



# M.DynaMix – a scalable portable parallel MD simulation package for arbitrary molecular mixtures

Alexander P. Lyubartsev, Aatto Laaksonen \*

*Division of Physical Chemistry, Arrhenius Laboratory, University of Stockholm, S-106 91 Stockholm, Sweden*

Received 8 September 1999; accepted 10 November 1999

## Abstract

A general purpose, scalable parallel molecular dynamics package for simulations of arbitrary mixtures of flexible or rigid molecules is presented. It allows use of most types of conventional molecular-mechanical force fields and contains a variety of auxiliary terms for inter- and intramolecular interactions, including an harmonic bond-stretchings. It can handle both isotropic or ordered systems. Besides an NVE<sup>MD</sup> ensemble, the simulations can also be carried out in either NVT or NPT ensembles, by employing the Nosé–Hoover thermostats and barostats, respectively. If required, the NPT ensemble can be generated by maintaining anisotropic pressures. The simulation cell can be either cubic, rectangular, hexagonal or a truncated octahedron, with corresponding periodic boundary conditions and minimum images. In all cases, the optimized Ewald method can be used to treat the Coulombic interactions. Double time-step or constrained dynamics schemes are included. An external electric field can be applied across the simulation cell. The whole program is highly modular and is written in standard Fortran 77. It can be compiled to run efficiently both on parallel and sequential computers. The inherent complexity of the studied system does not affect the scalability of the program. The scaling is good with the size of the system and with the number of processors. The portability of the program is good, it runs regularly on several common single- and multiprocessor platforms, both scalar and vector architectures included. © 2000 Elsevier Science B.V. All rights reserved.

PACS: 02.70.Ns; 34.20.Gj; 61.20.Ja; 82.20.Wt

Keywords: Parallel algorithms; Molecular dynamics; Computer simulations

## PROGRAM SUMMARY

*Title of the program:* M.DynaMix

*Catalogue identifier:* ADLW

*Program Summary URL:* <http://cpc.cs.qub.ac.uk/summaries/ADLW>

*Computer for which the program is designed and others on which it has been tested:* The program is not designed for any particular

computer. It has been tested on (a) single processor computers: IBM RISC 600; DEC Alpha; Pentium PC/Linux; Pentium PC/Windows with PGI (Portland Group) Fortran compiler; (b) parallel systems: Cray T3E, IBM SP2; Linux SMP; Linux network clusters; (c) vector processors: Fujitsu VX

*Installations:* Sample makefiles are provided for the architectures listed above

*Operating systems:* UNIX (preferred); Windows 98/NT (not thoroughly tested)

\* Corresponding author. E-mail: aatto@tom.fos.su.se.

*Programming languages:* Fortran 77, with a few auxiliary routines in C

*Memory required to execute with typical data:* Depends on the size of the simulated system, simulation parameters and architecture in hand (some data structures use distributed memory). As an example for a 10000 atoms system on a single-processor computer, 64 MB is suggested

*Number of processors used:* Arbitrary, but effective speedup depends on the communication bandwidth. For parallel computers with distributed memory (IBM SP2, Cray T3E) very good scaling is observed up to 32 processors. In general, the scaling properties increase with the number of particles

*Parallelization:* MPI protocol

*Number of bytes in distributed program, including test data, etc.:* 548 659 bytes

*Distribution format:* tar gzip file

*Keywords:* Parallel algorithms, molecular dynamics, computer simulations

#### *Nature of physical problem*

Many-body problem with interacting particles. Structural, thermodynamical and dynamical properties of molecular liquids and liquid mixtures, including organic molecules or biomacromolecules as solutes.

#### *Method of solution*

Numerical integration of classical (Newtonian) equations of motion, optionally modified for constant temperature or/and constant pressure simulations. Forces are calculated from standard molecular-mechanical force fields.

#### *Restrictions on the complexity of the problem*

The principal limitation is the size of the non-bonded neighbor lists, but it rarely reaches the limits of RAM memory available. In practice, typical systems consist of about  $10^4$  atoms, covered during several nanoseconds on modern parallel computers (of Cray T3E, IBM SP2 type).

#### *Typical running time*

Varies greatly depending on the complexity of the problem. Typically for a system of 10000 atoms, a simulation of 1 nano-second would take several days of CPU time using 16–32 processors.

## LONG WRITE-UP

### 1. Introduction

Within three decades, computer simulation methods, such as Molecular Dynamics (MD) and Monte Carlo (MC), have become important computational techniques for studying various types of condensed matter. Besides computational chemists, many experimentalists now also routinely use molecular simulations in their efforts to interpret observed data. Not only these computer experiments can provide a link between theory and experiment, but also provide a way to investigate complex many-body systems when analytical theories fail and experimental techniques are impossible to use or simply do not exist.

Investigators now routinely simulate molecular systems consisting of some 1000 particles, which in some cases (e.g., isotropic liquids) are sufficiently large to give a good description of the corresponding macroscopic properties. For other systems, a considerably larger number of particles is needed in order to describe them in a realistic way. Inclusion of complex bio- or organic molecules (e.g., carbohydrates, proteins, fragments of nucleic acids or membranes, etc.), immersed in a solvent will rapidly increase the total number of involved atoms by one or two, or even more orders of magnitude. Also, the larger the molecular systems grow, the longer simulations are needed to statistically follow various low-amplitude motions and possible slow conformational transitions. The recent rapid progress rate towards computer simulations of very much larger molecular systems than was possible a decade ago, and the possibility to apply more complex molecular models, is set to a large extent by the advances in microprocessor technology and new computer architectures, in parallel with the development of appropriate algorithms and software.

Computer simulations of many-particle systems are well suited for parallel computing. There are several ways to decompose the particle or interaction space into independently treatable parts. However, the optimal parallel scheme for a particular problem always depends both on the computer architecture in hand and on the system

under investigation (model, size, type of interactions, etc.). Electrostatic interactions, fast intramolecular motions due to explicit modeling of the light hydrogens, angle bending and fluctuations of the internal torsional-angle forces in macromolecules – all these features require their own special treatment in an effectively parallelized molecular computer simulation code.

Originally, from our own broad needs, we have developed a general purpose molecular dynamics simulation package, called *M.DynaMix* to emphasize that it is designed for simulations of arbitrary mixtures of both rigid and flexible molecules. It employs modern simulation techniques for high-quality simulations: a double time step algorithm for fast and slow modes, an optimized Ewald method for electrostatic interactions, a constant-temperature–constant-pressure algorithm. It is not built around any particular force field, but rather can make use of nearly all available conventional force fields, including AMBER [1], CHARMM [2] or GROMOS [3]. Also, most water models, rigid or flexible, can be plugged into the simulations of biomolecular systems in aqueous solutions. It is particularly important to have water models with a proper dielectric constant and diffusion coefficient when used as a solvent for biological systems. All these features make the code fairly universal and well suitable for simulation of both simple molecules and complex biological macromolecules. The program is written in standard Fortran 77 and is highly portable, being able to run on any parallel system with the MPI-library installed. Equally important is that the program is designed to be easily compiled to run on any single-processor computer without the MPI library. The code is constructed from a large number of modules, and the source code is extensively commented. Details about the program organization are given below, together with the algorithms used in the parallelization. The structure of the input files is also given (see sample in Appendix II). Finally, a short user manual is provided.

## 2. Molecular dynamics simulation method

Concerning details about the basic MD method itself the user is referred, for example, to the textbook by Allen and Tildesley [10] or the more recent one by Smit and Frenkel [11]. We will only discuss the details and strategies as implemented in this particular program package.

### 2.1. Force field

In the conventional molecular dynamics method, all the simulated atoms in the systems move according to the classical Newtonian equations. The forces acting on the atoms are defined from the gradients of the potential energy (force field), as functions of the distances between all interaction sites in the system (most often the atoms).

In our program we have implemented a typical core form of a standard force field, compatible with most available force fields, with the following functional form:

$$\begin{aligned}
 V = & \sum_{\text{covalent bonds}} K_b(r - r_{\text{eq}})^2 \\
 & + \sum_{\text{covalent angles}} K_a(\theta - \theta_{\text{eq}})^2 \\
 & + \sum_{\text{torsional angles}} \frac{1}{2} K_t(1 + \cos(m_t\phi - \gamma_t)) \\
 & + \sum_{i < j} \left\{ 4\varepsilon_{ij} \left[ \left( \frac{\sigma_{ij}}{r_{ij}} \right)^{12} - \left( \frac{\sigma_{ij}}{r_{ij}} \right)^6 \right] + \frac{q_i q_j}{r_{ij}} \right\}, \quad (1)
 \end{aligned}$$

where the first sum runs over all the covalent bonds, the second over all the covalent bond angles, the third over all the torsional angles and the fourth over all the non-bonded atom pairs.  $r_{ij}$  is the distance between the atoms  $i$  and  $j$ . The cross-terms of the Lennard-Jones parameters  $\sigma$  and  $\varepsilon$  are calculated by the usual Lorentz–Berthelot combination rules:

$$\varepsilon_{ij} = (\varepsilon_i \varepsilon_j)^{1/2}, \quad \sigma_{ij} = (\sigma_i + \sigma_j)/2. \quad (2)$$

The atoms in a molecular system are considered as non-bonded if they are found in different molecules, or within the same molecule but separated by three or more covalent bonds. In some cases, those atoms separated by exactly three covalent bonds, are considered as so called “1–4 interactions”, for which the electrostatic and Lennard-Jones terms are often scaled by some factor between 0 and 1. This possibility is implemented in our program as well, in order to allow use of different force fields.

In addition to the standard form of the force field, given above in Eq. (1), several additional potential terms are available in the package, and these can be used either alone or together with any of the other terms.

These are:

- (1) Morse type of potential for covalent bonds:

$$V_{\text{bond}} = D(1 - \exp(-\rho(r - r_{\text{eq}}))).$$

- (2) MM3 force field [12] type of potential for torsional angles:

$$V_{\text{tors}} = V_1 \frac{1 + \text{Cos}(\phi)}{2} + V_2 \frac{1 - \text{Cos}(2\phi)}{2} + V_3 \frac{1 + \text{Cos}(3\phi)}{2}.$$

- (3) Ryckaert–Bellemans [13] type of potential for torsional angles (intended originally for hydrocarbon chains and polymers):

$$V_{\text{tors}} = \sum_{i=1}^5 (V_i (\text{Cos}(\phi - 180))^i).$$

- (4) Improper torsional angle potential (for covalent bonds junctions) to enhance the planarity of the molecule:

$$V_{\text{imp}} = V_{\text{imp}} \frac{(\psi - \psi_{\text{eq}})^2}{2}.$$

- (5) A narrower additional potential well for hydrogen bonds to better control the hydrogen bond distances:

$$V_{\text{H-bond}} = \sum_{\text{H-bonds}} \left( \frac{C_{ij}}{r_{ij}^{12}} - \frac{D_{ij}}{r_{ij}^{10}} \right).$$

- (6) External electric field along  $z$ -axis, periodically fluctuating in time:

$$V_{\text{ext}} = E_{\text{ext}} \text{Cos}(\omega t) q_i.$$

- (7) Artificial harmonic potential:

$$V_{\text{harm}} = \sum_{\text{some atoms}} K (r_i - r_{i0})^2,$$

which can be used to fix a set of chosen atoms in a harmonic potential well of some effective radius. This may be useful in the simulation of macromolecules, if one wants to maintain the global structure intact but allow local fluctuations.

- (8) Potential for flexible SPC water model by Toukan and Rahman [4]. This water model includes some cross-terms for intramolecular motion not included in the standard form of the force field (Eq. (1)).

## 2.2. Integration of the equations of motion

Several MD schemes and statistical mechanical environments are implemented in the package:

- (1) Standard Newtonian NVE molecular dynamics with the Verlet Leap-frog algorithm [10]. If needed, the temperature can be controlled by scaling the velocities. Of course, the temperature scaling is recommended only in the initial states of those simulations where dynamical properties will be calculated.
- (2) Constant-temperature Nosé–Hoover molecular dynamics [5] in a constant-volume cell.
- (3) Constant-temperature–constant-pressure molecular dynamics [6] with isotropic cell fluctuations.
- (4) Constant-temperature–anisotropic constant-pressure molecular dynamics [6], allowing the simulation cell to fluctuate separately in the three cell coordinate directions.

In the case of flexible molecular models, the double (multiple) time-step algorithm by Tuckerman et al. [7] is implemented. Forces due to fast intramolecular motion as well as due to fast nearest short-range (typically, within 5 Å) non-bonded interactions, are recalculated at every short time step, while the more slowly fluctuating internal forces and those due to long-range non-bonded interactions are recalculated at each long time step. The ratio between the short and long time steps is determined by the studied system. Typical values are 0.2 and 2.0 fs for the short and long step, respectively. Our implementation of the multiple time step algorithm together with NVT or NPT molecular dynamics follows closely the algorithms given by Martyna et al. [8]. If rigid molecular models are used, or if the fast motion of the bond stretching needs to be frozen, a single time-step algorithm of leap-frog type is applied, using constrained dynamics by means of the SHAKE algorithm [9].

### 2.3. *Treatment of long-range electrostatic interactions*

Two ways of treating the long-range electrostatic interactions are currently implemented in the program: the Ewald method [10] and the reaction field [14]. In the Ewald method, the intermolecular Coulomb forces are divided into long-range and short range components. The long-range part is calculated in the reciprocal space, while the short-range part is treated alongside with Lennard-Jones forces. The convergence of each part is controlled by parameters, specified in the input file. The total CPU time depends also on the cut-off radius used for calculation of the electrostatic forces. The optimal choice of the Ewald sum convergence parameter leads to a scaling of the CPU time as  $O(N^{3/2})$  [16]. Besides Ewald and reaction field, it is also possible to choose a plain Coulomb-law term. This possibility is useful, for example, in vacuum simulations.

In the reaction field method, the electrostatic interactions are cut beyond a specified radius, and the long-range correction from the polarization of the medium outside the cut-off sphere is calculated within the mean-field theory, depending on the dielectric permittivity and ionic strength of the surrounding solution. The working formulas, implemented in the package, are taken from the work of Tirion et al. [14].

### 2.4. *Geometry of the simulation cell*

The program currently supports four types of simulation cell geometries with full translational symmetry: cubic, rectangular, hexagonal and truncated octahedron. Periodic boundary conditions together with the minimum image convention are applied in all cases. The Ewald summation method is implemented for all four cell geometries.

## 3. Implementation

### 3.1. *General organization of the program*

The entire program source code consists of two INCLUDE files and a number of FORTRAN files (modules), performing different tasks or groups of tasks. All the functions are extensively commented in the source files. The first INCLUDE file, dimpar.h, defines the boundaries of the working arrays. Dimensions specified in this file should be checked and edited if needed before the program is compiled, to fit the molecular system one is about to simulate into the available RAM memory. The second INCLUDE file, prcm.h, contains definitions of data structures and

COMMON blocks. The COMMON blocks are the main way to transfer data between different subroutines in the program. Other important FORTRAN files (modules, consisting of blocks of subroutines) in the program are:

- (1) main.f – The main program unit taking care of the initial tasks. It reads the input data, sets up the data structure and the initial state, and then starts the MD simulation.
- (2) input.f – Reads the input data.
- (3) setup.f – Sets up data structures, the internal units used by the program, and prepares the initial state of the system.
- (4) mdstep.f – Integrates the equations of motion.
- (5) forces.f – Calculates the forces (including generation of the neighbor lists).
- (6)
  - mpi.f – Subroutines needed in the communication between the nodes during a parallel execution. This is the only module containing calls to MPI library.
  - mpi\_cray.f – Cray version of mpi.f.
  - scalar.f – Sets up calls to dummy subroutines substituting the MPI calls for execution on a single-processor computer.
- (7) restart.f – Dumps or reads the restart file; dumps the trajectory file.
- (8) aver.f – Collects the averages of various physical quantities and reports the final values at the end.
- (9) tcf.f – A collection of subroutines for calculation of time correlation functions.
- (10) service.f – A collection of procedures needed at various stages of molecular dynamics. Contains calculation of temperature, molecular centers of mass, etc.
- (11) util.f – A collection of some other auxiliary procedures

The general organization of the computations in the *M.DynaMix* package is presented in Fig. 1. The program reads the main input file from the standard input (or from the file `md.input` in the case of parallel execution on some machines). This file specifies the studied system and all the simulation parameters. Additionally, files describing each molecular species (`.mol` files) are required. Their format is described in the README file included in the distribution.

During the execution, the program periodically dumps a restart file containing the current configuration of the system and all averages calculated to that moment. The program can be interrupted at any time and the execution can be continued later from the restart file. There are separate restart files to keep the radial distribution functions (RDF) and time correlation functions (TCF), if these quantities are calculated during the MD simulation. It should be stressed that particularly the TCF calculation can become rather memory-demanding and slow down the simulation.

The program produces an output file containing all the basic information and simulation results. The final configuration of the system can be dumped in the so-called *xyz*-format, compatible most of molecular viewers, such as XMOL [17]. During the simulation, trajectories of the atomic coordinates can be dumped. The trajectory files can be analyzed later, using a separate auxiliary analysis program called TRANAL.

### 3.2. Simulation setup

The start configuration of the simulated system can be taken from a specific input file (`.inp`) or generated automatically by the program. In the input file one can alternatively specify positions of all the atoms in the system, or positions of all molecular centers of mass (COM). If COM points are chosen, the atomic positions will be generated from the local atomic coordinates specified in the `.mol` files.

If automatic generation of the start configuration is chosen, the COM positions of the molecules are distributed on an FCC or cubic lattice. There is a possibility to place larger solute molecules at the center of the simulation cell and to distribute other molecules (smaller solvent molecules) outside a sphere or cylinder containing the macromolecule(s).

After generating the initial state, the program checks the configuration for any inconsistencies, reporting the cases possibly producing too large forces (e.g., cases when two atoms are too close to each other or some of

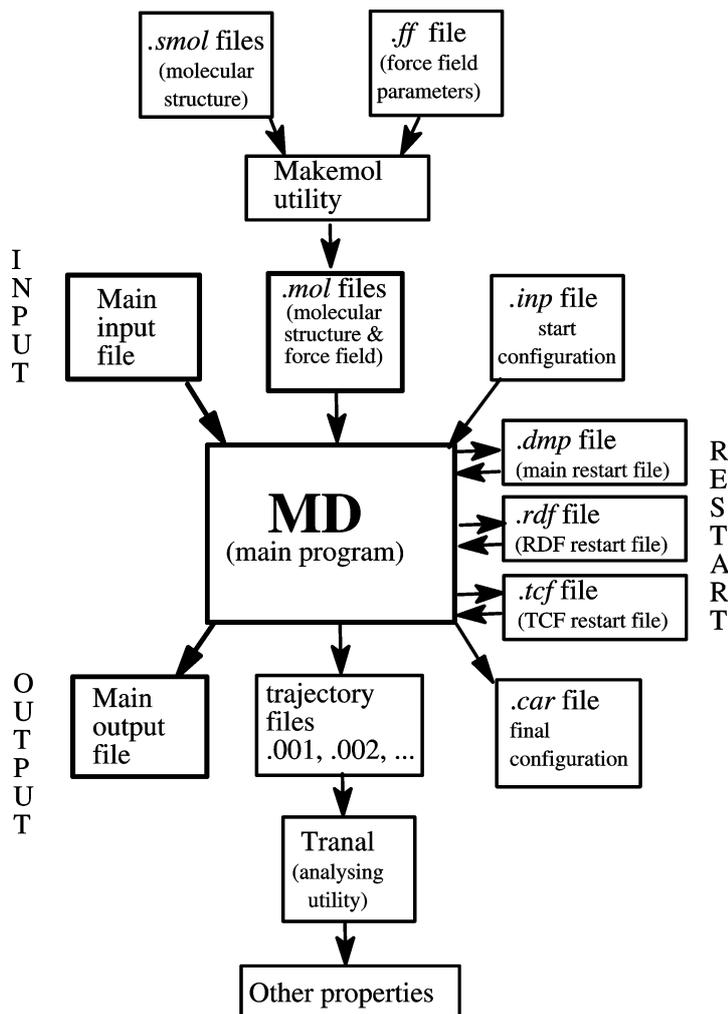


Fig. 1. General organization of computations using the M.DynaMix package.

the bonds are too long). In some cases it is still possible to start the simulation from such a bad configuration by specifying an option to cut all the forces exceeding a given level. Later, when the forces fluctuate at normal amplitudes, this option has no effect, since it only chops off the very high amplitudes.

### 3.3. Force calculations

The most time-consuming part of the molecular dynamics simulations is the calculation of the forces due to non-bonded interactions, which, in principle, requires a double sum over all the atom pairs, making it an  $O(N^2)$  problem. The application of a cut-off radius for the non-bonded interactions allows a considerable reduction of the CPU time for calculation of atom-atom interactions, by effectively setting the interactions between the particles separated by distances larger than the cut-off radius to zero.

To decide whether the forces between a given pair of atoms need be calculated or not can be difficult. In any case, one still should know the distances between all atom pairs, or at least to have a list of atom pairs with distances

less than the cut-off (so-called neighbor-list). In liquids this list must be regularly updated. In some other schemes, like the linked-cell (LC) method [18], the search for neighbors can be limited to the same cell and, because of the symmetry properties of the pairwise forces, only to half of the nearest cells. This leads to an  $O(N)$  algorithm, although a true  $O(N)$  algorithm is achieved only for very large  $N$ . For a more “normal” size of systems ( $10^3$ – $10^4$  particles), a periodic (e.g., each 10 MD steps) update of the neighbor list by looking through all the atom pairs is normally sufficient. Although the CPU time of this block scales as  $O(N^2)$ , the coefficient in front of it is small. For example, in the case of 2000 H<sub>2</sub>O molecules (6000 atoms), this part in our program consumes roughly 5% of the total CPU time.

### 3.4. Parallelization strategy

There are two popular main strategies to parallelize molecular dynamics programs, the “Replicated data” [18] (RD) and the “Domain decomposition” [19] (DD). In the RD method all the nodes keep the positions of all the particles in the system, while the calculation of different contributions to the forces is divided between the nodes and done in parallel. One of the advantages of using the RD method is its relatively simple and efficient distribution of the force calculations over the processors for nearly all types of force fields and molecular structures. An obvious weakness of the RD approach is the relatively high communication cost from collecting the contributions to the total forces and distributing the new particle positions again to all nodes at the end of each time step. However, these two global communication operations are unavoidable in most general parallel MD schemes when simulations are carried out on current distributed computer architectures.

In the DD approach, the particles are distributed between the nodes, with each of the nodes being responsible for the particles in the corresponding subcell. The communication overhead is normally lower, compared to the RD method, while other sources of overhead arise instead. These are for example due to monitoring the particle movement across the subcells, implementing the Ewald summation of electrostatic interactions, applying the Newton’s third law, etc. The LC scheme and also the DD method are most efficient when simulating systems with short-range interactions on massively parallel computers, but for a parallel simulation of complex biomolecular systems on computers with less than 100 processors, the RD method is normally to prefer. The RD scheme is implemented in our parallel MD program.

### 3.5. Parallelization of specific parts of the program

#### 3.5.1. Selecting the neighbors

To calculate the forces between the atoms, we first create a list of neighbors from all atom pairs in the system. For efficient parallelization, the whole list of atomic pairs:  $(I, J = 1, \dots, N, I < J)$  should be evenly divided between the available nodes. The condition  $I < J$  is set to avoid double counting of the interactions. This condition, however, makes it difficult to divide the atomic pairs equally between the nodes. To circumvent this problem, the following scheme has been applied. All particles are divided equally between the processors. This is done in a loop like:

```
DO I=TASKID, N, NUMTASK
```

where TASKID is the node number and NUMTASK the total number of available nodes. For each I, we have an inner loop over J with the following condition: I-J is even for  $I > J$  while J-I is odd for  $I < J$ . This way the atomic pairs will occur uniformly distributed among the nodes, and each of them will only be treated once. Thereafter, lists of the closest neighbors ( $r < 5 \text{ \AA}$ ) and those further away ( $5 < r < R_{\text{cut}}$ ) will be calculated to be stored on each node locally and later recalculated after about 10 long time steps.

#### 3.5.2. Non-bonded interactions

The neighbor lists are used for calculation of the Lennard-Jones and real-space part of the electrostatic contributions to the forces. Because each node treats approximately the same number of atom pairs, good load

balancing can be achieved. In addition, the use of local lists minimizes both the memory demand and the amount of inter-processor communication. The same code for force calculations and integration of the equations of motion is executed on each node, but with different data, taken from the local lists of neighbors. The great benefit of this implementation is that it works well with any number of nodes, even with one single node, making the code equally efficient on any conventional single-processor computer in sequential mode.

### 3.5.3. Ewald sum

Parallelization of the reciprocal part of the Ewald sum is straightforward. This contribution is expressed as a sum over the reciprocal space vectors. Each node calculates the force contributions from its own group of reciprocal vectors (fixed for each node), defined by the following DO-loop:

```

IBEG=NUMTASK-TASKID
DO IKV=IBEG, NKV, NUMTASK

```

where NKV is the total number of reciprocal vectors.

### 3.5.4. Intramolecular interactions

The intramolecular contributions to the forces due to the covalent bonds, covalent angles and torsional angles can be calculated independently for each particular degree of freedom. At the very beginning of the simulation, all the internal degrees of freedom are evenly distributed among the attached nodes. Each node receives a dedicated list of bonds, angles and torsional angles for calculation of the intramolecular contribution to the forces, in the same way as the reciprocal space vectors in the Ewald sum.

### 3.5.5. Summation of the forces

The different contributions to the forces acting on each atom, coming from interactions with other atoms, are first accumulated separately on each node. For example, in the double time-step algorithm, the fast fluctuating forces are accumulated into the arrays HX, HY, HZ, while the slower forces are added into the arrays GX, GY, GZ. When the force calculations are completed, these arrays contain the contributions to the atomic forces calculated on each particular node. To obtain the total force, acting on the given atom, contributions from all the nodes must be gathered and summed up.

The summation of the forces is schematically shown in Fig. 2. All the forces acting on a given atom are transferred to its dedicated node and summed up. The whole operation can be done by a single call of the MPI subroutine MPI\_REDUCE\_SCATTER. Appendix I contains a fragment of the subroutine CM\_ADDLF, taking care of the summation of slow forces. We would like to point out that this operation is faster and uses fewer cycles

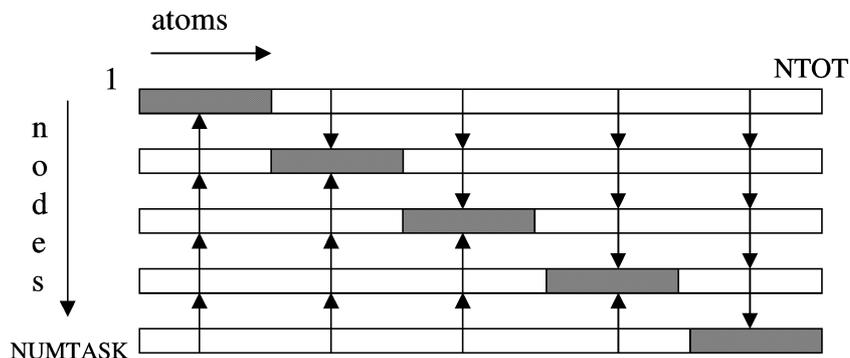


Fig. 2. Summation scheme for the forces. The total force, acting on each atom, is obtained as a sum of forces from each node. The grey areas show the nodes receiving the resulting total force acting on given atoms.

than the perhaps more frequently used `MPI_REDUCE` operation, which sums up the data from all the nodes to one single node. Moreover, using the above scheme, the integration of the equation of motion can be parallelized efficiently without any additional communication cost.

### 3.5.6. *Integrating the equations of motion*

As a result of the `MPI_REDUCE_SCATTER` operation, each node receives the total forces acting on the atoms it is hosting. The equations of motion are now solved locally for new positions and velocities. After completing the entire MD step, the new atomic positions are broadcast to all other nodes to allow the force calculations in the next MD step. We would like to add that, although the integration of the equations of motion consumes only a small fraction of total CPU time, the parallelization of this part has an essential effect on the total speed-up when the number of processors exceeds 16.

### 3.5.7. *Pros and Cons of the current scheme*

The parallelization scheme used in the program is insensitive to the details of the molecular structure of the simulated system. The implementation of the parallel algorithms is designed to work on the atoms and their coordinates only, regardless of their location. This allows us to write relatively simple and general code, suitable for simulation of very different kinds of molecular systems, essentially without any additional data structures and special operations for the parallelization. This gives an optimal scalability with the number of atoms and attached processors. Although the program is parallelized, it runs equally well on single-processor computers without any overhead. No unnecessary operations have to be executed when the program runs in the single-processor mode.

In spite of the simple and general structure of the code, the two global communication operations cannot be avoided. These are the final summation of the forces and the broadcasting of the new coordinates to all nodes. Currently, the overhead due to the communication becomes noticeable when the number of processors reaches 64 when we run our own applications on the parallel computers available to us. In practice, when using national super-computer resources, shared by many other users, it is normally not easy to access more than 16 to 32 nodes anyway, without needing to wait too long (our own experience). This range of number of nodes gives very good scaling for simulations of molecular systems consisting of an order of  $10^4$  atoms.

## 3.6. *Computation of physical quantities*

A variety of physical properties of the molecular system can be calculated directly while the simulation proceeds. Among these are the potential energy and different contributions contributing to it: long-range electrostatic interactions and short-range Lennard-Jones energies, both divided between the different components of the simulated system. Besides the intermolecular energies, all the internal energies are also calculated, including the average values of the bond lengths and angles. The pressure is calculated both from the atomic and molecular virials. In addition, the pressure can be determined along each coordinate axis, an important piece of information in anisotropic and ordered systems. Intermediate average values of all these quantities are calculated during specified time intervals and saved in the restart file. When the simulation is finished, all the final averages are calculated. The statistical errors of the final average values are determined from the variance of the intermediate averages.

Other properties, possible to calculate during the simulations, are the radial distribution functions (RDF) and time correlation functions (TCF). RDFs can be calculated for all pairs or for a selected set of different atom pairs. TCFs are calculated for each molecule type. Twelve types of TCF may be calculated: COM velocity of the molecules, angular velocity, dipole moment (if any) and its second Legendre polynomial, reorientational TCF defined by an arbitrary vector attached to the molecule and its second Legendre polynomial. The attached vectors of interest are normally those along some covalent bond, in-plane or out-of-plane of a planar molecule, etc. Finally, both the velocity and angular velocity TCFs can be calculated in the principal axis system of the molecule or in the laboratory coordinate system.

Calculation of some of physical quantities during the simulation run may require additional memory, or they may slow down the execution, especially if the program is running on a large number of processors. This is particularly true for the TCF calculations, but also for the RDF calculations as well as the average values and energies of covalent bonds, angles and torsional angles. The on-flight calculations of these properties can be turned off, and all the physical properties of the simulated system can be calculated afterwards while analyzing the trajectory files.

## 4. How to use the program

### 4.1. Compilation of the program

Before compiling the program, users may need to change the size of some of the working arrays, defined in the `dimpar.h` include file. This file also contains information about the purpose of the parameters and when one has to change them. Several pre-determined copies of this include file are available in the distribution, designed to run systems of different sizes and types.

The parameters defining array boundaries also define the total amount of required RAM memory. One of the most memory-demanding arrays is the one holding the neighbor lists containing the non-bonded atomic pairs within the cut-off distance. Its size is defined by the product of the parameters `NTOT` (maximum number of atoms) and `NBLMX` (maximum number of neighbors for each atom within the cut-off radius). Since the list of neighbors is distributed among the available nodes, this parameter can be decreased and optimized in the case of a parallel execution.

Yet another memory-consuming parameter is `MAXCF`, defining the time interval (number of time points) for the TCF calculations. This parameter can be set to 1 if no TCF calculations will be carried out during the simulations.

The array containing the intermediate averages of different physical quantities may also require a considerable amount of memory. Its size is `NRQS * LHIST`, where `NRQS` is the maximum number of different averages to be calculated and `LHIST` is the maximum number of time intervals to calculate the intermediate averages. This array may grow very big for large macromolecules, because of all the bond lengths, covalent and torsional angles are calculated and stored in this array. Accumulation of these quantities can be switched off by specifying an additional parameter, `noaver` (see files `MD.input` and `Extra_param` in the package distribution).

Each time the program is started it checks the allocated array boundaries based on the information it obtains from the input data, and it stops at every inconsistency, trying to tell how to fix the problem. If this happens, the corresponding parameters must be corrected in the `dimpar.h` file, and the program has to be recompiled.

Included in the program distribution are several Makefiles for different common computer architectures. If none is directly applicable, one should choose the most suitable one and edit it if necessary. Because the code uses only standard Fortran 77 statements, without any external libraries (except the MPI library for parallel execution), the only thing to do in most cases is to choose a proper compiler name and optimization parameters from the compiler manual. Sometimes problems may arise due to calling the CPU time counter (file `cpu_time.f`). Because accessing the CPU time is not a standard function in Fortran, this file may need some editing depending on the computer and compiler. The CPU time counter can always be turned off just by commenting out all the executable lines inside this particular file.

Another thing which may need adjusting is the linking of the MPI library. On most parallel computers the script `mpif77` should do the proper work. We also would like to mention that for the parallel Linux and Cray versions, a modified version of the `input.f` is used (generated after applying the file `input.patch`). This modified version then reads the program input from the file `md.input` rather than from the standard input as the normal version does. This is because some problems are encountered while reading the standard input on these particular parallel architectures. The name of the resulting executable file is `md` for single-processor computers and `mdp` for parallel computers.

#### 4.2. Preparation of the force field files

For each molecule type a specific topology file must be created, describing molecular structure and parameters of the force field. The file must have the extension .mol. The format of the .mol files is described in the documentation file README. The .mol files contain information about relative coordinates of all atoms,  $q$ ,  $\sigma$  and  $\epsilon$  parameters for all atoms, a full list of covalent bonds, angles and torsional angles with the corresponding force field parameters, and some additional optional force field parameters. Some examples of .mol files are given in the *moldb* directory. These files can be prepared or edited manually, which also provides good flexibility in changing force field parameters. For example, the Urey–Bradley term for covalent angles (which is included in the CHARMM force field) may be introduced as an additional bond, connecting the first and third atoms defining the covalent angle.

For large molecules, the procedure of writing .mol files may become rather tedious. The utility *makemol* is written to simplify this process. This utility creates a .mol file from two separate input files. One is a “simple molecular file” (.smol), containing only the atomic coordinates, corresponding atomic charges, the chemical atomic type and a list of bonds. The second one contains the force field parameters for the different chemical atomic types. This utility program goes through the list of atoms in the .smol file and finds first the Lennard-Jones parameters for each atom, and then it reads the bond list and inserts the parameters for covalent bonds. Finally, it generates lists of covalent angles and torsional angles from the bond list and finds the correct force field parameters for them, creating the final .mol file.

#### 4.3. Preparation of the input file

A sample input file, MD.input, comes with the distribution (see Appendix II). This file is extensively documented, and can be even used as a short manual for preparing the input file. Below are given a few more detailed explanations to some of the parameters.

The composition of the system is defined by the number of molecule types, the names of molecules of each type and the number of molecules of each type. The internal molecular structure and the force field are determined in the .mol files read separately by the MD program. Only a few additional parameters for adjusting some features in the force field can be specified through the input file, for example, the scaling of the “1–4” intramolecular interactions.

The type of the molecular dynamics algorithm is determined by the flags “Constant temperature”, “Constant pressure”, and “Constrained dynamics”. If the last of these parameters is set to .true., then the simple leap-frog algorithm with single time-step and the constrained dynamics will be applied. The constraints are applied on all covalent bonds specified in the .mol files. If the parameter “Constrained dynamics” is .false., then the double time-step algorithm will be applied. This algorithm can be converted into single time-step (Leap-frog) by setting the ratio between short and long time steps equal to one.

One of the features requiring some special care in setting up a simulation is specifying the parameters for Ewald summation of the electrostatic interactions. Normally, three parameters affect these calculations: the two cut-off radii in the real and in reciprocal spaces, and the convergence parameter (denoted as  $\alpha$ ) regulating the convergence of both the real and reciprocal parts of the Ewald sum. In the program, these parameters are set in the following way. First, the cut-off radius in real space,  $R_{\text{cut}}$ , has to be specified explicitly. Then, the convergence parameter  $\alpha$  is specified as the product  $\alpha R_{\text{cut}}$ . Note that this product entirely determines the precision of the real-space part,  $\text{erfc}(\alpha R_{\text{cut}})$ , which must be small enough. The third parameter,  $\text{FEXP} = (k_{\text{max}}\pi/\alpha L_{\text{box}})^2$ , determines the value of the cut-off in reciprocal space,  $k_{\text{max}}$ , and thereby the precision of the reciprocal part of the Ewald sum, which is equal to  $\exp[-\text{FEXP}]$ . Reasonable precision of the calculated total electrostatic energy is achieved at  $\alpha R_{\text{cut}} = 3$  and  $\text{FEXP} = 9$ . If the required precision is fixed both for the real and reciprocal parts of the Ewald sum, the only remaining parameter to vary is the cut-off radius of the real-space part of the Ewald sum. A small value for this parameter reduces the computation time for the real-space term but increases it for the reciprocal-space term, and *vice versa*. An optimal value of the cut-off radius may be set empirically from the condition that the CPU time for the real-space calculations is roughly equal that for the reciprocal-space calculations. The execution time of

different blocks in the program can be seen in the output. The optimal value of  $R_{\text{cut}}$  increases slowly as  $L^{1/6}$  with the box size. We recommend increasing  $R_{\text{cut}}$  gradually from 13 Å for the box length 30 Å, to 15 Å for the box length 60 Å (see details on optimization of the Ewald sum parameters in Ref. [16]).

There are options in the program to completely freeze the positions of the atoms of some molecules, or just to tie them loosely by placing some of the atoms in the molecules in harmonic potentials with a given effective radius. These features may be useful at various stages while simulating macromolecules.

Normally, when restarting the program, it reads the temperature and box size values from the restart file and simply ignores the values specified in the input file. However, if the flags in the input file, “Change the temperature” or “Change the density” are set to .true., the corresponding properties will be changed. The temperature will be changed by scaling all the velocities, and the density (i.e., the box sizes) by scaling the distances between molecular COMs uniformly.

Some of parameters are not specified in the input file by default. They can be introduced, if needed, after the line “Extra parameters”. The line “*add < n >*” says that *< n >* additional parameters follow. These parameters are introduced by a keyword followed by the parameter value(s). A detailed description of them is given in the file ExtraParam, included in the distribution. They describe additional features introduced in the latest version of the package. Additional new features, which will eventually appear in future releases of the program, will be introduced in the same way. Such a scheme allows us to retain the same format of the input file while upgrading the program and introducing new possibilities. Among a few additional parameters we wish to mention flags signaling ASCII format of the restart file (default is a binary format) and an option to abolish the calculation of average bond lengths and angles with corresponding energies. The first one is useful when the simulation is continued on another (not binary compatible) computer, while the second is used just to save memory.

Possible format errors in all input files are detected by the program and signaled to the user by specifying the file and the line number where the error is encountered.

#### 4.4. Start-up hints

Preparing a start configuration for the molecular system may sometimes be a bit tricky. This is usually not a problem for systems consisting of small molecules, in which case it is sufficient just to specify in the input file “start from an FCC lattice” at a given temperature and density. However, if large molecules, especially polymers, are included, it may sometimes become very difficult to prepare initial configurations so that any two atoms are not too close to each other, which can cause very large forces, enough to disintegrate the whole system (sometimes these would be enough to blow the whole of mother Earth to pieces). There are several possibilities to deal with this problem, either alone or combined.

- (1) An option available is to place any large solute (fragment of DNA, protein, etc.) in a cylindrical or spherical cavity of a specified radius at the center of the simulation cell. The solvent, normally consisting of much smaller molecules can then be distributed on a lattice around the cavity containing the large solute.
- (2) One can always start the simulation at a very low density (large box size), which essentially corresponds to gas phase conditions, thereby reducing the probability of overlapping atoms during the automatic generation of the initial configuration. Then, in sequences of short runs (100 steps per run is normally enough) the density can gradually be increased to a more normal level. The molecules have time to adjust themselves during this procedure.
- (3) The parameter “Cut the large forces” may be set to .true.. In this case, any total force acting on any atom growing above a specified level is simply cut to this level while maintaining its direction.
- (4) The initial coordinates of all the atoms, or alternatively all the molecular COMs can be given in a separate input file (with extension .inp), if a reasonable initial configuration is known from other sources. This file contains the Cartesian  $x$ -,  $y$ - and  $z$ -coordinates of each atom, one line per atom (or molecular COM). Lines, beginning with “#” can be given in the coordinate file as comments for documentational purposes.

Moreover, it is recommended that the simulation is always started at constant volume (parameter “constant pressure” set to `.false.`), because a non-equilibrium initial configuration may correspond to enormous pressures. Sometimes the Nosé constant-temperature algorithm should be turned off at start-up (set “constant-temperature” to `.false.`). In the latter case, the temperature may be regulated by simple velocity scaling. If the actual temperature deviates from the target value by more than some specified difference, the velocities of all the atoms are scaled to provide the proper temperature.

#### 4.5. Running the program

The program can be started from the following command line:

```
md < md.in > md.out
```

where `md.in` is the primary standard input file (see Section 4.3) and `md.out` is the program’s standard output file.

In the case of parallel execution on computers running Linux, or on a Cray, the input is always taken from the file `md.input`, and should not be specified in the command line.

The program starts, e.g.:

```
mpirun - np 2 mdp > md.out
```

This command above is used to start a job on a double-processor Linux computer. The exact form of the command to start a parallel job may differ on different architectures.

During the execution, the program regularly dumps a restart file after a specified number of steps. The program may be interrupted at any time and then restarted from the restart file without losing any calculated information. The program may also be started in the “check only” mode. In that case it only reads the restart file, calculates the final averages from the intermediate averages and reports the results, without running a simulation. The moment (time origin) when the final averaging is started can be changed in this run. In this way, the *equilibration* part and the *production* part can be specified *after* the simulation.

The program can also be set to dump the trajectories of all the atoms in the system, or just of those belonging to molecular types specified in the input file. The trajectory may be dumped either in a binary or in an ASCII (XMOL [17] compatible) format. The whole trajectory is stored in separate files with the extensions sequentially numbered: `.001`, `.002`, `.003`, ... The parameters defining how often configurations are dumped, and the number of configurations in each trajectory file, can be specified in the input file. The dumped trajectories may be used later to analyze results after the simulation is finished. The format of the binary trajectory file can be seen from subroutine TRACE in the `restart.f` file.

#### 4.6. Post-analysis of the trajectories

The module TRANAL can be used to extract and calculate additional properties of the simulated system calculated from the trajectory files. The main block of the module reads the trajectory files in binary format, and then calls different subroutines, performing calculations of different properties. At present, the following properties can be calculated from an analysis of the trajectory files:

- (1) Spatial distribution functions of an atom density in a local coordinate system attached to a given molecule. Output of SDF density may later be visualized with the GOpenMol package [20].
- (2) Three-body correlations (more exactly, density distribution of a third particle around two particles separated by a specified fixed distance).
- (3) Radial distribution functions.
- (4) Time dependence of the root-mean-square displacement (RMSD) of molecules and the self-diffusion coefficients.
- (5) Residence (life) times of specified atoms to be within a certain distance around another atom.

(6) Population distribution of specified torsion angles.

Users can easily add their own subroutines to calculate other properties, and put them into the main block of the TRANAL module, after the place where the atomic coordinates are read.

The TRANAL module is not included in the distribution but can be obtained from the authors on request.

## 5. Benchmarks and testing

The M.DynaMix package has been used by our group for several years as the main computer simulation package. It has been used in simulations of very different molecular systems. To give some idea of its applicability we may mention studies of ionic water solutions [21], double-helix DNA fragment surrounded by water and ions [22], nucleotides and nucleoside salts in solution [25], dicarboxylic acids in solution [24], *t*-butyl alcohol in aqueous solution [26], disaccharides in water and water–DMSO mixture [23].

Some benchmark results concerning the program performance are given in Table 1. The simulated systems are:

- (1) A periodic fragment of double helix B-DNA in ionic aqueous solution: DNA (635 atoms), 1050 waters and 40 sodium and 20 chloride ions (3845 atoms totally) [22].
- (2) An NaCl ion solution: 20 Na<sup>+</sup>, Cl<sup>-</sup> ion pairs and 1960 water molecules, 5920 atoms total [21].
- (3) A lipid bilayer. Two similar systems differing only in the size: (a) 64 DPPC lipid molecules, 50 atoms each (united atom GROMOS force field), and 1472 waters, 7616 atoms in total and (b) a 4 times larger fragment with 256 DPPC phospholipids and 5888 water molecules, 30464 atoms totally.

These molecular systems are very different in character and composition:

- (1) a large macromolecule surrounded by the solvent,
- (2) a homogeneous solution of small molecules,
- (3) an anisotropic ordered system consisting of a large number of larger molecules in water as solvent.

The simulations were performed on an IBM SP2 parallel computer with 150 MHz RISC 6000 processors (at the Royal Institute of Technology, Stockholm) and on the departmental Linux PC cluster of Beowulf type built from fourteen 400 MHz Pentium II processors.

The speed-up of the program depending on the number of processors is displayed in Fig. 3. The program shows excellent scaling properties for all three molecular systems running on on up to 32 processors. On our PC cluster the speed-up is somewhat worse because of the lower communication speed offered by the standard 100 Mbits Ethernet cards.

Table 1

CPU time (in min) for 1 ps simulation of 3 different systems on IBM SP2 depending on number of processors. Box sizes and cut-off distances  $R_{\text{cut}}$  are given in Å. Double time step 0.2/2 ps was used for systems 1 and 2; for system 3 constrained dynamics (SHAKE algorithm) with time step 2 fs was applied

System	Num. atoms	Box sizes	$R_{\text{cut}}$	Nodes				
				1	4	8	16	32
<i>IBM SP2</i>								
1. DNA	3825	33×33×34	13	60	16	8.5	4.5	2.8
2. Ion solution	5920	39×39×39	14	110	29	14	8.5	5
3. lipid bilayer	7616	44×44×64	14.5	150	38	22	12	6.5
<i>Pentium II PC cluster</i>								
2. Ion solution	5920	39×39×39	14	97	28	18		
3a. lipid bilayer	7616	43×43×68	14.5	124	35	23		
3b. “	30464	86×86×68	16	805	210	115		

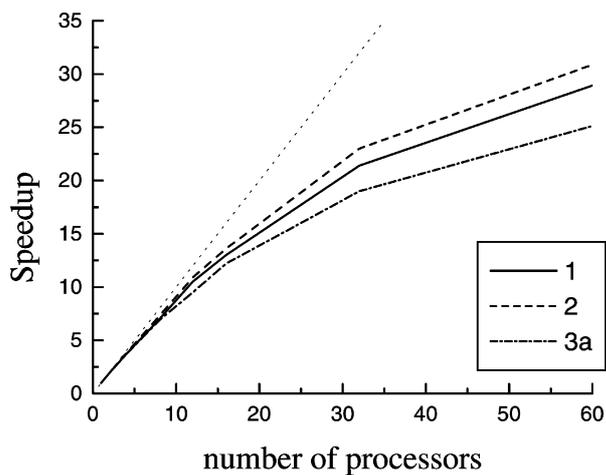


Fig. 3. Effective speed-up of computations on different number of processors of IBM SP2 compared to single-processor computations. For details on the simulated system, see the text.

### Acknowledgements

This work has been supported by the Swedish Council for Natural Sciences (NFR) and by a grant from the Swedish Strategic Foundation (SSF). The Center for Parallel Computers (PDC) in Stockholm, The High Performance Computer Centre North (HPC2N) in Umeå and the National Supercomputer Centre (NSC) in Linköping are gratefully thanked for generous allocations of computing time.

**APPENDIX I**

A fragment of the parallel code performing the force summation.

```

*
* 5.1 Sum up forces
* -----
C
C NAB(J) is the first atom for which node J is responsible
C NAE(J) is the last atom for which node J is responsible
C NABS(J) = NAB(J) - 1
C NAP(J) = NAE(J) - NABS(J) is the number of atoms for node J
C (these are set in module UNITS, file setup.f )
C
      DO J = 0, NUMTASK-1
        I0 = 3*NABS(J)
        I1 = I0+NAP(J)
        I2 = I1+NAP(J)
C Put X,Y,Z projections into a single buffer
        DO K = 1,NAP(J)
          I = NABS(J)+K
          BUFF(I0+K) = GX(I)
          BUFF(I1+K) = GY(I)
          BUFF(I2+K) = GZ(I)
        END DO! of K
      END DO! of J
C
C Perform global summation
C
      call MPI_REDUCE_SCATTER(BUFF,BUF2,NAP3,MPI_REAL,
+MPI_SUM,MPI_COMM_WORLD,ierr)
C Return results from buffer BUF2 to forces array GX
      DO J = NAB(TASKID),NAE(TASKID)
        I = J-NABS(TASKID)
        GX(J) = BUF2(I)
        GY(J) = BUF2(I+NAP(TASKID))
        GZ(J) = BUF2(I+NAP(TASKID)*2)
C Contribution to molecular virial coef.
        M = NNUM(J)
        WIRS = WIRS-GX(J)*(SX(J)-X(M))
+                                     -GY(J)*(SY(J)-Y(M))
+                                     -GZ(J)*(SZ(J)-Z(M))
      END DO! of J

```

**APPENDIX II**

## A sample standard input file

```

# Sample input file for the MD program, v. 4.3
#
# Simulated system consists of 246 flexible SPC water molecules,
# and 5 Na+ Cl- ion pairs
#
# Lines beginning with "#" are comments used for documentation
# Note, the comments are given before the corresponding parameters.
#
# This file can be used as an input (see directory sample for
# short version of the input file)
#
#
#           Output control parameter:
# Suitable values 2-10. The less number, the less you see of the output.
# Parameters higher than 7 are used mostly for debug purposes
5
#
#           Base file name for output files:
# Other files requested or created by the program will have this name with
# various extensions
5nacl
#
#           Path to the molecular database:
# This directory contains the *.mol files which describe the molecular
# structure and the force field
./moldb
#
# The program creates and updates periodically a restart file containing
# the current configuration of the system and the calculated averages.
# The program can be interrupted any time and continued later from the
# restart file without losing any information.
# If the "Check only" parameter is true, the program does not start a
# simulation run. If it is a new run, the program only checks the input.
# If it is continuation of the old run, the program gives the so far
# accumulated results.
#
# Read from           Dump           Check           Zero counter
# the restart file?  a restart file?  only?           of cpu time?
#           .f.           .t.           .f.           .t.
#
# The type of statistical MD ensemble. "Anisotropic NPT" means
# a separate pressure/volume control in each direction. Use this
# option only if the system is really anisotropic (a fragment of
# DNA, membrane, liquid crystal, etc). Note, that for "constant
# pressure"=.t. also the "constant temperature"
# must be .true. (God knows what an NPE ensemble is !!!).
# Constant temperature?  Constant pressure?  Anisotropic NPT?
#           .t.           .f.           .f.

```

```
#
# Number of molecular types:
#       3
#
# Types of molecules (models)
# The database directory (specified above) should contain
# the following files:
# H2O.mol, Na+.mol, Cl-.mol
# See the README file about the format of the .mol files
#       H2O       Na+       Cl-
#
# Specify for each molecular type:
# The number of molecules of each type:
#       246       5       5
#
#       Non-bonded intramolecular interactions:
# Calculate the intramolecular potential
# (LJ and electrostatic)
# for non-bonded atoms, separated by more than 3
# covalent bonds. Normally, it should be .true.
# Although unimportant for small molecules)
#       .t.       .t.       .t.
#
#       1 - 4 intramolecular interactions:
# Calculate the L-J and electrostatic terms for 1-4 intramolecular
# (i.e. separated by 3 covalent bonds) interactions
#       .f.       .f.       .f.
# Scaling factors for 1-4 L-J interactions
# These are 0 in AMBER, 1 in CHARMM and 0.25 in GROMOS
# Not important for small molecules
#       1.       1.       1.
# Scaling factor for 1-4 electrostatic interactions
#       1.       1.       1.
#
#       Intramolecular potential type
# In this case it should be 0 for all molecules except the water.
# For water 1 is "harmonic" and 2 "anharmonic" flexible SPC water
# (It tells the program to use a special subroutine to handle anharmonic
# bond stretching potentials)
#       2       0       0
#
#       Rules for an initial box size / density:
# - If one of the box side lengths is zero,
# the actual box size (cubic box)
# will be defined from the density.
# - If the density is also zero, the program runs a"vacuum
# simulation". Set the Ewald parameters (below) to 0 in the case of
# a vacuum simulation.
# - If all the three box sizes are not zero,
# they will define initial box size
# and shape, and so the actual density
#
```

```

#           Cell type:
# 0 - rectangular
# 1 - truncated octahedron: cube of side BOXL centred in 0,0,0 with
#   truncated corners:       $|x|+|y|+|z| < 0.75*BOXL$ 
# 2 - hexagonal along Z axis
#
# temperature ( K)      density (g/cm**3)      pressure (atm)
#   298.                1.03                  1.
#
#           box size (\AA)      cell type
#           0.   0.   0.                0
#
#           This is the long time step
#           Time step (s)      Small steps in one long time step
#           2.d-15                10
#
# Total (long)  Steps for interme-      take averages      dump restart file
# MD-steps     diate averaging          each .. steps      after .. steps
# 10000        1000                    1                  500
#
# Nose thermostat parameters:           Meaningful in constant-energy
#                                       simulations (if const. temp. = .f.):
# Thermostat (fs) Barostat (fs)        Simple velocity      delta T (K)
#                                       scaling?
#   30.            700.                  .f.                  20.
#
# Rcutoff sets the cut-off radius for the L-J and Real-space
# electrostatic forces
# Interactions inside the Rcut-fast are
# recalculated each short time step,
# Interactions between Rcut-fast and
# Rcutoff each long time step
#
# Rcutoff(\AA)      Rcut-fast forces      check neighbors after .. steps
#   10.              5.                  10
#=====
# Treatment of electrostatic interactions:
#
#           Ewald parameters:
# alpha and fexp are set from the following conditions:
# erfc(alpha*R) = required precision of the real-space Ewald
#   exp(-fexp) = required precision of reciprocal-space Ewald
# A rule of thumb:
#   alpha*R          fexp (m/s**2)
#   3.14159256      9.81
#
# If alpha*R above is negative -
# the reaction field method is used with
# -alpha*R as dielectric permittivity,
# fexp is the Debye screening length in \AA
# (setting the Debye length to 0 means infinite
# Debye length,i.e non-conducting solution)
# If alpha is set to zero (exactly: between -1 and 0),
# no special treatment of

```

```

# the electrostatic interactions (simple spherical cut-off)
#
# Be careful when playing with these parameters
# and try to understand what you want to do.
# Unreasonable values will result in a funny
# behavior of the system or too long computation time.
#
#=====
# Which of the molecules move:
# .f., means the molecules are fixed, but still
# interact with other molecules.
# .t.      .t.      .t.
#
# recalculate the list of intramolecular interactions?
# can be set to .f., if you have a large molecule with
# a stable conformation
#
# .t.
#
# If true, constrained dynamics with the SHAKE
# algorithm for specified molecular species will be used.
# It will keep all the bond lengths constant.
# No double time step algorithm will be used in this case
# which molecules considered as rigid (1/0)
#
# Constrained dynamics?  tolerance param  which molecule types are
#                        (not in effect  constrained?
#                        if false)      (1 - constrained; 0 - flexible)
# .f.                    1.d-4         1      1      1
#
# Initial state (from -1 to 4)
# Values:
# -1 take the initial center-of-mass coordinates of the
#     molecules from the *.inp file
# 0  take the initial atomic coordinates
#     from the *.inp file
# /* The .inp file should be written in a free
#     x y z format (one atom per line)
#     The order of atoms and molecules:
#     molecular type1      molecular type 2
#     mol1    mol2    mol3      mol1    mol2    mol3
#     at1 at2 at1 at2 at1 at2      at1 at2 at1 at2    ...  */
# 1 Start from an FCC lattice
# 2 Set a cylindrical hole along the z-axis.
#     Place the molecules marked ``1''
#     on the line below into the hole according initial
#     coordinates in .mol file, and distribute others out the hole
# 3 The same but for a spherical cavity
# 4 Start from a cubic lattice
#
# | set velocities to 0?
# | (if .false., Maxwell distribution at
# | startup or as it is at restart)

```

```

1 .f.
#
# Parameters for above in cases 2, 3 (specify 0 or 1)
#   0   0   0
#
#           If this parameter is set to .t., the atoms
#           described in the file "fixed.atoms" will be
#           put in a harmonic potential of this radius:
# Radius of a hole           Keep specified   Allowed   File
# for large molecules       configuration? deviation? name
#   10.                      .f.             3.       fixed.atoms
#
# If you want to change the temperature or density after the restart.
# Change temperature at restart?           Change density?
#   .f.                                   .f.
#
# When to start the final averaging
# Final averaging after ..                 Dump
# intermediate averagings                 XMOL config. file?
#   6                                     .f.
#
# This only makes sense if you have to start
# from a very bad configuration
# If the total force acting on an atom
# exceeds some high level defined by the given
# level, the force ( in the internal units) will be cut to this level
# Cut large forces?                       level (this is a reasonable value)
#   .f.                                   1.d-4
#
# RDFs have a separate restart file with an extension .rdf
# Calculate RDF?           Dump RDF restart file?           Read restart RDF file?
#   .t.                   .t.                             .f.
#
# RDF for all sites?
# (if .f., they should
# be specified below)           Cutoff-RDF(\AA)           Resolution of RDFs
#   .f.                         10.                             200
#
# dump trajectories (0/1/2)?           number of config. in a trajectory
# 0 - no trajectory                   file. Trajectory files have extensions
# 1 - unformatted files                 .001, .002, ...
# 2 - "XYZ" format                       interval(s)
#   0                                   1.d-14           500
# dump trajectory of molecules (1/0):
#   1   1   1
#
# TCF calculations
# -----
# Attention! Tcf calculations are currently
# NOT parallelized and may slow down parallel
# simulations quite considerably.
#

```

```

# TCFs have a separate restart file with extension .tcf
# NSTEG is number of points for calculation TCF
# JUMP - number of MD steps between the points for calculation of tcf.
# calculate tcf?  restart tcf?  dump tsf?  NSTEG  JUMP
#   .f.           .f.           .f.       200    5
#
# which of 12 tcf calculate (0/1/2):
# 12 types:
# 1 - velocity autocorrelations
# 2 - angular velocity autocorrelations
# 3 - 1 order Legendre polynom for dipole moment
# 4 - 2 order Legendre polynom for dipole moment
# 5 - 1 order ... tcf defined by vector below
# 6 - 2 order ... tcf defined by vector below
# 7 - X principal component of the velocity TCF
# 8 - Y principal component of the velocity TCF
# 9 - Z principal component of the velocity TCF
# 10 - X principal component of the angular velocity TCF
# 11 - Y principal component of the angular velocity TCF
# 12 - Z principal component of the angular velocity TCF
# if "2" specified for tcf 7-9 or 10-12, then tcf projections are
# calculated in molecular principal coordinate system; otherwise
# they calculated in laboratory coordinate system
#   1 1 1 1 1 1 2 2 2 0 0 0
#
#   unit vectors for the reorientational tcf
#   this vector is defined by 2 selected atoms on each molecule
#   1   1   1
#   2   1   1
#
# Other optional parameters
# if line "add <n>" is omitted, no optional parameters
# Description of optional parameters is given in file Extra_param
#
add      0
# The total number of RDFs to be calculated.
10
# RDFs are defined by their "global" site numbers
# First molecule, type H2O has 3 sites with site numbers 1, 2, 3
# Second molecule Na has one site with number 4
# and third molecule Cl has one site with number 5
#   0 - 0
# This means RDF between sites 1 and 1, i.e. 0 atoms on H2O molecule
#   1       1
#   0 - H
# symbol & followed by a number means that these RDF will be averaged
# (in this case: two hydrogens atoms (atoms 2 and 3 in H2O molecule)
# are equivalent)
&2
1 2
1 3
#   0 - Na

```

```
# atoms of the next molecular type (Na) have site number 4, and Cl is 5.
1 4
# O - Cl
1 5
# H - H
&3
2 2
2 3
3 3
# H - Na
&2
2 4
3 4
# H - Cl
&2
2 5
3 5
# Na - Na
4 4
# Na - Cl
4 5
# Cl - Cl
5 5
```

## References

- [1] S.J. Weiner, P.A. Kollman, D.T. Ngyuen, D.A. Case, *J. Comput. Chem.* 7 (1986) 230.
- [2] A.D. MacKerell, J. Wiorkiewicz-Kuczera, M. Karplus, *J. Amer. Chem. Soc.* 117 (1995) 11946.
- [3] W.F. van Gunsteren, H.J.C. Berendsen, *Groningen Molecular Simulation (GROMOS), Library Manual* (Groningen, 1987).
- [4] K. Toukan, A. Rahman, *Phys. Rev. B* 31 (1985) 2643.
- [5] S. Nosé, *Mol. Phys.* 52 (1984) 255.
- [6] G.J. Martyna, D.J. Tobias, M.L. Klein, *J. Chem. Phys.* 101 (1994) 4177.
- [7] M. Tuckerman, B.J. Berne, *J. Chem. Phys.* 97 (1992) 1990.
- [8] G.J. Martyna, M.E. Tuckerman, D.J. Tobias, M.L. Klein, *Mol. Phys.* 87 (1996) 1117.
- [9] J.P. Ryckaert, G. Ciccotti, H.J.C. Berendsen, *J. Comput. Phys.* 23 (1977) 327.
- [10] M.P. Allen, D.J. Tildesley, *Computer Simulation of Liquids* (Clarendon, Oxford, 1987).
- [11] B. Smit, D. Frenkel, *Understanding Molecular Simulation* (Academic Press, San Diego, 1996).
- [12] N.L. Allinger, Y.H. Yuh, J.-H. Lii, *J. Amer. Chem. Soc.* 111 (1989) 8551.
- [13] P. Ryckaert, A. Bellemans, *Chem. Phys. Lett.* 30 (1985) 123.
- [14] I.G. Tironi, R. Sperb, P.E. Smith, W.F. van Gunsteren, *J. Chem. Phys.* 102 (1995) 5451.
- [15] M. Tuckerman, B.J. Berne, G.J. Martyna, *J. Chem. Phys.* 97 (1992) 1990.
- [16] D. Fincham, *Molecular Simulations* 13 (1994) 1.
- [17] XMOl, v.1.3.1 (Research Equipment Inc., Minnesota Supercomputer Center, Inc.).
- [18] W. Smith, *Comput. Phys. Commun.* 62 (1991) 229.
- [19] D. Fincham, *Mol. Simul.* 1 (1987) 1.
- [20] <http://laaksonen.csc.fi/gopenmol/gopenmol.html>.
- [21] A.P. Lyubartsev, A. Laaksonen, *J. Phys. Chem.* 100 (1996) 16410.
- [22] A.P. Lyubartsev, A. Laaksonen, *J. Biomol. Struct. Dyn.* 16 (1998) 579.
- [23] A. Vishnyakov, G. Widmalm, J. Kowalewski, A. Laaksonen, *J. Amer. Chem. Soc.* 121 (1999) 5403.
- [24] J.A. Nilsson, A. Laaksonen, L.A. Eriksson, *J. Chem. Phys.* 109 (1998) 2403.
- [25] K. Kulinska, T. Kulinski, J. Stawinski, A. Laaksonen, *J. Biomol., Struct. Dyn.* 15 (1998) 987.
- [26] P.G. Kusalik, A. Lyubartsev, D. Bergman, A. Laaksonen, submitted (1999).