J Mol Model (2001) 7:306–317
DOI 10.1007/s008940100045

ORIGINAL PAPER

Erik Lindahl · Berk Hess · David van der Spoel

GROMACS 3.0: a package for molecular simulation and trajectory analysis

Received: 9 April 2001 / Accepted: 26 June 2001 / Published online: 25 August 2001

Abstract GROMACS 3.0 is the latest release of a versatile and very well optimized package for molecular simulation. Much effort has been devoted to achieving extremely high performance on both workstations and parallel computers. The design includes an extraction of virial and periodic boundary conditions from the loops over pairwise interactions, and special software routines to enable rapid calculation of $x^{-1/2}$. Inner loops are generated automatically in C or Fortran at compile time, with optimizations adapted to each architecture. Assembly loops using SSE and 3DNow! Multimedia instructions are provided for x86 processors, resulting in exceptional performance on inexpensive PC workstations. The interface is simple and easy to use (no scripting language), based on standard command line arguments with self-explanatory functionality and integrated documentation. All binary files are independent of hardware endian and can be read by versions of GROMACS compiled using different floating-point precision. A large collection of flexible tools for trajectory analysis is included, with output in the form of finished Xmgr/Grace graphs. A basic trajectory viewer is included, and several external visualization tools can read the GROMACS trajectory format. Starting with version 3.0, GROMACS is available under the GNU General Public License from http://www.gromacs.org.

Keywords Parallel molecular dynamics · Simulation · Algorithmic optimization · Assembly loops · Benchmark

E. Lindahl
Theoretical Physics, KTH, 10044 Stockholm, Sweden

B. Hess
Dept. of Biophysical Chemistry, Groningen University,
Nijenborgh 4, 9747 AG, Groningen, The Netherlands

D. van der Spoel (✉)
Dept. of Biochemistry, Uppsala University,
Husargatan 3, Box 576, 75123 Uppsala, Sweden
e-mail: spoel@xray.bmc.uu.se
Tel.: +46-18-4714205, Fax: +46-18-511755

Introduction

The GROMACS package is a versatile collection of programs and libraries for the simulation of molecular dynamics and the subsequent analysis of trajectory data. Although it is primarily targeted at biological molecules with complex bonded interactions, the very effective implementation of nonbonded force calculations makes GROMACS suitable for all kinds of molecular dynamics simulations based on pair potentials. Apart from normal potential functions like Lennard-Jones, Buckingham and Coulomb, it is possible to use arbitrary forms of interactions with spline-interpolated tables. The history of the package dates back to the late 1980s in the Berendsen/Van Gunsteren group at the University of Groningen, The Netherlands. As parallel computers were becoming more common, there was a need for an efficient parallel implementation of a general-purpose molecular dynamics code. The GROMOS software [1] was chosen as a basis and reference for the simulation algorithms, while the parallelization concepts were adopted from the earlier literature. [2, 3]

The limited availability of parallel programming tools (except for Transputer systems with the dedicated OCCAM language) and a wish to use more structured programming techniques led to a decision to implement the whole package from scratch in the C language. This was not a trivial choice, since at the time there was a large gap between the optimization capabilities of Fortran and C compilers. It has, however, enabled a faster development cycle and more flexible code, using standard C features like dynamic memory allocation and advanced debuggers. With the later standardization of ANSI C, this has had the further advantage that the GROMACS code can be compiled on essentially any computer where a C compiler is present. To address the performance issue, the inner loops of the GROMACS code were rewritten in Fortran around 1995. Since almost all the execution time is spent in this small part of the program, it provided a speedup of a factor of 3 on a Silicon Graphics R8000 machine. The gap between For-

tran and C has narrowed since then, but it is still present, and hence the Fortran inner loops have been retained in the GROMACS code. From this version, they have also been supplemented with assembly loops for common PC hardware. In the early stages of the GROMACS project two dedicated parallel computers were built, both based on the Intel i860 RISC processor and consisting of 32 nodes. These machines are described in some detail in earlier papers, [4, 5] with the essentials of the parallel algorithms covered in References [5, 6] Since then, GROMACS has turned into a pure software project, designed for UNIX workstations and massively parallel computers using MPI (Message Passing Interface) for communication.

In the current release, the GROMACS source consists of roughly 150,000 lines of code, predominantly in the C language. It is built as two general libraries of molecular dynamics subroutines, and about 75 executable programs, which are linked to the libraries. The only external software required besides the C compiler is the free FFTW (Fastest Fourier Transform in the West, http://www.fftw.org/) library. [7, 8] For parallel operation an MPI library is needed, such as the free LAM (Local area multicomputer, http://www.mpi.nd.edu/lam) or MPICH (http://www-unix.mcs.anl.gov/mpi/mpich). All massively parallel computers nowadays offer efficient machine-specific MPI implementations. The configuration of the source and makefiles is completely automated using GNU autoconf, optimization options and the use of Fortran or assembly inner-loops are usually deduced automatically from the computer architecture and operating system. Last but not least, a 200+ page user manual is available, [9] which contains an introduction to molecular simulation and many details about potentials, topologies etc.

Programming concepts

General

There are a few key concepts worth mentioning that have contributed significantly to the success of the GROMACS design. Although we decided on a third generation programming language as the backbone of the code, we did adopt some concepts from object-oriented languages. C structures are used throughout the software, each with specialized functions operating on them, providing encapsulation of data structures. Dedicated subroutines, including extensive error checking and an architecture abstraction layer, have been implemented to perform file I/O operations on most of these structures. By changing a single typedef (using an option to the configure script) it is possible to compile the entire package in either single or double floating-point precision.

It is difficult to design and maintain the architectural integrity of a large scientific software package, since the specification is usually a moving target. By enforcing strong typing of the data structures and providing a capable library for basic functionality that is used by all programs, it has nevertheless proven easy to add new functionality and optimizations without major code changes. For example, all programs call a function parse_common_args to parse command line arguments, passing it a number of special data structures that are initialized statically in the program code. Since this function is used in all programs, it was relatively easy to build a graphical (X/Motif) interface into this routine, which is now automatically part of all programs. Similarly, every single tool that reads trajectories has support for starting and finishing the analysis at times given as command line arguments, without any extra user code.

Extending the GROMACS code should be feasible for anyone with a working knowledge of the C language, and a basic understanding of the MD algorithm. There is no need for users to program in Fortran or assembly language. A template for building analysis tools is part of the source code.

Algorithmic optimization

Although versatility and a portable code base are nice benefits, speed is essentially everything for a molecular dynamics code, since a typical run might take weeks or months to complete. GROMACS is very fast for several reasons. The most important optimization in this context is that any unnecessary calculations must be avoided at the algorithm level. A good example of this is that we have been able to extract the computation of the virial from the inner loops of the program by rewriting it as a single sum over particles. [10, 11] This increases the performance by roughly 40% without sacrificing any results: the exact full virial necessary for the pressure calculation is still obtained.

Another example is that the image calculation connected to simulations employing periodic boundary conditions is not performed in the inner loop at all. Instead, the correct image is calculated when performing the neighbor search and stored as a translation vector in the Verlet neighborlist. [12] Using a minimum image convention there can be at most 11 different such translation vectors in a triclinic cell (including the zero translation in the central box), which we essentially treat as independent parts of the list. The translation vector only has to be added to the parent particle of the list once, outside the innermost pair interaction loops. This increases simulation performance significantly through better instruction scheduling, since conditional statements are completely avoided in the inner loops. A similar translation vector solution is used to remove all periodic boundary condition calculations from the bonded interactions.

Force calculation

Even after substantial algorithmic alterations, the calculation of pairwise nonbonded forces still dominates the

simulation, typically accounting for 90% of the run time. On most computers, the single most expensive part of these loops is the calculation of pairwise distances **r** from particle coordinates, due to the complexity of the square root operation (squaring the inter-particle vector produces $\mathbf{r}^2$). Further, most interactions do not depend directly on the distance **r** but rather $\mathbf{r}^{-1}$ (Coulomb) or integer multiples of $\mathbf{r}^2$ (e.g. Lennard-Jones). In GROMACS, this bottleneck has been circumvented by constructing a special software routine for the reciprocal square root operation $x^{-1/2}$ to calculate $\mathbf{r}^{-1}$ directly from $\mathbf{r}^2$. An initial table lookup yields an approximate result a with 12 bits of accuracy, and the full single precision result is obtained after a single Newton–Raphson iteration [13]

$$\boldsymbol{r}^{-1}=(1/2)a(3-\boldsymbol{r}^2a^2) \quad (1)$$

Double precision requires another iteration, but for most normal simulations single precision provides sufficient accuracy. This software approach more than doubles the total performance on normal processors without a hardware square root[1]. In addition, not all interactions require the inverse distance. For pure Lennard-Jones potentials we only need the square of the inverse distance, i.e., the reciprocal operation $1/x$ when starting from $\mathbf{r}^2$. If the hardware does not provide a fast instruction for this, it is accomplished with a similar table lookup and the iteration [13]

$$\boldsymbol{r}^{-2}=a(2-\boldsymbol{r}^2a) \quad (2)$$

In the most recent version of GROMACS we have extended this approach to a complete inner loop generator, which at build time constructs separate routines in either C or Fortran for all possible combinations of interactions. The optional software replacements for $x^{-1/2}$ and/or x^{-1} are inlined, explicit prefetching of coordinates and forces can be used, and it is even possible to vectorize the entire distance calculation to use faster library functions provided by hardware vendors. Apart from the normal routine, each interaction loop is also created in three special versions to further optimize for common cases like general solvents, water molecules, and most important, interactions between pairs of water molecules. This makes it possible to hide the latencies of coordinate fetching and table lookups almost completely, by treating several interactions in parallel. In the current version, this results in about 80 different specialized inner loops entirely without conditional statements. For each loop there is a specific neighborlist, although not all of them are used in one simulation. The resulting performance improvements depend very much on the type of system simulated, but it can be as high as another factor of 2.

This automatic inner loop generation provides very good performance on a broad range of hardware, but in many cases neither C nor Fortran compilers are able to take full advantage of the available hardware. This is especially significant for common PC processors like Intel Pentium III and AMD Athlon. These processors have dedicated multimedia instructions and registers that are primarily aimed at games and graphics applications, but since both x^{-1} and $x^{-1/2}$ are frequent operations in lighting processing, table lookups for these important functions have been implemented in hardware. Obviously, this makes the multimedia instructions very attractive for use in molecular simulation and scientific applications in general. In the current version of GROMACS, the entire inner loops have been implemented in assembly code to make use of either 3DNow! instructions [14] on AMD K6 Athlon and Duron processors, or Intel SSE instructions [15] on Pentium III and IV. Both of these architectures further support single instruction, multiple data programming (SIMD), which makes it possible to unroll the interaction calculation twice when using 3DNow! and fourfold with SSE support (see Fig. 1). This increases GROMACS' performance by another factor of 2 compared to the already fast software square root implementation[2]. Using SSE instructions, the entire assembly Lennard-Jones inner loop takes 27 clock cycles per pair interaction on a Pentium III. For comparison, the GNU C library square root alone takes more than 60 clock cycles.

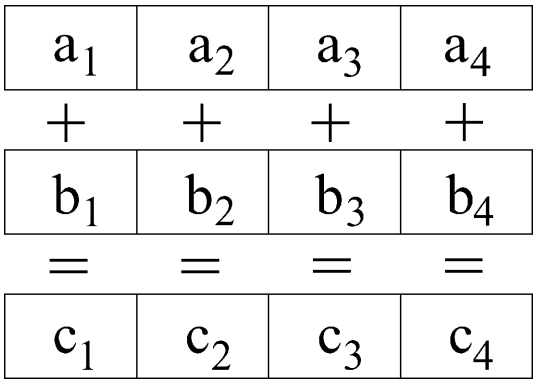

Fig. 1 The GROMACS assembly inner loops on x86 processors are implemented with SIMD instructions (single instruction, multiple data), demonstrated here by four parallel addition operations issued as a single instruction. The inner loops are thus effectively unrolled a factor of 4 if the CPU supports Intel SSE instructions and twice for AMD 3DNow! code. In practice the speedup is about a factor of 2 for both platforms due to differences in bus bandwidth and instruction pipeline length

File formats

The parameters and topology files used to create a GROMACS run input file are all in readable clear text and free layout format. Since the C preprocessor is used to scan topologies it is easy to include force fields, molecule definitions and even to introduce conditional parts

[1] Although many processors provide assembly instructions for the square root it is often only a mnemonic for a slow software calculation.

[2] The only twofold unrolling of 3DNow! compared to four with SSE is compensated for by a shorter pipeline and special instructions to perform the Newton–Raphson iterations.

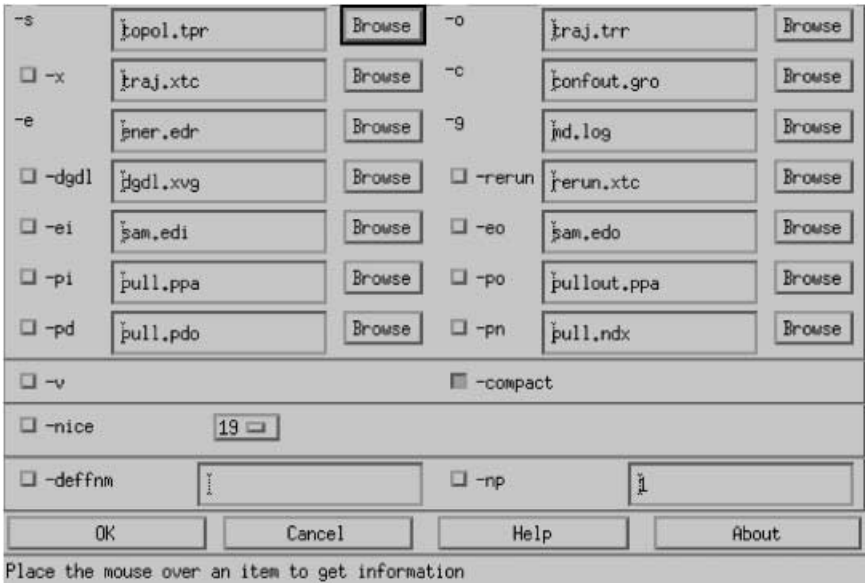

Fig. 2 Example X/Motif dialog box for the mdrun program, where input files and options can be selected graphically. Context sensitive help lines are displayed as the cursor is moved over the different objects, and more extensive help is available in a separate window

and macros. The input files for mdrun (the molecular dynamics program) as well as the energy and trajectory files are written in an architecture-independent format, making it possible to share files between big and small endian machines, and even to read double precision files with a single precision analysis program, and vice versa. Simulation coordinates can be written directly as compressed trajectories using only about 4 bytes for each coordinate triplet. This is achieved by a lossy compression algorithm that limits accuracy[3], but the level of compression can be selected by the user. The GROMACS libraries provide generic routines to read and write coordinate and/or velocity data, making them completely independent of the file formats. Every time a new format is added it is thus automatically supported by all programs in the package. It is also possible to compress most files with gzip or UNIX compress, and GROMACS will uncompress them on the fly upon reading.

User interface

The GROMACS user interface is designed to be simple and consistent – there is no scripting language involved. All programs are controlled by command line arguments and hence appear to the user as any other UNIX command. In each program the -h option yields a description of the function of the program and an overview of all arguments that may be given. This documentation is generated automatically from options and comments contained in the source code, and can be printed to the screen (using the -h option) or to a file in either HTML, LATEX or nroff (UNIX manual page) format. This ascertains that the documentation is kept up to date and that there are no differences between the various formats.

[3] Typically, coordinates are stored as integer multiples of picometers.

If Motif or lesstif is present on the build system, the configuration script automatically includes code for an X/Motif user interface to GROMACS; this means that in addition to the command line interface one can optionally select options by clicking in a dialog box (Fig. 2). The dialog box further features a context sensitive help line, which displays information about the field where the mouse cursor is positioned over. This may help new users to grasp an overview of the features in each program, while simultaneously providing a smooth transition to use of the command line interface later.

GROMACS includes a simple trajectory viewer (ngmx) for animation and general visualization that only requires standard X libraries, which are present on virtually every UNIX workstation. On systems that support OpenGL, users are encouraged to try external viewers, either the visual molecular dynamics (VMD) [16] program or gOpenMol. [17] Both of these natively support the GROMACS trajectory file formats and provide more advanced tools for visual inspection of simulation trajectories.

Most of the GROMACS analysis tools provide output data files formatted to serve as complete input for the Grace (formerly Xmgr) program, [18] a very capable graphing tool that is more or less a de facto standard under UNIX. Observables like energy, radius of gyration of a molecule, etc., are presented as functions of time in a graph including axis labels and legends.

Overview of features

Molecular dynamics

The GROMACS molecular dynamics code contains several algorithms that have been developed in the Berendsen group over the years, as well as many contributions from elsewhere. The leap-frog integrator is used

in two versions, for molecular dynamics and stochastic dynamics (velocity Langevin dynamics). [19] It was chosen after careful comparison with other integrators [20] due to its stability at long time steps and the advantage that it requires only one call to the constraining algorithm for each time step. For both algorithms GROMACS determines the kinetic energy from the velocity at the whole step, which is calculated as the average of the velocities at minus and plus half a step. This results in more accurate energies compared to the normal leap-frog velocities which are offset by half a step. There is also a first order integrator for Brownian dynamics (position Langevin dynamics). Each of these three integrators supports all features of GROMACS.

Simulations can be performed without periodic boundary conditions, or with periodic boundary conditions in a general triclinic cell. This means the program supports geometries like rectangular cells, the rhombic dodecahedron and the truncated octahedron, since all these are special cases of a triclinic cell. With periodic boundary conditions, a grid-based neighbor-searching is used to determine which interactions are within the cut-off distance. The GROMACS grid search algorithm and lattice summation methods are extremely efficient and just as fast for a triclinic cell as for a rectangular cell. This makes the rhombic dodecahedron the preferable cell for simulation of globular proteins in solvent; since its volume is only 71% of that of a cubic cell one can use fewer solvent molecules with the same periodic image distance. Pressure coupling is implemented for all types of simulation cells (including full support for cell deformation) using either the weak coupling scheme [21] or a Parinello–Rahman barostat. [22, 23] Temperature can be controlled groupwise, either with weak coupling or Nosé–Hoover thermostats. [24, 25] When periodic boundary conditions are applied, long-range interactions can be calculated using either the particle–particle particle–mesh algorithm [26] or the more accurate particle mesh Ewald [27, 28] sum. A reference (read inefficient) implementation of the standard Ewald algorithm [29] is also available.

GROMACS provides extensive support for constraint dynamics. The most generally applicable algorithm is still SHAKE, [30] in which the successive overrelaxation optimization [31] has been implemented. The default algorithm for pure bond constraints in GROMACS is, however, the non-iterative LINCS, [32] which is much more stable. LINCS makes it possible to use considerably longer time steps when the fast degrees of freedom due to hydrogen atoms are removed from the system (up to 6–7 fs when hydrogens are converted to dummy particles). [33] Both SHAKE and LINCS include support for free energy calculations where constraint lengths are changed. Water molecules can be constrained using the analytical SETTLE [34] algorithm leading to considerable performance improvement compared to SHAKE. Some additional algorithms and features implemented in the package are listed in appendix A.

Free energy calculations

GROMACS can calculate the free energy difference between two systems *A* and *B*, characterized by hamiltonians H_A and H_B, using a coupling parameter approach. The general Hamiltonian is constructed as a function of a parameter λ:

$$H=H(\lambda) \tag{3}$$

such that $H(0)=H_A$ and $H(1)=H_B$. The free energy difference between states *A* and *B* is obtained by performing simulations at several values of λ, from which $\langle \partial H(\lambda)/\partial\lambda \rangle$ is obtained. Integrating this quantity from $\lambda=0$ to 1 gives the free energy difference between states *A* and *B*. [35] This can be automated by changing the value of λ linearly with simulation time (a.k.a. the slow growth algorithm). It is, however, preferable to perform simulations at fixed λ values, such that equilibration of $\langle \partial H(\lambda)/\partial\lambda \rangle$ can be monitored. A numerical integration algorithm can then be used to compute the net free energy difference between states *A* and *B*.

The form of the potential at intermediate values of can be chosen freely, since these are non-physical states. The potential should be chosen such that $\langle \partial H(\lambda)/\partial\lambda \rangle$ is as smooth as possible. For the bonded interactions, GROMACS uses linear interpolation of force constants, bond lengths and angles with respect to the coupling parameter λ. Nonbonded interactions between pairs of particles of which at least one is perturbed are interpolated using the general soft-core potential

$$V_{sc}(r)=(1-\lambda)V^A(r_A)+\lambda V^B(r_B) \tag{4}$$

$$r_A=(\alpha\sigma_A{}^6\lambda^2+r^6)^{1/6} \tag{5}$$

$$r=[\alpha\sigma_B{}^6(1-\lambda)^2+r^6]^{1/6} \tag{6}$$

where V^A and V^B are the normal, hard-core, potentials and σ_A, σ_B the interaction radii of the pair of particles in state *A* and *B* respectively. The same σ's are used for the Van der Waals and electrostatics interactions. Normally the Lennard-Jones potential is used for the Van der Waals interactions, and then the σ's are conveniently chosen as the Lennard-Jones radii. Linear potential interpolation is obtained by setting the soft-core parameter α to zero. For $\alpha>0$ (normally a value around 1.5 is used), the singularities of the potential at $r=0$ for intermediate values of λ are removed. This makes it possible to let particles appear and disappear without encountering singularities in the potential, forces or $\partial H(\lambda)/\partial\lambda$. A more detailed description of the soft-core Lennard-Jones potential has been published by Beutler et al. [36] In GROMACS, the electrostatics and Van der Waals potentials are interpolated in the same manner to avoid creating multiple minima in the total non-bonded interaction between a pair of particles. Free energy calculations can be performed with all types of non-bonded electrostatics present in GROMACS, and with Lennard-Jones interactions. The Buckingham potential is not compatible with the soft-core formulation, but can be used with linear interpolation between the *A* and *B* states.

Force fields

The GROMOS87 [1] and GROMOS96 [37, 38] force fields for general biomolecular simulations are distributed with the current version of GROMACS, and fully integrated in the sense that complete topologies can be generated automatically from e.g. a Protein Data Bank file. The OPLS force field [39] has also been used in some GROMACS simulations. [40] Due to the clear text topology format it is straightforward to use any other united-atom or all-atom force field based on the general types of potential functions implemented in the code; there are for instance several different parameter sets for phospholipid molecules that have been used. [41, 42] Small molecules are easy to build and topology descriptions for e.g. a number of water models are available. This includes SPC, [43] SPC/E, [44] TIP3P and TIP4P, [45] TIP5P, [46] SPC/RF and TIP4P/RF [47] and two polarizable water models, of De Leeuw and Parker [48] and of Van Maaren and Van der Spoel. [49] Finally, work is in progress to include the OPLS [39] and AMBER [50, 51] force fields in the GROMACS package.

Potential functions that are implemented in the software include the Morse potential for bond stretching, [52] the Ryckaert–Bellemans potential for torsion angles [53] and the Buckingham potential for Van der Waals interactions. Electrostatic potentials include both normal Coulombic interactions, Coulomb modified by a reaction field, [54, 55] modified by a shift [56, 57] or switch [1] function. In addition, user-provided nonbonded pair potentials can be applied if they adhere to the following general form:

$$V_{ij}(\mathbf{r}_{ij})=A_{ij}f(\mathbf{r}_{ij})+B_{ij}g(\mathbf{r}_{ij})+(q_iq_j)/(4\pi\varepsilon_0)h(\mathbf{r}_{ij}) \quad (7)$$

The user should then provide input files containing tables of $f(x)$, $g(x)$ and $h(x)$ and their second derivatives, from which energy and force are calculated using a cubic spline interpolation algorithm.

Utility programs

The GROMACS package consists of about 75 executable programs. Most of them are analysis tools for trajectory or energy data generated by MD simulations. An overview of these analysis tools is available in appendix B. Here we briefly describe some of the most useful programs for setting up simulations. **pdb2gmx** generates molecular topology files from a Protein Data Bank coordinate file. The program automatically assigns bonds, angles, dihedrals and charges based on a residue topology database. Protonation states of ionizable groups are by default set according to pH 7, but it is also possible to select the states manually. For histidine, with a pK_a of about 6.5, the protonation state is determined from the hydrogen bonding pattern, or if the sidechain is solvent-exposed, a proton is put on the Nε only. The **pdb2gmx** program can also convert hydrogen atoms to dummy particles to remove the fastest degrees of freedom, in order to make it possible to use long time steps. **genbox** solvates molecules, either in a rectangular or triclinic box for simulations employing periodic boundary conditions or with a shell of solvent molecules. Mixed solvents of any composition can also be handled. **genion** places ions at positions of favorable electrostatic potential, and **x2top** generates a topology by detecting bonds and angles present directly from raw coordinates. Additional programs are available for checking the internal consistency of GROMACS files and for displaying the contents of binary files.

Other computational chemistry algorithms

A number of additional algorithms and methods in computational chemistry have been implemented in the GROMACS package as separate programs. The **nmrun** program performs normal mode analysis (NMA) [58, 59, 60] by calculating the Hessian matrix numerically from the forces. Separate programs to analyze the obtained Hessian are included. Principal component [61] or essential dynamics [62] analysis of molecular dynamics trajectories is also available. These algorithms can further be applied to collections of conformations of the same molecule including experimentally determined structures. [63, 64] Finally, a parallel implementation of the CONCOORD (prediction of protein conformational freedom from distance constraints) algorithm [65, 66] is provided in a master–slave algorithm that scales very well with the number of processors.

Benchmarks

GROMACS benchmarks

To present an overview of the GROMACS molecular dynamics performance and compare the simulation speed attainable on some of the most common hardware platforms we have constructed a benchmark consisting of a few typical systems, where possible from the published literature:

- The Villin headpiece, a 35-residue peptide the structure of which has been determined by NMR. [67] This was used in a peptide folding simulation of a microsecond by Duan and Kollman, [68] and we chose an essentially identical setup for this benchmark. The system was simulated with 3,000 water molecules in a truncated octahedron unit cell (slightly less than 10,000 atoms in total), using a group-based cut-off for both electrostatic and Lennard-Jones interactions at 0.8 nm. A time step of 2 fs was employed, LINCS [32] was used to constrain protein bonds involving hydrogens, while SETTLE [34] was used to maintain the water geometry. Neighbor-lists were used and updated after 20 fs. Temperature coupling [21] was applied with a time constant of 0.1 ps, and pressure cou-

pling with a time constant of 20 ps. For the peptide, the GROMOS96 force field [37] was used, while the water was described with the TIP3P model. [45]

- The larger lysozyme protein (pdb entry 2LZM) [69] using the GROMOS96 force field [37] in SPC water. [43] The total number of atoms was 23,207. All hydrogens on the protein were treated as dummy particles to remove the fast bond and angle vibrations, allowing a time step of 4 fs. [33] A rhombic dodecahedron simulation box was used, which is the most spherical alternative, and hence most suited to globular proteins. A twin-range group-based cut-off scheme was used for both Coulomb and Lennard-Jones. Interactions within 0.9 nm were calculated every step, and long-range forces out to 1.4 nm updated every five time steps during neighbor-list generation.
- To assess the performance of the lattice summation code, the lysozyme system was also benchmarked with the cut-off for Coulomb interactions replaced with the smooth Particle Mesh Ewald algorithm. [27, 28] The full direct and reciprocal space parts were calculated each step and a lattice spacing of 0.12 nm used. The direct space part of the Coulomb interaction was cut off at 0.9 nm while the Lennard-Jones interaction was calculated using a twin-range cut-off of 0.9 and 1.4 nm, as in the previous benchmark. Hence we can assess the computational cost of moving from a reasonable cut-off scheme to a much more accurate model employing PME.
- A phospholipid membrane, consisting of 1,024 dipalmitoylphosphatidylcholine (DPPC) lipids in a bilayer configuration with 23 water molecules per lipid, for a total of 121,856 atoms. The characteristics of this particular system have been described extensively by Lindahl and Edholm. [70, 71] It was simulated with a twin-range group-based cut-off of 1.8 nm for electrostatics and 1.0 nm for Van der Waals interactions. The long-range Coulomb forces between 1.0 nm and 1.8 nm were updated every tenth integration step during neighbor-list generation. The force field described by Berger et al. [42] was used for the lipids while the water was simulated with the SPC model. [43]
- A 6,000-unit polyethylene molecule modeled with anisotropic united atoms. [72] This means the bonded forces act on the position of the carbon, while the Lennard-Jones interaction site is a united atom displaced to the center of the CH_2 group. The force field of Boyd and coworkers was used, [73] with flexible bonds and a 1 fs time step. A 0.9 nm cut-off was employed for Lennard-Jones interactions and dispersion corrections applied. Although the anisotropic interaction sites increase the particle number to 12,000 they are easily implemented as dummy atoms and the computational cost is very close to that of a 6,000 particle system.

Six different machines were used for the full benchmark: 800 MHz AMD Athlon and dual 800 MHz Intel Pentium III processors running Linux (using gcc compilers), an SGI O200 equipped with a 270 MHz MIPS R12000, a Sun Ultra 10 with a 440 MHz Ultrasparc IIi, a Compaq ES40 with a 667 MHz Alpha 21264, and finally a single 395 MHz Power3 processor on an IBM SP2 winterhawk-2 node. The parallelization capabilities of the code were examined by running the lipid membrane benchmark using different number of processors on the IBM SP2 and on a low-cost Linux cluster of dual 800 MHz Pentium III machines using 100 Mbps ethernet communication. The parallel efficiency S is given by

$$S=t_N/(Nt_1) \tag{8}$$

where t_x is the performance on x processors. The results of the benchmarks are presented in Tables 1 and 2; thanks to the GROMACS SSE and 3DNow! assembly loops, the performance of the fast (expensive) Alpha and IBM processors is essentially matched by the PC hardware which is almost an order of magnitude cheaper, not to mention the performance of the dual processor Pentium III. This makes Linux and x86 processors a very attractive alternative for molecular dynamics simulations in most cases, as has been noted before. [74] The μs simulation of Duan and Kollman [68] could in principle be reproduced using a single dual Pentium III machine within 8 months. Furthermore, we see from the lysozyme benchmarks that the use of particle mesh Ewald is only slightly slower than using a cut-off.

For large-scale parallelization, the IBM Power Parallel architecture shows very good scaling. Due to better cache usage and the extremely high intra-node MPI bandwidth it even exhibits superscaling (S>100%) when no communication between different nodes is necessary.

Table 1 Benchmark system performance presented as ps/day (i.e. higher numbers are better) for a few common processor architectures. The hardware used was 800 MHz AMD Athlon and Intel Pentium III (P3) machines, a 270 MHz MIPS R12000 (SGI O200), a 440 MHz Sun Ultra 10, a Compaq ES40 (Alpha 21264/667 MHz single processor), and finally a single 395 MHz Power3 processor on an IBM SP2 node. The second column indicates the time steps Δt used for each system. Additional details on the simulation parameters are discussed in the benchmark section

System	Δt (fs)	Athlon	P3	P3-SMP	R12000	Ultra	Power3	Alpha
Villin	2	2412	2330	4080	1129	1115	2109	2982
Lys/Cut	4	622	662	1115	326	310	607	846
Lys/PME	4	456	455	608	311	250	393	709
DPPC	2	41	46	106	27	25	49	61
Polyethylene	1	1001	960	1385	910	760	1163	1670

Table 2 Parallel simulation performance versus number of processors *N* for the DPPC benchmark system on an IBM SP2 machine (395 MHz, four processors per node) and a Pentium III cluster (dual 800 MHz configurations) using switched 100 Mbps ethernet communication. The result is presented both as ps/day (higher is better) and scaling efficiency *S* is given by Equation (8). Due to better cache usage combined with high shared memory bandwidth, both the SP2 and the Pentium III exhibit significant superscaling (*S*>100%) on a single node. The Pentium performance, however, drops significantly when ethernet communication is involved

N	1	2	4	8	12	16	20	24	32
SP2/Power	49	105	205	387	537	694	796	884	1019
Scaling (%)	100	107	104	99	92	89	81	75	65
Pentium III	46	106	174	274	327	394			
Scaling (%)	100	116	95	75	60	54			

Table 3 Selected benchmark from other popular MD packages. Fastest reported results for each of these benchmarks (time in seconds, i.e. lower is better) are compared to running the same system with GROMACS on a single Pentium 3-800 MHz and a Compaq ES40 (Alpha 21264/667 MHz). Note: the AMBER benchmark (http://www.amber.ucsf.edu/) uses PME with a cut-off of 0.95 nm, the CHARMM benchmark [75] uses a shift function which shifts the potential to zero from 1.2–1.4 nm, while the GROMOS96 benchmark [76] uses a twin-range cut-off of 0.8/1.4 nm in which the long-range contribution to the forces and energy is updated every fifth MD step during neighbor searching and a reaction field is used for the Coulomb interactions. Furthermore the number of time steps was increased from 10 to 100 in the case of the GROMOS96 benchmark. For the GROMACS equivalent of the CHARMM benchmark the number of water molecules was increased to 4,126 since GROMACS does not have a real all-atom force field. This brings the total number of atoms to 14,026, identical to the original CHARMM benchmark

Benchmark	Other		GROMACS	
	Fastest machine(s)	Time	Pentium 3	Alpha
AMBER	Compaq Alpha ES40			
(4096 Water)	21264/667 MHz	62	36	27
CHARMM	NEC SX-4	813		
(MbCO+3830 Water)	ES40/667	1375	463	510
GROMOS96	DEC AS8400			
(Thrombin+5427 Water)	21264/575 MHz	168	31	26

It should be possible in general to increase the performance further by explicitly using thread and shared memory parallelization on SMP machines such as the IBM power nodes. On the SMP Pentium III we also find superscaling due to more effective cache usage. In fact, since the modern Pentium III processors only have 256 kb cache, performance is degraded considerably for large systems, which is amply demonstrated by the superscaling factor of 116% for the DPPC benchmark. Finally our parallel benchmark shows (Table 2) that 16 Pentium III CPUs effectively have the same parallel performance as eight IBM Power3 processors.

Other benchmarks

Since a considerable amount of time was spent optimizing GROMACS it is instructive to compare its performance to that of other popular packages. To this end we have run benchmark simulations from the AMBER suite, [50] from the CHARMM suite [75] and from GROMOS96. [76] We have attempted to mimic the original simulation as closely as possible. More in particular, the AMBER benchmark (4,096 water in a cubic box) uses the PME algorithm [27, 28] with a cut-off of 0.95 nm, the CHARMM benchmark (CO-MyoGlobin in a sphere of water molecules) uses a shift function to force the potential to zero over a range from 1.2 to 1.4 nm, while the GROMOS96 benchmark (Thrombin in a truncated octahedron box filled with water) uses a reaction field combined with a twin-range cut-off of 0.8/1.4 nm. All these features are present in GROMACS, allowing for a fair comparison. We have compared the fastest reported results for the other packages to GROMACS running on a single P3-800 and a Compaq ES40. The resulting numbers are presented in Table 3. Clearly, GROMACS is considerably faster than any other package. It is interesting to see that GROMACS on a single P3-800 is also considerably faster than CHARMM run on the NEC-SX4, one of the fastest supercomputers available today. The difference between GROMACS and AMBER is not as large as with the other packages, but this is due to the PME algorithm for which both packages use the same (original) implementation. The difference between the two is thus mainly due to faster calculation of the short-range forces. A comparison between GROMACS and AMBER without PME is given by the Villin benchmark (see previous paragraph). Finally it should be stressed here once more that the GROMACS code uses single precision while the other packages use double precision floating-point calculations. Depending on the architecture of the machine this can make a difference of between 0 (IBM) and 100% (Pentium, due to lack of double precision SSE routines). On other RISC architectures such as the Alpha processors the runtime would be increased by a relatively small amount. In general we

consider single precision to be accurate enough for molecular dynamics since the main error in the energy is determined by other factors such as cut-off, temperature scaling algorithms etc.

Conclusions and future development

Amidst the general molecular simulation packages available to researchers, we believe GROMACS stands out for a number of reasons. First, the code is extremely well tuned for the most common hardware in modern workstations, especially AMD Athlon/Duron and Intel Pentium III/IV processors, which also are common in modern Linux clusters. This, in combination with advanced algorithmic optimizations, makes it very hard to beat the GROMACS code performance-wise. Second, the package provides a broad range of options for molecular simulations, several of which make it possible to extend simulations much further using longer time steps and removal of fast degrees of freedom. There is also a wealth of analysis tools, and innovative features like hardware-independent and precision-independent trajectory formats. Third, the code is not merely free, but distributed under the GNU General Public License (GPL). This means any user is free to redistribute it, although we strongly suggest modifications are communicated back to the GROMACS authors for inclusion in the official distribution, in order to guarantee the scientific integrity of the software.

Several new features are planned for future GROMACS versions. A hybrid quantum mechanics/molecular mechanics (QM/MM) interface between GROMACS and the quantum chemistry package GAMESS-UK [77] is being developed that will allow the user to partition a system into a QM and MM region. This will provide a possibility to study bond-breaking and bond-forming processes in the QM region by solving the electronic Schrödinger equation. The MM region, which only influences the reaction via electrostatic interactions, is treated by conventional force-field methods. The main features of our hybrid will include geometry optimization, intrinsic reaction coordinate calculation, and transition-state optimization. A prominent application of the methodology is the elucidation of enzymatic reaction pathways. Other development efforts include a twin-level parallelization setup, using explicit multithreading inside each node and MPI to communicate between nodes. This will make both intra-node and inter-node parallelization on SMP machines (including the ones used in the benchmarks, Table 2) more efficient, because fewer, but larger, chunks of data are communicated between the nodes. Furthermore a new self-descriptive file format based on XML (extensible markup language) is planned. The most recent version of the official GROMACS distribution and additional on-line resources can be found on the project homepage http://www.gromacs.org/, hosted by the Department of Biophysical Chemistry, University of Groningen.

Acknowledgements We thank Gerrit Groenhof for a sneak preview of the QM/MM interface code that is currently under construction. Maija Lahtela-Kakkonen and the Center for Scientific Computing, Espoo, Finland are gratefully acknowledged for providing access to their IBM SP2 for benchmarking. Frans van Hoesel is acknowledged for creating and implementing the portable compressed trajectory format. Finally, we acknowledge stimulating discussions with Herman Berendsen and Alan Mark.

Appendix A. Additional package features

GROMACS implements a wide range of algorithms from molecular dynamics and related areas. Apart from the general architecture and methods discussed before, some of the most prominent features are:

- Support for Reaction field [55] potentials that can be applied to liquid systems, and the generalized reaction field (GRF) approach due to Tironi et al. [78] (note that this is different from the GRF of Alper and Levy), [79] which can be used for ionic systems. There is no torque due to a reaction field included in the software. [80]
- A force shifting algorithm that changes energies and forces to approach zero smoothly at the cut-off radius, [56] as well as a switch function, [1] in which the energy goes to zero smoothly, but not the forces. GROMACS can also apply analytic corrections to either energy or both energy and pressure for dispersion interactions beyond the cut-off. [81]
- Energy minimization using either steepest descent or conjugate gradients algorithms.
- Support for determining the potential of mean force when pulling a molecule (e.g. a lipid out of a membrane). [82] With this code it is possible to simulate atomic force microscopy experiments.
- NMR refinement based on time average distance restraints [83, 84] and ensemble averaged distance restraints. [85, 86]
- X-ray interaction with matter can be simulated to model the effects of radiation damage on a sample. [87]
- Shell molecular dynamics [88] introduces shell particles on nuclei or other sites in a molecule to represent the electronic degrees of freedom [49, 89] and hence accounts for polarizability in the simulation. The shell particles positions can be optimized at each time step using a steepest descent energy minimization.
- Dummy or virtual particles can be used, either to extend a model like in TIP4P [45] or TIP5P [46] water, or to remove rapid oscillations with small amplitude and hence allow for increased time steps. [33] Construction of dummy positions from constituting atoms and distribution of forces to constituting atoms is done according to Reference [21]
- Extended sampling techniques [90, 91] based on eigenvectors from essential dynamics or normal modes from normal mode analysis. The input files necessary for this type of simulation can be prepared with the WhatIF [92] software.

- Non-equilibrium molecular dynamics options, including accelerations on arbitrary groups of atoms and electric fields.
- Non-equilibrium methods for determining the shear viscosity of liquids. A cosine-shaped velocity profile is generated using a cosine-shaped acceleration profile. The viscosity can then be calculated from the ratio of the two amplitudes.
- Automated force field optimization using general coupling theory. [93, 94] This has been applied successfully to optimizations of parameters for novel water models. [47, 49]

Appendix B. Analysis tools

A large number of complete programs with extensive options to perform common trajectory analyses are distributed with the GROMACS package. Most of these analyses can be performed either on default, or 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:

- Detailed 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. [81]
- Transverse current autocorrelation functions to determine the shear viscosity from an equilibrium simulation.
- Root mean square deviations between a reference structure and a trajectory, or a complete RMSD matrix of a trajectory against itself or against another trajectory.
- Dielectric analysis to calculate dielectric constants and Kirkwood factors [95] and Cole–Cole plots.
- Radial distribution functions.
- Analysis of the orientation of solvent molecules around solutes.
- Hydrogen bond analysis based on geometric criteria.
- 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) [96] program. Pretty (PostScript) plots of the secondary structure as a function of time can be made.
- Solvent accessible surface of macromolecules can be computed from trajectories or structures using the algorithms of Eisenhaber et al. [97]
- Chemical shifts can be computed based on Φ/Ψ angles [98] or protein conformations. [99]
- Relaxation times for molecular motions can be calculated in a form directly comparable to NMR results. [47]
- Order parameters for lipid carbon tails. [100, 101]
- Density profiles, either along a box axis which is useful for interface studies, [102, 103] or radial for systems like a micel. [104]
- 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. [105] A time dependent Ramachandran plot can be visualized with an X 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. [106, 107]
- 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.

References

1. van Gunsteren WF, Berendsen HJC, Biomos BV (1987) Gromos-87 manual. Nijenborgh 4, 9747 AG Groningen, The Netherlands
2. Askew CR, Carpenter DB, Chalker JT, Hey AJG, Nicole DA (1986) Comput Phys Commun 42:21–26
3. Raine ARC, Fincham D, Smith W (1989) Comput Phys Commun 55:13–30
4. Bekker H, Berendsen HJC, Dijkstra EJ, Achterop S, van Drunen R, van der Spoel D, Sijbers A, Keegstra H, Reitsma B, Renardus MKR (1993) In: de Groot RA, Nadrchal J (eds) Physics computing 92. World Scientific, Singapore, pp 252–256
5. Berendsen HJC, van der Spoel D, van Drunen R (1995) Comput Phys Commun 91:43–56
6. van der Spoel D, Berendsen HJC (1994) In: van der Steen A. (ed) Aspects of computational science. National Computing Facilities Foundation: P.O. Box 93120, 2509 AC Den Haag, The Netherlands
7. Frigo M, Johnson SG (1998) In: ICASSP Conference Proceedings, Vol 3, Seattle, pp 1381–1384
8. Frigo M (1999) ACM SIGPLAN NOTICES 34:169–180
9. van der Spoel D, van Buuren AR, Apol E, Meulenhoff PJ, Tieleman DP, Sijbers ALTM, Hess B, Feenstra KA, Lindahl E, van Drunen R, Berendsen HJC (2001) Gromacs User Manual version 3.0. Nijenborgh 4, 9747 AG Groningen, The Netherlands Internet: http://www.gromacs.org
10. Bekker H, Berendsen HJC, Dijkstra EJ, Achterop S, van Drunen R, van der Spoel D, Sijbers A, Keegstra H, Reitsma B, Renardus MKR (1993) In: de Groot RA, Nadrchal J (eds) Physics computing 92. World Scientific, Singapore, pp 257–261
11. Bekker H, Dijkstra EJ, Renardus MKR, Berendsen HJC (1995) Mol Simul 14:137–152
12. Verlet L (1967) Phys Rev 159:98–103

13. Press WH, Teukolsky SA, Vetterling WT, Flannery BP (1992) Numerical recipes in C. Cambridge University Press, Cambridge
14. AMD (2000) 3DNow! technology manual. Advanced Micro Devices.
15. Intel (1999) Intel architecture software developer's manual, Vol. 2. Instruction set reference. Intel Corporation
16. Humphrey W, Dalke A, Schulten K (1996) J Mol Graph 14:33–38
17. Laaksonen L (2001) gOpenMol, Version 2.0. Center for scientific computing, Espoo, Finland http://www.csc.fi/~laaksone/gopenmol/gopenmol.html
18. Grace Development Team (2001) Grace 5.1.3. http://plasma-gate.weizmann.ac.il/Grace
19. van Gunsteren WF, Berendsen HJC (1988) Mol Simul 1:173–185
20. Berendsen HJC, van Gunsteren WF (1986) In: Ciccotti G, Hoover WG (eds) Molecular-dynamics simulation of statistical-mechanical systems. North-Holland, Amsterdam, pp 43–65
21. Berendsen HJC, Postma JPM, DiNola A, Haak JR (1984) J Chem Phys 81:3684–3690
22. Parinello M, Rahman A (1981) J Appl Phys 52:7182–7190
23. Nosé S, Klein ML (1983) Mol Phys 50:1055–1076
24. Nosé S (1984) Mol Phys 52:255–268
25. Hoover WG (1985) Phys Rev A 31:1695–1697
26. Hockney RW, Eastwood JW (1981) Computer simulation using particles. McGraw-Hill, New York
27. Darden T, York D, Pedersen L (1993) J Chem Phys 98: 10089–10092
28. Essman U, Perera L, Berkowitz ML, Darden T, Lee H, Pedersen LG (1995) J Chem Phys 103:8577–8592
29. Ewald PP (1921) Ann Phys 64:253–287
30. Ryckaert JP, Ciccotti G, Berendsen HJC (1977) J Comput Phys 23:327–341
31. Barth E, Kuczera K, Leimkuhler B, Skeel D (1996) J Comput Chem 16:1192–1209
32. Hess B, Bekker H, Berendsen HJC, Fraaije JGEM (1997) J Comput Chem 18:1463–1472
33. Feenstra KA, Hess B, Berendsen HJC (1999) J Comput Chem 20:786–798
34. Miyamoto S, Kollman PA (1992) J Comput Chem 13:952–962
35. Kirkwood JG (1935) J Chem Phys 3:300
36. Beutler TC, Mark AE, van Schaik RC, Gerber PR, van Gunsteren WF (1994) Chem Phys Lett 222:529–539
37. van Gunsteren WF, Billeter SR, Eising AA, Hünenberger PH, Krüger P, Mark AE, Scott WRP.; Tironi IG (1996) Biomolecular simulation: the GROMOS96 manual and user guide. Hochschulverlag AG an der ETH Zürich, Zürich, Switzerland
38. Scott WRP, Hünenberger PH, Tironi IG, Mark AE, Billeter SR, Fennen J, Torda AE, Huber T, Kruger P, van Gunsteren WF (1999) J Phys Chem A 103:3596–3607
39. Jorgensen WL, Tirado-Rives J (1988) J Am Chem Soc 110:1657–1666
40. van der Spoel D, van Buuren AR, Tieleman DP, Berendsen HJC (1996) J Biomol NMR 8:229–238
41. Tieleman DP, Berendsen HJC (1996) J Chem Phys 105: 4871–4880
42. Berger O, Edholm O, Jähnig F (1997) Biophys J 72:2002–2013
43. Berendsen HJC, Postma JPM, van Gunsteren WF, Hermans J (1981) In: Pullman B (ed) Intermolecular forces. Reidel, Dordrecht, pp 331–342
44. Berendsen HJC, Grigera JR, Straatsma TP (1987) J Phys Chem 91:6269–6271
45. Jorgensen WL, Chandrasekhar J, Madura JD, Impey RW, Klein ML (1983) J Chem Phys 79:926–935
46. Mahoney MW, Jorgensen WL (2000) J Chem Phys 112:8910–8922
47. van der Spoel D, van Maaren PJ, Berendsen HJC (1998) J Chem Phys 108:10220–10230
48. de Leeuw NH, Parker SC (1998) Phys Rev B 58:13901–13908
49. van Maaren PJ, van der Spoel D (2001) J Phys Chem B 105: 2618–2626
50. Weiner SJ, Kollman PA, Nguyen DT, Case DA (1986) J Comput Chem 7:230–252
51. Cornell WD, Cieplak P, Bayly CI, Gould IR, Merz Jr KM, Ferguson DM, Spellmeyer DC, Fox T, Caldwell JW, Kollman PA (1995) J Am Chem Soc 117:5179–5197
52. Morse PM (1929) Phys Rev 34:57–64
53. Ryckaert J, Bellemans A (1975) Chem Phys Lett 30:123–125
54. Watts RO (1974) Mol Phys 28:1069–1083
55. Neumann M, Steinhauser O (1980) Mol Phys 39:437–454
56. Berendsen HJC (1993) In: van Gunsteren WF, Weiner PK, Wilkinson AJ (eds) Computer simulation of biomolecular systems. ESCOM, Leiden, pp 161–181
57. van der Spoel D, van Buuren AR, Apol E, Meulenhoff PJ, Tieleman DP, Sijbers ALTM, Hess B, Feenstra KA, van Drunen R, Berendsen HJC (1999) Gromacs User Manual version 2.0. Nijenborgh 4, 9747 AG Groningen, The Netherlands. Internet: http://www.gromacs.org
58. Gõ N, Noguti T, Nishikawa T (1983) Proc Natl Acad Sci USA 80:3696–3700
59. Brooks B, Karplus M (1983) Proc Natl Acad Sci USA 80: 6571–6575
60. Levitt M, Sander C, Stern PS (1983) Proc Natl Acad Sci USA 10:181–199
61. Garcia AE (1992) Phys Rev Lett 68:2696–2699
62. Amadei A, Linssen ABM, Berendsen HJC (1993) Proteins: Struct Funct Gen 17:412–425
63. van der Spoel D, de Groot BL, Hayward S, Berendsen HJC, Vogel H J (1996) Protein Sci 5:2044–2053
64. de Groot BL, Hayward S, van Aalten DMF, Amadei A, Berendsen HJC (1998) Proteins: Struct Funct Gen 31:116–127
65. de Groot BL, van Aalten DMF, Scheek RM, Amadei A, Vriend G, Berendsen HJC (1997) Proteins: Struct Funct Gen 29:240–251
66. de Groot BL, Vriend G, Berendsen HJC (1999) J Mol Biol 286:1241–1249
67. McKnight CJ, Matsudaira PT, Kim PS (1997) Nature Struct Biol 4:180–184
68. Duan Y, Kollman PA (1998) Science 282:740–744
69. Weaver LH, Matthews BW (1987) J Mol Biol 193:189
70. Lindahl E, Edholm O (2000) J Chem Phys 113:3882–3893
71. Lindahl E, Edholm O (2000) Biophys J 79:426–433
72. Toxvaerd S (1990) J Chem Phys 93:4290–4295
73. Pant PK, Han J, Smith GD, Boyd RH (1993) J Chem Phys 99: 597–604
74. Rives JT, Jorgensen WL (1996) J Comput Chem 17:1385–1386
75. Brooks BR, Hodoscek MC (1992) Des Autom News 7:16–22
76. Bonvin AMJJ, Mark AE, van Gunsteren WF (2000) Comput Phys Commun 128:550–557
77. Gamess-UK is a package of ab initio programs written by M.F. Guest, J.H. van Lenthe, J. Kendrick, K. Schoeffel, P. Sherwood, and R.J. Harrison, with contributions from R.D. Amos, R.J. Buenker, M. Dupuis, N.C. Handy, I.H. Hillier, P.J. Knowles, V. Bonacic-Koutecky, W. von Niessen, A.P. Rendell, V.R. Saunders, and A. Stone. The package is derived from the original gamess code due to M. Dupuis, D. Spangler and J. Wendoloski, NRCC software catalog, Vol. 1, Program No. QG01 (GAMESS), 1980.
78. Tironi IG, Sperb R, Smith PE, van Gunsteren WF (1995) J Chem Phys 102:5451–5459
79. Alper H, Levy RM (1993) J Chem Phys 99:9847–9852
80. Roberts JE, Woodman BL, Schnitker J (1996) Mol Phys 88: 1089–1108
81. Allen MP, Tildesley DJ (1987) Computer simulations of liquids. Oxford Science Publications, Oxford
82. Marrink SJ, Bergera O, Tieleman DP, Jähnig F (1998) Biophys J 74:931–943
83. Torda AE, Scheek RM, van Gunsteren WF (1989) Chem Phys Lett 157:289–294
84. Torda AE, Scheek RM, van Gunsteren WF (1990) J Mol Biol 214:223–235

85. Scheek RM, Torda AE, Kemmink J, van Gunsteren WF (1991) Computational In: Hoch J, Poulsen FM, Redfield C (eds) Aspects of the Study of Biological Macromolecules by Nuclear Magnetic Resonance. Plenum, New York, pp 209–217
86. Kemmink J, Scheek RM (1995) J Biomol NMR 6:33–40
87. Neutze R, Wouts R, van der Spoel D, Weckert E, Hajdu J (2000) Nature 406:752–757
88. Dick BG, Overhauser AW (1958) Phys Rev 112:90–103
89. Jordan PC, van Maaren PJ, Mavri J, van der Spoel D, Berendsen HJC (1995) J Chem Phys 103:2272–2285
90. de Groot BL, Amadei A, van Aalten DMF, Berendsen HJC (1996) J Biomol Struct Dyn 13:741–751
91. de Groot BL, Amadei A, Scheek RM, van Nuland NAJ, Berendsen HJC (1996) Proteins: Struct Funct Genet 26:314–322
92. Vriend G (1990) J Mol Graph 8:52–56
93. Chung JW (1993) Force field optimisation using coupling theory. Master's thesis, University of Groningen
94. Njo SL, Gunsteren WF, Müller-Plathe F (1995) J Chem Phys 102:6199–6207
95. Neumann M (1986) J Chem Phys 85:1567–1580
96. Kabsch W, Sander C (1983) Biopolymers 22:2577–2637
97. Eisenhaber F, Lijnzaad P, Argos P, Sander C, Scharf M (1995) J Comput Chem 16:273–284
98. Wishart DS, Nip AM (1998) Biochem Cell Biol 76:153–163
99. Williamson MP, Asakura T (1993) J Magn Reson Ser B 101:63–71
100. Marrink SJ, Berkowitz M, Berendsen HJC (1993) Langmuir 9:3122–3131
101. Egberts E, Marrink SJ, Berendsen HJC (1994) Eur Biophys J 22:423–436
102. van Buuren AR, Berendsen HJC (1994) Langmuir 10:1703–1713
103. van Buuren AR, de Vlieg J, Berendsen HJC (1995) Langmuir 11:2957–2965
104. Tieleman DP, van der Spoel D, Berendsen H JC (2000) J Phys Chem B 104:6380–6388
105. Ramachandran GN, Ramakrishnan C, Sasisekharan V (1963) J Mol Biol 7:95–99
106. Torda AE, van Gunsteren WF (1994) J Comput Chem 15: 1331–1340
107. Daura X, Gademann K, Jaun B, Seebach D, van Gunsteren WF, Mark AE (1999) Angew Chem Int Ed Engl 38:236–240