GROMACS: Fast, Flexible, and Free

DAVID VAN DER SPOEL,[1] ERIK LINDAHL,[2] BERK HESS,[3] GERRIT GROENHOF,[4]
ALAN E. MARK,[4] HERMAN J. C. BERENDSEN[4]

[1]*Department of Cell and Molecular Biology, Uppsala University, Husargatan 3,
Box 596, S-75124 Uppsala, Sweden*
[2]*Stockholm Bioinformatics Center, SCFAB, Stockholm University, SE-10691 Stockholm, Sweden*
[3]*Max-Planck Institut für Polymerforschung, Ackermannweg 10, D-55128 Mainz, Germany*
[4]*Groningen Biomolecular Sciences and Biotechnology Institute, University of Groningen,
Nijenborgh 4, NL-9747 AG Groningen, The Netherlands*

Abstract: This article describes the software suite GROMACS (Groningen MAchine for Chemical Simulation) that was developed at the University of Groningen, The Netherlands, in the early 1990s. The software, written in ANSI C, originates from a parallel hardware project, and is well suited for parallelization on processor clusters. By careful optimization of neighbor searching and of inner loop performance, GROMACS is a very fast program for molecular dynamics simulation. It does not have a force field of its own, but is compatible with GROMOS, OPLS, AMBER, and ENCAD force fields. In addition, it can handle polarizable shell models and flexible constraints. The program is versatile, as force routines can be added by the user, tabulated functions can be specified, and analyses can be easily customized. Nonequilibrium dynamics and free energy determinations are incorporated. Interfaces with popular quantum-chemical packages (MOPAC, GAMES-UK, GAUSSIAN) are provided to perform mixed MM/QM simulations. The package includes about 100 utility and analysis programs. GROMACS is in the public domain and distributed (with source code and documentation) under the GNU General Public License. It is maintained by a group of developers from the Universities of Groningen, Uppsala, and Stockholm, and the Max Planck Institute for Polymer Research in Mainz. Its Web site is http://www.gromacs.org.

© 2005 Wiley Periodicals, Inc. J Comput Chem 26: 1701–1718, 2005

Key words: GROMACS; molecular simulation software; molecular dynamics; free energy computation; parallel computation

Introduction

GROMACS is an acronym for GROningen MAchine for Chemical Simulation. This sounds strange for a software package, but the name arises from a collaborative project of our group in the Department of Chemistry at the University of Groningen, The Netherlands, with the Department of Computer Science in Groningen, in the early 1990s, to construct a dedicated parallel computer system for molecular simulation. Until that time the Fortran package GROMOS had been developed in our group, mainly by van Gunsteren, who continued the development of GROMOS after he moved to the ETH in Zürich, Switzerland, in 1990.[1]

The hardware project concerned the construction of a system of 32 processors (Intel i860, a 40 MHz, 80 Mflop RISC processor), communicating in a ring structure by bidirectional dual-memory access, and with a theoretical peak performance of 2.5 Gflop/s. Effective communication required a different method of neighbor searching and a distribution of the neighbor list over processors. In fact, a complete software renewal was necessary, and it was decided to write the GROMACS software in C. Communication was done in C-routines that made use of calls supplied by the PVM (Parallel Virtual Machine) library; later, this was replaced by MPI (Message Passing Initiative). These have proven to be good choices, because ANSI C and MPI became a standard, and most other languages popular with computer scientists at the time (as Pascal and OCCAM for parallel systems) have become obsolete. This renewal gave the opportunity to redesign the software and incorporate several innovations that simplified and accelerated the computation. (See refs. 2–5 for the GROMACS version 1.0 of the late 1994s.)

Correspondence to: H. J. C. Berendsen; e-mail: H.J.C.Berendsen@rug.nl

The hardware project was quite successful, and resulted in a prototype and a final machine (Amphisbaena,[6] constructed by CHESS Engineering B.V., Haarlem, The Netherlands). Its performance exceeded the vector supercomputers of the time, CRAY Y/MP and NEC SX-3, by a factor of 3 to 6, and it served the group well for many years. However, with the upcoming mass-produced personal computers in the later 90s, and the rapid increase in processor speed, systems quickly became technically obsolete and the construction of special-purpose machines was no longer a cost-effective option. Since then, the GROMACS software was further developed and optimized, especially for use on PC-clusters.

An early design decision was the choice to work with *particle decomposition* rather than *domain decomposition* to distribute work over the processors. In the latter case, spatial domains are assigned to processors, which enables finding spatial neighbors quickly by local communication only, but complications due to particles that move over spatial boundaries are considerable. Domain decomposition is a better choice only when linear system size considerably exceeds the range of interaction, which is seldom the case in molecular dynamics. With particle decomposition each processor computes the forces and coordinate/velocity updates for an assigned fraction of the particles, using a precomputed neighborlist evenly distributed over processors.[7] The force F_{ij} arising from the pair interaction between particles i and j, which is needed for the velocity update of both particles i and j, is computed only once and communicated to other processors. Every processor keeps in its local memory the complete coordinate set of the system rather than restricting storage to the coordinates it needs. This is simpler and saves communication overhead, while the memory claim is usually not a limiting factor at all, even for millions of particles. The neighborlist, on the other hand, which can contain up to 1000 times the number of particles, is distributed over the processors. Communication is essentially restricted to sending coordinates and forces once per time step around the processor ring. These choices have proven to be robust over time and easily applicable to modern processor clusters.

While retaining algorithmic choices with proven validity and efficiency from GROMOS,[8] such as leap-frog integration of the equations of motion,[9] double-range neighborlist using charge groups, incorporation of SHAKE[10,11] to maintain holonomic constraints, weak coupling to temperature and pressure baths,[12] and stochastic dynamics algorithms,[13,14] new concepts[15] could be incorporated. These included (1) the computation of the virial of the force as a single, rather than double sum over particles, thus increasing the performance by 40% by avoiding an inner loop calculation;[3,16] (2) the use of triclinic boxes, which encompass all possible periodic constructs,[17] and (3) the storage of the translation vector that defines the image of each particle in the neighborlist, thus avoiding conditional statements in the inner loop over particle pairs. Further efficiency-enhancing features, incorporated in version 3.0, are described in a review in 2001:[18] (4) the use of a special software routine to evaluate the inverse square root, (5) force/energy evaluation by cubic spline interpolation from tabulated values, (6) fast grid-based neighbor searching, (7) a new and more robust linear constraint solver LINCS,[19] and (8), the use of multimedia (3DNow! and SSE) instructions on Pentium (III and higher), Athlon, and Duron processors.

GROMACS does not have a force field of its own. It now supports the GROMOS96,[20] Encad,[21] and OPLS-AA[22,23] force fields, while Amber[24] is in a testing stage.

As GROMACS has been developed entirely in the course of publicly funded projects, it was decided to offer the software under the GNU General Public License (GPL).[25] This guarantees not only open access for the scientific community, but also availability of the source code. The latter we consider a prerequisite for good science; a scientist is obliged to report in full and take full responsibility for his or her work, which (s)he cannot do using black-box software and proprietary force field information. The GNU GPL allows the incorporation of GROMACS parts into software packages only if these are offered under GPL themselves.

The following sections describe the scientific concepts and the programming concepts on which GROMACS is based. Then we demonstrate the capabilities with some recent applications. Future developments are then indicated. The Appendix describes recent benchmarks and lists the capabilities of the utility and analysis tools provided with GROMACS.

Scientific Concepts

It is the aim of GROMACS to provide a versatile and efficient MD program with source code, especially directed towards the simulation of biological (macro)molecules in aqueous and membrane environments, and able to run on single processors as well as on parallel computer systems. It does not only provide microcanonical Hamiltonian mechanics, but also stochastic dynamics (SD) including Langevin and Brownian dynamics, and energy minimization (EM). Various coupling methods to temperature and pressure baths are included, also allowing anisotropic pressures and triclinic box changes in response to pressure tensor fluctuations. External forces can be applied to enforce nonequilibrium dynamics or "steered MD." Atoms can be organized in special groups for the purpose of selective participation in the dynamics or detailed analysis of energies. There are also provisions for the use of *virtual sites,* which are massless interaction sites constructed from atom positions. The program package includes a large variety of analysis tools, ranging from extensive graphical trajectory analyses to normal mode and principal component analysis of structural fluctuations. See the Appendix for a list of features.

System Definition

The system is defined by its size and shape, the number and types of molecules it contains, and the coordinates and velocities of all atoms. Velocities may also be generated from a Maxwellian distribution.

The usual geometry for the simulated system is a rectangular box with periodic boundary conditions. GROMACS allows a general triclinic box shape, which encompasses all possible space-filling constructs, including the approximately spherical *rhombic dodecahedron* and *truncated octahedron.* It is also possible to study an isolated system, either as such or by choosing a sufficiently large box. Recently, an algorithm has been published that constructs a minimal-volume triclinic box shape given a minimum distance between any atom of an arbitrary macromolecule and any

atom of any of its images.[26] Such optimization of solvent volume is to be used in conjunction with a simulation under rotational constraint.[27] It easily saves a factor of 2 in computer time spent on the solvent. These features are not yet incorporated into GROMACS.

Force Field

The force field used for intramolecular interactions is in principle not part of GROMACS, but the organization of force and energy evaluations, although versatile, does set limitations to the allowed types of force field. GROMACS is compatible with GROMOS-96,[20] including the 45a3 and 53a5/6 parameter sets,[28,29] Encad,[21] OPLS,[22,23] and AMBER,[24] and similar in structure to CHARMM.[30] Forces and energies are computed on the basis of three different types of interaction:

- *Bonded interactions* between two, three, or four particles, based on predefined, fixed lists. The two-particle bonded interactions include harmonic, cubic, and Morse potentials. The harmonic term $V_b(r_{ij}) = (k/2)(r_{ij} - b)^2$ can—as in GROMOS—also be replaced by its approximation $(k/8b^2)(r_{ij}^2 - b^2)^2$, which avoids a square-root operation. The three-particle bond angle interaction is harmonic, either in the angle θ itself (which gives a discontinuous force when θ reaches 180 degrees), or in the cosine of the angle. The four-particle dihedral interaction allows either a periodic function of the dihedral angle ϕ (as in GROMOS, AMBER, and CHARMM) in conjunction with a special 1–4 Lennard–Jones interaction, or a power series (up to the 5th power) of $\cos \phi$ without 1–4 interaction. The latter allows incorporation of the Ryckaert–Bellemans[31,32] potential for alkanes, but also the Fourier series used in OPLS. There is also a harmonic *improper dihedral* function to prevent deviations from a flat geometry or accidental jumps to mirror images. Some special interactions have been implemented, including a fourth-order polynomial for bond-angle potentials and the FENE (Finite Extendible Nonlinear Elastic) potential[33] for coarse-grained polymer simulations $V(r) = -0.5kb^2 \log(1 - r^2/b^2)$.

Instead of applying interaction functions, bond lengths and angles can be *constrained* to given values by applying the traditional SHAKE[10] or the faster and more robust LINCS[19] algorithm. For water the SETTLE[34] algorithm is used.

- *Nonbonded interactions* between particle pairs, based on a regularly updated volatile list of pairs. These interactions are pair-additive and centro-symmetric. They consist of either a Lennard–Jones 6–12 potential or a Buckingham potential, where the r^{-12} repulsion is replaced by an exponential term, and a Coulomb term. The Coulomb interaction has a fixed dielectric constant and can be modified by a *reaction field*,[35,36] mimicking the effect of a homogeneous dielectric environment beyond the cutoff radius, including the effect of ionic strength.[37] The nonbonded functions can be modified by a *shift function* or a *switch function* that causes forces and their derivatives to be continuous at the cutoff radius. For use in conjunction with Ewald summation (see below), the Coulomb interaction is modified with an error function. It is also possible to use arbitrary tabulated

functions for the nonbonded interactions: the energy values are obtained from cubic spline interpolation, and the forces are exact derivatives of the energy and have continuous derivatives themselves. This feature would, for example, allow a distance-dependent dielectric constant as is used in CHARMM.

Long-range interactions can be chosen in a variety of ways. A single or a twin-range cutoff can be employed, both preferably in conjunction with the use of neutral charge groups to define the cutoff; the latter prevents strong artifacts near the cutoff range. However, the cutoff range is not allowed to exceed half the smallest box size to prevent violation of the minimum image convention. Alternatively, and preferable for most applications, a *lattice sum* evaluation for the Coulomb forces can be chosen in conjunction with a properly modified short-range part. The original but inefficient Ewald summation[38] is included for checking purposes or for use on small systems; normally, one employs the efficient *Particle-Mesh-Ewald* (PME) method of Darden et al.[39,40] The full anisotropic virial is computed and charge and dipole corrections are made to the Ewald sum. A version of the *Particle-Particle Particle-Mesh* (PPPM or P^{3}M) method of Hockney and Eastwood[41] is still available in GROMACS, but is in practice superseded by PME.

- *Special interactions* can be defined to impose position, angle, or distance restraints on the motion of the system. *Position restraints* are harmonic interactions of specified atoms with fixed positions. The force constant can be specified for each spatial dimension. They can be used during equilibration procedures of "bad" structures, such as a macromolecule put into a hole cut from a water configuration: position-restraining the macromolecular atoms allows the water molecules to find reasonable configurations and prevents drastic rearrangements within the macromolecule. They can also be used to define a shell of particles around a central region of interest with the aim to mimic an infinite environment with a limited number of particles; the force constant of the restraining potential should then vary from very weak near the central region to strong near the outer boundary. One should, of course, be aware of the limitations of such an environment that restricts diffusional freedom and lacks correct long-range interactions. *Angle restraints* can be put on the angle between two pairs of particles or between one pair of particles with the z-axis. They can be used to incorporate experimental data on orientation, if available. *Distance restraints* add a penalty to the potential energy when the distance between specified pairs of particles exceeds a threshold value. They are used to impose experimental restraints on the motion of the system, as obtained (usually) from Nuclear Overhauser Effect (NOE) data in high-resolution NMR. The penalty potential depends on three distance parameters r_0, r_1, and r_2: it is zero between r_0 and r_1, increases quadratically below r_0 and above r_1, until r_2, after which it increases linearly. Averaging over more than one pair of atoms is possible to allow for uncertain assignments in NMR (as two distinct methylene hydrogens) or for rapid motion (as for the protons in a rotating methyl group). Averaging of r^{-3} over time (with a specified time constant) is also possible, as is ensemble averaging over multiple copies of the simulated system.

Orientation restraints are another type of restraint that can be obtained from NMR experiments, but on partially ordered molecules. Such experiments include residual dipolar couplings and chemical-shift anisotropies. Vectors between atom pairs can be restrained with respect to the orientation of the molecule. As for distance restraints this can also be done with time and/or ensemble averaging.[42]

In addition to restraining potentials, it is also possible to define *freeze groups.* Particles that belong to a freeze group are kept stationary in the dynamics.

Polarizable Force Fields

Polarizable force fields can be incorporated into GROMACS if they make use of a *shell model*[43,44] or *charge-on-spring model.*[45] In these models a massless charge (the "shell") is connected to another particle by a harmonic interaction. The position of the shell is iteratively adjusted until the net force on the shell vanishes, thus introducing a dipole moment proportional to the local electric field acting on the shell. GROMACS has been used[46] to perform MD on the "Mobile Charge Densities in Harmonic Oscillators" (MCDHO) shell-type model for water of Saint-Martin et al.[47] GROMACS also allows, without any modification, to treat shell-polarizable models as extended systems by giving the shells a small mass and performing normal dynamics with small time steps, instead of iterating the shell positions. When the shells are kept cold, they move close to their minimum energy surface, similar to the behavior of wave function coefficients in Car–Parrinello *ab initio* MD.[48] If the shell charge is relatively large, and hence the shell displacement is small, the shell model is identical to a polarizable model based on induced dipoles.

GROMACS can be used to realize *flexible constraints.*[46] These are adjustments of atom positions (in bonds) such that the net force in the bond direction on the atom, including velocity-dependent forces as the centrifugal force, vanishes. This is the proper treatment for degrees of freedom as bond vibrations that are in their quantum-mechanical ground state. Not the atomic coordinate, nor the deviation from a constrained bond length, but the deviation from the flexible constraint position is a separable quantum coordinate.[46]

QM/MM Hybrid Interactions

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 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 (QM/MM).[49] The reacting parts of the system are treated quantum mechanically, with the remainder being modeled using the force field. In GROMACS, interactions between the two subsystems are either handled as described by Field et al.[50] or within the ONIOM approach by Svensson et al.[51]

In the first approach, there are electrostatic interactions between the electrons of the QM region and the MM nuclei, electrostatic interactions between QM and MM nuclei, and Lennard–Jones interactions between the QM and MM atoms. 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 in the QM region. The force on this cap atom, which is present in the QM calculation only, is distributed over the two atoms of the bond. The advantage of this method is that the MM environment can induce polarization in the QM region. However, this might not be compatible with the use of a nonpolarizable MM force field. Furthermore, the use of standard Lennard–Jones parameters for QM nuclei could easily lead to an overestimation of the polarization, as these parameters already include the polarization implicitly to some extent.

In the ONIOM approach the energy and gradients are first evaluated for the isolated QM subsystem at the desired level of *ab initio* theory. Then, the energy and gradients of the total system, including the QM subsystem, are calculated using a molecular mechanics force field and added to the energies and gradients just calculated for the isolated QM subsystem. Finally, to correct for counting the interactions inside the QM region twice, a force-field calculation is performed on the isolated QM subsystem and the energies and gradients are subtracted. The ONIOM scheme has the advantage that it is not restricted to a two-layer QM/MM model, but can handle more than two layers, with each layer described at a different level of theory. Furthermore, ONIOM does not suffer from the inconsistencies in the other QM/MM approach, but has the disadvantage that MM layers do not induce polarization of the QM region if the force field lacks polarization terms.

GROMACS provides both QM/MM approaches with interfaces to several Quantum Chemistry Packages (Mopac,[52] Gamess-UK,[53] Gaussian[54]).

Removing Fast Degrees of Freedom

Obtaining as much sampling as possible within the available computer time is essential if one is interested in conformational dynamics of macromolecules such as the folding of proteins. With GROMACS, several tricks can be played to obtain the longest possible time step without loosing (much) accuracy. Using a united-atom instead of an all-atom force field saves roughly a factor of 4 in the number of interactions (e.g., the nine interactions between two methylene groups are reduced to 2). Constraining bond lengths allows a time step of 2 fs instead of 0.5 fs or less. The maximum time step is limited by the shortest oscillation period found in the system, and with constrained bond lengths this turns out to be about 13 fs for bond-angle vibrations involving hydrogen atoms.[55] In many cases such bond-angle motions can be removed by defining the hydrogen as a *virtual* particle, adding its mass to the heavy atom it is connected to, and reconstructing the hydrogen position from the position of three nearby atoms. There are six
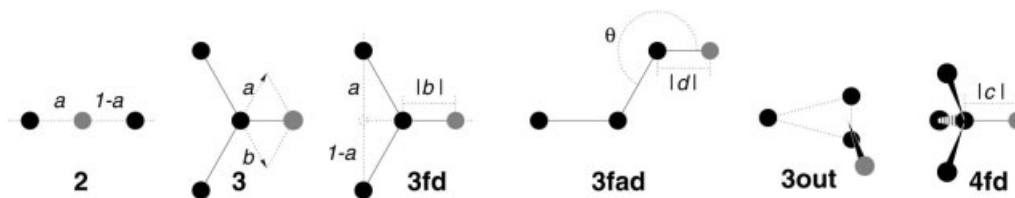
Figure 1. The six different types of virtual particle construction in GROMACS. The atoms are shown as black circles, the virtual particles in gray. The first four are in-plane constructions; the last two are out of plane.

virtual-site constructions defined in GROMACS (see Fig. 1) that can be used for this purpose. Other simplifying constructions are possible for planar ring systems. But when a free rotation is still possible, as an XOH-group that can rotate around the X—O bond, or a water molecule rotating around the symmetry axis of the molecule, the rotation cannot be quenched. The only option to slow down such motions is to increase the mass of the hydrogen atom at the expense of the mass of neighboring heavy atoms. For a classical system, the thermodynamic equilibrium properties do not depend on the distribution of masses of the particles, and increasing hydrogen masses (e.g., to 4) only influences the fast dynamics, but not the thermodynamics and slow dynamics of the system. Time steps can routinely be extended to 4 fs, and sometimes even longer.[55]

Dynamic Integration

The integration follows the leap-frog discretization, which is time-reversible and symplectic,[56] also when holonomic constraints are imposed.[57] These two properties are considered important for long-term stability of the simulation. An algorithm is symplectic when it preserves areas and volumes in phase space, which is a property of Hamiltonian mappings.

Microcanonical (i.e., constant total energy) Hamiltonian mechanics is very inconvenient for most applications, and a form of temperature control and often also of pressure control is desirable. In a microcanonical simulation the temperature will change (mostly increase) during equilibration, during conformational changes, and as a result of errors in the integration and force evaluation. Of the various thermostatting methods, the *weak-coupling*[12] and the Nosé–Hoover[58–60] methods are included in GROMACS, while coupling to a stochastic bath can be accomplished by standard Langevin dynamics, incorporated into GROMACS with the third-order algorithm of ref. 14 that reduces to the leap-frog scheme in the case of zero friction. With weak coupling the system responds smoothly with a first-order exponential decay to disturbances, as does stochastic coupling; Nosé–Hoover thermostats involve a quite undesirable second-order oscillatory response (see Fig. 2).[61] Langevin dynamics produces a canonical ensemble, but imposes an extra friction that influences dynamic properties;[61,62] Nosé–Hoover coupling produces a canonical ensemble in configurational space, but not in the full phase space of the particles, and weak coupling produces an ensemble in configurational space that is intermediate between microcanonical (no coupling) and canonical (very strong coupling with time constant

equal to the time step). The latter case corresponds to the simple velocity-scaling method of Woodcock.[63] Morishita[64] has derived distribution functions and fluctuation equations for intermediate cases of weak coupling. Under the weak coupling scheme there is a very slow transfer of energy to collective degrees of freedom that are uncoupled from the system, as the total linear momentum. In long simulations this may lead to the "flying ice cube" effect;[65] a cold system with a collective velocity. In GROMACS, this effect is prevented by removing any center-of-mass motion.

Nonequilibrium Dynamics

It is possible to exert external forces on groups of atoms, to perform nonequilibrium simulations. Defined groups of atoms can each be subjected to a (constant) *acceleration,* which allows the determination of transport properties by linear response theory[66] or the perturbation of a system in a desired direction. A special case is implemented for the determination of viscosities: a transverse cosine acceleration profile can be imposed, with acceleration in the x-direction of the ith particle proportional to $\cos kz_i$, where $k = 2\pi/L_z$ and L_z is the box height. This perturbation fits the period-
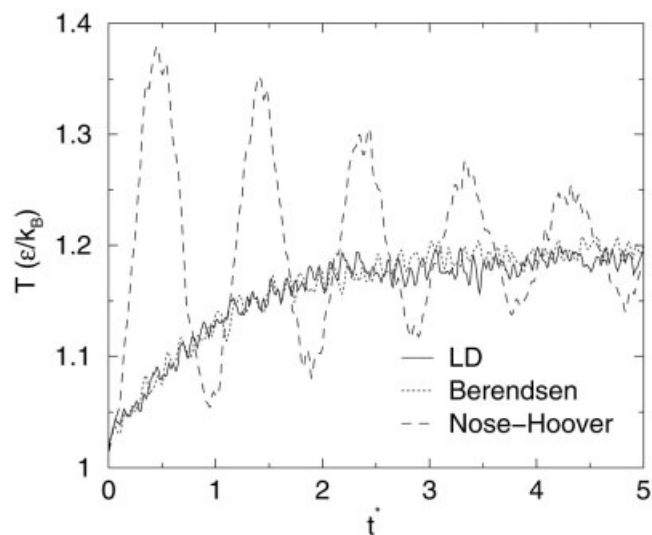

Figure 2. Response of three different thermostats to a temperature jump from 1 to 1.2 for a system of 8000 Lennard–Jones particles. Time and temperature are in reduced units (from Hess[61]).

icity of the box and produces a steady-state velocity profile, which is monitored by Fourier transforming the particle velocities. This velocity profile is subtracted from the velocities to determine temperatures, and the profile is not subjected to thermostating. The method, which has been known since 1973,[66–69] yields the shear viscosity η from the amplitude of the velocity profile. The method has been implemented into GROMACS and compared to other methods, such as pressure or momentum fluctuations and sliding boundary conditions,[61,70] with the conclusion that (for water) the periodic nonequilibrium method is to be preferred because it is more accurate than the fluctuation methods and does not share some problems of the sliding boundary method.

GROMACS also allows the incorporation of an electric field $E(t)$ (at present only a homogeneous field is allowed), by imposing an acceleration $q_i E(t)/m_i$ on each particle. The field can be chosen stationary or periodic in time, the latter optionally damped with a Gaussian function. This allows the measurement of dielectric or conductivity responses in a system. For an application involving the voltage-dependent insertion of alamethicin in interfaces (see Tieleman et al.[71]).

Several methods that have been incorporated to facilitate free energy calculations (see the next subsection), can be used to impose nonequilibrium processes and realize a variety of "steered" dynamics. These include constraining the distance between the centers of mass of two groups of atoms, imposing an umbrella potential, and AFM pulling by a moving spring connected to an atom.

Free Energy

There are two types of free energy, which one would like to obtain from simulations. The first type is a *difference* in Gibbs free energy between two thermodynamic states of a system, such as a macromolecule with bound ligand in solution *minus* that macromolecule in solution plus a free ligand in solution with standard concentration, all at the same temperature and pressure, yielding the binding constant of the ligand. Through the use of thermodynamic cycles (see, e.g., ref. 72), such questions can be reduced to the difference in Gibbs free energy between two systems with different Hamiltonians. The two systems may have a different atomic composition (with a physically impossible "alchemical" transition and meaningless free-energy difference). The second type is really a *potential of mean force* and concerns a *free energy profile* as a function of some (one or more) reaction coordinate(s) that can be treated as a restraint or constraint in the system.

GROMACS allows thermodynamic integration from system A to system B using a *coupling parameter* λ, defined in such a way that $\lambda = 0$ corresponds to system A and $\lambda = 1$ corresponds to B. The value of λ can be manipulated during a MD run: it is possible to perform equilibrium runs for given values of λ, and collect free energy derivatives, but it is also possible to perform a slow-growth integration, changing λ gradually during the simulation. During the λ-path *soft-core* potentials may be defined for the nonbonded interactions, to avoid unnecessary barriers along the path. The soft-core potentials modify the interaction distance in the nonbonded Lennard–Jones and Coulomb interactions in such a way that the potentials are unmodified at $\lambda = 0$ or 1, but have the singularities at zero distance removed for intermediate values of

λ.[18,73] Differences in constraints or in long-range interactions are properly handled.

GROMACS supports three different ways to compute potentials of mean force:

AFM Pulling

An atom or group of atoms is connected to a slowly retracting spring. The rate constant and spring constant can be varied to study, for example, the unbinding of a ligand from a protein.

Constraint Force

The distance between the centers of mass of two groups of atoms can be constrained and the constraint force monitored.

Umbrella Sampling

A simple umbrella potential can be inserted, with a harmonic umbrella potential that acts on the center of mass of a group of atoms.

In all three cases the distance can be defined in one, two, or three dimensions. Integration of the measured forces during equilibrium (or very slow) simulations yields the change in potential of mean force. When the change is not slow, the measured work W done exceeds the free energy change; in principle, the change in potential of mean force V^{mf} can then be extracted from an ensemble-average using Jarzynski's equation:[74,75]

$$\Delta V^{\text{mf}} = -k_B T \ln\langle \exp(-W/k_B T)\rangle, \tag{1}$$

where the averaging is over an equilibrium ensemble of initial states. However, in practice, the exponential averaging yields very poor statistics[76] and the available computer time is better spent on approximating an equilibrium path.

The use of distances between centers of mass of groups of particles guarantees that no metric tensor corrections to the constraint force[77–79] are necessary.

A Note on Units

GROMACS uses the following convenient and consistent set of units, of which the first four are chosen[80] and the others derived:

mass: u (unified atomic mass unit, $1.660\,538\,86(28) \times 10^{-27}$ kg)
length: nm (nanometer, 10^{-9} m)
time: ps (picosecond, 10^{-12} s)
charge: e (elementary charge, $1.602\,176\,53(14) \times 10^{-19}$ C)
energy: kJ/mol (kiloJoule per mol, $1.660\,538\,86(28) \times 10^{-21}$ J)
force: kJ mol^{-1} nm^{-1} ($1.660\,538\,86(28) \times 10^{-12}$ N)
pressure: kJ mol^{-1} nm^{-3} ($1.660\,538\,86(28) \times 10^{6}$ Pa ≈ 16.6 bar)
electric factor: $f_{\text{el}} = 138.935\,457(12)$ kJ mol^{-1} nm e^{-2}, as in $V = f_{\text{el}} q^2/r$
electric potential: kJ mol^{-1} e^{-1} = $0.010\,364\,269(1)$ Volt, as in $\Phi = f_{\text{el}} q/r$

Pressure is communicated in bar. Units as Ångström and kcal/mol can be used as inputs, but they are converted to the units given

above. With apology to our kcal-loving US friends, we do not recommend the use of such non-SI units, which are now practically obsolete in European education.

Implementation and Programming

GROMACS is written primarily in C, and an ANSI C compiler is, in fact, the only *requirement* for building the package. The reasons for this are partly historical; when GROMACS was first conceived of in the early 1990s, scientific code was almost universally written in Fortran for performance reasons. A key idea of the GROMACS project was, however, to develop a portable and efficient parallel molecular dynamics implementation,[5] and to facilitate interaction with communication libraries the C language was chosen for the project. This was not an easy decision, because C performance was still significantly behind Fortran, but it was outweighed by benefits such as structured programming, abstract data types, and fully dynamic memory allocation, which has enabled a faster and more efficient development.

The performance penalty of the C code was addressed in 1995 by isolating the handful of speed-sensitive inner loops and rewriting them in Fortran. Linking the two languages is trivial, but recent C compilers have largely closed the gap to Fortran, so the choice is entirely optional. This mixed layout has been quite successful and provides high performance in the core code, while the majority of the program is maintained in a higher level language. In recent versions of GROMACS, this idea has been extended further with manually tuned assembly inner loops for most common architectures.[18]

Portability

The ability to build GROMACS with only a C compiler greatly facilitates porting. Because essentially all operating system kernels are written in C, compilers are present even on the most esoteric hardware imaginable. For example, the GROMACS project initially included dedicated hardware using 32 Intel i860 CPUs,[2,4] and recently the software has been ported to embedded architectures with extremely low cost and power requirements (e.g., Fujitsu FR-V 555 and ClearSpeed CS301).

GROMACS makes full use of GNU-style autoconfiguration scripts, which means the source will configure and compile entirely automatically on any POSIX-like architecture, including systems where it has not been run before. Installing GROMACS from source is literally as easy as

```
$ tar -xzf gromacs-3.3.tar.gz $ cd gromacs-3.3
  $ ./configure ;
make ; make install
```

If *any* ANSI-compliant C compiler fails to build the package it is considered a high-priority bug, which will be fixed in the next patch release. The standard configuration scheme has also made it straightforward to provide RPM and other binary packages for several architectures at the project Web site http://www.gromacs.org. GROMACS can also be compiled as a native Microsoft Windows program by using the Visual Studio Project files available from the ftp site.

Fast Fourier Transforms are used heavily both in the algorithms PME[39,40] and PPPM,[41] as well as several analysis programs. For this, GROMACS relies on the freely available FFTW library (Fastest Fourier Transform in the West[81]), which has proven to be one of the fastest implementations available. Parallel runs require a standard MPI communication library, for example, the free LAM-MPI[82] or MPICH[83] implementations. The GROMACS libraries have recently been overhauled and made fully reentrant. This makes it safe to call GROMACS routines from multithreaded applications, and future program versions will support multithreaded SMP parallelization of GROMACS itself as well as a twin-level mixed thread/MPI parallelization mode.

The User Interface Layer

A guiding principle for GROMACS has been to create programs that mimic the way the standard UNIX environment works. Instead of a complex scripting language there is a large collection of programs each responsible for a specific task, and options are controlled with command line arguments. In the source, this has been implemented as a separate virtual interface layer. When a program starts it only needs to specify the type of input and output it accepts, and then call a single routine to do all the argument parsing. This provides the user with a highly consistent interface and enables some features like automatically running heavy programs at low priority, but more important, it makes it possible to automatically extract documentation strings for the program and options from the source code. The resulting help information is always available through the "-h" option; it can be viewed on the UNIX man page generated for each program, in the GROMACS manual, or why not the online HTML documentation—all guaranteed to be consistent and up to date with respect to the source. The file options are also used to generate shell completions for bash, tcsh, and zsh: after adding an option (e.g., "-p" to read a topology) you can just hit tab and get a list of all files in the current director that are compatible with the option in question.

The interface layer is not limited to simple argument parsing. It has also been used to implement an X11/Motif graphical interface to the programs. This extension is enabled automatically if the appropriate libraries are found at compile time, and provides dialog boxes and context-sensitive help without adding a single line of code outside the interface layer. The whole concept is also equally useful for output data: many programs automatically generate postscript (EPS), pixmap (XPM), structures (e.g., PDB), or finished Xmgrace graphs (XVG), and the corresponding viewer applications can be configured to start automatically.

Programs have been designed with simplicity but generality in mind; by default, tasks like topology building from coordinate files is completely automatic for all force fields in the distribution, but by adding command-line switches the user can choose to select protonation states, disulphide bridges, or terminus types interactively. Virtual interaction sites can be generated on the fly without any user interaction, and only run parameters that differ from their default values have to be specified. Finally, the programs do rather excessive amounts of double and triple checking to ensure input data and parameters are consistent and correct.

Code Structure and Algorithms

The GROMACS source is written with maintainability in mind, and makes heavy use of structures and abstract data types to enforce encapsulation. The design is largely object-oriented (despite using C instead of C++ for portability reasons); concepts such as the topology, trajectory, or system state are normally handled as abstract objects with dedicated manipulation functions. The only significant exception to this is the core nonbonded kernels where the goal has instead been to maximize performance at any cost.

GROMACS can be compiled in either single or double precision by adding a single flag to the configuration script. By taking care of summation order and sometimes using intermediate double precision variables it has been possible to use single precision by default while still conserving energy perfectly in microcanonical ensemble runs. Modern CPU architectures are equally fast for single and double precision calculations, but the reduced amount of data doubles cache efficiency and has enabled the use of multimedia instructions that are often only available in single precision. The only remaining application where double precision is strongly recommended is energy minimization performed before a normal mode calculation,[84] to make sure the Hessian matrix is positive definite (no negative eigenvalues).

One of the most common tasks in a simulation or analysis is to perform operations on a subset of the system, for example, to restrain alpha carbons or calculate the diffusion of a certain molecule. GROMACS supports this by introducing a general *index group* concept throughout the source code. This enables default features like limiting trajectory output to a certain subgroup and separating all nonbonded energies into pairwise contributions. All routines that perform actions like restraining, pulling, or rotating atoms have an optional index group argument to restrict the action to those atoms, and when calculating RMS (root-mean-square) differences one group is used for fitting and an arbitrary list of groups for the evaluation.

The GROMACS simulation box and periodic boundary conditions are universally implemented as triclinic matrix operations. Because any crystal type can be represented with a triclinic cell, this means GROMACS is entirely general when it comes to the system shape. All routines work equally well with, for example, a truncated octahedron or dodecahedron periodic boundary condition. Both the grid-based neighbor searching and nonbonded interactions are, for instance, equally fast regardless of the system shape, and by using anisotropic pressure coupling it is even possible to simulate free or forced transitions between different crystal cell geometries.[85,86]

The code only makes very general assumptions about potential functions for particle interactions. *Bonded interactions* are simply defined by one or more atoms, an enumerated index for the type of the interaction, and a set of parameters. This includes bonds, angles, and torsion terms, but also, for example, water polarization shells, position, and distance restraints. The code already supports 17 different bonded interaction forms, but all that is required to extend it further is to (1) write the code to calculate the new interaction as a separate function, and (2) add the type, name, and parameters to the list of interactions. The whole propagation of data from the force field files to the routine where it is evaluated will be handled automatically.

Nonbonded interactions are currently limited to pairwise contributions, defined through dynamic neighbor lists.[87] The group concept makes it trivial to make exceptions in the neighbor searching, for instance, to exclude all interactions in a frozen part of the system. Simple potentials such as standard Coulomb, Reaction-Field,[35] Lennard–Jones, and Buckingham are implemented in analytical form, and more complex interactions (including user-defined ones) are supported through an efficient implementation of cubic spline interpolation with flexible accuracy. These tabulated interactions are only about 30% slower than the simplest possible Coulomb potential ($1/r$) in GROMACS, so the only reason for writing new analytical interactions is if they for some reason cannot be formulated with tables. On the code level, each individual interaction can have a unique set of nonbonded parameters, but they have usually been calculated from combination rules and/or scaling factors in the topology.

The Shift Concept and Single-Sum Virial

The computationally most expensive part of molecular simulations is the evaluation of nonbonded forces. Because most are performed under periodic boundary conditions and minimum image convention, a significant part of the time for each pairwise interaction is often spent on calculating which periodic image is closest. Triclinic boundary conditions require almost twice as many floating-point operations as rectangular boxes, not to mention that the necessary conditional statements are particularly bad on superscalar CPUs where they inhibit instruction pipelining.

In GROMACS, these problems were worked around and performance improved significantly by introducing the concept of *nonbonded shift* during neighbor searching. Instead of calculating the closest image of each neighbor *j*-particle to the *i*-particle "owning" a neighborlist, the *i*-particle itself is iteratively placed in each possible image location and all local *j*-particle neighbors added to a neighborlist where the translational shift of the *i*-particle is also recorded. Figure 3 illustrates the process for an imaginary two-dimensional system. Theoretically, there are up to 11 possible such shifts vectors for a triclinic cell, but in practice, many of them fall outside the list cutoff, so particles only have neighbors for a handful of different shifts.

The main advantage is, however, not a slightly more efficient neighbor searching, but the idea that the different images of the *i*-particle can be treated as separate neighborlists with fixed shifts. For each such list, the *i*-particle is first shifted to the correct image position, and all nonbonded interactions evaluated without having to calculate any periodic images in the innermost loop over neighbors. In practice, the translation vectors are stored as relative translation indices to account for changing cell shapes when pressure coupling is enabled. This idea removes not only the conditional statements from the inner loop, but all references to the periodic boundary conditions—including the extra cost of triclinic instead of rectangular cell geometry.

The shift vectors further make it possible to completely extract the evaluation of the virial tensor Ξ from the inner loops.[3,16] With the minimum image vector distance defined as $\mathbf{r}_{ij}'' = (\mathbf{r}_i + \delta_i - $
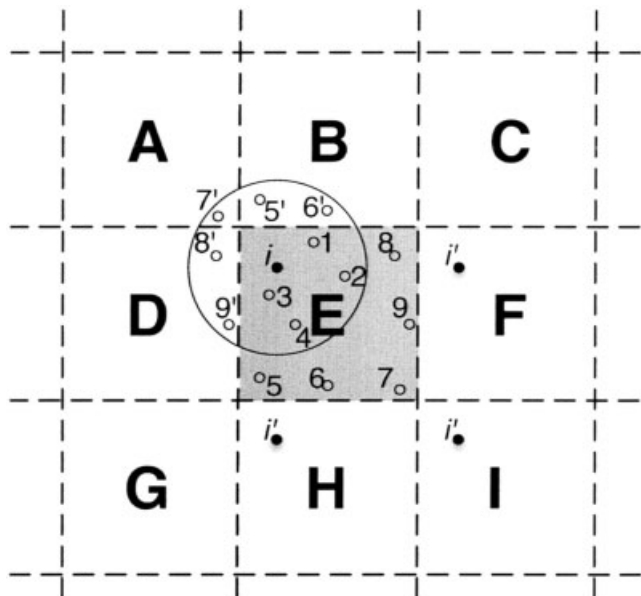
Figure 3. The shift concept for nonbonded interactions. The actual simulation cell (*E*) is shaded, but due to periodic boundary conditions the particle *i* sometimes interacts with periodic images (primed numbers) of the original locations. Instead of calculating the image for each pairwise evaluation, an iteration over image locations for the main particle is done and *j* particle indices added to neighborlists with separate shift indices. In this case, the particle *i* would have neighbors 1, 2, 3, 4 for central shift (cell **E**); neighbors 8, 9 when shifted to cell **F**; neighbors 5, 6 when shifted to cell **H**; and finally, neighbor 7 when shifted to cell **I**. When evaluating nonbonded energies in each list, the *i* particle is moved to the shift location and all local interactions calculated without having to check which image is closer. Note that real three-dimensional neighborlists will be much longer.

$\mathbf{r}_j$), where δ_i is the shift vector and the pairwise force $\mathbf{F}_{ij}$, it follows that

$$\Xi = -\frac{1}{2} \sum_{i<j}^{N} \mathbf{r}_{ij}^n \otimes \mathbf{F}_{ij}$$

$$= -\frac{1}{4} \sum_{i=1}^{N} \sum_{j=1}^{N} (\mathbf{r}_i + \delta_i) \otimes \mathbf{F}_{ij} - \mathbf{r}_j \otimes \mathbf{F}_{ij}$$

$$= -\frac{1}{4} \left[\sum_{i=1}^{N} (\mathbf{r}_i + \delta_i) \otimes \sum_{j=1}^{N} \mathbf{F}_{ij} - \sum_{j=1}^{N} \mathbf{r}_j \otimes \sum_{i=1}^{N} \mathbf{F}_{ij} \right]$$

$$= -\frac{1}{4} \left[2 \sum_{i=1}^{N} \mathbf{r}_i \otimes \mathbf{F}_i + \sum_{i=1}^{N} \delta_i i \otimes \mathbf{F}_i \right]. \tag{2}$$

The virial can thus be expressed as a combination of single sums over the particle positions and particle shifts in vector products with the total force on each particle. No evaluation whatsoever

has to be done in the innermost loop over neighbors, and in the outer loop it is only necessary to sum the total force for each shift. This saves another nine multiplications and nine additions per interaction, and results in roughly 40% performance improvement while the full accuracy virial is still obtained for pressure calculation.

A similar *bonded shift* concept is used for all bonded interactions in GROMACS, but because those account for a much smaller fraction of the run time the performance difference is less dramatic, although it hides the additional complexity of the triclinic boundary conditions.

Optimization of the Square Root Operation

Even after extracting conditionals and the virial calculation, the inner loops still account for the largest fraction of simulation runtime. It turns out that the single most expensive floating-point operation is the calculation of $1/r$ from the squared distance r^2. This corresponds to first taking a square root and then reciprocal of the result. Division and square roots are incredibly slow on most hardware, and to make things worse, the square root is calculated by taking the inverse square root and then multiplying with the initial value; in other words, the reciprocal is entirely superfluous!

In practice, inverse square roots are calculated by performing a limited-accuracy table lookup followed by several steps of Newton–Raphson refinement (see, e.g., Numerical Recipes[88]). Even CPUs that provide hardware SQRT instructions use a microcoded version of this algorithm, but it is usually not pipelined and thus very slow.

To improve this, GROMACS performs the inverse square root operation in software on platforms where it is faster. Because the operation will be performed billions of times, a larger table is used (wasting some cache) to get 12 bits of accuracy in the initial lookup value a. The input argument does not have to be checked because r^2 cannot be negative, and a single step of Newton–Raphson refinement is enough to achieve single-precision accuracy if rare errors in the least significant bit are acceptable:

$$\frac{1}{r} = a(3 - ar^2)/2 \tag{3}$$

This more than doubles performance on processors without hardware square-root instructions, and even when double precision forces another iteration step it can be a significant speedup due to our larger table and fewer argument checks.

Specialized Nonbonded Kernels

The nonbonded inner loops in GROMACS have gradually evolved to higher performance. Apart from the algorithmic optimizations mentioned above, a lot of speed is gained simply by avoiding unnecessary calculations. For example, atoms without van der Waals interactions can be handled in a faster Coulomb-only interaction routine.

In version 3 of GROMACS, this idea was taken to another level by borrowing a successful approach from the BLAS[89]/LAPACK[90] libraries: A large set of separate nonbonded kernels were written, each responsible for a very specific type of interaction, for exam-

ple, Reaction-Field Coulomb combined with tabulated van der Waals interactions. Because most biomolecular simulations contain large amounts of water, special routines were created for interactions between water and other atoms and even between pairs or water molecules. This leads to better cache usage and hides floating-point latencies. Initially, it was only implemented for the SPC[91] or TIP3P[92] water models, but GROMACS 3.3 adds support for accelerated TIP4P[92] nonbonded kernels. Any similar three or four particle water model (e.g., SPC/E[93]) can also be used.

In practice, the roughly 80 different nonbonded functions are written in either C or Fortran at build time using a small kernel generator program. The software inverse square root is optionally inlined, and coordinates and/or forces can be prefetched. Because the interaction forms are defined in the generator source, it guarantees consistency and eliminates the risk of bugs in functions that are used only rarely, not to mention that it saves programming time.

To fully exploit the hardware, many of the GROMACS nonbonded kernels have been tuned for specific hardware or even implemented in assembly code. The lack of conditional statements and streamlined core loops simplified this task considerably. Initially, it started as a test to see whether the SSE multimedia instructions in ×86 CPUs could be used to accelerate molecular dynamics. From a marketing point of view, molecular simulation is irrelevant compared to games, but luckily enough it turns out that inverse square roots are very common in shadow and lighting calculations, and the GROMACS SSE kernels more than doubled the already high performance on cheap PC hardware.

The current release of GROMACS provides accelerated assembly kernels for eight different combinations of architecture and precision. Most of them rely on multimedia instructions: 3DNow on Athlon CPUs,[94] SSE (single) and SSE2 (double) instructions on ia32 Pentium and AthlonXP CPUs,[95] 64-bit SSE and SSE2 kernels for Opteron processors and Altivec/VMX[96] on PowerPC processors. The most recent addition is Itanium2; while this architecture does not provide any multimedia instructions, it supports a powerful concept called software pipelining that has been used to completely hide instruction latencies: The sequential particle interaction is split into several "pipeline" stages, analogous to running multiple laps in a single loop. The Itanium2 kernels achieve over 90% of the theoretical hardware peak performance in both single and double precision; a complete Coulombic interaction including forces is calculated in only 12 clock cycles, and when only the potential is required it is reduced to as little as seven cycles.

File Formats and I/O Layer

The GROMACS library provides general I/O functions that can read and write any coordinate, trajectory, energy or topology format supported in the package. For instance, any program that needs to read or write coordinates just specifies a "generic trajectory" argument. The format of the file will be determined and converted by the I/O layer, and where applicable, the user will see options to start/stop reading at arbitrary frames in the file. Similarly, output formats are set automatically based on the file name extensions provided by the user. This way, all GROMACS programs support all formats, and if a program needs, for example,

atom or residue names it can get them from either a topology, a coordinate file, or even a trajectory format that includes names. Files that do not require random access can be compressed and decompressed automatically on the fly with gzip or compress.

To generate binary data in a fully portable way, GROMACS uses a built-in version of the POSIX External Data Representation (XDR, RFC1014).[97] All files written by the package can be shared between big and small endian hosts, machines with different floating-point precision or even non-IEEE floating point formats. Single precision versions of GROMACS can read double precision files and vice versa, but obviously the practical precision is limited to the stored data.

By default, GROMACS conserves space for trajectory storage by separating trajectories into a full precision file that stores both coordinates and velocities more seldom (for restarting runs), while coordinates for data analysis are normally stored in a lossy compression format. The loss is introduced by truncating coordinates to integer units of picometers (or any other scaling factor). Water atoms are sorted to utilize that hydrogen coordinates can be stored more efficiently by their offset from the oxygen, and finally, the bits are compressed with a lossless algorithm. With the standard values, as little as 4 *bytes* is required for each atom coordinate triplet in the final trajectory, that is, more than an order of magnitude compression. Even the lossless compression step is highly efficient; gzip is not able to shrink the files further. Just as the normal trajectories, the compressed files can be read and written by all programs using the GROMACS library. Interaction energies and other statistical data such as temperature, pressure, or distance restraint violations are written to portable binary *energy files*. A nice feature of these is that they store not only instantaneous values but exact averages, fluctuations, and drifts based on every single time step in the simulation, even when the energy frames are only written, for example, every 1000 steps. This is actually not a trivial problem; early versions of GROMACS used the classical variance formula

$$\sigma^2 = \sum_{i=1}^{N} x_i^2 - \frac{1}{N} \left(\sum_{i=1}^{N} x_i \right)^2, \tag{4}$$

but for long trajectories this leads to large numerical errors. Instead, the current implementation stores the partial sum and variance

$$X_{1,m} = \sum_{i=1}^{m} x_i, \tag{5}$$

$$\sigma_{1,m}^2 = \sum_{i=1}^{m} \left(x_i - \frac{X_{1,m}}{m} \right)^2, \tag{6}$$

from which the average and variance of the values stored between steps m and $m + k$ can be calculated as

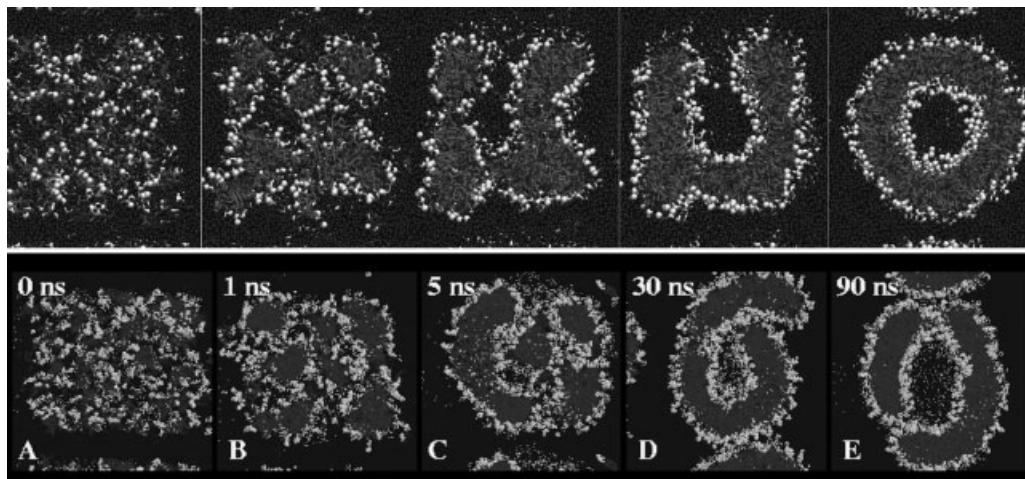$$\langle x \rangle = \frac{X_{1,m+k} - X_{1,m}}{k}, \tag{7}$$

Figure 4. Snapshots from a coarse-grained simulation (upper panel) and a simulation in atomic detail (lower panel) of lipids (DPPC) aggregating in water, from random distribution (left, at time 0) and after (effective) times of 1, 5, 30, and 90 ns. The system contains 1017 DPPC and 106,000 water molecules, totaling about 370,000 atoms. In the coarse-grained simulation there are 12 particles per lipid and one particle per four water molecules, totaling 38,700 particles. The coarse-grained simulation took half a processor day, while the detailed simulation took about two processor years! (Courtesy of S.-J. Marrink[106] and A. H. de Vries;[107] the lower panel is from ref. 107, reproduced with permission of the American Chemical Society.)

$$\sigma^2 = \left[\sigma_{1,m+k}^2 - \sigma_{1,m}^2 - \frac{m(m+k)}{k} (\langle x \rangle_{1,m} - \langle x \rangle_{1,m+k})^2 \right] / k. \quad (8)$$

GROMACS includes programs that use these formulas both to extract and report energy statistics (*g_energy*) and to concatenate/split energy files (*eneconv*). There is a similar kitchensink-style program for manipulating coordinate trajectories called *trjconv*.

Selected Applications

In this section we review some simulations that have been done with GROMACS that are either interesting from a scientific point of view or employ some of the special (and sometimes unique) features of the code. A number of high profile biological applications have been possible mainly due to the high performance of the code. In other cases special additions have been implemented.

Membrane Simulations

Simulations of phospholipids, phospholipid membranes, and surfactants have long been a special interest in the group,[98–101] and the advent of the GROMACS hardware and software made it possible to simulation very large membranes[102] or very long time scales. In particular, the simulations of spontaneous aggregation of dodecylphosphocholine lipids (DPC) into micelles[103] and of dipalmitoylphosphatidylcholine (DPPC) into bilayers[104] were interesting because they showed a plausible picture of the process. In the latter case it was found that the rate-limiting step is due to breaking a water pore spanning the lipid bilayer. These applications were taken a step further by Marrink and Mark when they

developed a course-grained model for lipids and water, where four water molecules are treated as a single particle and DPPC lipids are represented by 12 beads.[105] Simplifying the model in this manner has two advantages: a smaller numbers of particles and the possibility to take large time steps in the integration algorithm (here 50 fs). Although one looses the detailed atomic information, one gains in time and length scale. With this model the spontaneous formation of vesicles was observed, and interesting differences between the inner and outer membrane of the vesicles (diffusion rate, composition) were observed.[106] Finally, simulations of the formation of a DPPC vesicle were performed in atomic detail as well[107] (see Fig. 4).

Membrane Protein Simulations

Whereas protein simulations in solvent have become commonplace, membrane proteins are still a more specialized topic. In part, this is due to the fact that one needs to treat lipids in atomic detail as well as the protein, although there are examples of membranes represented by a matrix of hydrophobic particles[108,109] as well. The added complexity (including the problem which force field to use[110,111]) plus the usually large number of atoms make that the simulation of membrane proteins still is more of a challenge than simple liquids or solvated proteins. One of the first large membrane protein simulation using GROMACS was performed by Tieleman et al. on a porin trimer.[112–114] Other interesting applications are the transport of small molecules through Aquaporin and GlpF (a sugar transporter),[115] in particular because these processes were simulated in real time (ns).[116] Similar studies describe water transport through Gramicidin[117] and, again, through Aquaporin.[118] In the latter case, the energetics of the
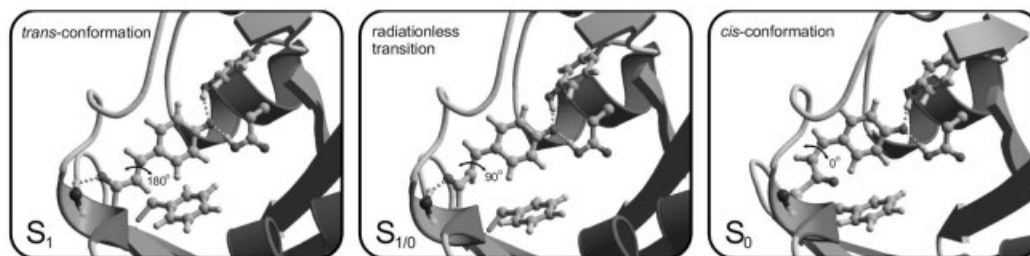
Figure 5. Snapshots of a photo-isomerization simulation. In the *trans* conformation, the p-coumaric acid chromophore, which is covalently linked to PYP via a cysteine, is electronically excited. In the excited state (S1), the chromophore rapidly decays from the Franck–Condon region on the S1 surface to a local minimum via a 90 deg. rotation of the double bond (indicated by the double arrow). There it encounters the S1/S0 intersection seam, where a radiationless transition takes the QM/MM trajectory to the ground-state (S0). Back on S0, relaxation leads to the isomerized conformation of the chromophore. The isomerization is the essential step for reaching the signaling state of PYP.

process are also described based on the free energy of water diffusion.

Interaction of Molecules with Ionizing X-rays

The advent of X-ray free electron lasers (FEL) will enable a whole range of new experiments in physics, chemistry, and biology within a few years.[119–121] In the field of biological applications, one could envision performing structural studies on large biomolecules, biomolecular aggregates, or nanocrystals. Molecular dynamics simulations of a protein molecule in an FEL beam are encouraging as to the feasibility of such experiments.[122] In this work interactions between X-rays and T4-lysozyme were implemented in GROMACS, to simulate the damage on a protein due to radiation, which manifests itself mainly through the photoelectric effect. In a recent follow-up, interactions between an electron gas (representing the electrons liberated through the photoelectric effect and secondary electron cascades[123]) were added to the model.[124] In this model the nonlinear Poisson–Boltzmann equation for the combination of point charges on the atoms and the electron gas is solved using an iterative scheme. The results show that the electrons retained in the sample (water clusters) undergoing radiation damage slow down the eventual Coulomb explosion of the sample, something that was also found in a much simpler hydrodynamic model,[125] which, however, does not give the same kind of detail. This is beneficial for experiments that will have to be performed in the fs time range, in the sense that it seems that the Coulomb explosion is slightly slower than estimated beforehand.[122]

Combined Quantum Mechanics and Classical Mechanics

The embedding of a quantum mechanical subsystem (QM) within a classical representation of the local environment (MM) has extended the range of applicability of MD simulation into areas of research concerned with chemical reactivity in condensed phases. Examples include chemical reactions in solution and in enzymes. Furthermore, simulations are no longer limited to the ground state, as photo-induced processes can also be studied using hybrid QM/MM schemes. In a recent application[126] we have used a combination of high level CASSCF *ab initio* calculations, surface-hopping techniques, and classical molecular dynamics (MD) simulation techniques to directly simulate the process of photo-isomerization of a chromophore within photoactive yellow protein (PYP),[127] a bacterial photoreceptor protein (Fig. 5). Within this approach it has not only been possible to explicitly follow the dynamics in the excited state, but also to predict with high accuracy the radiationless transitions between the excited state and ground state, the relaxation time of the system, the quantum yield, fine details of the fluorescence decay spectra, and the structural properties of intermediates occurring during the process of isomerization, which had previously been observed in low temperature crystallographic studies. This work has opened up the possibility to use similar approaches to study a wide range of photo-activated processes in proteins. In another application, the QM/MM interface was used to calculate proton affinities of side chains within PYP.[128,129]

Protein Folding-Related Simulations

The topic of protein folding has long been one of the challenges pursued by the GROMACS authors.[130–133] More recently, longer simulations have become possible, in the range of 50 ns and longer. Under these circumstances, processes like folding of α-helices can, in principle, be simulated. However, due to the marginal stability of helices one either has to simulate longer, or perform multiple simulations, and hope to see a folding process in one of them. This has been done in the extreme by the Pande group at Stanford, who use screen savers on desktop computers around the world to study protein folding.[134] The latest version of their screen-saver program is based on GROMACS, and because of this they were able for the first time to study proteins in explicit solvent[135] (see Fig. 6). Further long simulations are necessary to test the stability in different force fields, however.[136] Zhou et al. have recently presented an interesting study of the hydrophobic collapse of a multidomain protein,[137] using some of the artificial tricks possible in GROMACS, that is, they switched off the interactions between part of the protein and solvent to speed up the
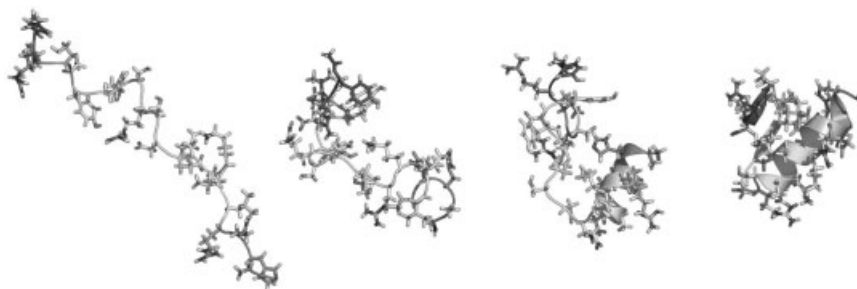
Figure 6. A sequence of structures of the 23-residue protein BBA5 in explicit water in a folding pathway generated in a Folding@Home project using GROMACS.[135] Ten thousand trajectories were produced, totaling 100-μs simulated time, which allowed for proper statistics and analysis of the folding process. Thirteen folding events were recorded.

dewetting transition at the interface between the domains of the protein.

Outlook

Plans for future implementation of algorithms abound. It is easy to write down a list of things one wishes to implement; however, considering limited time of the main developers, and unwillingness of granting agencies to sponsor code development one has to prioritize. We therefore give a short list of things that will almost certainly be implemented in the near future.

1. Better scaling parallel code. This will be the main feature of the 4.0 version of GROMACS. It will include combined multithreading/MPI, dynamic distribution of particles over processors, parallelization of constraint algorithms, and better parallelization of PME. A collaboration between the main developers and the German Max-Planck institutes has been established to this end.
2. XML (eXtensible Markup Language) based file formats, which, in principle, would give "future-proof" files. Although not inherently difficult to implement, progress in this area is somewhat slow due to the ambition to have a file format that can be used without change by different codes. See also the remark below.
3. An interface with the EDEN (Electron DENsity) program[138,139] will be developed that will allow the use of GROMACS and EDEN for real space refinement of models to electron density.[140]
4. Improvement of interfaces to popular force fields.
5. As the need arises, additional bonded interaction functions will be added.
6. To facilitate the communication with other scientific software, we envision to modularize the source code into small libraries with a well-defined application program interface. By doing this, parts of GROMACS can be incorporated into other free software packages, for example, the PyMOL graphics program[141] could allow a minimization function by calling the GROMACS minimizer. Other parts of the code that would be

useful to share include the analysis tools, which could be useful for other modeling codes or for structural analysis.

We wish to make a final remark on an outlook on the future that is not in our hands. The uncoupling of force fields from simulation packages would be a valuable general asset, adding much flexibility to user's applications. A standardized description or *ontology* for force fields, as a defined XML namespace, would be the way to ensure wide acceptance and unequivocal (possibly automated) implementation. We recommend the scientific community to consider such definitions.

Acknowledgments

Appendix A: Benchmark Results

GROMACS includes a small suite of benchmarks that we continuously run on new interesting hardware to determine which architectures and vendors provide the best price/performance ratio, and

Table 1. GROMACS Stimulation Performance Quoted in Picoseconds/Day on Single PC CPUs.

Simulation setup						Performance, ps/day		
Syst	FF	virtH	Water	Coulomb	LJ	ia32	×86-64	ppc
Vil	G	no	TIP3P	cutoff 0.8	cutoff 0.8	9744	9574	14,385
Vil	G	yes	TIP3P	cutoff 0.8	cutoff 0.8	16,900	16,895	23,681
Vil	G	yes	TIP3P	RF 1.0	cutoff 1.0	10,308	9719	12,934
Vil	G	yes	TIP3P	PME 0.9	twin 0.9/1.4	4624	4849	5101
Lys	G	no	SPC	cutoff 1.0	cutoff 1.0	2312	2002	3373
Lys	G	yes	SPC	cutoff 1.0	cutoff 1.0	3933	3635	5391
Lys	G	yes	SPC	twin 0.8/1.4	twin 0.8/1.4	3059	3076	4243
Lys	G	no	SPC	RF 1.0	cutoff 1.0	2140	1897	2534
Lys	G	no	SPC	RF 1.0	switch 1.0	1859	1742	2288
Lys	G	no	SPC	PME 0.9	cutoff 0.9	1010	1129	1108
Lys	G	no	SPC	PME 0.9	switch 0.9	958	1088	1097
Lys	G	no	SPC	table 1.0	table 1.0	1300	1302	1585
Lys	O	no	TIP3P	cutoff 1.0	cutoff 1.0	1927	1742	2626
Lys	O	yes	TIP3P	cutoff 1.0	cutoff 1.0	3372	3200	4565
Lys	O	no	TIP4P	cutoff 1.0	cutoff 1.0	1553	1549	1997
Lys	O	no	TIP4P	PME 0.9	cutoff 0.9	813	922	945
Lys	O	no	TIP3P	PME 0.9	cutoff 0.9	908	1018	987

Vil: Villin headpiece; Lys: Lysozyme; G: Gromos96; O: OPLS-AA; twin: twin range.
Force field is either Gromos96 (G) or OPLS-AA/L (O). The virtH column specifies whether hydrogens were modeled with virtual interaction sites to enable a 4-fs time step; otherwise, a 2-fs step was used. For both Coulomb and Lennard–Jones interactions the setup is described in the text, while the number denotes the cutoff in nanometers—two numbers are given for twin-range cutoff setups. This is by no means an exhaustive list of alternatives, but it includes the most common choices of algorithms and cutoffs. Due to their small size and simple interactions, the first two Villin (Vil) systems spend a fairly large time outside the nonbonded kernels routines.

how well different networks scale. The results are publicly available in the benchmarks section at www.gromacs.org, but to tell the truth, it has gotten a bit boring lately—there is simply no competition to cheap PC boxes using ia32, ×86-64, or PowerPC processors. So, instead of just citing standard numbers that would anyway be outdated soon, a slightly different benchmark was created for this article. Small or medium-size proteins are by far the most common simulated molecules, so we figured it would be useful to have a summary of performance with the various algorithms we have covered for precisely these systems. For some cases one of the PC architectures perform significantly better than the others, and it is also quite useful for estimating the performance of, for example, different force fields and water models, or fast/simple vs. slow/accurate simulation protocols.

For the small system we selected the Villin headpiece (PDB code 1VII), solvated with 3000 SPC water molecules in a truncated octahedron box, totaling just under 10,000 atoms. The medium-size protein is represented by hen egg white Lysozyme (2LZM), surrounded by 7156 water molecules in a rhombic dodecahedron, which is the most spherical of available shapes. Depending on the force field and water model, this system contains between 23,207 and 31,275 atoms. Internally in GROMACS, both these simulation boxes are represented as triclinic systems. Both the GROMOS96 and OPLS-AA-1 force fields were used, as well as the SPC/TIP3P and TIP4P water models. The basic timestep was 2 fs, with all bonds constrained. In some simulations the hydrogens were modeled as a virtual interaction site, which enabled a 4-fs time step.

Neighborlists were updated every 20 fs, that is, every 5 or 10 steps depending on the time step length. Berendsen temperature coupling with a time constant of 0.1 ps was used, but no pressure coupling. When PME was used the lattice spacing was restricted to be less than 0.12 nm in all directions, and the reciprocal energy evaluated every step. For twin-range neighbor searching, the nonbonded forces inside the first cutoff were evaluated every step, and the part between the two cutoffs evaluated during neighborlist updating. Other algorithms used include reaction-field electrostatics (RF), Lennard–Jones interactions switched off continuously from 0.2 nm inside the cutoff, and user-defined interactions using cubic spline interpolation tables read from files.

The machines used in the benchmark were a 2.8-GHz Intel Xeon with a 512-kb cache memory and a 800-MHz bus (ia32), a 2-GHz AMD Opteron with a 1024 kb cache running in 64-bit mode (×86-64), and a 2.5-GHz Apple Macintosh PowerPC 970 (ppc). Intel compilers (32 and 64 bit versions, respectively) were used on the Intel/AMD machines and IBM compilers on the PowerPC.

The results mostly speak for themselves (Table 1), but there are a couple of noteworthy observations. First, the three architectures are almost on par considering the PowerPC machine is cutting edge while the ×86 alternatives are several months old. The OPLSAA/L simulations are on average 10–15% slower, explained by the extra hydrogen atoms and significantly more dihedrals. The TIP4P overhead compared to three-particle water models is fairly low, and because it is mostly due to loading extra coordinates, it is

even less evident for the complex interactions. The cutoff interactions in PowerPC is a special case, because the SPC/TIP3P kernel has been manually fine tuned, which we have not yet had time to do for TIP4P. Reaction-field is only slightly slower than plain cutoff Coulomb, and when combined with switched Lennard–Jones it provides relatively accurate simulations with very little speed compromise. PME is definitely better, but the performance a bit lower due to the charge spreading and interpolation. Unfortunately, these operations are slow on modern CPUs because they cause a large amount of cache misses. There might be a way to improve this slightly, though: because the GROMACS direct-space interactions are quite fast it would make sense to extend the cutoff in combination with a coarser reciprocal-space grid. We are currently investigating which combinations to use to maintain accuracy. Finally, the user table interactions means GROMACS still provides quite impressive performance for arbitrary forms of the direct-space interaction potentials!

Appendix B: Analysis Tools

A number of programs with options to perform common trajectory analyses are distributed with the GROMACS package. Most of these analyses can be performed either on default or on user-defined atom selections. Advanced atom selections can be stored in a so-called index file, which in turn, is generated by an interactive program (make_ndx). This approach yields a very flexible way of analyzing trajectories. The analysis tasks automated in this way include, among others:

- Analysis of energy components, including interactions between user-defined groups of atoms. Fluctuations and drift are automatically calculated. The reported statistics is not calculated from the data points stored in the energy file, but as the true values based on all the simulation time steps.
- Mean square displacement of groups of atoms and velocity autocorrelation functions that can be used to determine diffusion coefficients.[69]
- Transverse current autocorrelation functions to determine the shear viscosity from an equilibrium simulation.
- Root mean square deviations (RMSD) between a reference structure and a trajectory, or a complete RMSD matrix of a trajectory against itself or against another trajectory.
- RMSD of pairs of distances, that is, the differences in distances between all atomic pairs in a reference structure and those in the trajectory. The advantage of this algorithm over traditional RMSD is that it is independent of the superposition, and hence, can be used for very different structures.
- Dielectric analysis to calculate dielectric constants and Kirkwood factors[142] and Cole–Cole plots.
- Radial distribution functions.
- Analysis of the orientation of solvent molecules around solutes.
- Hydrogen bond analysis based on geometric criteria and hydrogen bond lifetime analysis.[143]
- Analysis of formation and breaking of salt bridges.
- Protein secondary structure can be analyzed using a GROMACS front-end to the dictionary of secondary structure in proteins (DSSP[144]) program. Pretty (PostScript) plots of the secondary structure as a function of time can be made.[130]
- Solvent-accessible surface of macromolecules can be computed from trajectories or structures using the algorithms of Eisenhaber et al.[145]
- Chemical shifts can be computed based on ϕ/ψ angles[146] or protein conformations.[147,148]
- Relaxation times for molecular motions can be calculated in a form directly comparable to NMR results.[149]
- Order parameters for lipid carbon tails.[98,99]
- Density profiles, either along a box axis, which is useful for interface studies,[100,150] or radial for systems like a micel.[101]
- Analysis of a bundle of axes: the distance of the axes to the center and two different tilt angles. This is especially useful for transmembrane helices.
- Ramachandran plots.[151] A time-dependent Ramachandran plot can be visualized with an X-windows program that displays the Ramachandran plot for all frames in a molecular dynamics trajectory as a function of time.
- A special program to study all dihedral angles in proteins, as well as a more general tool for angle- and dihedral analysis.
- Analysis of bond length distributions.
- A clustering program to combine similar conformations of a molecule, with support for several different algorithms.[152,153]
- Cluster size analysis tool for nucleation and condensation analysis.
- The radius of gyration of arbitrary sets of atoms can be computed, and a principal component analysis of the moment of inertia applied to make the result independent of the orientation.
- Positional root-mean-square fluctuations that can be converted to B-factors.
- Essential Dynamics (ED) analysis or principal component analysis[154,155] can be done in cartesian space or in dihedral space[156] using GROMACS tools. There are also tools to analyze the covariance matrix by making projections of trajectories on the eigenvectors. In addition, one can use GROMACS to perform preferential sampling of phase space defined by certain of the principal components.[157,158]

References

1. Christen, M.; Hünenberger, P. H.; Bakowies, D.; Baron, R.; Bürgi R.; Geerke, D. P.; Heinz, T. N.; Kastenholz, M. A.; Kräutler, V.; Oostenbrink, C.; Peter, C.; Trzesniak, D.; van Gunsteren, W. F. J Comput Chem 2005, 26, 1719.
2. 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. In Physics Computing 1992; de Groot, R. A.; Nadrchal, J., Eds.; World Scientific: Singapore, 1993, p. 252.
3. 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. In Physics Computing 1992; de Groot, R. A.; Nadrchal, J., Eds.; World Scientific: Singapore, 1993, p. 257.
4. Berendsen, H. J. C.; van der Spoel, D.; van Drunen, R. Comput Phys Commun 1995, 91, 43.
5. van der Spoel, D.; Berendsen, H. J. C. In Aspects of Computational Science; van der Steen, A. J., Ed.; National Computing Facilities Foundation (NCF): Den Haag, The Netherlands, 1995, p. 367.

6. van Drunen, R.; van Teylingen, C.; Kroontje, M.; Berendsen, H. J. C. Int Conf on Parallel and Distributed Processing Techn Applic 1996 (PDPTA'96).

7. Bekker, H.; Dijkstra, E. J.; Berendsen, H. J. C. In Parallel Computing: From Theory to Sound Practice; Joosen, W.; Milgrom, E., Eds.; IOS Press: Amsterdam, 1992, p. 268.

8. van Gunsteren, W. F.; Berendsen, H. J. C. Angew Chem Int Ed English 1990, 29, 992.

9. Berendsen, H. J. C.; van Gunsteren, W. F. In Molecular Dynamics Simulation of Statistical Mechanical Systems; Ciccotti, G.; Hoover, W., Eds.; Soc Italiana di Fisica, Bologna; North Holland: Amsterdam, 1986, p. 43.

10. Ryckaert, J.-P.; Ciccotti, G.; Berendsen, H. J. C. J Comput Phys 1977, 23, 327.

11. Berendsen, H. J. C.; van Gunsteren, W. F. In The Physics of Superionic Conductors and Electrode Materials; Perram, J. W., Ed.; NATO ASI Series B 92, Plenum: New York, 1983, p. 221.

12. Berendsen, H. J. C.; Postma, J. P. M.; van Gunsteren, W. F.; DiNola, A.; Haak, J. R. J Chem Phys 1984, 81, 3684.

13. van Gunsteren, W. F.; Berendsen, H. J. C. Mol Phys 1982, 45, 637.

14. van Gunsteren, W. F.; Berendsen, H. J. C. Mol Simul 1988, 1, 173.

15. Bekker, H. Molecular Dynamics Simulation Methods Revised, Ph.D. Thesis Groningen 1996 (http://www.ub.rug.nl/eldoc/dis/science/h.bekker/).

16. Bekker, H.; Dijkstra, E. J.; Renardus, M. K. R.; Berendsen, H. J. C. Mol Simul 1995, 14, 137.

17. Bekker, H. J Comput Chem 1997, 18, 1930.

18. Lindahl, E.; Hess, B.; van der Spoel, D. J Mol Model 2001, 7, 306.

19. Hess, B.; Bekker, H.; Berendsen, H. J. C.; Fraaije, J. G. E. M. J Comput Chem 1997, 18, 1463.

20. van Gunsteren, W. F.; Billeter, S. R.; Eising, A. A.; Hünenberger, P. A.; Krüger, P.; Mark, A. E.; Scott, W. R. P.; Tironi, I. G. Biomolecular Simulation, the GROMOS96 Manual and User Guide; Hochschulverlag AG der ETH: Zürich, 1996.

21. Levitt, M. Energy Calculations and Dynamics Program; Molecular Applications Group: Stanford; and Yeda: Rehovot, Israel, 1990.

22. Jorgensen, W. L.; Maxwell, S.; Tirado–Rives, J. J Am Chem Soc 1996, 118, 11225.

23. Kaminski, G. A.; Friesner, R. A.; Tirado–Rives, T.; Jorgensen, W. L. J Phys Chem B 2001, 105, 6474.

24. Case, D. A.; Cheatman, T. E., III; Darden, T.; Gohlke, H.; Luo, R.; Merz, K. M., Jr.; Onufriev, A.; Simmerling, C.; Wang, B.; Woods, R. J. J Comput Chem 2005, 26, 1668. (See also: http://amber.scripps.edu/amber8.ffparms.tar.gz).

25. GNU General Public License: (http://www.opensource.org/licenses/gpl-license.php).

26. Bekker, H.; van den Berg, J. P.; Wassenaar, T. A. J Comput Chem 2004, 25, 1037.

27. Amadei, A.; Chillemi, G.; Ceruso, M. A.; Grottesi, A.; Di Nola, A. J Chem Phys 2000, 112, 9.

28. Schuler, L. D.; Daura, X.; van Gunsteren, W. F. J Comput Chem 2001, 22, 1205.

29. Oostenbrink, C.; Villa, A.; Mark, A. E.; van Gunsteren, W. F. J Comput Chem 2004, 25, 1656.

30. Brooks, B. R.; Bruccoleri, R. E.; Olafson, B. D.; States, D. J.; Swaminathan, S.; Karplus, M. J Comp Chem 1983, 4, 187.

31. Ryckaert, J.-P.; Bellemans, A. Chem Phys Lett 1975, 30, 123.

32. Ryckaert, J.-P.; Bellemans, A. Faraday Dis Chem Soc 1978, 66, 95.

33. Warner, H. R., Jr. Ind Eng Chem Fundam 1972, 11, 379.

34. Miyamoto, S.; Kollman, P. A. J Comput Chem 1992, 13, 1463.

35. Watts, R. O. Mol Phys 1974, 28, 1069.

36. Neumann, M.; Steinhauser, O. Mol Phys 1980, 39, 437.

37. Tironi, I. G.; Sperb, R.; Smith, P. E.; van Gunsteren, W. F. J Chem Phys 1995, 102, 5451.

38. Ewald, P. P. Ann Phys 1921, 64, 253.

39. Darden, T.; York, D.; Pedersen, L. J Chem Phys 1993, 98, 10089.

40. Essmann, U.; Perera, L.; Berkowitz, M. L.; Darden, T.; Lee, H.; Pedersen, L. G. J Chem Phys 1995, 103, 8577.

41. Hockney, R. W.; Eastwood, J. W. Computer Simulation Using Particles; McGraw–Hill: New York, 1981.

42. Hess, B.; Scheek, R. M. J Magn Res 2003, 164, 19.

43. Jordan, P. C.; van Maaren, P. J.; Mavri, J.; van der Spoel, D.; Berendsen, H. J. C. J Chem Phys 1995, 103, 2272.

44. van Maaren, P. J.; van der Spoel, D. J Phys Chem B 2001, 105, 2618.

45. Yu, H. B.; Hansson, T.; van Gunsteren, W. F. J Chem Phys 2003, 118, 221.

46. Hess, B.; Saint–Martin, H.; Berendsen, H. J. C. J Chem Phys 2002, 116, 9602.

47. Saint–Martin, H.; Hernández–Cobos, J.; Bernal–Uruchurtu, M. I.; Ortega–Blake, I.; Berendsen, H. J. C. J Chem Phys 2000, 112, 7919.

48. Car, R.; Parrinello, M. Phys Rev Lett 1985, 55, 2471.

49. Warshel, A.; Levitt, M. J Mol Biol 1976, 109, 227.

50. Field, M.; Bash, P. A.; Karplus, M. J Comput Chem 1990, 11, 700.

51. Svensson, M.; Humbel, S.; Froes, R. D. J.; Matsubara, T.; Sieber, S.; Morokuma, K. J Phys Chem 1996, 100, 19357.

52. Dewar, M. J. S. J Mol Struct 1983, 100, 41.

53. Guest, M. F.; Harrison, R. J.; van Lenthe, J. H.; van Corler, L. C. H. Theor Chim Acta 1987, 71, 117.

54. Frisch, M. J.; Trucks, G. W.; Schlegel, H. B.; Gill, P. M. W.; Johnson, B. G.; Robb, M. A.; Cheeseman, M. J. R.; Keith, T. A.; Petersson, G. A.; Montgomery, J. A.; Raghavachari, K.; Al-Laham, M. A.; Zakrzewski, V. G.; Ortiz, J. V.; Foresman, J. B.; Ciosloswki, J.; Stefanof, B. B.; Nanayakkara, A.; Challacombe, M.; Peng, C. Y.; Ayala, P. Y.; Chen, W.; Wong, M. W.; Andres, J. L.; Replogle, E. S.; Gomperts, R.; Martin, R. L.; Fox, D. J.; Binkley, J. S.; Defrees, D. J.; Baker, J.; Stewart, J. P.; Head–Gordon, M.; Gonzalez, C.; Pople, J. A. Gaussian 98, Revision A.7; Gaussian, Inc.: Pittsburgh, PA, 1998.

55. Feenstra, K. A.; Hess, B.; Berendsen, H. J. C. J Comput Chem 1999, 20, 786.

56. Leimkuhler, B. J. In Computational Molecular Dynamics: Challenges, Methods, Ideas; Deuflhard, P.; Hermans, J.; Leimkuhler, B.; Mark, A. E.; Reich, S.; Skeel, R. D. Eds.; Lecture Notes in Computational Science and Engineering 4; Springer Verlag: Heidelberg, 1999, p. 349.

57. Leimkuhler, B.; Skeel, R. D. J Comput Phys 1994, 112, 117.

58. Nosé, S. Mol Phys 1984, 52, 255.

59. Nosé, S. J Chem Phys 1984, 81, 511.

60. Hoover, W. G. Phys Rev A 1985, 31, 1695.

61. Hess, B. Thesis, University of Groningen, Jan 2002 (http://www.ub.rug.nl/eldoc/dis/science/b.hess/).

62. Schneider, T.; Stoll, E. Phys Rev B 1978, 17, 1302.

63. Woodcock, L. V. Chem Phys Lett 1971, 10, 257.

64. Morishita, T. J Chem Phys 2000, 113, 2976.

65. Harvey, S. C.; Tan, R. K.-Z.; Cheatham, T. E., III. J Comput Chem 1998, 19, 726.

66. Berendsen, H. J. C. In Computer Simulations in Material Science; Meyer, M.; Pontikis, V., Eds.; Kluwer: Dordrecht, 1991, p. 139.

67. Gosling, E. M.; McDonald, I. R.; Singer, K. Mol Phys 1973, 26, 1475.

68. Ciccotti, G.; Jacucci, G.; McDonald, I. R. J Stat Phys 1979, 21, 1.

69. Allen, M. P.; Tildesley, D. J. Computer Simulation of Liquids; Oxford Science Publ: New York, 1987.

70. Hess, B. J Chem Phys 2002, 116, 209.

71. Tieleman, D. P.; Berendsen, H. J. C.; Sansom, M. S. P Biophys J 2001, 80, 331.

72. Leach, A. R. Molecular Modelling, Principles and Applications; Prentice Hall: Englewood Cliffs, NJ, 2001.

73. Beutler, T. C.; Mark, A. E.; van Schaik, R. C.; Gerber, P. R.; van Gunsteren, W. F. Chem Phys Lett 1994, 222, 529.

74. Jarzynski, C. Phys Rev Lett 1997, 78, 2690.

75. Jarzynski, C. Phys Rev E 1997, 56, 5018.

76. Ytreberg, F. M.; Zuckerman, D. M. J Comp Chem 2004, 25, 1749.

77. den Otter, W. K.; Briels, W. J. J Chem Phys 1998, 109, 4139.

78. Sprik, M.; Ciccotti, G. J Chem Phys 1998, 109, 7737.

79. den Otter, W. J. J Chem Phys 2000, 112, 7283.

80. CODATA most recent fundamental constants (http://physics.nist.gov/cuu/Constants/index.html).

81. Frigo, M.; Johnson, S. G. In ICASSP Conference Proceedings, Seattle, 1998, vol. 3, p. 1381.

82. Squyres, J.; Lumsdaine, A. In Proceedings of the 10th European PVM/MPI Users' Group Meeting; Springer-Verlag: Venice, 2003, p. 379.

83. Gropp, W.; Lusk, E.; Doss, N.; Skjellum, A. Parallel Comput 1996, 22, 789.

84. Levitt, M.; Sander, C.; Stern, P. S. Proc Natl Acad Sci USA 1983, 10, 181.

85. Parrinello, M.; Rahman, A. J Appl Phys 1981, 52, 7182.

86. Nosé, S.; Klein, M. L. Mol Phys 1983, 50, 1055.

87. Verlet, L. Phys Rev 1967, 98, 98.

88. Press, W. H.; Teukolsky, S. A.; Vetterling, W. T.; Flannery, B. P. Numerical Recipes in C; Cambridge University Press: Cambridge, 1992.

89. Lawson, C. L.; Hanson, R. J.; Kincaid, D.; Krogh, F. T. ACM Trans Math Soft 1979, 5, 308.

90. Anderson, E.; Bai, Z.; Bischof, C.; Blackford, S.; Demmel J.; Dongarra, J.; DuCroz, J.; Greenbaum, A.; Hammerling, S.; McKenney, A.; Sorensen, D. LAPACK Users' Guide; SIAM: Philadelphia, 1999, 3rd ed.

91. Berendsen, H. J. C.; Postma, J. P. M.; van Gunsteren, W. F.; Hermans, J. In Intermolecular Forces; Pullman, B., Ed.; D. Reidel Publishing Company: Dordrecht, 1981, p. 331.

92. Jorgensen, W. L.; Chandrasekhar, J.; Madura, J. D.; Impey, R. W.; Klein, M. L. J Chem Phys 1983, 79, 926.

93. Berendsen, H. J. C.; Grigera, J. R.; Straatsma, T. P. J Phys Chem 1987, 91, 6292.

94. 3DNow! Technology Manual; Advanced Micro Devices, Inc., 2000. http://www.amd.com.

95. Intel Architecture Software Developer's Manual, Volume 2: Instruction Set Reference: Intel Corporation, 1999. http://www.intel.com.

96. Altivec Technology Programming Interface Manual, Motorola Literature Distribution, Denver, 1999.

97. XDR: External Data Representation Standard Sun Microsystems, Inc., Mountain View, 1987.

98. Marrink, S. J.; Berkowitz, M.; Berendsen, H. J. C. Langmuir 1993, 9, 3122.

99. Egberts, E.; Marrink, S. J.; Berendsen, H. J. C. Eur Biophys J 1994, 22, 423.

100. van Buuren, A. R.; Berendsen, H. J. C. Langmuir 1994, 10, 1703.

101. Tieleman, D. P.; van der Spoel, D.; Berendsen, H. J. C. J Phys Chem B 2000, 104, 6380.

102. Lindahl, E.; Edholm, O. Biophys J 2000, 79, 426.

103. Marrink, S. J.; Tieleman, D. P.; Mark, A. E. J Phys Chem B 2000, 104, 12165.

104. Marrink, S. J.; Lindahl, E.; Edholm, O.; Mark, A. E. J Am Chem Soc 2001, 123, 8638.

105. Marrink, S.-J.; de Vries, A. H.; Mark, A. E. J Phys Chem B 2004, 108, 750.

106. Marrink, S. J.; Mark, A. E. J Am Chem Soc 2003, 125, 15233.

107. de Vries, A. H.; Mark, A. E.; Marrink, S. J. J Am Chem Soc 2004, 126, 4488.

108. Åqvist, J.; Luzhkov, V. Nature 2000, 404, 881.

109. Edman, K.; Royant, A.; Larsson, G.; Jacobsson, F.; Taylor, T.; van der Spoel, D.; Landau, E. M.; Pebay–Peyroula, E.; Neutze, R. J Biol Chem 2004, 279, 2147.

110. Tieleman, D. P.; Berendsen, H. J. C. J Chem Phys 1996, 105, 4871.

111. Berger, O.; Edholm, O.; Jähnig, F. Biophys J 1997, 72, 2002.

112. Tieleman, D. P.; Berendsen, H. J. C. Biophys J 1998, 74, 2786.

113. Robertson, K. M.; Tieleman, D. P. FEBS Lett 2002, 528, 53.

114. Robertson, K. M.; Tieleman, D. P. Biochem Cell Biol 2002, 80, 517.

115. de Groot, B. L.; Grubmüller, H. Science 2001, 294, 2353.

116. Berendsen, H. J. C. Science 2001, 294, 2304.

117. de Groot, B. L.; Tieleman, D. P.; Pohl, P.; Grubmueller, H. Biophys J 2002, 82, 2934.

118. Vidossich, P.; Cascella, M.; Carloni, P. Proteins 2004, 55, 924.

119. Wiik, B. H. Nucl Inst Meth Phys Res B 1997, 398, 1.

120. Hajdu, J.; Hodgson, K.; Miao, J.; van der Spoel, D.; Neutze, R.; Robinson, C. V.; Faigel, G.; Jacobsen, C.; Kirz, J.; Sayre, D.; Weckert, E.; Materlik, G.; Szöke, A. In LCLS: The First Experiments 2000, 35; SSRL, SLAC: Stanford, 2000.

121. Winick, H. J Elec Spec Rel Phenom 1995, 75, 1.

122. Neutze, R.; Wouts, R.; van der Spoel, D.; Weckert, E.; Hajdu, J. Nature 2000, 406, 752.

123. Ziaja, B.; Szöke, A.; van der Spoel, D.; Hajdu, J. Phys Rev B 2002, 66, 024116.

124. Bergh, M.; Tîmneanu, N.; van der Spoel, D. Phys Rev E 2004, 70, 051904.

125. Hau–Riege, S. P.; London, R. A.; Szöke, A. Phys Rev E 2004, 69, 051906.

126. Groenhof, G.; Bouxin–Cademartory, M.; Hess, B.; de Visser, S. P.; Berendsen, H. J. C.; Olivucci, M.; Mark, A. E.; Robb, M. A. J Am Chem Soc 2004, 126, 4228.

127. Hellingwerf, K. J.; Hedriks, J.; Gensch, T. J Phys Chem 2003, 107, 1082.

128. Groenhof, G.; Lensink, M. F.; Berendsen, H. J. C.; Snijders, J. G.; Mark, A. E. Proteins 2002, 48, 202.

129. Groenhof, G.; Lensink, M. F.; Berendsen, H. J. C.; Mark, A. E. Proteins 2002, 48, 212.

130. van der Spoel, D.; Vogel, H. J.; Berendsen, H. J. C. Proteins 1996, 24, 450.

131. van der Spoel, D.; van Buuren, A. R.; Tieleman, D. P.; Berendsen, H. J. C. J Biomol NMR 1996, 8, 229.

132. van der Spoel, D.; Berendsen, H. J. C. Biophys J 1997, 72, 2032.

133. van der Spoel, D. Biochem Cell Biol 1998, 76, 164.

134. Shirts, M. R.; Pande, V. S. Science 2000, 86, 4983.

135. Rhee, Y. M.; Sorin, E. J.; Jayachandran, G.; Lindahl, E.; Pande, V. S. Proc Natl Acad Sci USA 2004, 101, 6456.

136. van der Spoel, D.; Lindahl, E. J Phys Chem B 2003, 117, 11178.

137. Zhou, R. H.; Huang, X. H.; Margulis, C. J.; Berne, B. J. Science 2004, 305, 1605.

138. Szöke, A. Acta Crystallogr A 1993, 49, 853.

139. Somoza, J. R.; Szöke, H.; Goodman, D. M.; Beran, P.; Truckses, D.; Kim, S. H.; Szöke, A. Acta Crystallogr A 1995, 51, 691.

140. Chapman, M. S. Acta Crystallogr A 1995, 51, 69.

141. Python Molecular Graphics Program (http://pymol.sourceforge.net).

142. Neumann, M. J Chem Phys 1986, 85, 1567.

143. Luzar, A.; Chandler, D. Nature 1996, 379, 55.

144. Kabsch, W.; Sander, C. Biopolymers 1983, 22, 2577.

145. Eisenhaber, F.; Lijnzaad, P.; Argos, P.; Sander, C.; Scharf, M. J Comput Chem 1995, 16, 273.

146. Wishart, D. S.; Nip, A. M. Biochem Cell Biol 1998, 76, 153.

147. Williamson, M. P.; Asakura, T. J Magn Reson Ser B 1993, 101, 63.
148. Case, D. A.; Dyson, H. J.; Wright, P. E. Methods Enzymol 1994, 239, 392.
149. van der Spoel, D.; van Maaren, P. J.; Berendsen, H. J. C. J Chem Phys 1998, 108, 10220.
150. van Buuren, A. R.; de Vlieg, J.; Berendsen, H. J. C. Langmuir 1995, 11, 2957.
151. Ramachandran, G. N.; Ramakrishnan, C.; Sasisekharan, V. J Mol Biol 1963, 7, 95.
152. Torda, A. E.; van Gunsteren, W. F. J Comput Chem 1994, 15, 1331.
153. Daura, X.; Gademann, K.; Jaun, B.; Seebach, D.; van Gunsteren, W. F.; Mark, A. E. Angew Chemie Int Ed 1999, 38, 236.
154. Garcia, A. E. Phys Rev Lett 1992, 68, 2696.
155. Amadei, A.; Linssen, A. B. M.; Berendsen, H. J. C. Proteins 1993, 17, 412.
156. Mu, Y.; Nguyen, P. H.; Stock, G. Proteins 2005, 58, 45.
157. de Groot, B. L.; Amadei, A.; van Aalten, D. M. F.; Berendsen, H. J. C. J Biomol Struct Dyn 1996, 13, 741.
158. de Groot, B. L.; Amadei, A.; Scheek, R. M.; van Nuland, N. A. J.; Berendsen, H. J. C. Proteins 1996, 26, 314.