculates the *velocity autocorrelation function*.

$$C_{\boldsymbol{v}}(\tau) \;=\; \left\langle\boldsymbol{v}_i(\tau)\cdot\boldsymbol{v}_i(0)\right\rangle_{i\in A} \tag{8.15}$$

The self diffusion coefficient can be calculated using the Green-Kubo relation [106]:

$$D_A \;=\; \frac{1}{3}\int_0^\infty \left\langle\mathbf{v}_i(t)\cdot\mathbf{v}_i(0)\right\rangle_{i\in A}\,dt \tag{8.16}$$

[1] $P_0(x)=1$, $P_1(x)=x$, $P_2(x)=(3x^2-1)/2$

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. 8.6), which can also be used for the computation of diffusion constants. However, Allen & Tildesley [106] 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 of type A is calculated as follows by `gmx 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 [154].

The viscosity of a liquid can be related to the correlation time of the Pressure tensor $\boldsymbol{P}$ [155, 156]. `gmx energy` can compute the viscosity, but this is not very accurate [137], and actually the values do not converge.

8.6 Mean Square Displacement

`gmx msd`
To determine the self D_A of particles of type A, one can use the Einstein relation [106]:

$$\lim_{t \to \infty} \langle \|\mathbf{r}_i(t) - \mathbf{r}_i(0)\|^2 \rangle_{i \in A} \;=\; 6 D_A t \tag{8.18}$$

This *mean square displacement* and D_A are calculated by the program `gmx msd`. Normally an index file containing atom numbers is used and the MSD is averaged over these atoms. For molecules consisting of more than one atom, $\mathbf{r}_i$ can be taken as the center of mass positions of the molecules. In that case, you should use an index file with molecule numbers. The results will be nearly identical to averaging over atoms, however. The `gmx msd` 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.

8.7 Bonds/distances, angles and dihedrals

`gmx distance, gmx angle, gmx gangle`
To monitor specific *bonds* in your modules, or more generally distances between points, the program `gmx distance` can calculate distances as a function of time, as well as the distribution of the distance. With a traditional index file, the groups should consist of pairs of atom numbers, for example:

```
[ bonds_1 ]
 1     2
 3     4
 9    10
```

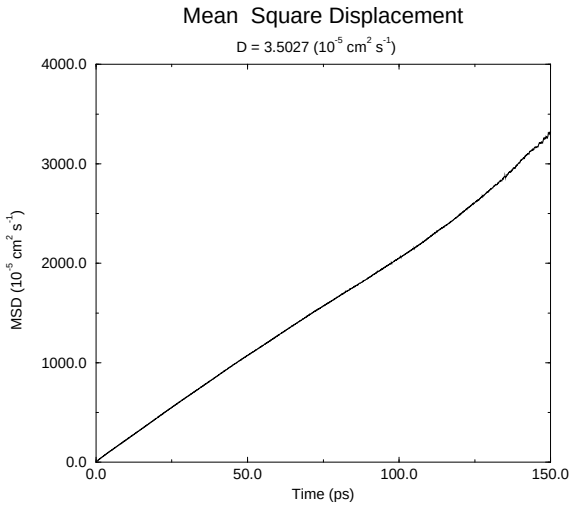


Figure 8.4: Mean Square Displacement of SPC-water.

```
[ bonds_2 ]
12    13
```

Selections are also supported, with first two positions defining the first distance, second pair of positions defining the second distance and so on. You can calculate the distances between CA and CB atoms in all your residues (assuming that every residue either has both atoms, or neither) using a selection such as:

```
name CA CB
```

The selections also allow more generic distances to be computed. For example, to compute the distances between centers of mass of two residues, you can use:

```
com of resname AAA plus com of resname BBB
```

The program `gmx 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.

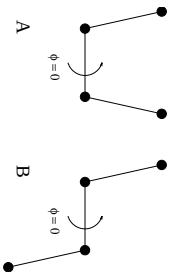


Figure 8.5: Dihedral conventions: A. “Biochemical convention”. B. “Polymer convention”.

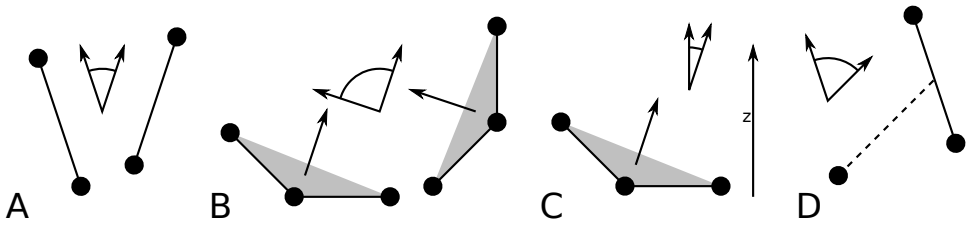


Figure 8.6: Angle options of `gmx gangle`: A. Angle between two vectors. B. Angle between two planes. C. Angle between a vector and the z axis. D. Angle between a vector and the normal of a sphere. Also other combinations are supported: planes and vectors can be used interchangeably.

The program `gmx gangle` provides a selection-enabled version to compute angles. This tool can also compute angles and dihedrals, but does not support all the options of `gmx angle`, such as autocorrelation or other time series analyses. In addition, it supports angles between two vectors, a vector and a plane, two planes (defined by 2 or 3 points, respectively), a vector/plane and the z axis, or a vector/plane and the normal of a sphere (determined by a single position). Also the angle between a vector/plane compared to its position in the first frame is supported. For planes, `gmx gangle` uses the normal vector perpendicular to the plane. See Fig. 8.6A, B, C) for the definitions.

8.8 Radius of gyration and distances

`gmx gyrate, gmx distance, gmx mindist, gmx mdmat, gmx xpm2ps`
To have a rough measure for the compactness of a structure, you can calculate the *radius of gyration* with the program `gmx 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:

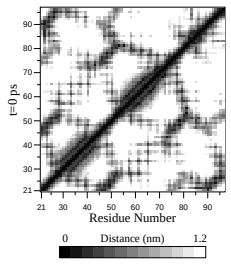


Figure 8.7: A minimum distance matrix for a peptide [157].

- The *distance between the geometrical centers* of two groups can be calculated with the program `gmx distance`, as explained in sec. 8.7.

- The *minimum distance* between two groups of atoms during time can be calculated with the program `gmx mindist`. It also calculates the *number of contacts* between these groups within a certain radius r_{max}.

- To monitor the *minimum distances between amino acid residues* within a (protein) molecule, you can use the program `gmx 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.

8.9 Root mean square deviations in structure

`gmx rms, gmx 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 `gmx rms` by least-square fitting the structure

to the reference structure ($t_2 = 0$) and subsequently calculating the $RMSD$ (eqn. 8.20).

$$RMSD(t_1, t_2) = \left[\frac{1}{M} \sum_{i=1}^{N} m_i \|\mathbf{r}_i(t_1) - \mathbf{r}_i(t_2)\|^2 \right]^{\frac{1}{2}} \tag{8.20}$$

where $M = \sum_{i=1}^{N} m_i$ and $\mathbf{r}_i(t)$ 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. A matrix can also be made with the $RMSD$ as a function of t_1 and t_2, which gives a nice graphical interpretation 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 `gmx 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.

8.10 Covariance analysis

Covariance analysis, also called principal component analysis or essential dynamics [158], can find correlated motions. It uses the covariance matrix C of the atomic coordinates:

$$C_{ij} = \left\langle M_{ii}^{\frac{1}{2}} (x_i - \langle x_i \rangle) M_{jj}^{\frac{1}{2}} (x_j - \langle x_j \rangle) \right\rangle \tag{8.22}$$

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:

$$R^T C R = \text{diag}(\lambda_1, \lambda_2, \ldots, \lambda_{3N}) \quad \text{where} \quad \lambda_1 \geq \lambda_2 \geq \ldots \geq \lambda_{3N} \tag{8.23}$$

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 $p_i(t)$:

$$\mathbf{p}(t) = R^T M^{\frac{1}{2}} (\mathbf{x}(t) - \langle \mathbf{x} \rangle) \tag{8.24}$$

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:

$$\mathbf{x}^f(t) = \langle \mathbf{x} \rangle + M^{-\frac{1}{2}} R_{*i}\, p_i(t) \tag{8.25}$$

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 (see below). 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.

A good measure for the overlap has been defined in [159]. 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 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:

$$A^{\frac{1}{2}} = R\,\mathrm{diag}(\lambda_1^{\frac{1}{2}}, \lambda_2^{\frac{1}{2}}, \ldots, \lambda_{3N}^{\frac{1}{2}})\,R^T \tag{8.26}$$

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:

$$\begin{aligned} d(A,B) &= \sqrt{\mathrm{tr}\left(\left(A^{\frac{1}{2}} - B^{\frac{1}{2}}\right)^2\right)} && (8.27)\\ &= \sqrt{\mathrm{tr}\left(A + B - 2A^{\frac{1}{2}}B^{\frac{1}{2}}\right)} && (8.28)\\ &= \left(\sum_{i=1}^{N}\left(\lambda_i^A + \lambda_i^B\right) - 2\sum_{i=1}^{N}\sum_{j=1}^{N}\sqrt{\lambda_i^A\lambda_j^B}\left(R_i^A \cdot R_j^B\right)^2\right)^{\frac{1}{2}} && (8.29) \end{aligned}$$

where tr is the trace of a matrix. We can now define the overlap s as:

$$s(A,B) = 1 - \frac{d(A,B)}{\sqrt{\mathrm{tr}A + \mathrm{tr}B}} \tag{8.30}$$

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:

$$\mathrm{overlap}(\mathbf{v},\mathbf{w}) = \frac{1}{n}\sum_{i=1}^{n}\sum_{j=1}^{m}(\mathbf{v}_i \cdot \mathbf{w}_j)^2 \tag{8.31}$$

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 that the the principal components of random diffusion are cosines with the number of periods equal to half the principal component index [160, 159]. The eigenvalues are proportional to the index to the power -2. The cosine content is defined as:

$$\frac{2}{T}\left(\int_0^T \cos\left(\frac{i\pi t}{T}\right) p_i(t)\mathrm{d}t\right)^2 \left(\int_0^T p_i^2(t)\mathrm{d}t\right)^{-1} \tag{8.32}$$

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 `gmx covar`. The principal components and overlap (and many more things) can be plotted and analyzed with `gmx anaeig`. The cosine content can be calculated with `gmx analyze`.

8.11 Dihedral principal component analysis

`gmx angle, gmx covar, gmx anaeig`

Principal component analysis can be performed in dihedral space [161] using GROMACS. You start by defining the dihedral angles of interest in an index file, either using `gmx mk_angndx` or otherwise. Then you use the `gmx angle` program with the `-or` flag to produce a new `.trr` file containing the cosine and sine of each dihedral angle in two coordinates, respectively. That is, in the `.trr` file you will have a series of numbers corresponding to: cos(ϕ_1), sin(ϕ_1), cos(ϕ_2), sin(ϕ_2), ..., cos(ϕ_n), sin(ϕ_n), and the array is padded with zeros, if necessary. Then you can use this `.trr` file as input for the `gmx covar` program and perform principal component analysis as usual. For this to work you will need to generate a reference file (`.tpr`, `.gro`, `.pdb` etc.) containing the same number of "atoms" as the new `.trr` file, that is for n dihedrals you need $2n/3$ atoms (rounded up if not an integer number). You should use the `-nofit` option for `gmx covar` since the coordinates in the dummy reference file do not correspond in any way to the information in the `.trr` file. Analysis of the results is done using `gmx anaeig`.

8.12 Hydrogen bonds

`gmx hbond`

The program `gmx hbond` analyzes 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:

$$\begin{array}{rcrcl} r & \leq & r_{HB} & = & 0.35 \text{ nm} \\ \alpha & \leq & \alpha_{HB} & = & 30^o \end{array} \tag{8.33}$$

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 `gmx hbond` analyzes 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:

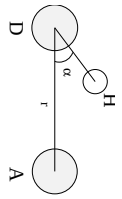


Figure 8.8: Geometrical Hydrogen bond criterion.

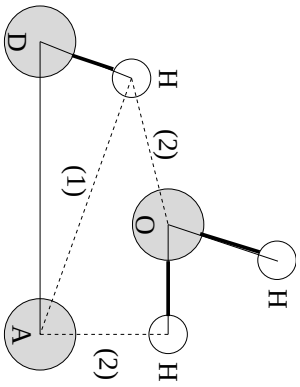


Figure 8.9: Insertion of water into an H-bond. (1) Normal H-bond between two residues. (2) H-bonding bridge via a water molecule.

- 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.
- 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:

$$C(\tau) \;=\; \langle s_i(t)\; s_i(t+\tau)\rangle \tag{8.34}$$

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}:

$$\tau_{HB} \;=\; \int_0^\infty C(\tau)d\tau \tag{8.35}$$

Both the integral and the complete autocorrelation 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}. A more complete analysis is given in ref. [162]; one of the more fancy option is the Luzar and Chandler analysis of hydrogen bond kinetics [163, 164].

- An H-bond existence map can be generated of dimensions *# H-bonds*×*# frames*. The ordering is identical to the index file (see below), but reversed, meaning that the last triplet in the index file corresponds to the first row of the existence map.
- 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.

8.13 Protein-related items

`gmx do_dssp, gmx rama, gmx wheel`
To analyze structural changes of a protein, you can calculate the radius of gyration or the minimum residue distances over time (see sec. 8.8), or calculate the RMSD (sec. 8.9).

You can also look at the changing of *secondary structure elements* during your run. For this, you can use the program `gmx do_dssp`, which is an interface for the commercial program `DSSP` [165]. For further information, see the `DSSP` manual. A typical output plot of `gmx 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 `gmx rama`. A typical output is given in Fig. 8.12.

When studying α-helices it is useful to have a *helical wheel* projection of your peptide, to see whether a peptide is amphipathic. This can be done using the `gmx wheel` program. Two examples are plotted in Fig. 8.13.

8.14 Interface-related items

`gmx order, gmx density, gmx potential, gmx traj`
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 `gmx order` can calculate order parameters using the equation:

$$S_z = \frac{3}{2}\langle \cos^2 \theta_z \rangle - \frac{1}{2} \tag{8.36}$$

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

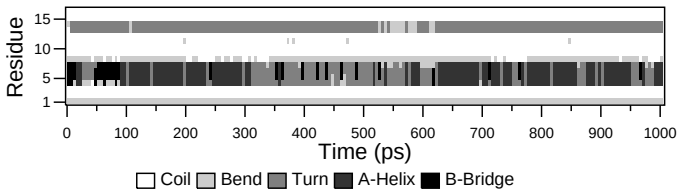


Figure 8.10: Analysis of the secondary structure elements of a peptide in time.

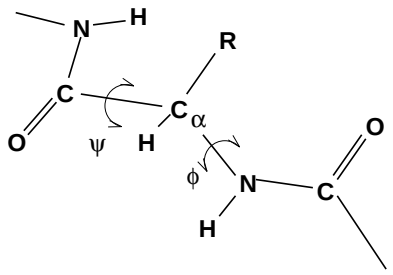


Figure 8.11: Definition of the dihedral angles ϕ and ψ of the protein backbone.

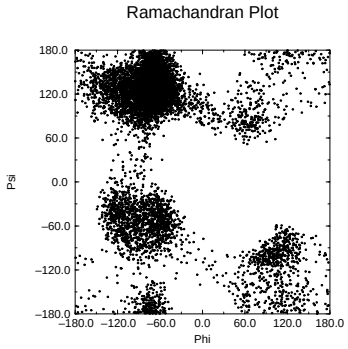


Figure 8.12: Ramachandran plot of a small protein.

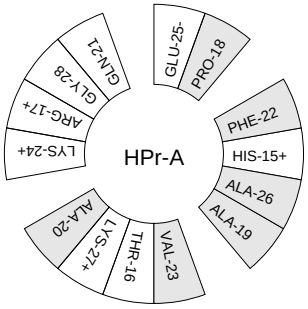


Figure 8.13: Helical wheel projection of the N-terminal helix of HPr.

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 the double integral of the charge density ($\rho(z)$):

$$\psi(z) - \psi(-\infty) = -\int_{-\infty}^{z} dz' \int_{-\infty}^{z'} \rho(z'')dz''/\epsilon_0 \tag{8.37}$$

where the position $z = -\infty$ is far enough in the bulk phase such 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 `gmx potential` divides the box in slices and sums all charges of the atoms in each slice. It then integrates this charge density to give the electric field, which is in turn integrated to give the potential. Charge density, electric field, and potential are written to `xvgr` input files.

The program `gmx traj` is a very simple analysis program. All it does is print the coordinates, velocities, or forces of selected 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 interfacial systems.

The program `gmx density` calculates the mass density of groups 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.

Appendix A

Technical Details

A.1 Mixed or Double precision

GROMACS can be compiled in either mixed or double precision. Documentation of previous GROMACS versions referred to "single precision", but the implementation has made selective use of double precision for many years. Using single precision for all variables would lead to a significant reduction in accuracy. Although in "mixed precision" all state vectors, i.e. particle coordinates, velocities and forces, are stored in single precision, critical variables are double precision. A typical example of the latter is the virial, which is a sum over all forces in the system, which have varying signs. In addition, in many parts of the code we managed to avoid double precision for arithmetic, by paying attention to summation order or reorganization of mathematical expressions. The default configuration uses mixed precision, but it is easy to turn on double precision by adding the option `-DGMX_DOUBLE=on` to `cmake`. Double precision will be 20 to 100% slower than mixed 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.

The energies in mixed 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 the virial can be two orders of magnitude larger than the average. Using cut-offs 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 partial use of single precision. Since MD is chaotic, trajectories with very similar starting conditions will diverge rapidly, the divergence is faster in mixed precision than in double precision.

For most simulations, mixed precision is accurate enough. In some cases double precision is required to get reasonable results:

- normal mode analysis, for the conjugate gradient or l-bfgs minimization and the calculation and diagonalization of the Hessian

- long-term energy conservation, especially for large systems

A.2 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. Some other useful environment variables are listed in the following sections. Most environment variables function by being set in your shell to any non-NULL value. Specific requirements are described below if other values need to be set. You should consult the documentation for your shell for instructions on how to set environment variables in the current shell, or in config files for future shells. Note that requirements for exporting environment variables to jobs run under batch control systems vary and you should consult your local documentation for details.

Output Control

1. `GMX_CONSTRAINTVIR`: print constraint virial and force virial energy terms.
2. `GMX_MAXBACKUP`: GROMACS automatically backs up old copies of files when trying to write a new file of the same name, and this variable controls the maximum number of backups that will be made, default 99. If set to 0 it fails to run if any output file already exists. And if set to -1 it overwrites any output file without making a backup.
3. `GMX_NO_QUOTES`: if this is explicitly set, no cool quotes will be printed at the end of a program.
4. `GMX_SUPPRESS_DUMP`: prevent dumping of step files during (for example) blowing up during failure of constraint algorithms.
5. `GMX_TPI_DUMP`: dump all configurations to a `.pdb` file that have an interaction energy less than the value set in this environment variable.
6. `GMX_VIEW_XPM`: `GMX_VIEW_XVG`, `GMX_VIEW_EPS` and `GMX_VIEW_PDB`, commands used to automatically view `.xvg`, `.xpm`, `.eps` and `.pdb` file types, respectively; they default to `xv`, `xmgrace`, `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.
7. `GMX_VIRIAL_TEMPERATURE`: print virial temperature energy term
8. `GMX_LOG_BUFFER`: 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.
9. `GMX_LOGO_COLOR`: set display color for logo in `ngmx`.
10. `GMX_PRINT_LONGFORMAT`: use long float format when printing decimal values.

11. GMX_COMPELDUMP: Applies for computational electrophysiology setups only (see section 6.6). The initial structure gets dumped to .pdb file, which allows to check whether multimeric channels have the correct PBC representation.

Debugging

1. GMX_PRINT_DEBUG_LINES: when set, print debugging info on line numbers.
2. GMX_DD_NST_DUMP: number of steps that elapse between dumping the current DD to a PDB file (default 0). This only takes effect during domain decomposition, so it should typically be 0 (never), 1 (every DD phase) or a multiple of nstlist.
3. GMX_DD_NST_DUMP_GRID: number of steps that elapse between dumping the current DD grid to a PDB file (default 0). This only takes effect during domain decomposition, so it should typically be 0 (never), 1 (every DD phase) or a multiple of nstlist.
4. GMX_DD_DEBUG: general debugging trigger for every domain decomposition (default 0, meaning off). Currently only checks global-local atom index mapping for consistency.
5. GMX_DD_NPULSE: over-ride the number of DD pulses used (default 0, meaning no override). Normally 1 or 2.

Performance and Run Control

1. GMX_DO_GALACTIC_DYNAMICS: planetary simulations are made possible (just for fun) by setting this environment variable, which allows setting epsilon_r = -1 in the .mdp file. Normally, epsilon_r must be greater than zero to prevent a fatal error. See www.gromacs.org for example input files for a planetary simulation.
2. GMX_ALLOW_CPT_MISMATCH: when set, runs will not exit if the ensemble set in the .tpr file does not match that of the .cpt file.
3. GMX_CUDA_NB_EWALD_TWINCUT: force the use of twin-range cutoff kernel even if rvdw = rcoulomb after PP-PME load balancing. The switch to twin-range kernels is automated, so this variable should be used only for benchmarking.
4. GMX_CUDA_NB_ANA_EWALD: force the use of analytical Ewald kernels. Should be used only for benchmarking.
5. GMX_CUDA_NB_TAB_EWALD: force the use of tabulated Ewald kernels. Should be used only for benchmarking.
6. GMX_CUDA_STREAMSYNC: force the use of cudaStreamSynchronize on ECC-enabled GPUs, which leads to performance loss due to a known CUDA driver bug present in API v5.0 NVIDIA drivers (pre-30x.xx). Cannot be set simultaneously with GMX_NO_CUDA_STREAMSYNC.
7. GMX_CYCLE_ALL: times all code during runs. Incompatible with threads.
8. GMX_CYCLE_BARRIER: calls MPI_Barrier before each cycle start/stop call.

9. GMX_DD_ORDER_ZYX: build domain decomposition cells in the order (z, y, x) rather than the default (x, y, z).

10. GMX_DD_USE_SENDRECV2: during constraint and vsite communication, use a pair of MPI_SendRecv calls instead of two simultaneous non-blocking calls (default 0, meaning off). Might be faster on some MPI implementations.

11. GMX_DLB_BASED_ON_FLOPS: do domain-decomposition dynamic load balancing based on flop count rather than measured time elapsed (default 0, meaning off). This makes the load balancing reproducible, which can be useful for debugging purposes. A value of 1 uses the flops; a value ¿ 1 adds (value - 1)*5% of noise to the flops to increase the imbalance and the scaling.

12. GMX_DLB_MAX_BOX_SCALING: maximum percentage box scaling permitted per domain-decomposition load-balancing step (default 10)

13. GMX_DD_RECORD_LOAD: record DD load statistics for reporting at end of the run (default 1, meaning on)

14. GMX_DD_NST_SORT_CHARGE_GROUPS: number of steps that elapse between re-sorting of the charge groups (default 1). This only takes effect during domain decomposition, so should typically be 0 (never), 1 (to mean at every domain decomposition), or a multiple of nstlist.

15. GMX_DETAILED_PERF_STATS: when set, print slightly more detailed performance information to the .log file. The resulting output is the way performance summary is reported in versions 4.5.x and thus may be useful for anyone using scripts to parse .log files or standard output.

16. GMX_DISABLE_SIMD_KERNELS: disables architecture-specific SIMD-optimized (SSE2, SSE4.1, AVX, etc.) non-bonded kernels thus forcing the use of plain C kernels.

17. GMX_DISABLE_CUDA_TIMING: timing of asynchronously executed GPU operations can have a non-negligible overhead with short step times. Disabling timing can improve performance in these cases.

18. GMX_DISABLE_GPU_DETECTION: when set, disables GPU detection even if mdrun was compiled with GPU support.

19. GMX_DISABLE_PINHT: disable pinning of consecutive threads to physical cores when using Intel hyperthreading. Controlled with mdrun -nopinht and thus this environment variable will likely be removed.

20. GMX_DISRE_ENSEMBLE_SIZE: the number of systems for distance restraint ensemble averaging. Takes an integer value.

21. GMX_EMULATE_GPU: emulate GPU runs by using algorithmically equivalent CPU reference code instead of GPU-accelerated functions. As the CPU code is slow, it is intended to be used only for debugging purposes. The behavior is automatically triggered if non-bonded calculations are turned off using GMX_NO_NONBONDED case in which the non-bonded calculations will not be called, but the CPU-GPU transfer will also be skipped.

22. GMX_ENX_NO_FATAL: disable exiting upon encountering a corrupted frame in an .edr file, allowing the use of all frames up until the corruption.

23. GMX_FORCE_UPDATE: update forces when invoking mdrun -rerun.

24. GMX_GPU_ID: set in the same way as the mdrun option -gpu_id, GMX_GPU_ID allows the user to specify different GPU id-s, which can be useful for selecting different devices on different compute nodes in a cluster. Cannot be used in conjunction with -gpu_id.

25. GMX_IGNORE_FSYNC_FAILURE_ENV: allow mdrun to continue even if a file is missing.

26. GMX_LJCOMB_TOL: when set to a floating-point value, overrides the default tolerance of 1e-5 for force-field floating-point parameters.

27. GMX_MAX_MPI_THREADS: sets the maximum number of MPI-threads that mdrun can use.

28. GMX_MAXCONSTRWARN: if set to -1, mdrun will not exit if it produces too many LINCS warnings.

29. GMX_NB_GENERIC: use the generic C kernel. Should be set if using the group-based cutoff scheme and also sets GMX_NO_SOLV_OPT to be true, thus disabling solvent optimizations as well.

30. GMX_NB_MIN_CI: neighbor list balancing parameter used when running on GPU. Sets the target minimum number pair-lists in order to improve multi-processor load-balance for better performance with small simulation systems. Must be set to a positive integer, the default value is optimized for NVIDIA Fermi and Kepler GPUs, therefore changing it is not necessary for normal usage, but it can be useful on future architectures.

31. GMX_NBLISTCG: use neighbor list and kernels based on charge groups.

32. GMX_NBNXN_CYCLE: when set, print detailed neighbor search cycle counting.

33. GMX_NBNXN_EWALD_ANALYTICAL: force the use of analytical Ewald non-bonded kernels, mutually exclusive of GMX_NBNXN_EWALD_TABLE.

34. GMX_NBNXN_EWALD_TABLE: force the use of tabulated Ewald non-bonded kernels, mutually exclusive of GMX_NBNXN_EWALD_ANALYTICAL.

35. GMX_NBNXN_SIMD_2XNN: force the use of 2x(N+N) SIMD CPU non-bonded kernels, mutually exclusive of GMX_NBNXN_SIMD_4XN.

36. GMX_NBNXN_SIMD_4XN: force the use of 4xN SIMD CPU non-bonded kernels, mutually exclusive of GMX_NBNXN_SIMD_2XNN.

37. GMX_NO_ALLVSALL: disables optimized all-vs-all kernels.

38. GMX_NO_CART_REORDER: used in initializing domain decomposition communicators. Rank reordering is default, but can be switched off with this environment variable.

39. GMX_NO_CUDA_STREAMSYNC: the opposite of GMX_CUDA_STREAMSYNC. Disables the use of the standard cudaStreamSynchronize-based GPU waiting to improve performance when using CUDA driver API ealier than v5.0 with ECC-enabled GPUs.

40. GMX_NO_INT, GMX_NO_TERM, GMX_NO_USR1: disable signal handlers for SIGINT, SIGTERM, and SIGUSR1, respectively.

41. GMX_NO_NODECOMM: do not use separate inter- and intra-node communicators.

42. GMX_NO_NONBONDED: skip non-bonded calculations; can be used to estimate the possible performance gain from adding a GPU accelerator to the current hardware setup – assuming that this is fast enough to complete the non-bonded calculations while the CPU does bonded force and PME computation.

43. GMX_NO_PULLVIR: when set, do not add virial contribution to COM pull forces.

44. GMX_NOCHARGEGROUPS: disables multi-atom charge groups, *i.e.* each atom in all non-solvent molecules is assigned its own charge group.

45. GMX_NOPREDICT: shell positions are not predicted.

46. GMX_NO_SOLV_OPT: turns off solvent optimizations; automatic if GMX_NB_GENERIC is enabled.

47. GMX_NSCELL_NCG: the ideal number of charge groups per neighbor searching grid cell is hard-coded to a value of 10. Setting this environment variable to any other integer value overrides this hard-coded value.

48. GMX_PME_NTHREADS: set the number of OpenMP or PME threads (overrides the number guessed by mdrun.

49. GMX_PME_P3M: use P3M-optimized influence function instead of smooth PME B-spline interpolation.

50. GMX_PME_THREAD_DIVISION: PME thread division in the format "x y z" for all three dimensions. The sum of the threads in each dimension must equal the total number of PME threads (set in GMX_PME_NTHREADS).

51. GMX_PMEONEDD: if the number of domain decomposition cells is set to 1 for both x and y, decompose PME in one dimension.

52. GMX_REQUIRE_SHELL_INIT: require that shell positions are initiated.

53. GMX_REQUIRE_TABLES: require the use of tabulated Coulombic and van der Waals interactions.

54. GMX_SCSIGMA_MIN: the minimum value for soft-core σ. **Note** that this value is set using the sc-sigma keyword in the .mdp file, but this environment variable can be used to reproduce pre-4.5 behavior with respect to this parameter.

55. GMX_TPIC_MASSES: should contain multiple masses used for test particle insertion into a cavity. The center of mass of the last atoms is used for insertion into the cavity.

56. GMX_USE_GRAPH: use graph for bonded interactions.

57. GMX_VERLET_BUFFER_RES: resolution of buffer size in Verlet cutoff scheme. The default value is 0.001, but can be overridden with this environment variable.

58. GMX_VERLET_SCHEME: convert from group-based to Verlet cutoff scheme, even if the cutoff_scheme is not set to use Verlet in the .mdp file. It is unnecessary since the -testverlet option of mdrun has the same functionality, but it is maintained for backwards compatibility.

59. MPIRUN: the mpirun command used by g_tune_pme.

60. MDRUN: the mdrun command used by g_tune_pme.

61. GMX_NSTLIST: sets the default value for nstlist, preventing it from being tuned during mdrun startup when using the Verlet cutoff scheme.

62. GMX_USE_TREEREDUCE: use tree reduction for nbnxn force reduction. Potentially faster for large number of OpenMP threads (if memory locality is important).

Analysis and Core Functions

1. GMX_QM_ACCURACY: accuracy in Gaussian L510 (MC-SCF) component program.

2. GMX_QM_ORCA_BASENAME: prefix of .tpr files, used in Orca calculations for input and output file names.

3. GMX_QM_CPMCSCF: when set to a nonzero value, Gaussian QM calculations will iteratively solve the CP-MCSCF equations.

4. GMX_QM_MODIFIED_LINKS_DIR: location of modified links in Gaussian.

5. DSSP: used by do_dssp to point to the dssp executable (not just its path).

6. GMX_QM_GAUSS_DIR: directory where Gaussian is installed.

7. GMX_QM_GAUSS_EXE: name of the Gaussian executable.

8. GMX_DIPOLE_SPACING: spacing used by g_dipoles.

9. GMX_MAXRESRENUM: sets the maximum number of residues to be renumbered by grompp. A value of -1 indicates all residues should be renumbered.

10. GMX_FFRTP_TER_RENAME: Some force fields (like AMBER) use specific names for N- and C- terminal residues (NXXX and CXXX) as .rtp entries that are normally renamed. Setting this environment variable disables this renaming.

11. GMX_PATH_GZIP: gunzip executable, used by g_wham.

12. GMX_FONT: name of X11 font used by ngmx.

13. GMXTIMEUNIT: the time unit used in output files, can be anything in fs, ps, ns, us, ms, s, m or h.
14. GMX_QM_GAUSSIAN_MEMORY: memory used for Gaussian QM calculation.
15. MULTIPROT: name of the multiprot executable, used by the contributed program do_multiprot.
16. NCPUS: number of CPUs to be used for Gaussian QM calculation
17. GMX_ORCA_PATH: directory where Orca is installed.
18. GMX_QM_SA_STEP: simulated annealing step size for Gaussian QM calculation.
19. GMX_QM_GROUND_STATE: defines state for Gaussian surface hopping calculation.
20. GMX_TOTAL: name of the total executable used by the contributed do_shift program.
21. GMX_ENER_VERBOSE: make g_energy and eneconv loud and noisy.
22. VMD_PLUGIN_PATH: where to find VMD plug-ins. Needed to be able to read file formats recognized only by a VMD plug-in.
23. VMDDIR: base path of VMD installation.
24. GMX_USE_XMGR: sets viewer to xmgr (deprecated) instead of xmgrace.

A.3 Running GROMACS in parallel

By default GROMACS will be compiled with the built-in thread-MPI library. This library handles communication between threads on a single node more efficiently than using an external MPI library. To run GROMACS in parallel over multiple nodes, e.g. on a cluster, you need to configure and compile GROMACS with an external MPI library. All supercomputers are shipped with MPI libraries optimized for that particular platform, and there are several good free MPI implementations; OpenMPI is usually a good choice. Note that MPI and thread-MPI support are mutually incompatible.

In addition to MPI parallelization, GROMACS supports also thread-parallelization through OpenMP. MPI and OpenMP parallelization can be combined, which results in, so called, hybrid parallelization. It can offer better performance and scaling in some cases.

See www.gromacs.org for details on the use and performance of the different parallelization schemes.

A.4 Running GROMACS on GPUs

As of version 4.6, GROMACS has native GPU support through CUDA. Note that GROMACS only off-loads the most compute intensive parts to the GPU, currently the non-bonded interactions, and does all other parts of the MD calculation on the CPU. The requirements for the CUDA code are

an Nvidia GPU with compute capability ≥ 2.0, i.e. at least Fermi class. In many cases `cmake` can auto-detect GPUs and the support will be configured automatically. To be sure GPU support is configured, pass the `-DGMX_GPU=on` option to `cmake`. The actual use of GPUs is decided at run time by `mdrun`, depending on the availability of (suitable) GPUs and on the run input settings. A binary compiled with GPU support can also run CPU only simulations. Use `mdrun -nb cpu` to force a simulation to run on CPUs only. Only simulations with the Verlet cut-off scheme will run on a GPU. To test performance of old tpr files with GPUs, you can use the `-testverlet` option of `mdrun`, but as this doesn't do the full parameter consistency check of `grommp`, you should not use this option for production simulations. Getting good performance with GROMACS on GPUs is easy, but getting best performance can be difficult. Please check www.gromacs.org for up to date information on GPU usage.

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 [166].

B.1.1 Virial

In a system with , 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.

In a triclinic system, there are 27 possible images of i; when a 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 with respect to 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 intra-molecular 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.

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, gray

5 Pick one of the gray 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, gray

7 If any gray atom remains, go to [5]

8 If any white atom remains, go to [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 method again

Find a representation of the bonds as a bidirectional graph.

B.1.4 Virial from Covalent Bonds

Since the covalent bond force gives a contribution to the virial, we have:

$$
\begin{aligned}
b &= \|\boldsymbol{r}_{ij}^n\| && \text{(B.15)}\\
V_b &= \frac{1}{2}k_b(b-b_0)^2 && \text{(B.16)}\\
\boldsymbol{F}_i &= -\nabla V_b && \text{(B.17)}\\
&= k_b(b-b_0)\frac{\boldsymbol{r}_{ij}^n}{b} && \text{(B.18)}\\
\boldsymbol{F}_j &= -\boldsymbol{F}_i && \text{(B.19)}
\end{aligned}
$$

The virial contribution from the bonds then is:

$$
\begin{aligned}
\Xi_b &= -\frac{1}{2}(\boldsymbol{r}_i^n \otimes \boldsymbol{F}_i \;+\; \boldsymbol{r}_j \otimes \boldsymbol{F}_j) && \text{(B.20)}\\
&= -\frac{1}{2}\boldsymbol{r}_{ij}^n \otimes \boldsymbol{F}_i && \text{(B.21)}
\end{aligned}
$$

B.1.5 Virial from SHAKE

An important contribution to the virial comes from shake. Satisfying the constraints a force **G** that is exerted on the particles "shaken." If this force does not come out of the algorithm (as in standard SHAKE) it can be calculated afterward (when using *leap-frog*) by:

$$\Delta \boldsymbol{r}_i = \boldsymbol{r}_i(t+\Delta t) - [\boldsymbol{r}_i(t) + \mathbf{v}_i(t-\frac{\Delta t}{2})\Delta t + \frac{\boldsymbol{F}_i}{m_i}\Delta t^2] \tag{B.22}$$

$$\mathbf{G}_i = \frac{m_i \Delta \boldsymbol{r}_i}{\Delta t^2} \tag{B.23}$$

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:

$$\Xi = -\frac{1}{2}\sum_i^{N_c} \boldsymbol{r}_i \otimes \boldsymbol{F}_i \tag{B.24}$$

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 uses special inner loops to calculate non-bonded interactions for water molecules with other atoms, and yet another set of loops for interactions between pairs of water molecules. There highly optimized loops for two types of water models. For three site models similar to SPC [81], *i.e.*:

1. There are three atoms in the molecule.
2. The whole molecule is a single charge group.
3. The first atom has Lennard-Jones (sec. 4.1.1) and Coulomb (sec. 4.1.3) interactions.
4. Atoms two and three have only Coulomb interactions, and equal charges.

These loops also works for the SPC/E [167] and TIP3P [125] water models. And for four site water models similar to TIP4P [125]:

1. There are four atoms in the molecule.
2. The whole molecule is a single charge group.

3. The first atom has only Lennard-Jones (sec. 4.1.1) interactions.
4. Atoms two and three have only Coulomb (sec. 4.1.3) interactions, and equal charges.
5. Atom four has only Coulomb interactions.

The benefit 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-sub-expression elimination, or moving code around.

B.2.2 Fortran Code

Unfortunately, on a few platforms Fortran compilers are still better than C-compilers. 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 that 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 look-up 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 [168] the $1/\sqrt{x}$ function can be evaluated using the Newton-Raphson iteration scheme. The inverse function is:

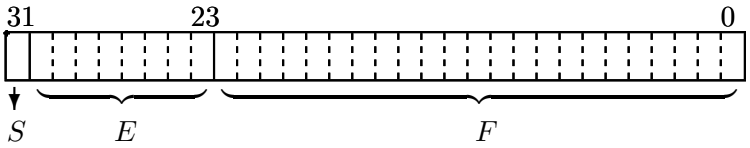
$$X(y) = \frac{1}{y^2} \tag{B.26}$$

So instead of calculating:

$$Y(a) = q \tag{B.27}$$

the equation:

$$X(q) - a = 0 \tag{B.28}$$



$$Value = (-1)^S(2^{E-127})(1.F)$$

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 [168] 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.

B.3.4 Specification of the look-up 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 (eqn. B.34):

$$\frac{\varepsilon_n}{y} = \sqrt{\varepsilon_{r_{n+1}} \frac{2f'}{yf''}} \tag{B.35}$$

So for the look-up 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 look-up 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 look-up table follows from:

$$\frac{\Delta x}{x} = -2 \frac{\Delta Y}{Y} \tag{B.40}$$

So, using the floating-point accuracy (eqn. B.36):

$$\frac{\Delta x}{x} = -2 \sqrt{\frac{2}{3} 2^{-24}} \tag{B.41}$$

This defines the length of the look-up table which should be ≥ 12 bit.

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 look-up 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 look-up table is length of look-up 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 (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 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.

$$E' = \frac{127 - E}{2} + 126 \tag{B.51}$$

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 look-up table are independent. The fraction look-up 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 look-up 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 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:

$$y_{n+1} = \frac{y_n}{2}(3 - ay_n^2) \tag{B.57}$$

Where y_0 can be found in the look-up 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.

Appendix C

Averages and fluctuations

C.1 Formulae for averaging

Note: this section was taken from ref [169].

When analyzing a MD trajectory averages $\langle x \rangle$ and fluctuations

$$\left\langle (\Delta x)^2 \right\rangle^{\frac{1}{2}} \;=\; \left\langle [x - \langle x \rangle]^2 \right\rangle^{\frac{1}{2}} \tag{C.1}$$

of a quantity x are to be computed. The variance σ_x of a series of N_x values, $\{\mathrm{x}_i\}$, can be computed from

$$\sigma_x \;=\; \sum_{i=1}^{N_x} x_i^2 \;-\; \frac{1}{N_x} \left(\sum_{i=1}^{N_x} x_i \right)^2 \tag{C.2}$$

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

$$\sigma_x \;=\; \sum_{i=1}^{N_x} [x_i - \langle x \rangle]^2 \tag{C.3}$$

with

$$\langle x \rangle \;=\; \frac{1}{N_x} \sum_{i=1}^{N_x} x_i \tag{C.4}$$

Using eqns. C.2 and C.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. C.1 requires only one sequential scan of the series $\{\mathrm{x}_i\}$. However, one may cast eqn. C.2 in another form, containing partial sums, which allows for a sequential update algorithm. Define the partial sum

$$X_{n,m} \;=\; \sum_{i=n}^{m} x_i \tag{C.5}$$

and the partial variance

$$\sigma_{n,m} = \sum_{i=n}^{m} \left[x_i - \frac{X_{n,m}}{m-n+1} \right]^2 \tag{C.6}$$

It can be shown that

$$X_{n,m+k} = X_{n,m} + X_{m+1,m+k} \tag{C.7}$$

and

$$\sigma_{n,m+k} = \sigma_{n,m} + \sigma_{m+1,m+k} + \left[\frac{X_{n,m}}{m-n+1} - \frac{X_{n,m+k}}{m+k-n+1} \right]^2 * \frac{(m-n+1)(m+k-n+1)}{k} \tag{C.8}$$

For $n = 1$ one finds

$$\sigma_{1,m+k} = \sigma_{1,m} + \sigma_{m+1,m+k} + \left[\frac{X_{1,m}}{m} - \frac{X_{1,m+k}}{m+k} \right]^2 \frac{m(m+k)}{k} \tag{C.9}$$

and for $n = 1$ and $k = 1$ (eqn. C.8) becomes

$$\sigma_{1,m+1} = \sigma_{1,m} + \left[\frac{X_{1,m}}{m} - \frac{X_{1,m+1}}{m+1} \right]^2 m(m+1) \tag{C.10}$$

$$= \sigma_{1,m} + \frac{[\, X_{1,m} - m x_{m+1} \,]^2}{m(m+1)} \tag{C.11}$$

where we have used the relation

$$X_{1,m+1} = X_{1,m} + x_{m+1} \tag{C.12}$$

Using formulae (eqn. C.11) and (eqn. C.12) the average

$$\langle x \rangle = \frac{X_{1,N_x}}{N_x} \tag{C.13}$$

and the fluctuation

$$\left\langle (\Delta x)^2 \right\rangle^{\frac{1}{2}} = \left[\frac{\sigma_{1,N_x}}{N_x} \right]^{\frac{1}{2}} \tag{C.14}$$

can be obtained by one sweep through the data.

C.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.

C.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. C.7 and C.8 the average and standard deviation over part of the trajectory can be computed as:

$$X_{m+1,m+k} = X_{1,m+k} - X_{1,m} \tag{C.15}$$

$$\sigma_{m+1,m+k} = \sigma_{1,m+k} - \sigma_{1,m} - \left[\frac{X_{1,m}}{m} - \frac{X_{1,m+k}}{m+k} \right]^2 \frac{m(m+k)}{k} \tag{C.16}$$

or, more generally (with $p \geq 1$ and $q \geq p$):

$$X_{p,q} = X_{1,q} - X_{1,p-1} \tag{C.17}$$

$$\sigma_{p,q} = \sigma_{1,q} - \sigma_{1,p-1} - \left[\frac{X_{1,p-1}}{p-1} - \frac{X_{1,q}}{q} \right]^2 \frac{(p-1)q}{q-p+1} \tag{C.18}$$

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. C.11 and C.12:

$$X_{1,p-1} = X_{1,p} - x_p \tag{C.19}$$

$$\sigma_{1,p-1} = \sigma_{1,p} - \frac{[\, X_{1,p-1} - (p-1)x_p \,]^2}{(p-1)p} \tag{C.20}$$

C.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:

$$X^{AB}_{1,n+m} = X^A_{1,n} + X^B_{1,m} \tag{C.21}$$

When we want to compute the partial variance from the two components we have to make a correction $\Delta\sigma$:

$$\sigma^{AB}_{1,n+m} = \sigma^A_{1,n} + \sigma^B_{1,m} + \Delta\sigma \tag{C.22}$$

if we define x_i^{AB} as the combined and renumbered set of data points we can write:

$$\sigma^{AB}_{1,n+m} = \sum_{i=1}^{n+m} \left[x_i^{AB} - \frac{X^{AB}_{1,n+m}}{n+m} \right]^2 \tag{C.23}$$

and thus

$$\sum_{i=1}^{n+m}\left[x_i^{AB}-\frac{X_{1,n+m}^{AB}}{n+m}\right]^2=\sum_{i=1}^{n}\left[x_i^{A}-\frac{X_{1,n}^{A}}{n}\right]^2+\sum_{i=1}^{m}\left[x_i^{B}-\frac{X_{1,m}^{B}}{m}\right]^2+\Delta\sigma \tag{C.24}$$

or

$$\begin{aligned}\sum_{i=1}^{n+m}\left[(x_i^{AB})^2-2x_i^{AB}\frac{X_{1,n+m}^{AB}}{n+m}+\left(\frac{X_{1,n+m}^{AB}}{n+m}\right)^2\right] \;-\\ \sum_{i=1}^{n}\left[(x_i^{A})^2-2x_i^{A}\frac{X_{1,n}^{A}}{n}+\left(\frac{X_{1,n}^{A}}{n}\right)^2\right] \;-\\ \sum_{i=1}^{m}\left[(x_i^{B})^2-2x_i^{B}\frac{X_{1,m}^{B}}{m}+\left(\frac{X_{1,m}^{B}}{m}\right)^2\right] \;=\; \Delta\sigma\end{aligned} \tag{C.25}$$

all the x_i^2 terms drop out, and the terms independent of the summation counter i can be simplified:

$$\begin{aligned}\frac{\left(X_{1,n+m}^{AB}\right)^2}{n+m}-\frac{\left(X_{1,n}^{A}\right)^2}{n}-\frac{\left(X_{1,m}^{B}\right)^2}{m} \;-\\ 2\,\frac{X_{1,n+m}^{AB}}{n+m}\sum_{i=1}^{n+m}x_i^{AB}+2\,\frac{X_{1,n}^{A}}{n}\sum_{i=1}^{n}x_i^{A}+2\,\frac{X_{1,m}^{B}}{m}\sum_{i=1}^{m}x_i^{B} \;=\; \Delta\sigma\end{aligned} \tag{C.26}$$

we recognize the three partial sums on the second line and use eqn. C.21 to obtain:

$$\Delta\sigma \;=\; \frac{\left(mX_{1,n}^{A}-nX_{1,m}^{B}\right)^2}{nm(n+m)} \tag{C.27}$$

if we check this by inserting $m=1$ we get back eqn. C.11

C.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:

$$X_{m,n}^{S} \;=\; \sum_{i=m}^{n}\sum_{s=1}^{S}x_i^s \;=\; \sum_{s=1}^{S}\sum_{i=m}^{n}x_i^s \;=\; \sum_{s=1}^{S}X_{m,n}^{s} \tag{C.28}$$

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:

$$\sigma_{m,n}^{S} \;=\; \sum_{s=1}^{S}\sigma_{m,n}^{s}+\Delta\sigma \tag{C.29}$$

if we fill in eqn. C.6:

$$\sum_{i=m}^{n}\left[\left(\sum_{s=1}^{S}x_i^s\right)-\frac{X_{m,n}^{S}}{m-n+1}\right]^2=\sum_{s=1}^{S}\sum_{i=m}^{n}\left[(x_i^s)-\frac{X_{m,n}^{s}}{m-n+1}\right]^2+\Delta\sigma \tag{C.30}$$

which we can expand to:

$$\sum_{i=m}^{n}\left[\sum_{s=1}^{S}(x_i^s)^2+\left(\frac{X_{m,n}^S}{m-n+1}\right)^2-2\left(\frac{X_{m,n}^S}{m-n+1}\sum_{s=1}^{S}x_i^s+\sum_{s=1}^{S}\sum_{s'=s+1}^{S}x_i^s x_i^{s'}\right)\right]$$
$$-\sum_{s=1}^{S}\sum_{i=m}^{n}\left[(x_i^s)^2-2\frac{X_{m,n}^s}{m-n+1}x_i^s+\left(\frac{X_{m,n}^s}{m-n+1}\right)^2\right]=\Delta\sigma \tag{C.31}$$

the terms with $(x_i^s)^2$ cancel, so that we can simplify to:

$$\frac{\left(X_{m,n}^S\right)^2}{m-n+1}-2\frac{X_{m,n}^S}{m-n+1}\sum_{i=m}^{n}\sum_{s=1}^{S}x_i^s-2\sum_{i=m}^{n}\sum_{s=1}^{S}\sum_{s'=s+1}^{S}x_i^s x_i^{s'}-$$
$$\sum_{s=1}^{S}\sum_{i=m}^{n}\left[-2\frac{X_{m,n}^s}{m-n+1}x_i^s+\left(\frac{X_{m,n}^s}{m-n+1}\right)^2\right]=\Delta\sigma \tag{C.32}$$

or

$$-\frac{\left(X_{m,n}^S\right)^2}{m-n+1}-2\sum_{i=m}^{n}\sum_{s=1}^{S}\sum_{s'=s+1}^{S}x_i^s x_i^{s'}+\sum_{s=1}^{S}\frac{\left(X_{m,n}^s\right)^2}{m-n+1}=\Delta\sigma \tag{C.33}$$

If we now expand the first term using eqn. C.28 we obtain:

$$-\frac{\left(\sum_{s=1}^{S}X_{m,n}^s\right)^2}{m-n+1}-2\sum_{i=m}^{n}\sum_{s=1}^{S}\sum_{s'=s+1}^{S}x_i^s x_i^{s'}+\sum_{s=1}^{S}\frac{\left(X_{m,n}^s\right)^2}{m-n+1}=\Delta\sigma \tag{C.34}$$

which we can reformulate to:

$$-2\left[\sum_{s=1}^{S}\sum_{s'=s+1}^{S}X_{m,n}^s X_{m,n}^{s'}+\sum_{i=m}^{n}\sum_{s=1}^{S}\sum_{s'=s+1}^{S}x_i^s x_i^{s'}\right]=\Delta\sigma \tag{C.35}$$

or

$$-2\left[\sum_{s=1}^{S}X_{m,n}^s\sum_{s'=s+1}^{S}X_{m,n}^{s'}+\sum_{s=1}^{S}\sum_{i=m}^{n}x_i^s\sum_{s'=s+1}^{S}x_i^{s'}\right]=\Delta\sigma \tag{C.36}$$

which gives

$$-2\sum_{s=1}^{S}\left[X_{m,n}^s\sum_{s'=s+1}^{S}\sum_{i=m}^{n}x_i^{s'}+\sum_{i=m}^{n}x_i^s\sum_{s'=s+1}^{S}x_i^{s'}\right]=\Delta\sigma \tag{C.37}$$

Since we need all data points i to evaluate this, in general this is not possible. We can then make an estimate of $\sigma_{m,n}^S$ using only the data points that are available using the left hand side of eqn. C.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.

Bibliography

[1] Bekker, H., Berendsen, H. J. C., Dijkstra, E. J., Achterop, S., van Drunen, R., van der Spoel, D., Sijbers, A., Keegstra, H., Reitsma, B., Renardus, M. K. R. Gromacs: A parallel computer for molecular dynamics simulations. In *Physics Computing 92* (Singapore, 1993). de Groot, R. A., Nadrchal, J., eds. . World Scientific.

[2] 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.

[3] Lindahl, E., Hess, B., van der Spoel, D. GROMACS 3.0: A package for molecular simulation and trajectory analysis. J. Mol. Mod. 7:306–317, 2001.

[4] van der Spoel, D., Lindahl, E., Hess, B., Groenhof, G., Mark, A. E., Berendsen, H. J. C. GROMACS: Fast, Flexible and Free. J. Comp. Chem. 26:1701–1718, 2005.

[5] Hess, B., Kutzner, C., van der Spoel, D., Lindahl, E. GROMACS 4: Algorithms for Highly Efficient, Load-Balanced, and Scalable Molecular Simulation. J. Chem. Theory Comput. 4(3):435–447, 2008.

[6] Pronk, S., Páll, S., Schulz, R., Larsson, P., Bjelkmar, P., Apostolov, R., Shirts, M. R., Smith, J. C., Kasson, P. M., van der Spoel, D., Hess, B., Lindahl, E. GROMACS 4.5: a high-throughput and highly parallel open source molecular simulation toolkit. Bioinformatics 29(7):845–854, 2013.

[7] 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.

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

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

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

[11] van Gunsteren, W. F., Karplus, M. Effect of constraints on the dynamics of macromolecules. Macromolecules 15:1528–1544, 1982.

[12] 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.

[13] 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.

[14] Geman, S., Geman, D. Stochastic relaxation, Gibbs distributions and the Bayesian restoration of images. IEEE Trans. Patt. Anal. Mach. Int. 6:721, 1984.

[15] 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.

[16] 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.

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

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

[19] 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.

[20] Hockney, R. W., Goel, S. P., Eastwood, J. Quiet High Resolution Computer Models of a Plasma. J. Comp. Phys. 14:148–158, 1974.

[21] Verlet., L. Computer experiments on classical fluids. I. Thermodynamical properties of Lennard-Jones molecules. Phys. Rev. 159:98–103, 1967.

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

[23] Swope, W. C., Andersen, H. C., Berens, P. H., Wilson, K. R. A computer-simulation method for the calculation of equilibrium-constants for the formation of physical clusters of molecules: Application to small water clusters. J. Chem. Phys. 76:637–649, 1982.

[24] Tuckerman, M., Berne, B. J., Martyna, G. J. Reversible multiple time scale molecular dynamics. J. Chem. Phys. 97(3):1990–2001, 1992.

[25] 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.

[26] Andersen, H. C. Molecular dynamics simulations at constant pressure and/or temperature. J. Chem. Phys. 72:2384, 1980.

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

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

[29] Bussi, G., Donadio, D., Parrinello, M. Canonical sampling through velocity rescaling. J. Chem. Phys. 126:014101, 2007.

[30] 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.

[31] Basconi, J. E., Shirts, M. R. Effects of temperature control algorithms on transport properties and kinetics in molecular dynamics simulations. J. Chem. Theory Comput. 9(7):2887–2899, 2013.

[32] Cooke, B., Schmidler, S. J. Preserving the Boltzmann ensemble in replica-exchange molecular dynamics. J. Chem. Phys. 129:164112, 2008.

[33] Martyna, G. J., Klein, M. L., Tuckerman, M. E. Nosé-Hoover chains: The canonical ensemble via continuous dynamics. J. Chem. Phys. 97:2635–2643, 1992.

[34] Martyna, G. J., Tuckerman, M. E., Tobias, D. J., Klein, M. L. Explicit reversible integrators for extended systems dynamics. Mol. Phys. 87:1117–1157, 1996.

[35] Holian, B. L., Voter, A. F., Ravelo, R. Thermostatted molecular dynamics: How to avoid the Toda demon hidden in Nosé-Hoover dynamics. Phys. Rev. E 52(3):2338–2347, 1995.

[36] Eastwood, M. P., Stafford, K. A., Lippert, R. A., Jensen, M. O., Maragakis, P., Predescu, C., Dror, R. O., Shaw, D. E. Equipartition and the calculation of temperature in biomolecular simulations. J. Chem. Theory Comput. ASAP:DOI: 10.1021/ct9002916, 2010.

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

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

[39] Tuckerman, M. E., Alejandre, J., López-Rendón, R., Jochim, A. L., Martyna, G. J. A Liouville-operator derived measure-preserving integrator for molecular dynamics simulations in the isothermal-isobaric ensemble. J. Phys. A. 59:5629–5651, 2006.

[40] Yu, T.-Q., Alejandre, J., Lopez-Rendon, R., Martyna, G. J., Tuckerman, M. E. Measure-preserving integrators for molecular dynamics in the isothermal-isobaric ensemble derived from the liouville operator. Chem. Phys. 370:294–305, 2010.

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

[42] 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.

[43] 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.

[44] 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.

[45] 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.

[46] Andersen, H. C. RATTLE: A "Velocity" version of the SHAKE algorithm for molecular dynamics calculations. J. Comp. Phys. 52:24–34, 1983.

[47] 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.

[48] Hess, B. P-LINCS: A parallel linear constraint solver for molecular simulation. J. Chem. Theory Comput. 4:116–122, 2007.

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

[50] Goga, N., Rzepiela, A. J., de Vries, A. H., Marrink, S. J., Berendsen, H. J. C. Efficient algorithms for Langevin and DPD dynamics. J. Chem. Theory Comput. 8:3637–3649, 2012.

[51] Byrd, R. H., Lu, P., Nocedal, J. A limited memory algorithm for bound constrained optimization. SIAM J. Scientif. Statistic. Comput. 16:1190–1208, 1995.

[52] Zhu, C., Byrd, R. H., Nocedal, J. L-BFGS-B: Algorithm 778: L-BFGS-B, FORTRAN routines for large scale bound constrained optimization. ACM Trans. Math. Softw. 23:550–560, 1997.

[53] Levitt, M., Sander, C., Stern, P. S. The normal modes of a protein: Native bovine pancreatic trypsin inhibitor. Int. J. Quant. Chem: Quant. Biol. Symp. 10:181–199, 1983.

[54] 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.

[55] 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.

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

[57] Bennett, C. H. Efficient Estimation of Free Energy Differences from Monte Carlo Data. J. Comp. Phys. 22:245–268, 1976.

[58] Shirts, M. R., Chodera, J. D. Statistically optimal analysis of multiple equilibrium simulations. J. Chem. Phys. 129:124105, 2008.

[59] Hukushima, K., Nemoto, K. Exchange Monte Carlo Method and Application to Spin Glass Simulations. J. Phys. Soc. Jpn. 65:1604–1608, 1996.

[60] Sugita, Y., Okamoto, Y. Replica-exchange molecular dynamics method for protein folding. Chem. Phys. Lett. 314:141–151, 1999.

[61] Seibert, M., Patriksson, A., Hess, B., van der Spoel, D. Reproducible polypeptide folding and structure prediction using molecular dynamics simulations. J. Mol. Biol. 354:173–183, 2005.

[62] Okabe, T., Kawata, M., Okamoto, Y., Mikami, M. Replica-exchange Monte Carlo method for the isobaric-isothermal ensemble. Chem. Phys. Lett. 335:435–439, 2001.

[63] Chodera, J. D., Shirts, M. R. Replica exchange and expanded ensemble simulations as gibbs sampling: Simple improvements for enhanced mixing. J. Chem. Phys. 135:194110, 2011.

[64] 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.

[65] 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.

[66] Lange, O. E., Schafer, L. V., Grubmuller, H. Flooding in GROMACS: Accelerated barrier crossings in molecular dynamics. J. Comp. Chem. 27:1693–1702, 2006.

[67] Lyubartsev, A. P., Martsinovski, A. A., Shevkunov, S. V., Vorontsov-Velyaminov, P. N. New approach to Monte Carlo calculation of the free energy: Method of expanded ensembles. J. Chem. Phys. 96:1776–1783, 1992.

[68] Liem, S. Y., Brown, D., Clarke, J. H. R. Molecular dynamics simulations on distributed memory machines. Comput. Phys. Commun. 67(2):261–267, 1991.

[69] Bowers, K. J., Dror, R. O., Shaw, D. E. The midpoint method for parallelization of particle simulations. J. Chem. Phys. 124(18):184109–184109, 2006.

[70] Qiu, D., Shenkin, P., Hollinger, F., Still, W. The GB/SA Continuum Model for Solvation. A Fast Analytical Method for the Calculation of Approximate Born Radii. J. Phys. Chem. A. 101:3005–3014, 1997.

[71] Hawkins, D., Cramer, C., Truhlar, D. Parametrized Models of Aqueous Free Energies of Solvation Based on Pairwise Descreening of Solute Atomic Charges from a Dielectric Medium. J. Phys. Chem. 100:19824–19839, 1996.

[72] Onufriev, A., Bashford, D., Case, D. Exploring protein native states and large-scale conformational changes with a modified Generalized Born model. PROTEINS: Struct. Funct. Gen. 55(2):383–394, 2004.

[73] Larsson, P., Lindahl, E. A High-Performance Parallel-Generalized Born Implementation Enabled by Tabulated Interaction Rescaling. J. Comp. Chem. 31(14):2593–2600, 2010.

[74] Schaefer, M., Bartels, C., Karplus, M. Solution conformations and thermodynamics of structured peptides: molecular dynamics simulation with an implicit solvation model. J. Mol. Biol. 284(3):835–848, 1998.

[75] 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.

[76] van der Spoel, D., van Maaren, P. J. The origin of layer structure artifacts in simulations of liquid water. J. Chem. Theory Comput. 2:1–11, 2006.

[77] 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.

[78] 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.

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

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

[81] 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.

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

[83] Warner Jr., H. R. Kinetic theory and rheology of dilute suspensions of finitely extendible dumbbells. Ind. Eng. Chem. Fundam. 11(3):379–387, 1972.

[84] Bulacu, M., Goga, N., Zhao, W., Rossi, G., Monticelli, L., Periole, X., Tieleman, D., Marrink, S. Improved angle potentials for coarse-grained molecular dynamics simulations. J. Chem. Phys. 123(11).

[85] Brooks, B. R., Bruccoleri, R. E., Olafson, B. D., States, D. J., Swaminathan, S., Karplus, M. CHARMM: a program for macromolecular energy, minimization, and dynamics calculation. J. Comp. Chem. 4:187–217, 1983.

[86] Lawrence, C. P., Skinner, J. L. Flexible TIP4P model for molecular dynamics simulation of liquid water. Chem. Phys. Lett. 372:842–847, 2003.

[87] Jorgensen, W. L., Tirado-Rives, J. Potential energy functions for atomic-level simulations of water and organic and biomolecular systems. Proc. Natl. Acad. Sci. USA 102:6665–6670, 2005.

[88] Bulacu, M., van der Giessen, E. Effect of bending and torsion rigidity on self-diffusion in polymer melts: A molecular-dynamics study. JCTC 9(8):3282–3292, 2013.

[89] Scott, R. A., Scheraga, H. Conformational analysis of macromolecules. J. Chem. Phys. 44:3054–3069, 1966.

[90] Pauling, L. The nature of chemical bond. Ithaca and New York: Cornell University Press. 1960.

[91] 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.

[92] Hess, B., Scheek, R. M. Orientation restraints in molecular dynamics simulations using time and ensemble averaging. J. Magn. Reson. 164:19–27, 2003.

[93] Thole, B. T. Molecular polarizabilities with a modified dipole interaction. Chem. Phys. 59:341–345, 1981.

[94] Lamoureux, G., Roux, B. Modeling induced polarization with classical drude oscillators: Theory and molecular dynamics simulation algorithm. J. Chem. Phys. 119:3025–3039, 2003.

[95] Lamoureux, G., MacKerell, A. D., Roux, B. A simple polarizable model of water based on classical drude oscillators. J. Chem. Phys. 119:5185–5197, 2003.

[96] Noskov, S. Y., Lamoureux, G., Roux, B. Molecular dynamics study of hydration in ethanol-water mixtures using a polarizable force field. J. Phys. Chem. B. 109:6705–6713, 2005.

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

[98] Beutler, T. C., Mark, A. E., van Schaik, R. C., Greber, P. R., van Gunsteren, W. F. Avoiding singularities and numerical instabilities in free energy calculations based on molecular simulations. Chem. Phys. Lett. 222:529–539, 1994.

[99] Pham, T. T., Shirts, M. R. Identifying low variance pathways for free energy calculations of molecular transformations in solution phase. J. Chem. Phys. 135:034114, 2011.

[100] Pham, T. T., Shirts, M. R. Optimal pairwise and non-pairwise alchemical pathways for free energy calculations of molecular transformation in solution phase. J. Chem. Phys. 136:124120, 2012.

[101] 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.

[102] 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.

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

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

[105] Ballenegger, V., Cerdà, J. J., Holm, C. How to convert SPME to P3M: Influence functions and error estimates. J. Chem. Theory Comput. 8(3):936–947, 2012.

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

[107] Wennberg, C. L., Murtola, T., Hess, B., Lindahl, E. Lennard-Jones Lattice Summation in Bilayer Simulations Has Critical Effects on Surface Tension and Lipid Properties. J. Chem. Theory Comput. 9:3527–3537, 2013.

[108] Oostenbrink, C., Villa, A., Mark, A. E., Van Gunsteren, W. F. A biomolecular force field based on the free enthalpy of hydration and solvation: The GROMOS force-field parameter sets 53A5 and 53A6. Journal of Computational Chemistry 25(13):1656–1676, 2004.

[109] Cornell, W. D., Cieplak, P., Bayly, C. I., Gould, I. R., Merz, K. R. Jr., Ferguson, D. M., Spellmeyer, D. C., Fox, T., Caldwell, J. W., Kollman, P. A. A Second Generation Force Field for the Simulation of Proteins, Nucleic Acids, and Organic Molecules. J. Am. Chem. Soc. 117(19):5179–5197, 1995.

[110] Kollman, P. A. Advances and Continuing Challenges in Achieving Realistic and Predictive Simulations of the Properties of Organic and Biological Molecules. Acc. Chem. Res. 29(10):461–469, 1996.

[111] Wang, J., Cieplak, P., Kollman, P. A. How Well Does a Restrained Electrostatic Potential (RESP) Model Perform in Calculating Conformational Energies of Organic and Biological Molecules? J. Comp. Chem. 21(12):1049–1074, 2000.

[112] Hornak, V., Abel, R., Okur, A., Strockbine, B., Roitberg, A., Simmerling, C. Comparison of Multiple Amber Force Fields and Development of Improved Protein Backbone Parameters. PROTEINS: Struct. Funct. Gen. 65:712–725, 2006.

[113] Lindorff-Larsen, K., Piana, S., Palmo, K., Maragakis, P., Klepeis, J. L., Dorr, R. O., Shaw, D. E. Improved side-chain torsion potentials for the AMBER ff99SB protein force field. PROTEINS: Struct. Funct. Gen. 78:1950–1958, 2010.

[114] Duan, Y., Wu, C., Chowdhury, S., Lee, M. C., Xiong, G., Zhang, W., Yang, R., Cieplak, P., Luo, R., Lee, T., Caldwell, J., Wang, J., Kollman, P. A Point-Charge Force Field for Molecular Mechanics Simulations of Proteins Based on Condensed-Phase Quantum Mechanical Calculations. J. Comp. Chem. 24(16):1999–2012, 2003.

[115] García, A. E., Sanbonmatsu, K. Y. α-Helical stabilization by side chain shielding of backbone hydrogen bonds. Proc. Natl. Acad. Sci. USA 99(5):2782–2787, 2002.

[116] MacKerell, J. A. D., Feig, M., Brooks III, C. L. Extending the treatment of backbone energetics in protein force fields: limitations of gas-phase quantum mechanics in reproducing protein conformational distributions in molecular dynamics simulations. J. Comp. Chem. 25(11):1400–15, 2004.

[117] MacKerell, A. D., Bashford, D., Bellott, Dunbrack, R. L., Evanseck, J. D., Field, M. J., Fischer, S., Gao, J., Guo, H., Ha, S., Joseph-McCarthy, D., Kuchnir, L., Kuczera, K., Lau, F. T. K., Mattos, C., Michnick, S., Ngo, T., Nguyen, D. T., Prodhom, B., Reiher, W. E., Roux, B., Schlenkrich, M., Smith, J. C., Stote, R., Straub, J., Watanabe, M., Wiorkiewicz-Kuczera, J., Yin, D., Karplus, M. All-atom empirical potential for molecular modeling and dynamics studies of proteins. J. Phys. Chem. B. 102(18):3586–3616, 1998.

[118] Feller, S. E., MacKerell, A. D. An improved empirical potential energy function for molecular simulations of phospholipids. J. Phys. Chem. B. 104(31):7510–7515, 2000.

[119] Foloppe, N., MacKerell, A. D. All-atom empirical force field for nucleic acids: I. Parameter optimization based on small molecule and condensed phase macromolecular target data. J. Comp. Chem. 21(2):86–104, 2000.

[120] MacKerell, A. D., Banavali, N. K. All-atom empirical force field for nucleic acids: II. application to molecular dynamics simulations of DNA and RNA in solution. J. Comp. Chem. 21(2):105–120, 2000.

[121] Bjelkmar, P., Larsson, P., Cuendet, M. A., Hess, B., Lindahl, E. Implementation of the CHARMM force field in GROMACS: Analysis of protein stability effects from correction maps, virtual interaction sites, and water models. J. Chem. Theory Comput. 6:459–466, 2010.

[122] Rühle, V., Junghans, C., Lukyanov, A., Kremer, K., Andrienko, D. Versatile Object-Oriented toolkit for Coarse-Graining applications. J. Chem. Theory Comput. 5(12):3211–3223, 2009.

[123] Bereau, T., Wang, Z.-J., Deserno, M. Solvent-free coarse-grained model for unbiased high-resolution protein-lipid interactions. (submitted).

[124] Wang, Z.-J., Deserno, M. A systematically coarse-grained solvent-free model for quantitative phospholipid bilayer simulations. J. Phys. Chem. B. 114(34):11207–11220, 2010.

[125] 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.

[126] IUPAC-IUB Commission on Biochemical Nomenclature. Abbreviations and Symbols for the Description of the Conformation of Polypeptide Chains. Tentative Rules (1969). Biochemistry 9:3471–3478, 1970.

[127] Mahoney, M. W., Jorgensen, W. L. A five-site model for liquid water and the reproduction of the density anomaly by rigid, nonpolarizable potential functions. J. Chem. Phys. 112:8910–8922, 2000.

[128] Ryckaert, J. P., Bellemans, A. Molecular dynamics of liquid alkanes. Far. Disc. Chem. Soc. 66:95–106, 1978.

[129] 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.

[130] 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.

[131] Neumann, R. M. Entropic approach to Brownian Movement. Am. J. Phys. 48:354–357, 1980.

[132] Jarzynski, C. Nonequilibrium equality for free energy differences. Phys. Rev. Lett. 78(14):2690 – 2693, 1997.

[133] O. Engin, M. S. A. Villa, Hess, B. Driving forces for adsorption of amphiphilic peptides to air-water interface. J. Phys. Chem. B.

[134] Kutzner, C., Czub, J., Grubmüller, H. Keep it flexible: Driving macromolecular rotary motions in atomistic simulations with GROMACS. J. Chem. Theory Comput. 7:1381–1393, 2011.

[135] Kutzner, C., Grubmüller, H., de Groot, B. L., Zachariae, U. Computational electrophysiology: the molecular dynamics of ion channel permeation and selectivity in atomistic detail. Biophys. J. 101:809–817, 2011.

[136] 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.

[137] Hess, B. Determining the shear viscosity of model liquids from molecular dynamics. J. Chem. Phys. 116:209–217, 2002.

[138] Dewar, M. J. S. Development and status of MINDO/3 and MNDO. J. Mol. Struct. 100:41, 1983.

[139] Guest, M. F., Harrison, R. J., van Lenthe, J. H., van Corler, L. C. H. Computational chemistry on the FPS-X64 scientific computers - Experience on single- and multi-processor systems. Theor. Chim. Act. 71:117, 1987.

[140] Frisch, M. J., Trucks, G. W., Schlegel, H. B., Scuseria, G. E., Robb, M. A., Cheeseman, J. R., Montgomery, J. A. Jr., Vreven, T., Kudin, K. N., Burant, J. C., Millam, J. M., Iyengar, S. S., Tomasi, J., Barone, V., Mennucci, B., Cossi, M., Scalmani, G., Rega, N., Petersson, G. A., Nakatsuji, H., Hada, M., Ehara, M., Toyota, K., Fukuda, R., Hasegawa, J., Ishida, M., Nakajima, T., Honda, Y., Kitao, O., Nakai, H., Klene, M., Li, X., Knox, J. E., Hratchian, H. P., Cross, J. B., Bakken, V., Adamo, C., Jaramillo, J., Gomperts, R., Stratmann, R. E., Yazyev, O., Austin, A. J., Cammi, R., Pomelli, C., Ochterski, J. W., Ayala, P. Y., Morokuma, K., Voth, G. A., Salvador, P., Dannenberg, J. J., Zakrzewski, V. G., Dapprich, S., Daniels, A. D., Strain, M. C., Farkas, O., Malick, D. K., Rabuck, A. D., Raghavachari, K., Foresman, J. B., Ortiz, J. V., Cui, Q., Baboul, A. G., Clifford, S., Cioslowski, J., Stefanov, B. B., Liu, G., Liashenko, A., Piskorz, P., Komaromi, I., Martin, R. L., Fox, D. J., Keith, T., Al-Laham, M. A., Peng, C. Y., Nanayakkara, A., Challacombe, M., Gill, P. M. W., Johnson, B., Chen, W., Wong, M. W., Gonzalez, C., Pople, J. A. Gaussian 03, Revision C.02. Gaussian, Inc., Wallingford, CT, 2004.

[141] Car, R., Parrinello, M. Unified approach for molecular dynamics and density-functional theory. Phys. Rev. Lett. 55:2471–2474, 1985.

[142] Field, M., Bash, P. A., Karplus, M. A combined quantum mechanical and molecular mechanical potential for molecular dynamics simulation. J. Comp. Chem. 11:700, 1990.

[143] Maseras, F., Morokuma, K. IMOMM: A New Ab Initio + Molecular Mechanics Geometry Optimization Scheme of Equilibrium Structures and Transition States. J. Comp. Chem. 16:1170–1179, 1995.

[144] Svensson, M., Humbel, S., Froes, R. D. J., Matsubara, T., Sieber, S., Morokuma, K. ONIOM a multilayered integrated MO + MM method for geometry optimizations and single point energy predictions. a test for Diels-Alder reactions and Pt(P(t-Bu)3)2 + H2 oxidative addition. J. Phys. Chem. 100:19357, 1996.

[145] Praprotnik, M., Delle Site, L., Kremer, K. Adaptive resolution molecular-dynamics simulation: Changing the degrees of freedom on the fly. J. Chem. Phys. 123:224106, 2005.

[146] Praprotnik, M., Delle Site, L., Kremer, K. Multiscale simulation of soft matter: From scale bridging to adaptive resolution. Annu. Rev. Phys. Chem. 59:545–571, 2008.

[147] Junghans, C., Poblete, S. A reference implementation of the adaptive resolution scheme in ESPResSo. Comp. Phys. Comm. 181:1449–1454, 2010.

[148] Fritsch, S., Junghans, C., Kremer, K. Structure formation of toluene around c60: Implementation of the adaptive resolution scheme (adress) into gromacs. J. Chem. Theory Comput. 8:398–403, 2012.

[149] Praprotnik, M., Poblete, S., Kremer, K. Statistical physics problems in adaptive resolution computer simulations of complex fluids. J. Stat. Phys. 145:946–966, 2011.

[150] Delle Site, L. Some fundamental problems for an energy-conserving adaptive-resolution molecular dynamics scheme. Phys. Rev. **E** 76.

[151] Poblete, S., Praprotnik, M., Kremer, K., Delle Site, L. Coupling different levels of resolution in molecular simulations. J. Chem. Phys. 132:114101, 2010.

[152] Fritsch, S., Poblete, S., Junghans, C., Ciccottii, G., Delle Site, L., Kremer, K. Adaptive resolution molecular dynamics simulation through coupling to an internal particle reservoir. Phys. Rev. Lett. 108:170602, 2012.

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

[154] 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.

[155] Smith, P. E., van Gunsteren, W. F. The Viscosity of SPC and SPC/E Water. Comp. Phys. Comm. 215:315–318, 1993.

[156] 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.

[157] 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.

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

[159] Hess, B. Convergence of sampling in protein simulations. Phys. Rev. **E** 65:031910, 2002.

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

[161] Mu, Y., Nguyen, P. H., Stock, G. Energy landscape of a small peptide revelaed by dihedral angle principal component analysis. PROTEINS: Struct. Funct. Gen. 58:45–52, 2005.

[162] van der Spoel, D., van Maaren, P. J., Larsson, P., Timneanu, N. Thermodynamics of hydrogen bonding in hydrophilic and hydrophobic media. J. Phys. Chem. B. 110:4393–4398, 2006.

[163] Luzar, A., Chandler, D. Hydrogen-bond kinetics in liquid water. Nature 379:55–57, 1996.

[164] Luzar, A. Resolving the hydrogen bond dynamics conundrum. J. Chem. Phys. 113:10663–10675, 2000.

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

[166] 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.

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

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

[169] 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.

Index

E

F

G

H

I

K

L

M

N

O

P

V

W

X