Universität Dortmund
Physikalische Chemie IIa
Otto-Hahn Str. 6
D–44221 Dortmund, FRG

User's Guide and Manual

MOSCITO 4

Performing Molecular Dynamics Simulations

Dietmar Paschek and Alfons Geiger

April 7, 2003

Contents

1 Introduction

1.1 Preliminaries

We hope that we don't have to convince you that molecular dynamics simulation is a powerful technique and that you are quite familiar with its foundations and limitations and willing to run a simulation. The books by Allen and Tildesley [1] and Frenkel and Smit [2] are recommended as comprehensive introductionary texts[1]. An extensive overview over the field of molecular modelling is given by Leach [4].

The *MOSCITO 4* simulation program is designed to perform molecular dynamics simulations of rigid and/or flexible molecules using classical molecular mechanics force–fields. *MOSCITO 4* could perhaps stand for 'MOlecular Simulation of Charged InteracTing Objects' (If you have a better idea aboout the *MOSCITO 4* akronym, please send us a note). This shall emphasise in some way that the long ranged Coulomb interactions are treated in a rather sophisticated way, which is rigorously correct for periodic systems. An important feature of *MOSCITO 4* is the employed flexible force field approach (in a subsequent chapter the definition will be given in detail), which allows the treatment of a wide variety of different problems in the field of molecular physics.

MOSCITO 4 runs efficiently on rather small computers like stand-alone PC's (The authors favour strongly the use of the Linux operating system) as well as on large "Supercomputers" like the IBM SP2. So, the provided methods allow an efficient treatment of a wide spectrum of MD–simulation tasks.

To achieve maximum flexibility, the aspect of force–field parameterisation has been separated completely from the simulation kernel. This concept was favoured, since the interaction potential is the most crucial and important input to a simulation. Its quality depends strongly on the care which has been spent on its development. Some serious thoughts about the potential parameterisation should thus be in front of any simulation. There are certainly good and highly reliable general purpose force-fields available, such as AMBER, OPLS, CHARMM and GROMOS, but one should be able to check easily whether they give an appropriate description of your system or not. Additionally one should be able to fine–tune Potential–parameters by systematically improving specific interactions, such as torsional potentials etc. *MOSCITO 4* gives you the opportunity to do so. Moreover, the current version of *MOSCITO 4* provides programs, which allow one to perform molecule parameterisations based on the freely available Cornell et al. forcefield [5], which is part of the AMBER package.

This manual comprises the documentation for the *MOSCITO 4* software. It is meant to introduce the user to algorithms, provide references for the computational methods. An extensive part of this manual will deal with "example applications", where the use of the provided software will be demonstrated in detail.

[1] A German textbook on molecular dynamics simulation by Haberlandt et al. [3] deals particularly with applications in physical chemistry.

1.2 Citation Form

The *MOSCITO 4* simulation package should be cited as follows:

> MOSCITO 4, D. Paschek and A. Geiger, Department of Physical Chemistry, University of Dortmund, (2002).

1.3 Obtaining *MOSCITO 4*

The newest version of the *MOSCITO 4* molecular dynamics simulation package is always available through the Internet from the *MOSCITO 4* home page at:

`http://ganter.chemie.uni-dortmund.de/MOSCITO`

The *MOSCITO 4* package can be directly downloaded using a browser like netscape and saved as a source–file. If you have hints, suggestions and/or comments you can send us an e–mail:

Dietmar Paschek
Physikalische Chemie II a
Universität Dortmund
D-44221 Dortmund
`paschek@pc2a.chemie.uni-dortmund.de`

Alfons Geiger
Physikalische Chemie II a
Universität Dortmund
D-44221 Dortmund
`geiger@pc2a.chemie.uni-dortmund.de`

we are also interested in suggestions for improving this manual.

1.4 Disclaimer

1.5 Acknowledgements

We would like to thank the following people who have contributed to *MOSCITO 4* in any possible way: Andreas Appelhagen, Ralf Baumert, Oliver Biermann, Robert Bieshaar, Frank Eikelschulte, Amelie Rehtanz, Ingo Köper, Sascha Nonn, Frank Schmauder and Ralf Schmelter. Ralf Schmelter has to be thanked extensively for contributing the Intel Assembler coding of "`force_ew_fast`" and some tuning of PME– and neighborlist routines. Sascha Nonn has contributed his X11-based crd-file trajectory viewer.

2 Installation

2.1 Source distribution

To install *MOSCITO 4* you have to create a directory somewhere in your directory tree (e.g. `/home/myself/MOSCITO`) and to unpack the `moscito_source.tgz` archive file (which is gnuzipped) there:

```
                   gzcat moscito_source.tgz | tar -xvf -
or alternatively   tar -xvzf moscito_source.tgz
```

You will obtain the complete *MOSCITO 4* sub–directory tree which is organized as follows:

`./Machines`	System dependent Makefile headers.
`./MosView-0.84b`	X11-based crd-file trajectory viewer contributed by Sascha Nonn.
`./bin`	All executables.
`./doc`	The documentation.
`./examples`	Examples for reference and testing.
`./forcefield`	Several programs and forcefield-libraries to conveniently create and handle moscito system-files.
`./include`	The header files.
`./mostools`	The *MOSCITO 4* utility program sources.
`./parallel`	The parallel *MOSCITO 4* MD program sources.
`./potential`	Some programs to determine point charge models and torsion potentials from ab initio calculations.
`./src_drop`	Sources for a MD code without periodic boundary conditions.
`./src_fft`	The *netlib* fast Fourier transform routines.
`./src_mos`	The *MOSCITO 4* MD program sources.
`./src_pme`	The particle mesh Ewald sum routines.
`./src_xrd`	The XRD libraries for compact coordinate files.

If you would like to use all the features the *MOSCITO 4* system provides, you should definitely compile all the utility programs in the `./mostools` and `./forcefield` directories (they are discussed in detail in subsequent chapters) as well. All the mentioned programs are compiled automatically when using *make* in the *MOSCITO 4* main directory (see `Makefile` therein for details). However, before invoking the *make*-command, the appropriate configuration has to be selected in the `Makefile` by activating one of the given switches (shown in Figure 2.1). One of the corresponding Makefile-headers (which reside in `./Machines`) will then by copied to the main dir as `Makefile.system` and will then be included by any other Makefile. All system dependent compiler options are defined here.

```
#####################################################
#
#Switches for the MOSCITO progs:
#
#GENERIC=TRUE
#SUNSPARC=TRUE
#ALPHA_AXP=TRUE
#RS6000=TRUE
#RS6000_PWR2=TRUE
#RS6000_PPC604_fftw=TRUE
#Linux=TRUE
#Linux_i486=TRUE
#Linux_i686=TRUE
#Linux_i686_fftw=TRUE
Linux_i686_dfftw=TRUE
#SuSE_8.1=TRUE
#RedHat=TRUE
#
#Switches for the MOSVIEW code:
#
MOSVIEW=TRUE
#
#####################################################
```

Figure 2.1: Head section of the Makefile in the *MOSCITO 4* main directory.

If your System has a modern Intel/AMD hardware running Linux with a GNU (gcc/g77) (e.g. gcc-2.95.3) compiler you should be able to compile the whole system straight out of the box. We recommend using the *FFTW* library for fast Fourier transformation written by Frigo and Johnson [6] and published under the GPL. The libs can be obtained from `http://www.fftw.org` and are available for several platforms. The routines have turned out to perform quite effectively on Intel/AMD architectures. Moreover, *FFTW* is already part of several common Linux distributions like e.g. SuSE 7.2. For such a case the "`Linux_i686_fftw=TRUE`" option will do well right from the start.

In case *FFTW* is not available and you cannot install it for some reason you can alternatively use the *netlib* FFT routines which reside in the `./src_fft` subdir. To choose them one would use the "`Linux_i686=TRUE`" option.

If your architecture is not present Fig 2.1, try the "`GENERIC=TRUE`"-option using the `./Machines/Makefile.system_GENERIC`–file and see what will happen. If this doesn't work you have to find out the necessary compiler commands/directives on your own and do the compilation by hand. Please note that due to dependencies it is important to do compilation in the correct order: First, compile the FFT and PME–routines, then the *MOSCITO 4* simulation program and finally the utility programs.

To allow access to the set of *MOSCITO 4* programs from any point of your directory-tree you should add "`/home/myself/MOSCITO/bin`" to your `$PATH`–variable. Now you should be able to run *MOSCITO 4* from anywhere in your system you like. Finally, a `make clean` can be used to tidy up the directory tree (Note that the binaries will be discarded as well!).

2.1.1 Fixed array dimensions

As most of the *MOSCITO 4* code is written in FORTRAN77, the array sizes are hardcoded statically. In order to do this the most consistent way, the size-limits are defined in one single file `./include/mos_const.h` which is included by all of the programs (see file for a more detailed description of the constants). If you have a system which exceeds the default limits you have perhaps to increase some of the values. Typically, moscito stops with a message indicating which limit has been exceeded.

2.1.2 Some Intel/AMD specific code

The real space force evaluation has been directly coded in Intel Assembler (`.src_mos/force_ew_fast_i386.S`) yielding some significant performace gain. In section 4.1.5 the keywords are explained which enable using this subroutine.

2.1.3 Some notes concerning recent Red Hat and SuSE Linux distributions

If you use a Red Hat (7.x) or SuSE (8.x) Linux distribution you have to apply some minor modifications. Since the employed gcc3.x *cpp* behaves differently you have to replace

```
CPPOPT  = -P
```

by

```
CPPOPT  = -P -traditional -undef
```

in your `./Machines/Makefile.system_Linux`... to ensure that the Fortran source code is processed correctly or simply activate the `SuSE_8.1=TRUE` or `RedHat=TRUE`-switch.

2.2 Binary distribution

For Linux systems running on an Intel/AMD hardware we provide a binary distribution of the *MOSCITO 4* package (`moscito_binaries.tgz`) on our WWW-site.

2.3 Basic testing

Go to the `./example` subdir and start the `RUN.examples` shell script. This will invoke five subsequent MD simulation runs of several molecular systems with different sizes. A detailed description of the systems under consideration will be written to ⟨STDOUT⟩. Moreover, the produced averages for pressure, energies etc. are compared with reference data from our lab. The differences of the printed data should be in any case zero. Otherwise, the build has been errornous or your system has some severe problems with numerical accuracy. In addition, detailed program-timings are indicated. Please don't hesitate to send us the results together with some information on the system that you use (processor-type, clock-rate, OS, compiler, etc.) so that we can add it to our *MOSCITO 4* -benchmark list available at:

`http://ganter.chemie.uni-dortmund.de/MOSCITO/moscito_bench.shtml`

3 MD–Simulation: Basics

Molecular Dynamics simulation in the presented sense is a method to compute static and dynamical equilibrium properties of *classical* many–body systems. As the name indicates, the constituent "bodies" are molecules forming a certain ensemble, governed by the laws of classical statistical mechanics. This *classical* approach works rather well in the high temperature limit $h\nu \leq k_B T$, where quantum effects play a minor role. This chapter will focus on the molecular dynamics techniques which are implemented in *MOSCITO 4* .

3.1 MD–Algorithm

The concept of classical molecular dynamics in general is based on the idea of solving Newton's classical equations of motion for an interacting N–body system:

$$\begin{aligned} \frac{d^2\mathbf{r}_i(t)}{dt^2} &= m_i^{-1}\mathbf{f}_i \\ \mathbf{f}_i &= -\frac{\partial\mathcal{V}(\mathbf{r}_1....\mathbf{r}_N)}{\partial\mathbf{r}_i}. \end{aligned}$$

$\mathbf{f}_i$ is the force acting on particle i and $\mathcal{V}$ is the total interaction potential. Facing the fact that an analytical treatment fails if more than two particles are involved, this procedure has to be done numerically using finite differences.

A number of different integration schemes to solve this set of differential equations have been proposed. Luckily, one of the most simple ones is perhaps the most robust and usually the best: The Verlet algorithm. In *MOSCITO 4* it is implemented in its so called leapfrog form, where velocities $\boldsymbol{v}_i$ are calculated at half integer time-steps:

$$\begin{aligned} \boldsymbol{v}_i(t_n + \Delta t/2) &= \boldsymbol{v}_i(t_n - \Delta t/2) + \Delta t\; m_i^{-1}\mathbf{f}_i(\mathbf{r}_1...\mathbf{r}_N, t_n) + O(\Delta t^3) \\ \mathbf{r}_i(t_n + \Delta t) &= \mathbf{r}_i(t_n) + \boldsymbol{v}_i(t_n + \Delta t/2)\Delta t + O(\Delta t^3). \end{aligned}$$

This gives rise to a slight disadvantage of the leapfrog algorithm. It is not possible to calculate the total energy rigorously, because kinetic and potential energy are calculated at two different times. However, it shows consistently small errors of $O(\Delta t^3)$ for velocities and positions. As shown by a number of authors, the presented scheme is perfectly adequate for the MD problem. The achieved accuracy is maybe not impressive, but a better, more exact algorithm wouldn't do much better due to the chaotic nature of the problem. However, the algorithm's properties of good short-time and long-time energy conservation and time reversibility make it superior to other schemes, especially at *large* time-steps.

3.2 Constraint dynamics

The time-step Δt and consequently the efficiency of a MD–simulation is limited by the highest possible frequencies ν_{max} occurring in the system

$$\Delta t \ll \nu_{max}^{-1}.$$

If the system contains degrees of freedom corresponding to high frequency modes with small amplitude and little or no correlation to all other degrees of freedom, it is found to be a good approximation to separate these particular degrees of freedom and to freeze them using constrained dynamics. This approach has been proven to work well for bond lengths and to a lesser extent for bond bending interactions.

Several methods have been proposed to calculate the constraints. In *MOSCITO 4* the SHAKE–scheme for distance constraints according to Ryckaert et al.[7] has been implemented. A distance constraint between two particles at a time-step t is defined by

$$\sigma_k = (\mathbf{r}_i - \mathbf{r}_j)^2 - d_{ij}^2 = 0.$$

Thus, the equations of motions for the particles change in the following way introducing constraint–forces $\mathbf{g}_i$ on particle i

$$m_i \frac{d^2 \mathbf{r}_i}{dt^2} = \mathbf{f}_i + \mathbf{g}_i = -\frac{\partial \mathcal{V}(\mathbf{r}_1 .... \mathbf{r}_N)}{\partial \mathbf{r}_i} - \sum_k \lambda_k \frac{\partial \sigma_k}{\partial \mathbf{r}_i}. \tag{3.3}$$

The Lagrange multipliers λ_k have to be determined, so that all constraints are fulfilled. The SHAKE procedure solves the problem by correcting the positions obtained from a constraint–free dynamics $\mathbf{r}_i'$ using

$$\mathbf{r}_i = \mathbf{r}_i' + \delta \mathbf{r}_i.$$

The correction is performed along the distance constraint vector $\mathbf{r}_{ij} = \mathbf{r}_i - \mathbf{r}_j$ at the initial time-step t which leads to

$$\delta \mathbf{r}_i(t + \Delta t) = -\frac{\Delta t^2}{m_i} \sum_k \lambda_k \left(\frac{\partial \sigma_k}{\partial \mathbf{r}_i} \right)_t = -\frac{2\Delta t^2}{m_i} \sum_k \lambda_k \mathbf{r}_{ij}(t). \tag{3.5}$$

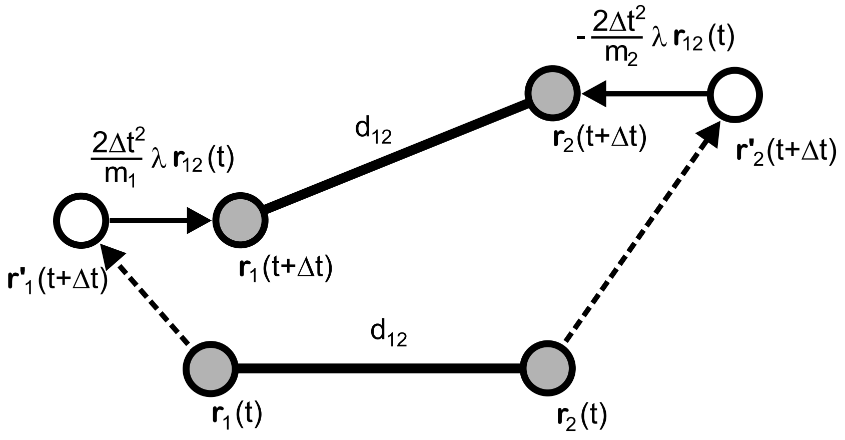
The problem is to determine the multipliers λ_k that enable all the constraints to be satisfied simultaneously. This can be done algebraically only in simple cases. Suppose, we wish to fix the bond–length in a diatomic molecule. There is just one constraint in this case, so we can write:

$$\mathbf{r}_1(t + \Delta t) = \mathbf{r}_1'(t + \Delta t) + \frac{2\Delta t^2}{m_1} \lambda \mathbf{r}_{12}(t) \tag{3.6}$$

$$\mathbf{r}_2(t + \Delta t) = \mathbf{r}_2'(t + \Delta t) - \frac{2\Delta t^2}{m_2} \lambda \mathbf{r}_{12}(t) \tag{3.7}$$

A third equation is derived from the requirement that the bond length is kept fixed

$$|\mathbf{r}_1(t + \Delta t) - \mathbf{r}_2(t + \Delta t)|^2 = |\mathbf{r}_1(t) - \mathbf{r}_2(t)|^2 = d_{12}^2. \tag{3.8}$$



$-\frac{2\Delta t^2}{m_2}\lambda \mathbf{r}_{12}(t)$
$\frac{2\Delta t^2}{m_1}\lambda \mathbf{r}_{12}(t)$
d_{12}
$\mathbf{r}_2(t+\Delta t)$
$\mathbf{r'}_2(t+\Delta t)$
$\mathbf{r'}_1(t+\Delta t)$
$\mathbf{r}_1(t+\Delta t)$
d_{12}
$\mathbf{r}_1(t)$
$\mathbf{r}_2(t)$

Figure 3.1: Example application of the SHAKE procedure to a diatomic molecule.

We now have three equations and three unknowns. Subtracting gives:

$$\mathbf{r}_{12}(t+\Delta t) = \mathbf{r}'_{12}(t+\Delta t) + 2\lambda\Delta t^2 \left(\frac{1}{m_1} + \frac{1}{m_2}\right) \mathbf{r}_{12}(t) \tag{3.9}$$

Squaring both sides and imposing the constraint gives:

$$\begin{aligned} \mathbf{r}'_{12}(t+\Delta t)^2 + 4\lambda\Delta t^2 \left(\frac{1}{m_1} + \frac{1}{m_2}\right) \mathbf{r}_{12}(t) & \\ + 2\lambda^2\Delta t^4 \left(\frac{1}{m_1} + \frac{1}{m_2}\right)^2 \mathbf{r}_{12}(t)^2 &= d_{12}^2 \end{aligned} \tag{3.10}$$

Solving this quadratic equation for λ enables the new positions $\mathbf{r}_1(t+\Delta t)$ and $\mathbf{r}_2(t+\Delta t)$ to be determined. However, in the general case with more than two atoms and more than one constraint per atom, the algebraic approach becomes rather complicated. The SHAKE procedure simplifies the problem by ignoring the terms which are quadratic in λ and solving it in an iterative manner. This is necessary, since satisfying one constraint may cause another one to be violated. Usually only a few iteration cycles are necessary to satisfy the constraints in an acceptable tolerance. This depends, of course, on the length of the time-step used in simulation. The tolerance should be tight enough to ensure that the fluctuations in the simulation are small compared to fluctuations due to other sources, such as cutoffs. Angle constraints can be easily accommodated in the SHAKE scheme by recognising that an angle constraint simply corresponds to an additional distance constraint. A systematic problem with SHAKE arises when planar structures of more than three atoms shall be constrained. This is due to the fact that out–of–plane distortions converge very poorly, since there are no directions available along which these displacements could be shifted back efficiently.

Normally SHAKE will be terminated if the constraints are satisfied within a given tolerance. The procedure will fail and the program stops, if one of the following two things happens:

1. An excessive number (100) of iterations is required due to badly defined constraints. *MOSCITO 4* will stop with the following error–message occurring:

```
SHAKE:   molecule no. 4
         number of iteration has exceeded maximum
         (100) -- simulation has been stopped !
```

2. The positional shift is so large in the unconstrained step that no correction along $\mathbf{r}_{12}(t)$ can be found that will produce a distance d_{12} between the corrected positions of atom 1 and 2. This may happen in an artificial starting configuration where strong forces occur due to an overlapping of particles. The corresponding message is:

```
SHAKE:   molecule no. 4
         deviation too large.
         simulation has been stopped !
```

3.3 Periodic boundary conditions and minimum image convention

Simulations of condensed phases are only possible in so called *periodic boundary conditions*[1]. Otherwise, the always present surface effects would affect the system properties in a non–predictable manner. The periodic boundary conditions are achieved by replicating the central simulation cell infinitely in space. Thus, as a molecule leaves the central box, one of its images enters through the opposite face. The central box has no walls at its boundaries; therefore the system has no surface. The only requirement for the shape of the central cell is a space-filling geometry. In *MOSCITO4* , actually, only rectangular systems can be considered.

Along with the periodic boundary conditions comes the definition of the *minimum image convention* (MIC). This means, considering a particle i, that all other particle positions are projected to a cell, where i is placed in its center. In rectangular systems the distance between two particles according to the minimum image convention is usually calculated by the “nint–trick”

$$\Delta x_{ij} = x_j - x_i - L_x \operatorname{anint}\Big((x_j - x_i)/L_x\Big) \;,$$

where L_x is the length of the box in the x–direction. Of course, the same procedure has to be applied also to the y– and z–directions. The procedure allows the calculation of the minimum distance between two image particles. The minimum image convention is just one technique among others to handle intermolecular distances in periodic systems. Different schemes have been proposed and were implemented (eg. in the MOLDY simulation program by K. Refson). The evaluation of intermolecular distances in *MOSCITO4* is rigorously based on the MIC. However, the necessary data–type conversion operations are computationally demanding. Therefore, significant speedup can be achieved by consequently reducing the number of MIC–operations. In most cases it is fully sufficient to evaluate only one MIC translation–vector per molecule–molecule pair. Practically, every time an atom–atom pair in the pair–list is found to belong to a new molecule–molecule pair, a new MIC translation–vector is created. Note, however, that this assumption is violated when relatively small systems of large molecules are to be considered (for more details see section 4.1.5).

[1]Sometimes also referred to as toroidal boundary conditions.

The application of the minimum image technique unambiguously requires a restriction to distances lower than $L_{x,y,z}/2$. This is the reason why the maximum cutoff–radius for the force evaluations is typically $L/2$. The sometimes mentioned argument for $r_c < L/2$ cutoff radii to avoid an artifactual behavior being introduced by an interaction of a particle with its own periodic image has no serious background. In fact, methods which treat long range interactions like the Ewald summation *are based* on the assumption that a particle interacts with *all* virtual duplicates.

3.4 The *MOSCITO 4* force field model

The fundamental assumption underlying Molecular Dynamics simulation is the validity of the Born–Oppenheimer approximation. Therefore, it is assumed that the electronic degrees of freedom always stay in equilibrium with the changing configuration of the nuclei. Consequently, the total interaction potential can be expressed as a function of the positions of the nuclei. To be suitable for MD–simulations the interaction potential and forces have to be given in an analytical form. Both, analytical form and parameterisation are typically referred to as a molecular mechanical force field. Note that the potentials discussed here are based on a pairwise additive approach[2]. Force fields which neglect couplings between individual terms are often considered as *diagonal* force fields. This type is used mostly for MD simulations. As mentioned before, there are no particular forcefield restrictions, when working with the *MOSCITO 4* simulation program. The given analytical form is flexible enough to handle state of the art molecular force fields like AMBER and CHARMm. As we will see in a later chapter, the Cornell et al. [5] force field can be easily used together with *MOSCITO 4* .

[2]This is not exactly the truth: The bond–bending term is a 3-body– and the dihedral terms are in fact 4-body–interactions.

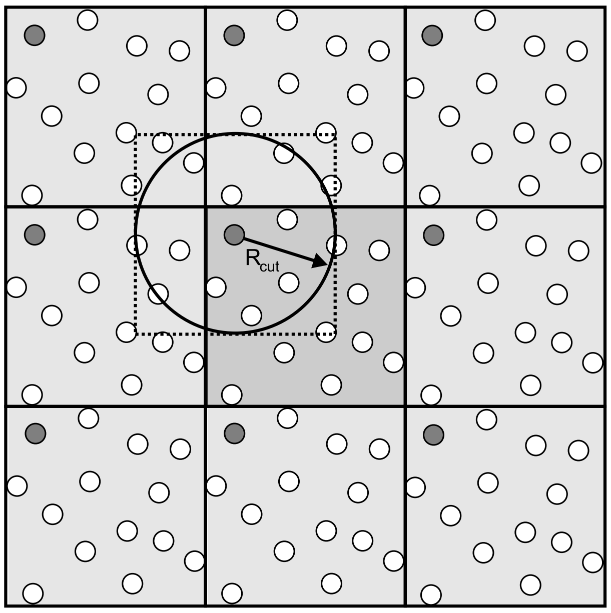


Figure 3.2: Periodic boundary conditions. The minimum image of the grey shaded particle is indicated.

The force field model is given in terms of the potential energy. For reasons of convenience, the total interaction energy is typically divided into an inter– and intramolecular part

$$E = E_{inter} + E_{intra}.$$

To approximate the potential interaction function the so called *interaction–sites* are formally introduced.

In the case of intermolecular interactions these interaction–sites carry the non–bonded interactions like Pauli–repulsion, electrostatic and dispersive interaction. They are modelled by point charges q_i and Lennard–Jones interactions

$$\begin{aligned} E_{inter} &= \underbrace{\frac{1}{4\pi\epsilon_0}\sum_n\sum_\kappa\sum_{m>n}\sum_\lambda \frac{q_{n\kappa}q_{m\lambda}}{r_{n\kappa m\lambda}}}_{\text{Coulomb term}} \\ &+ \underbrace{\sum_n\sum_\kappa\sum_{m>n}\sum_\lambda 4\,\epsilon_{n\kappa m\lambda}\left\{\left(\frac{\sigma_{n\kappa m\lambda}}{r_{n\kappa m\lambda}}\right)^{12} - \left(\frac{\sigma_{n\kappa m\lambda}}{r_{n\kappa m\lambda}}\right)^{6}\right\}}_{\text{Lennard-Jones term}} . \qquad (3.12) \end{aligned}$$

Typically these sites reside on the positions of the atoms, but this need not be the case. *MOSCITO 4* allows introduction of massless *virtual* sites which have to be defined in a framework of real atoms. A generalised force–shifting procedure which conserves total force and torque redistributes the forces to the atoms.

To be able to model an intramolecular potential function adequately, it is necessary to introduce additional terms which represent the covalent forces:

$$
\begin{aligned}
E_{intra} \; = \; & \underbrace{\frac{1}{4\pi\epsilon_0} \sum_n \sum_\kappa \sum_{\lambda>\kappa} \frac{q_{n\kappa} q_{n\lambda}}{r_{n\kappa\lambda}} f_{n\kappa\lambda}}_{\text{Coulomb term}} \\
+ \; & \underbrace{\sum_n \sum_\kappa \sum_{\lambda>\kappa} 4 \, \epsilon_{n\kappa\lambda} \left\{ \left(\frac{\sigma_{n\kappa\lambda}}{r_{n\kappa\lambda}} \right)^{12} - \left(\frac{\sigma_{n\kappa\lambda}}{r_{n\kappa\lambda}} \right)^{6} \right\}}_{\text{Lennard-Jones term}} \\
+ \; & \underbrace{\frac{1}{2} \sum_n \sum_g k^b_{ng} \left(r_{ng(\kappa\lambda)} - r^0_{ng} \right)^2}_{\text{bond–stretching}} \\
+ \; & \underbrace{\sum_n \sum_g k^{bm}_{ng} \left[\left(1 - \exp\left(-\alpha^{bm}_{ng} \left(r_{ng(\kappa\lambda)} - r^0_{ng} \right) \right) \right)^2 - 1 \right]}_{\text{bond–stretching (Morse)}} \\
+ \; & \underbrace{\frac{1}{2} \sum_n \sum_g k^a_{ng} \left(\phi_{ng(\kappa\lambda\omega)} - \phi^0_{ng} \right)^2}_{\text{bond–bending}} \\
+ \; & \underbrace{\sum_n \sum_g k^{al}_{ng} \left[1 + \cos\left(\phi_{ng(\kappa\lambda\omega)} \right) \right]}_{\text{linear bond–bending}} \\
+ \; & \underbrace{\frac{1}{2} \sum_n \sum_g k^{di}_{ng} \left(\psi_{ng(\kappa\lambda\omega\tau)} - \psi^0_{ng} \right)^2}_{\text{improper dihedral}} \\
+ \; & \underbrace{\sum_n \sum_g k^{dp}_{ng} \left[1 + \cos\left(m_{ng} \psi_{ng(\kappa\lambda\omega\tau)} - \psi^0_{ng} \right) \right]}_{\text{proper dihedral}} \qquad (3.13)
\end{aligned}
$$

Harmonic and Morse–type bond stretching terms, and angular bond-bending terms exist for this purpose [3]. To model torsion potentials appropriately, two types of dihedral potentials exist: (1) an harmonic *improper* dihedral potential, which will be used if only small distortions of a given geometry will be allowed (eg. to define planar structures like benzene rings); (2) a series of cosines forming *proper* dihedral potentials with freely tunable phase ψ^0_{ng} and multiplicity m_{ng} which will be used in more typical situations. The intramolecular nonbonded interaction in terms of Coulomb and Lennard-Jones potentials can differ substantially from the definitions used for the intermolecular interactions. A special set of Lennard-Jones parameters and a modified Coulomb-interaction reduced by a factor of $f_{n\kappa\lambda}$ is sometimes necessary to be able to parametrise torsional potentials properly. *MOSCITO 4* allows *any* nonbonded

[3]For 180° bond angles a singularity free treatment according to F. Müller–Plathe [8] has been implemented called 'linear bond bending'.

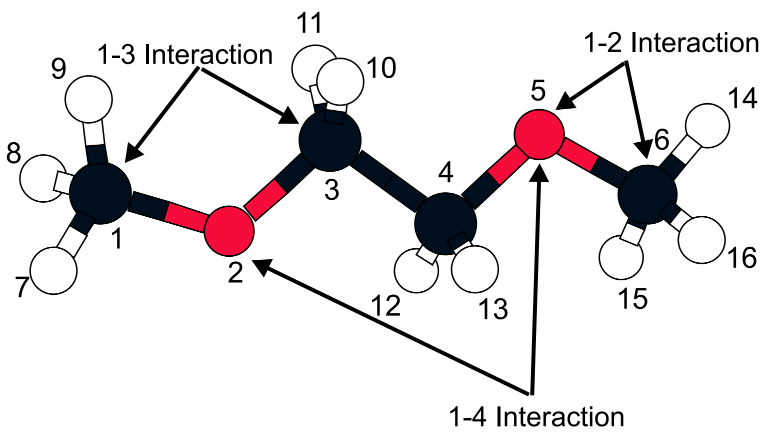


Figure 3.3: Specific 1-x–interactions in a molecular model. The nonbonded interaction of these combinations are typically treated in special way.

intramolecular interaction pair to be modified this way. Normally nonbonded 1-2 and 1-3–interactions (see for example figure 3.3) are completely neglected. This avoids unphysical repulsive forces between chemically bounded centers. So, the local geometry is mainly determined by bonded interactions. As mentioned before, explicitly modified 1-4 interactions are important for torsional potentials. In very rare cases 1-5 interactions have to be modified.

In typical force fields the Lennard–Jones interactions are parameterised for pair–interactions of identical sites. The cross-terms are obtained by applying so called combination rules. The most frequently used are the Lorentz–Berthelot combination rules

$$\sigma_{ij} \;=\; \frac{1}{2}(\sigma_{ii}+\sigma_{jj}) \quad ; \quad \epsilon_{ij} = (\epsilon_{ii}\epsilon_{jj})^{\frac{1}{2}} \,.$$

The OPLS–forcefield, as well as the GROMOS forcefield use a geometrical mean as well for Lennard-Jones σ

$$\sigma_{ij} \;=\; (\sigma_{ii}\sigma_{jj})^{\frac{1}{2}} \quad ; \quad \epsilon_{ij} = (\epsilon_{ii}\epsilon_{jj})^{\frac{1}{2}} \,.$$

Both approaches can be used for the definition of a molecular model in *MOSCITO 4* . Additionally, specific terms based on an approach different from the two mentioned here can be defined for explicit pairs.

3.5 Handling electrostatic interactions

For most molecular systems electrostatic interactions play an important role. Because of their long–range nature they afford a particularly different treatment than short–range interactions, which can be simply truncated. The electrostatic force evaluation in *MOSCITO 4* is based on the Ewald–method and on a recently developed modification, the particle-mesh Ewald (PME) approach.

3.5.1 The concept of Ewald summation

The Ewald ansatz is based on a solution of the Poisson equation for a periodic system. The Poisson equation relates the electrostatic potential $\phi(\mathbf{r})$ to the charge distribution $\rho(\mathbf{r})$

$$-\epsilon_0 \nabla^2 \phi(\mathbf{r}) = \rho(\mathbf{r}).$$

In reciprocal space the differential equation has an algebraic form

$$\epsilon_0 k^2 \hat{\phi}(\mathbf{k}) = \hat{\rho}(\mathbf{k})$$

and can be readily solved. $\mathbf{k}$ is the reciprocal lattice vector with

$$\mathbf{k} = 2\pi \, (l/L_x, m/L_y, n/L_z).$$

l, m, n are integers and L_x, L_y, L_z are the dimensions of the simulation box.

Due to the nature of the force field concept, the charge distribution $\rho(\mathbf{r})$ is expressed in terms of a sum of point charges q_i

$$\rho(\mathbf{r}) = \int_V \sum_{\mathbf{n}} \sum_i q_i \, \delta(\mathbf{r}_i + \mathbf{n} - \mathbf{r}) \, d\mathbf{r}. \tag{3.15}$$

$\mathbf{n}$ are lattice vectors generating an infinite real–space lattice. Consequently, the Coulomb contribution to the total energy of the system is

$$E_{Coul} = \frac{1}{2} \sum_i q_i \, \phi(\mathbf{r}_i), \tag{3.16}$$

where $\phi(\mathbf{r}_i)$ is the electrostatic potential at the position of point charge i, defined by all point charges (except i) in the entire inifinite volume

$$\phi(\mathbf{r}) = \frac{1}{4\pi\epsilon_0} \sum_{\mathbf{n}} \sum_i \frac{q_i}{|\mathbf{r}_i + \mathbf{n} - \mathbf{r}|}. \tag{3.17}$$

Due to the slow convergence of this sum, P.P. Ewald [9] proposed a different concept which is based on splitting the charge distribution into two parts. The idea is to turn the character of the interaction into a short–range behavior by adding a Gaussian charge cloud

$$\rho_i^{Gauss}(\mathbf{r}) = -q_i \left(\frac{\alpha^2}{\pi}\right)^{\frac{3}{2}} \exp\left(-\alpha^2 \, (\mathbf{r}_i - \mathbf{r})^2\right) \tag{3.18}$$

with opposite sign to each point charge . This will result in a fast converging “real space” sum. Of course, it is necessary to add Gaussian charge clouds with the original sign to restore the original charge distribution again. The contribution of the latter part of the charge distribution to the electrostatic potential is evaluated in reciprocal space by applying Fourier transformation (“reciprocal lattice sum”). Due to the smoothness of this Gaussian charge distribution, it turns out that the Fourier sum converges rapidly. So, the methodology can be summarised as an efficient replacement of a slowly converging sum by two rapidly converging ones. A careful choice of the width of the Gaussian charge distribution α is important for the method's efficiency. α is therefore sometimes considered as the Ewald “convergence parameter”.

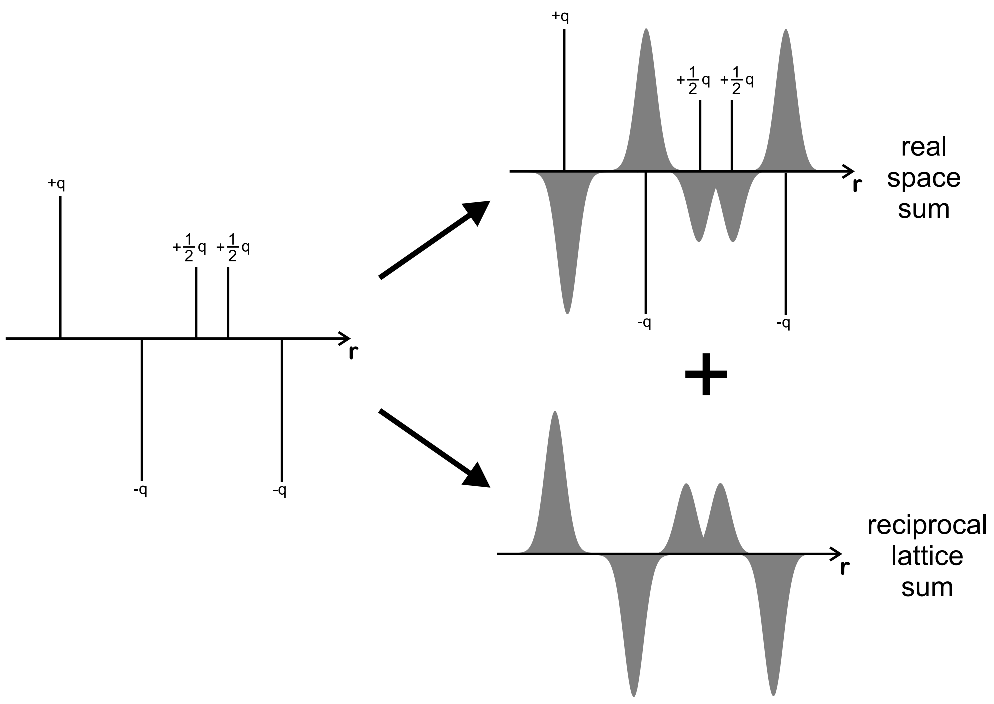


Figure 3.4: The original charge distribution is superimposed by Gaussian charge clouds of opposite sign to achieve a short-range behavior. The potential according to the Gaussian "screening background" is calculated as a reciprocal lattice sum.

Real space sum

We can obtain the electrostatic potential at a distance r to a Gaussian charge distribution

$$\rho^{\,\mathrm{Gauss}}(r) \;=\; q\,\left(\frac{\alpha^2}{\pi}\right)^{\frac{3}{2}} \exp\!\left(-\alpha^2 r^2\right) \tag{3.19}$$

by directly writing down the Poisson equation, while making use of the system's spherical symmetry

$$-\,\epsilon_0\,\frac{1}{r}\,\frac{\partial^2\, r\phi^{\,\mathrm{Gauss}}(r)}{\partial r^2} \;=\; \rho^{\,\mathrm{Gauss}}(r) \tag{3.20}$$

or equally

$$\frac{\partial^2\, r\phi^{\,\mathrm{Gauss}}(r)}{\partial r^2} \;=\; \frac{-q}{\epsilon_0}\, r\,\left(\frac{\alpha^2}{\pi}\right)^{\frac{3}{2}} \exp\left(-\alpha^2 r^2\right). \tag{3.21}$$

Partial integration yields[4]

$$\begin{aligned}\frac{\partial\, r\phi^{\,\mathrm{Gauss}}(r)}{\partial r} &= \frac{-1}{\epsilon_0}\int_{\infty}^{r} r\,\rho^{\,\mathrm{Gauss}}(r)\,dr \\ &= \frac{-q}{\epsilon_0}\left(\frac{\alpha^2}{\pi}\right)^{\frac{3}{2}}\int_{\infty}^{r} r\,\exp\left(-\alpha^2 r^2\right)dr \\ &= \frac{q}{2\pi\epsilon_0}\sqrt{\frac{\alpha^2}{\pi}}\,\exp\left(-\alpha^2 r^2\right). \end{aligned} \tag{3.22}$$

A second partial integration gives

$$\begin{aligned} r\phi^{\,\mathrm{Gauss}}(r) &= \frac{q}{2\pi\epsilon_0}\sqrt{\frac{\alpha^2}{\pi}}\int_{0}^{r}\exp\left(-\alpha^2 r^2\right)dr \\ &= \frac{q}{4\pi\epsilon_0}\,\mathrm{erf}\,(\alpha\, r) \end{aligned} \tag{3.23}$$

where in the last line, the definition of the error function

$$\mathrm{erf}(x) \equiv (2/\sqrt{\pi})\int_{0}^{x}\exp(-y^2)\,dy$$

has been applied. Hence, the electrostatic potential due to a Gaussian charge distribution is

$$\phi^{\,\mathrm{Gauss}}(r) = \frac{q}{4\pi\epsilon_0}\,\frac{1}{r}\,\mathrm{erf}\,(\alpha\, r). \tag{3.25}$$

Consequently, the electrostatic potential of a point charge, which is surrounded by a Gaussian charge distribution with opposite sign can be expressed as

$$\begin{aligned}\phi^{\,\mathrm{real}}(r) &= \frac{q}{4\pi\epsilon_0}\left\{\frac{1}{r} - \frac{1}{r}\,\mathrm{erf}\,(\alpha\, r)\right\} \\ &= \frac{q}{4\pi\epsilon_0}\,\frac{1}{r}\,\mathrm{erfc}\,(\alpha\, r). \end{aligned} \tag{3.26}$$

Therefore, the real space part of the potential energy can be expressed as[5]

$$E^{\,\mathrm{real}}_{\,\mathrm{Coul}} \equiv \frac{1}{4\pi\epsilon_0}\sum_{i}\sum_{j>i} q_i q_j \frac{1}{r_{ij}}\,\mathrm{erfc}\,(\alpha\, r_{ij}). \tag{3.27}$$

[4] The integration boundary is chosen so that the integration constant vanishes.

[5] If the minimum image convention is applied, distances larger than half a box length are not allowed. Typically, the interaction is truncated at a cutoff–distance $r_{\mathrm{cut}} \leq L/2$. In this case, a summation over all lattice vectors $\mathbf{n}$ doesn't have to be performed.

Reciprocal lattice sum

As indicated in figure 3.4, the total Gaussian charge distribution $\rho^{\text{rec}}(\mathbf{r})$ consists of a periodic sum of Gaussians at the positions $\mathbf{r}_j$ as defined by

$$\rho^{\text{rec}}(\mathbf{r}) = \sum_{\mathbf{n}} \sum_j q_j \left(\frac{\alpha^2}{\pi}\right)^{\frac{3}{2}} \exp\left(-\alpha^2 |\mathbf{r}_j + \mathbf{n} - \mathbf{r}|^2\right) .$$

Fourier–transforming the charge density yields

$$\begin{aligned}
\hat{\rho}^{\text{rec}}(\mathbf{k}) &= \frac{1}{V}\int_V \rho^{\text{rec}}(\mathbf{r}) e^{-i\mathbf{k}\cdot\mathbf{r}}\, d\mathbf{r} \\
&= \frac{1}{V}\int_V \left\{ \sum_{\mathbf{n}} \sum_j q_j \left(\frac{\alpha^2}{\pi}\right)^{\frac{3}{2}} \exp\left(-\alpha^2 |\mathbf{r}_j + \mathbf{n} - \mathbf{r}|^2\right) \right\} e^{-i\mathbf{k}\cdot\mathbf{r}}\, d\mathbf{r} \\
&= \frac{1}{V}\int_V \left\{ \sum_j q_j \left(\frac{\alpha^2}{\pi}\right)^{\frac{3}{2}} \exp\left(-\alpha^2 |\mathbf{r}_j - \mathbf{r}|^2\right) \right\} e^{-i\mathbf{k}\cdot\mathbf{r}}\, d\mathbf{r} \\
&= \frac{1}{V}\sum_j q_j \exp\left(-\frac{k^2}{4\alpha^2}\right) e^{-i\mathbf{k}\cdot\mathbf{r}_j} \qquad (3.29)
\end{aligned}$$

Inserting the Poisson equation leads to

$$\hat{\phi}^{\text{rec}}(\mathbf{k}) = \frac{1}{\epsilon_0 k^2}\frac{1}{V}\sum_j q_j \exp\left(-\frac{k^2}{4\alpha^2}\right) e^{-i\mathbf{k}\cdot\mathbf{r}_j}. \qquad (3.30)$$

Note, that the expression is defined only for $k \neq 0$. This is a direct consequence of the conditional convergence of the Ewald sum. Neglecting the $k = 0$–term, however, is identical to the assumption of embedding the periodic system in a medium with infinite dielectric constant, the so called *tin–foil* boundary conditions[6].

The electrostatic potential at a position $\mathbf{r}$ due to the Gaussian charge distribution is now given by

$$\begin{aligned}
\phi^{\text{rec}}(\mathbf{r}) &= \sum_{\mathbf{k}\neq 0} \hat{\phi}^{\text{rec}}(\mathbf{k}) e^{i\mathbf{k}\cdot\mathbf{r}} \\
&= \frac{1}{\epsilon_0 V}\sum_{\mathbf{k}\neq 0}\sum_j \frac{q_j}{k^2} \exp\left(-\frac{k^2}{4\alpha^2}\right) e^{-i\mathbf{k}\cdot(\mathbf{r}-\mathbf{r}_j)}. \qquad (3.31)
\end{aligned}$$

Finally, the reciprocal lattice part of the electrostatic interaction can be expressed as

$$\begin{aligned}
E^{\text{rec}}_{\text{Coul}} &\equiv \frac{1}{2}\sum_i q_i \phi^{\text{rec}}(\mathbf{r}_i) \\
&= \frac{1}{2}\sum_{\mathbf{k}\neq 0}\sum_i\sum_j \frac{q_i q_j}{\epsilon_0 V k^2} \exp\left(-\frac{k^2}{4\alpha^2}\right) e^{-i\mathbf{k}\cdot(\mathbf{r}_i-\mathbf{r}_j)} \qquad (3.32)
\end{aligned}$$

[6] In the case of a surrounding medium with vacuum permittivity, a so called surface–dipole term has to be added. Actually, *MOSCITO 4* allows only *tin–foil* boundary conditions.

For computational convenience this can be rewritten

$$\begin{aligned} E^{\text{rec}}_{\text{Coul}} &= \frac{1}{2\epsilon_0 V} \sum_{\mathbf{k}\neq 0} \frac{1}{k^2} \exp\left(-\frac{k^2}{4\alpha^2}\right) \sum_i q_i \, e^{-i\mathbf{k}\cdot\mathbf{r}_i} \cdot \sum_j q_j \, e^{i\mathbf{k}\cdot\mathbf{r}_j} \\ &= \frac{1}{2\epsilon_0 V} \sum_{\mathbf{k}\neq 0} \frac{1}{k^2} \exp\left(-\frac{k^2}{4\alpha^2}\right) |S(\mathbf{k})|^2 \end{aligned} \tag{3.33}$$

with $S(\mathbf{k})$ as the static structure factor of the charge distribution

$$S(\mathbf{k}) \equiv \sum_i q_i \, e^{i\mathbf{k}\cdot\mathbf{r}_i}. \tag{3.34}$$

Corrections for self-interaction

There are two problems which are introduced by the reciprocal lattice term. The first one is that it considers an interaction of a particle i with a charge cloud of the same sign centred at the same position[7]. The second one arises if we have a molecular system where particular pair interactions (eg. 1-2– and 1–3–interactions) shall *not* be considered or others shall be modified. So, we have to apply corrections; the so called *self-interaction* and *molecular self-interaction* terms.

Self-interaction: The correction is generally done by subtracting the interaction energy. Regarding the point charge with the surrounding Gaussian charge cloud we can calculate the interaction readily

$$\phi^{\text{Gauss}}(r=0) = \frac{2q}{4\pi\epsilon_0} \sqrt{\frac{\alpha^2}{\pi}} \tag{3.35}$$

So, the self–energy contribution is

$$\begin{aligned} E^{\text{self}}_{\text{Coul}} &= \frac{1}{2} \sum_i q_i \, \phi^{\text{self}}(\mathbf{r}_i) \\ &= \frac{1}{4\pi\epsilon_0} \sqrt{\frac{\alpha^2}{\pi}} \sum_i q_i^2 \end{aligned} \tag{3.36}$$

Note, that $E^{\text{self}}_{\text{Coul}}$ is a constant. Given the fact that the charges do not vary during a simulation run, this term has to be computed only once at the beginning.

Molecular self interaction: If particular pair interactions within a molecule shall be switched off, we have to subsequently subtract the interaction from the reciprocal lattice term. As outlined in section 3.5.1 the electrostatic potential of a Gaussian charge distribution is

$$\phi^{\text{Gauss}}(r) = \frac{q}{4\pi\epsilon_0} \frac{1}{r} \operatorname{erf}(\alpha r) \tag{3.37}$$

[7] This is due to the fact, that the condition $i \neq j$ is not fulfilled in the reciprocal lattice sum.

Consequently, the total molecular self energy is given by

$$
\begin{aligned}
E^{\,\mathrm{mol.self}}_{\,\mathrm{Coul}} &= \frac{1}{2} \sum_{\mathrm{pairs}(i,j)} q_i\, \phi^{\mathrm{Gauss}}(\mathbf{r}_j) + q_j\, \phi^{\mathrm{Gauss}}(\mathbf{r}_i) \\
&= \frac{1}{4\pi\epsilon_0} \sum_{\mathrm{pairs}(i,j)} \frac{q_i q_j}{r_{ij}}\, \mathrm{erf}\,(\alpha\, r_{ij}) \qquad (3.38)
\end{aligned}
$$

To achieve a consequent separation of inter– and intramolecular contributions to the electrostatic potential, *MOSCITO 4* assumes that *all* intramolecular interactions are *always* switched off. The electrostatic part of the (in some cases different) intramolecular interaction is then subsequently added in a second step. Due to the fact that intermolecular distances may change during a simulation, this correction term has to be calculated for every time-step. Note, that in contrast to the *self* correction the *molecular self* interaction also causes corrections on the forces.

Total electrostatic energy

To summarise, the total electrostatic energy for a molecular system is given by

$$
\begin{aligned}
E_{\,\mathrm{Coul}} &= \underbrace{\frac{1}{4\pi\epsilon_0} \sum_n \sum_\kappa \sum_{m>n} \sum_\lambda q_{n\kappa} q_{m\lambda} \frac{\mathrm{erfc}\,(\alpha r_{n\kappa m\lambda})}{r_{n\kappa m\lambda}}}_{\text{real space term}} \\
&+ \underbrace{\frac{1}{2\epsilon_0 V} \sum_{\mathbf{k}\neq 0} \frac{1}{k^2} \exp\left(-\frac{k^2}{4\alpha^2}\right) |S(\mathbf{k})|^2}_{\text{reciprocal lattice term}} \\
&- \underbrace{\frac{1}{4\pi\epsilon_0} \sqrt{\frac{\alpha^2}{\pi}} \sum_n \sum_\kappa q_{n\kappa}^2}_{\text{self interaction}} \\
&- \underbrace{\frac{1}{4\pi\epsilon_0} \sum_n \sum_\kappa \sum_{\lambda>\kappa} q_{n\kappa} q_{n\lambda} \frac{\mathrm{erf}\,(\alpha r_{n\kappa\lambda})}{r_{n\kappa\lambda}}}_{\text{molecular self interaction}} \\
&+ \underbrace{\frac{1}{4\pi\epsilon_0} \sum_n \sum_\kappa \sum_{\lambda>\kappa} \frac{q_{n\kappa} q_{n\lambda}}{r_{n\kappa\lambda}} f_{n\kappa\lambda}\,.}_{\text{intramolecular interaction}} \qquad (3.39)
\end{aligned}
$$

Note, that for all intramolecular distances the application of the minimum image convention is forbidden.

3.5.2 Smooth particle mesh Ewald

Due to an $O(N^2)$ behaviour of the classical Ewald approach with a fixed cutoff radius, it is almost impossible to realize it for increasingly large systems. Alternative schemes, assigning the charge distribution to a periodic grid like the particle-particle-particle-mesh (P^3M) approach according to Hockney and Eastwood [10] enable the use of highly efficient fast Fourier transform (FFT) techniques and have been shown to overcome this limitation in principle. The particle mesh Ewald (PME) method was inspired by this idea and achieves an accuracy comparing well with standard Ewald summation at moderate numerical effort, and it has the advantage that it can be implemented quite easily in existing Ewald codes [11]. In *MOSCITO 4* we use the smooth particle mesh Ewald technique using cardinal B–spline interpolation outlined by Essmann et al. [12]. Therefore it is useful to express the charge–weighted structure factor in terms of scaled coordinates u_α

$$S(\mathbf{k}) = \sum_i q_i \exp\left[2\pi i \left(\frac{k_x u_{ix}}{K_x} + \frac{k_y u_{iy}}{K_y} + \frac{k_z u_{iz}}{K_z}\right)\right], \tag{3.40}$$

where K_α are integers representing the number of grid points of a real space mesh in α direction and k_α are integers. The grid scaled fractional coordinates for an atom i can be written as

$$u_{i\alpha} = K_\alpha \frac{(\mathbf{r}_i)_\alpha}{L_\alpha} \tag{3.41}$$

where L_α ($\alpha = x, y, z$) are the box lengths of a rectangular MD–cell.

$S(\mathbf{k})$ is a discrete Fourier transform of a set of charges placed irregularly within the unit cell. It is now highly advantageous to express this charge–distribution on a regular grid of points. This enables the use of fast Fourier transform algorithms. The smooth PME–ansatz does this by spline interpolation of the exponentials

$$\exp\left[2\pi i \left(\frac{k_\alpha u_{i\alpha}}{K_\alpha}\right)\right] \approx b_\alpha(k_\alpha) \sum_{l_\alpha} M_n(u_{i\alpha} - l_\alpha) \exp\left[2\pi i \left(\frac{k_\alpha l_\alpha}{K_\alpha}\right)\right], \tag{3.42}$$

where n is the order of spline interpolation and $M_n(u_{i\alpha} - l_\alpha)$ defines the coefficients of the cardinal B–spline at the scaled fractional coordinates $u_{i\alpha}$. The sum over l_α, representing the grid points, is only over a finite range of integers, since the functions $M_n(u)$ are zero outside the interval $0 \le u \le n$. For any real number u, let $M_2(u)$ denote the the linear hat function given by $M_2(u) = 1 - |u - 1|$. For n greater than 2, define $M_n(u)$ by the recursion

$$M_n(u) = \frac{u}{n-1} M_{n-1}(u) + \frac{n-u}{n-1} M_{n-1}(u-1).$$

The complex coefficients

$$\begin{aligned} b_\alpha(k_\alpha) &= \exp\left[2\pi i \left(\frac{(n-1)k_\alpha}{K_\alpha}\right)\right] \\ &\times \left\{\sum_{l_\alpha=0}^{n-2} M_n(l_\alpha + 1) \exp\left[2\pi i \left(\frac{k_\alpha l_\alpha}{K_\alpha}\right)\right]\right\} \end{aligned} \tag{3.43}$$

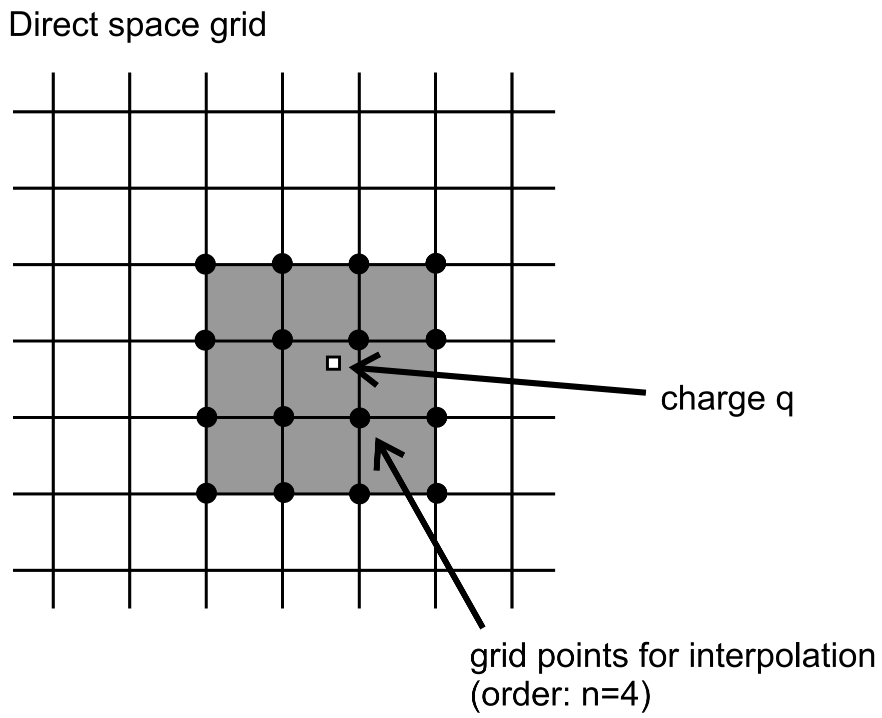


Figure 3.5: Example for an assignment of a point charge q to the real space mesh. The interpolation order is four. In the three-dimensional case the charge would be distributed over 64 mesh sites.

are independent of the charge coordinates $u_{i\alpha}$ and can therefore be calculated once at the beginning of a simulation. By inserting Eq. 3.42 in Eq. 3.40 we can approximate the structure factor $S(\mathbf{k})$ by

$$S(\mathbf{k}) \approx \tilde{S}(\mathbf{k}) = b_x(k_x)\, b_y(k_y)\, b_z(k_z)\, \mathcal{F}\left[Q\right](k_x, k_y, k_z), \tag{3.44}$$

where the array Q is given by

$$Q(m_x, m_y, m_z) = \sum_i q_i\, M_n(u_{ix} - m_x) M_n(u_{iy} - m_y) M_n(u_{iz} - m_z) \tag{3.45}$$

and $\mathcal{F}\left[Q\right](k_x, k_y, k_z)$ stands for the discrete Fourier transform at the grid point (k_x, k_y, k_z) of the array Q

$$\begin{aligned} \mathcal{F}[Q](k_x, k_y, k_z) &= \sum_{m_x=0}^{K_x-1} \sum_{m_y=0}^{K_y-1} \sum_{m_z=0}^{K_z-1} Q(m_x, m_y, m_z) \\ &\quad \times \exp\left[2\pi i \left(\frac{k_x m_x}{K_x} + \frac{k_y m_y}{K_y} + \frac{k_z m_z}{K_z}\right)\right] . \end{aligned}$$

Inserting the approximated structure factor $\tilde{S}(\mathbf{k})$ into Eq. 3.33 and using the fact that

$$\mathcal{F}[Q](-k_x, -k_y, -k_z) = K_x K_y K_z \mathcal{F}^{-1}[Q](k_x, k_y, k_z),$$

the approximate reciprocal lattice energy can be written as

$$\begin{aligned}
E^{\mathrm{rec}}_{\mathrm{Coul}} &\approx \frac{1}{2}\sum_{k_x=0}^{K_x-1}\sum_{k_y=0}^{K_y-1}\sum_{k_z=0}^{K_z-1} B(k_x,k_y,k_z)\, C(k_x,k_y,k_z)\, |\mathcal{F}[Q](k_x,k_y,k_z)|^2 \\
&= \frac{1}{2}\sum_{k_x=0}^{K_x-1}\sum_{k_y=0}^{K_y-1}\sum_{k_z=0}^{K_z-1} \mathcal{F}^{-1}[\Theta_{\mathrm{rec}}](k_x,k_y,k_z) \\
&\quad \times \mathcal{F}[Q](k_x,k_y,k_z)\, K_xK_yK_z\, \mathcal{F}^{-1}[Q](k_x,k_y,k_z) \\
&= \frac{1}{2}\sum_{k_x=0}^{K_x-1}\sum_{k_y=0}^{K_y-1}\sum_{k_z=0}^{K_z-1} Q(k_x,k_y,k_z)(\Theta_{\mathrm{rec}}\otimes Q)(k_x,k_y,k_z) \qquad (3.47)
\end{aligned}$$

with

$$\begin{aligned}
B(k_x,k_y,k_z) &= |b_x(k_x)|^2\, |b_y(k_y)|^2\, |b_z(k_z)|^2 \\
C(k_x,k_y,k_z) &= \frac{1}{\epsilon_0 V k^2}\exp\left(-\frac{k^2}{4\alpha^2}\right) \\
\Theta_{\mathrm{rec}} &= \mathcal{F}[BC],
\end{aligned}$$

where $\otimes$ denotes the convolution product. Regarding the fact that $M_n(u)$ is $2-n$ times differentiable (consider $n > 2$ for practical applications) and Θ_{rec} does not depend on the particle positions, the forces on the particle positions can be obtained straightforwardly

$$\begin{aligned}
(\mathbf{f}^{\mathrm{rec}}_i)_\alpha &= \frac{\partial E^{\mathrm{rec}}}{\partial(\mathbf{r}_i)_\alpha} = \sum_{k_x=0}^{K_x-1}\sum_{k_y=0}^{K_y-1}\sum_{k_z=0}^{K_z-1} \frac{\partial Q(k_x,k_y,k_z)}{\partial(\mathbf{r}_i)_\alpha} \\
&\quad \times(\Theta_{\mathrm{rec}}\otimes Q)(k_x,k_y,k_z). \qquad (3.48)
\end{aligned}$$

The calculation at every time-step is now carried out in the following way:

1. The grid scaled coordinates $u_{i\alpha}$ are calculated and the array Q is filled according to Eq. 3.42. This step is referred to as *charge assignment*. M_n and its derivatives are also obtained and stored.

2. $\mathcal{F}[Q]$ is calculated via an inverse 3DFFT. The array containing Q is overwritten by this procedure.

3. Using the transformed Q array as well as B, the approximate expression for $E^{\mathrm{rec}}_{\mathrm{Coul}}$ is calculated using Eq. 3.47. At the same time the transformed Q array is overwritten by the product of itself and with the arrays B and C.

4. This new array is then transformed in place by the forward 3DFFT to arrive at the convolution $\Theta_{\mathrm{rec}}\otimes Q$. Finally, the forces are computed using the previously stored derivatives of the M_n functions to recast $\partial Q/\partial(\mathbf{r}_i)_\alpha$.

Note that the smooth PME algorithm conserves energy, *but not* momentum [12]. Therefore, the total electrostatic forces are not zero, but instead a random quantity of the order of the rms error in the force. This leads typically to a Brownian motion of the system's center of mass and can lead to the occurrence of *box flow* (see section 3.6).

3.6 Temperature

In the canonical ensemble the total temperature is constant. In the microcanonical ensemble, however, it will fluctuate. The temperature T of a system is related to the kinetic energy by

$$T \equiv \frac{2}{k_B\,(3N - N_c - 3)} \sum_{i=1}^{N} \frac{1}{2}\, m_i\, |\boldsymbol{v}_i|^2 . \tag{3.49}$$

$\boldsymbol{v}_i$ are the velocities of the N particles in the system, N_c is the total number of constraints applied. Three degrees of freedom have to be subtracted because the system's total linear momentum is always considered to be zero. If this assumption is violated, because due to numerical instabilities *box–flow* has occurred, the total linear momentum has some unspecified value. Note, that this will leave less and less energy in the internal degrees of freedom. So, this will cause the system to transform into a very fast translating amorphous solid. This undesirable situation can be avoided by consequently monitoring the total linear momentum and by removing it if the box–flow phenomena has occurred.

3.7 Pressure

A correct pressure and pressure tensor evaluation is important for a wide variety of applications. It can be directly obtained from the virial. *MOSCITO 4* makes use of a molecular pressure tensor description

$$VP_{\alpha\beta} = \sum_n m_n\,(\boldsymbol{v}_n)_\alpha\,(\boldsymbol{v}_n)_\beta \tag{3.50}$$

$$+\sum_n \sum_\kappa \sum_{m>n} \sum_\lambda (\boldsymbol{r}_{nm})_\alpha\,(\boldsymbol{f}_{n\kappa m\lambda})_\beta , \tag{3.51}$$

where V is the volume of the system, m_n and $\boldsymbol{v}_n$ are the molecular mass and velocity of the center of mass, respectively. $\boldsymbol{r}_{nm}$ is the vector between the center of mass of molecules n and m, and $\boldsymbol{f}_{n\kappa m\lambda}$ is the force between atom κ in molecule n and atom λ in molecule m. The constraint forces and the intermolecular interactions make no contribution to the molecular pressure tensor. In the case of the the electrostatic interactions treated with the Ewald sum, there are two contributions to the potential, one in the real space (which is pairwise additive) and another in the reciprocal space (which is not). The appropriate expression according to

[13, 14] is

$$
\begin{aligned}
VP_{\alpha\beta} = {} & \sum_{n}\sum_{\kappa}\sum_{m>n}\sum_{\lambda} (\mathbf{r}_{nm})_{\alpha} \left(\frac{\mathbf{r}_{n\kappa m\lambda}}{r_{n\kappa m\lambda}}\right)_{\beta} \\
& \underbrace{\times\ 24 \left(\frac{\epsilon_{n\kappa m\lambda}}{\sigma_{n\kappa m\lambda}}\right) \left[2\left(\frac{\sigma_{n\kappa m\lambda}}{r_{n\kappa m\lambda}}\right)^{13} - \left(\frac{\sigma_{n\kappa m\lambda}}{r_{n\kappa m\lambda}}\right)^{7} \right]}_{\text{Lennard-Jones term}} \\
& + \frac{1}{4\pi\epsilon_0}\sum_{n}\sum_{\kappa}\sum_{m>n}\sum_{\lambda} \frac{(\mathbf{r}_{nm})_{\alpha}(\mathbf{r}_{n\kappa m\lambda})_{\beta}}{r^3_{n\kappa m\lambda}} \\
& \underbrace{\times\ q_{n\kappa}\, q_{m\lambda} \left(\frac{2\alpha}{\sqrt{\pi}}\, r_{n\kappa m\lambda} \exp\left(-\alpha^2 r^2_{n\kappa m\lambda}\right) + \operatorname{erfc}\left(-\alpha\, r_{n\kappa m\lambda}\right) \right)}_{\text{real space term}} \\
& + \underbrace{\frac{1}{2\epsilon_0 V}\sum_{\mathbf{k}\neq 0}\frac{1}{k^2}\exp\left(-\frac{k^2}{4\alpha^2}\right)|S(\mathbf{k})|^2 \left(\delta_{\alpha\beta} - \frac{2\mathbf{k}_{\alpha}\mathbf{k}_{\beta}}{k^2} - \frac{\mathbf{k}_{\alpha}\mathbf{k}_{\beta}}{2\alpha^2}\right)}_{\text{reciprocal lattice term}} \\
& + \frac{1}{2\epsilon_0 V}\sum_{n}\sum_{\kappa}(\mathbf{r}_n - \mathbf{r}_{n\kappa})_{\beta}\ q_{n\kappa}\sum_{\mathbf{k}\neq 0}\frac{1}{k^2}\exp\left(-\frac{k^2}{4\alpha^2}\right) \\
& \underbrace{\times\ i\,\mathbf{k}_{\alpha}\left(S(\mathbf{k})\, e^{-i\mathbf{k}\cdot\mathbf{r}_{n\kappa}} - S(-\mathbf{k})\, e^{i\mathbf{k}\cdot\mathbf{r}_{n\kappa}} \right)}_{\text{Correction to the reciprocal lattice term}} \quad . \qquad (3.52)
\end{aligned}
$$

If the PME method is applied, the last correction term can be obtained directly from the reciprocal lattice force components according to

$$VP_{\alpha\beta} = \ldots + \sum_{n=1}\sum_{\kappa=1}(\mathbf{r}_n - \mathbf{r}_{n\kappa})_{\beta}\,(\mathbf{f}^{rec}_{n\kappa})_{\alpha} \qquad (3.53)$$

3.8 Berendsen–Ensemble

Several methods for performing MD–simulations at constant temperature (NVT) and constant pressure/temperature (NPT) conditions have been proposed. *MOSCITO 4* uses a very simple (and *not* rigorous) method to achieve this: The *weak coupling* scheme according to Berendsen et al.[15]. The equations of motion are modified such that the system responds with a first order relaxation towards the preset reference values

$$
\begin{aligned}
\frac{d\,T(t)}{dt} &= \frac{1}{\tau_T}\left(T_0 - T(t)\right), \\
\frac{d\,p(t)}{dt} &= \frac{1}{\tau_p}\left(p_0 - p(t)\right).
\end{aligned}
$$

We will refer to the generated NVT and NPT ensembles as *Berendsen ensembles*.

Temperature: The temperature control is achieved by rescaling the velocities at each timestep t_n with a factor s_T according to

$$s_T = \Big(1 + K_T \, \Delta t \, (T_0/T(t_n) - 1)\Big)^{\frac{1}{2}} . \tag{3.54}$$

T_0 is the reference temperature and K_T determines the coupling to the external thermal reservoir. A K_T of 0 will result in a completely uncoupled simulation, while $K_T = \Delta t^{-1}$ will rescale the velocities completely to the desired temperature value. This *strongest* form of *weak coupling* will, of course, cause a strong perturbation on the trajectory, leading to an *isokinetic* ensemble. However, sometimes it is useful to apply this procedure to overcome highly strained initial conditions without introducing large amounts of kinetic energy into the system. K_T is related to τ_T by

$$K_T \;=\; \frac{2c_V^{df}}{k_B \tau_T}, \tag{3.55}$$

where c_V^{df} is the heat capacity per degree of freedom, which is not accurately known initially. However, $c_V^{df} = k_B/2$ serves as a practicable approximation. The value of K_T should be chosen sufficiently large to achieve the desired average temperature value, but on the other hand sufficiently small to avoid disturbance due to a coupling to the temperature reservoir. Considering aqueous model systems, *weak coupling* is normally well achieved at a τ_T of 0.5 ps.

Pressure: In analogy to the temperature coupling, the pressure is controlled by scaling the molecular centers of mass and the box dimensions by a factor

$$s_p = \Big(1 + K_p \, \Delta t \, (p(t_n) - p_0)\Big)^{\frac{1}{3}} , \tag{3.56}$$

where p_0 is the desired reference pressure. K_p defines the coupling strength. In contrast to the temperature scaling, where the strong coupling limit is leading to an isokinetic ensemble, strong pressure coupling will destroy the system. This will be due to the occurence of large forces, when particles "overlap" according to a strong position rescaling. K_p is related to τ_p by

$$K_p \;=\; \frac{\kappa_T}{\tau_p}, \tag{3.57}$$

where κ_T is the isothermal compressibility. Since the definition of the pressure depends on the kinetic energy, the pressure coupling should not be stronger than the temperature coupling

$$\tau_T \;\leq\; \tau_p. \tag{3.58}$$

MOSCITO 4 allows an anisotropic pressure scaling, where a separate scaling is performed for each box–dimension. A K_p of 0 will, of course switch off the coupling completely. Note, that pressure coupling can only be done *in combination* with temperature coupling, because no proper NPH ensemble is generated by this scheme.

4 Setting up a MD–Simulation

If you have successfully completed the installation procedure, everything is prepared for setting up a simulation run. Therefore you need *three* files containing the simulation parameters and the information about the system to be simulated. These have to reside in the present working directory.

- The PARAMETER–file, which has to be called `moscito.par` or has to be specified at the command line. It contains all informations specifying the simulation run like number of steps, temperature, etc..
- The SYSTEM–file (its name has to be specified in the PARAMETER–file or at the command line). It contains the complete force–field and the molecular topology information.
- The STRUCTURE–file, which has to be called `mosin.str` or `mos.structure` and which holds the Cartesian coordinates, forces and velocities of the complete system at a certain time step.

The PARAMETER– and SYSTEM–files are fully free format and you can introduce as many blank lines, spaces and comments as you like. (Everything that follows a '#' will be ignored). The file format is *not* case sensitive because all characters will always be converted to lower-case. The required syntax will be outlined in the following sections.

4.1 Simulation Control: PARAMETER–file

The PARAMETER –file contains the specifications which are necessary to handle a simulation run. They are introduced by a keyword followed by a number of possible options, which have to be separated by commata or blanks. Note that only one keyword per line is allowed. A typical PARAMETER–file is shown in figure 4.1. The following subsections explain the actions which correspond to the given keywords.

4.1.1 Forcefield declaration

The complete forcefield declaration is done in a separate file, the SYSTEM–file. Note, that the definition of a SYSTEM–file is essential for performing a simulation. If none is specified, *MOSCITO 4* will not start at all.

sysfile $1: This keyword specifies the filename $1 of the SYSTEM–file in the present directory. It contains the information concerning the molecular system, connectivity and force field (The outline of a system file is discussed in detail in section 4.2). Additionally, several analysing tools need information (e.g. explicit atom–atom pairs for which pair correlation functions shall be obtained), which also have to be defined in the SYSTEM–file. Note that $1 can only refer to filenames in lower-case!

4.1.2 Startup configuration

The startup configuration tells *MOSCITO 4* which structural and thermodynamical system information shall be used at the very beginning of a simulation run.

structurefile (velocities): The structurefile option invokes the program to read the complete system to be simulated from a STRUCTURE–file file named `mos.structure`. If, in addition, the keyword velocities is specified, the atomic velocities are also read from the STRUCTURE–file. Otherwise, *MOSCITO 4* will assume a random Maxwell–Boltzmann distribution corresponding to the temperature defined by the temperature keyword.

restart: This feature allows restarting a simulation run that has been terminated accidentally. It uses the information contained in the STRUCTURE–file and a separate checkpoint–file `mos.chk` (counters and buffers for average values are stored here). This procedure is meant to ensure continuity, if very long simulations have to be handled. (The automatic generation of these files is controlled by the restartdata command). Keep in mind that you need a STRUCTURE–file and `mos.chk`–file defining the *same* state of a simulation run! Note, without having activated structurefile velocities this option will have no effect. Normally this key will be rarely used.

stop momentum: This option will remove the total linear and angular momenta from the system before starting the simulation. This procedure is sometimes helpful, if, due to numerical instabilities *box–flow* has occurred or a single molecule shall be simulated *in vacuo*.

Note that it is, in principle, possible to start a simulation *without* having a STRUCTURE–file. The build–section, defined in the SYSTEM–file allows creation of a simple molecular crystal structure for a quick startup. Given this case, again a random Maxwell-Boltzmann distribution of velocities is generated.

4.1.3 Force calculation

This section deals with keywords, setting up the intermolecular force evaluation. Note, that there are two alternative neighborlist models available:

rcut $1: Radius of the cutoff–sphere for all nonbonded intermolecular site–site interactions (Lennard-Jones and screened Coulomb interactions). All pairs with distances larger than $1 will be neglected. Note, that for intramolecular interactions *no* cutoff is applied! Typical values for molecular systems are in the range of 8.0 Å to 9.0 Å. Since Ewald Summation is used for the electrostatic interactions and isotropic tail corrections are applied to correct the Lennard-Jones truncation, the exact value for the cutoff radius should not affect the simulation results (But keep in mind that it

```
#..........................Forcefield declaration
sysfile  spc-e.system
#..........................Startup configuration
structurefile velocities
#restart
stop  momentum
#..........................Force calculation
rcut              8.0
#neighborlist auto
neighborlist linkcell auto
rcutnb            9.5
#..........................SHAKE setup
shake             1.0e-4
#..........................Ewald summation setup
#ewald   grid newton
#ewald   grid fast
ewald   grid
alpha   5.37
#kspace 5 5 5 27
kspace pme 20 20 20 4
conserve
#..........................MD run specifications
timestep          1.0
steps             1000
#..........................Weak coupling control
temperature       300.0
pressure          0.1
scale  temperature  2.0e-3
scale  pressure     5.0e-7
#scale  pressure 5.0e-7 independently
#scale  stop        500
#..........................MD-Output setup
firststep     1
crddata      10
xtcdata      10
veldata      10
sysdata      10
logdata     100
restartdata 200
```

Figure 4.1: Example `moscito.par`–file controlling a small simulation run of a system of SPC/E water molecules defined in an already existing `mos.structure`–file. Coordinates and velocities are read. 1000 MD–steps are performed using the NPT–weak coupling scheme. The particle mesh Ewald method is employed, based on $20 \times 20 \times 20$ grid, applying 4'th order B–spline interpolation.

affects, of course, the simulation's performance).

Dimension of $1: Å

neighborlist ($1) (auto): This key invokes an application of a *Verlet*–type neighborlist, which will result in a considerable saving of computer time and is therefore highly recommended. All intermolecular pairs with distances smaller than rcutnb are considered here. The parameter $1 controls the update frequency of the neighborlist (A value

$1 = 10 for example will cause the list to be updated statically every 10'th time-step). Alternatively, the neighborlist auto keyword can be used. This feature ensures that the neighborlist is updated automatically, if one site could have crossed the distance rcutnb minus rcut with respect to any other particle (this is the safest option, so far).

neighborlist linkcell ($1) (auto): Using the linkcell option, the *Verlet*–type neighborlist is constructed via a more sophisticated linkcell algorithm. The computional cost of this approach scales linearly with the number of particles $O(N)$ (Instead of $O(N^2)$ for the conventional approach) and is significantly faster even for *moderately large* systems (>2000 interaction sites). However, due to a minor overhead, the simple approach is superior for small systems.

rcutnb $1: Radius of the cutoff–sphere to determine the intermolecular atom–atom pairs which will be considered in the neighborlist. It should be typically 1.5 Å to 2.0 Å larger than the cutoff–radius rcut for the site–site interactions. However, rcutnb has to be *smaller* than a half of the smallest box–length! Note, that the difference between rcut and rcutnb affects significantly the update frequency, when the neighborlist is updated automatically. An optimum performance is achieved in most cases when rcutnb is selected according to the given interval.

Dimension of $1: Å

4.1.4 SHAKE setup

shake $1: The parameter $1 defines the relative tolerance for the position restraining allowed by the SHAKE–procedure (in terms or constrained bond lengths). Taking into account that this value may influence the obtained system trajectory in a systematic way, a tight criterion is advised for MD–simulations. A value of 10^{-4} is therefore recommended. However, if due to an unfavourable starting geometry, SHAKE does not converge, a larger value may be used for an initial small period of time (Alternatively, you have always the opportunity to switch to a smaller time-step).

Dimension of $1: dimensionless

4.1.5 Ewald summation setup

The concept of Ewald summation is an intrinsic part of the *MOSCITO 4* MD–code. So, there is no alternative method available and the Ewald set of keywords must be specified!

ewald (grid) (grid newton) (grid fast): Specifies the procedure for the real–space part of the Ewald–sum. If no additional keyword is specified, the errorfunction is estimated via a polynomial approximation [16]. Alternatively, the ewald grid keyword enables a linear interpolation of potential and force from values stored in "lookup–tables". This procedure is the fastest available. A slightly more exact approximation is possible by using the Newton–Gregory forward difference interpolation scheme [1] activated by ewald grid newton. In both cases the interpolation is done in r^2–space. The accuracy achieved by ewald grid is usually fully sufficient and therefore recommended. A third option is given by ewald grid fast keyword. In this case the number of minimum image translations (which are rather CPU time–consuming) is minimimzed. This is based on the experience that the atom–atom minimum–image translations between two small molecules in a large system will allways be the same. However, a check will not *not performed* and this procedure may lead to wrong results if the assumption fails! Since this method has been implemented in assembler for Intel architectures it is by far the fastest available. But, it should be treated with care.

alpha $1 : The parameter $1 (denoted as α') determines the *relative* width of gaussian charge distribution. It is significantly responsible to balance real space and reciprocal lattice part of the total Ewald summation. The Ewald convergence parameter α in eq. 3.39 is related to α' by

$$\alpha = \frac{\alpha'}{2\,r_{cut}} \,, \tag{4.1}$$

where r_{cut} is the cutoff–radius defined by the key rcut. This procedure was chosen, since it guarantees suffiently small values at the cutoff–distance and enables simulations with fixed accuracy. A value of $\alpha' = 5.37$ leads to a reasonable accuracy [2] ϵ with

$$\epsilon \approx \frac{4}{\alpha'^2} \exp\left(-\alpha'^2/4\right) \tag{4.2}$$

of $1.0\ 10^{-4}$ in the real space sum. Note, that a proper setup of the reciprocal lattice sum requires a consideration of the chosen values for α' and r_{cut}!

Dimension of $1: dimensionless

kspace $1 $2 $3 $4 : This keyword activates the conventional Ewald technique to calculate the reciprocal lattice term. The first three parameters ($1–$3) specify the maximum integer values $l_{max}, m_{max}, n_{max}$ for the considered reciprocal lattice vectors $\mathbf{k}_{max} = 2\pi(l_{max}/L_x, n_{max}/L_x, m_{max}/L_z)$, where L_x, L_y, L_z are the box lengths. Assuming, that the reciprocal lattice sum should exhibit the same accuracy as the real space term, these values can be easily determined according to [2]

$$(l_{max}, m_{max}, n_{max}) = \left\lceil \frac{\alpha'^2}{4\pi\, r_{cut}} \,(L_x, L_y, L_z) \right\rceil \tag{4.3}$$

The last parameter $4 (denoted as n^2_{cut}) specifies the squared reciprocal lattice vector cutoff according to

$$k^2_{cut} = 4\pi^2 n^2_{cut} / \min\left\{L^2_x, L^2_y, L^2_z\right\}. \tag{4.4}$$

Of course, n^2_{cut} should be chosen such that cutoff sphere fits optimal into the reciprocal lattice determined by $(l_{max}, m_{max}, n_{max})$.

Example:

The squared reciprocal integer cutoff $n^2_{cut} = 27$ is optimal, if your reciprocal lattice vectors are defined by $(l_{max}, m_{max}, n_{max}) = (5, 5, 5)$. But, consider that also $n^2_{cut} = 27$, if you have a system which is elongated by factor of two in z–direction and therefore $(l_{max}, m_{max}, n_{max}) = (5, 5, 10)$!

kspace pme $1 $2 $3 $4: This option invokes the application of the PME–method for the reciprocal lattice term. The first three parameters ($1–$3) specify the number of grid points in the x–, y– and z–direction, respectively. $4 determines the spline–interpolation order, which has to be in the range between 4 and 10. Given a 4'th order interpolation, a spacing between mesh sites of 1.0 Å to 1.2 Å is advised [12]. This setup will lead to an accuracy comparing well with standard Ewald summation.

conserve: The smooth particle mesh Ewald algorithm *does not* conserve momentum! Consequently the the total net force is of the order the rms–fluctuation in the force [12]. When choosing low accuracy for the force estimation (which is in most cases a rather reasonable approximation), the resulting Brownian motion of the total momentum is unacceptably high and leads to a flux increasing approximately proportional to squareroot of the simulation length. To avoid this, the conserve–keyword invokes the total net force to be subtracted every time–step. This leads to stable simulations even at low PME–accuracy.

4.1.6 MD run specifications

timestep $1: Defines the time-step Δt between two configurations. The value of $1 is typically in the range between 0.5 fs and 2.0 fs . If dealing with completely rigid molecules like SPC/E–water, or flexible molecules, where all bonds are constrained, a time-step of 2.0 fs is appropriate. If only bonds to hydrogen atoms are fixed, a time-step of 1.0 fs is advised. For totally flexible molecules the time-step should not exceed 0.5 fs [17] .
Dimension of $1: fs

steps $1: Specifies the number of time-steps the simulation will extend to. This integer has to be multiplied by the actual time-step Δt to get the total simulation time.

4.1.7 Weak coupling control

This section deals with simulations of NPT– and NVT–ensembles realized by the weak coupling method.

temperature $1: This keyword determines the desired average temperature T_0. This essential key must be specified!
Dimension of $1: K

pressure $1: This keyword determines the desired average pressure p_0. This essential key must be specified!
Dimension of $1: MPa

scale temperature $1: Determines the coupling to the thermal–reservoir. $1 has to be specified in terms of $\Delta t\, K_T$. Note that according to this formulation, a change of the integration time-step will affect the coupling strength. However, a value of "1" will result in total velocity rescaling. A coupling of "0" will lead to a totally uncoupled simulation. This is the default value if this keyword is not specified.

Example:
Given a coupling constant of $2.0\ 10^{-3}$, a heat capacity per degree of freedom of

$k_B/2$ and a time-step of 1.0 fs, the temperature relaxation time τ_T is 0.5 ps.

Dimension of $1: dimensionless

scale pressure $1: Determines the coupling to the pressure–reservoir. $1 has to be specified in terms of $\Delta t\, K_p$. A coupling of "0" will switch off coupling completely. This is the default value.

Example:

Consider a simulation of an aqueous model–system with a coupling constant of $5.0\ 10^{-7}$ MPa^{-1}. Given the isothermal compressibility for water at 293 K and 1 atm of $\kappa_T = 45.91\, 10^{-6}$ bar^{-1} [18] and a time-step of 1.0 fs, we obtain a reasonable pressure relaxation time τ_p of 1.09 ps.

Dimension of $1: MPa^{-1}

scale pressure $1 independently : Defines the pressure–coupling the same way as above. But, box–dimensions relax individually due to the corresponding pressure tensor components.

Dimension of $1: MPa^{-1}

scale stop $1: This keyword enables a switch–off of any scaling at a certain time-step $1. If this key is disabled, the scaling persists until the simulation stops.

4.1.8 MD-Output control

The aim of a typical MD–simulation is to create a system–trajectory representing the system at equilibrium. *MOSCITO 4* allows one to store several aspects concerning this trajectory in different files. The specific file–formats will be discussed in detail in section 4.3. This section deals with keywords controlling the simulation output:

firststep $1: Specifies the time-step at which any trajectory output will be produced. The intention of this is to enable an initial equilibration period where no data is aquired. But, typically one will perform a separate equilibration run, so this value will be mainly set to '1'.

crddata $1: This option invokes the configurational part of the system–trajectory to be written to an unformatted file named `mos.crd` (an alternative filename can be specified at the command line). It contains the xyz–coordinates of all atomic sites as well as the box–dimensions. The parameter $1 specifies the output frequency (e.g. A "1" means that every configuration is written, and so forth. . .).

xtcdata $1: This option invokes the configurational part of the system–trajectory to be written to an unformatted file named `mos.xtc` (an alternative filename can be specified at the command line). It contains the xyz–coordinates of all atomic sites as well as the box–dimensions. The parameter $1 specifies the output frequency (e.g. A "1" means that every configuration is written, and so forth. . .).

veldata $1: This option invokes the momentum part of the system–trajectory to be written to an unformatted file named `mos.vel` (an alternative filename can be specified at the command line). It contains the components of all atomic velocities. The parameter $1 specifies the output frequency.

sysdata $1: This key invokes the thermodynamic data (e.g. Temperature, pressure, box-dimensions, potential energy…) to be written to a formatted file called `mos.data` (an alternative filename can be specified at the command line). The parameter `$1` specifies the output frequency.

logdata $1: A LOG–file named `mos.log` (an alternative filename can be specified at the command line) is created every time the *MOSCITO 4* program is started. It contains all informations regarding the present simulation run (Definition of the forcefield, simulation parameters…). Additionally, with the logdata key it is possible to generate a protocol of rolling averages of the same properties as written to the `mos.data`–file. `$1` characterises the length of every sampling period.

restartdata $1: Every `$1`'th step *MOSCITO 4* writes a formatted file called `mos.structure_out` or `mosout.str` (an alternative filename can be specified at the command line), which contains the complete configuration. Additionally, all counters and buffers are saved in `mos.chk` to enable a smooth restart of a simulation.

4.2 Forcefield Definition: SYSTEM–file

The SYSTEM–file contains all information about the force field and the specific molecular setup. It is generally organised in closed environments and commands slightly resembling to LaTeX–coding. There is only *one* command per line allowed. Each line may contain up to 256 characters and up to 20 *words*. A *word* is a string of characters, which can be separated by blanks, commas and "=". An environment is defined by a region limited by a `begin{`*environment–name*`}` and an `end{`*environment–name*`}` command. Environments are also referred to as "sections" and "subsections". The overall structure of a SYSTEM–file can be divided in three different parts or "chapters", which have to be defined in the given order:

1. The *header–chapter*, where interaction sites are defined by their mass, charge and Lennard-Jones parameters.

2. The *molecule–chapter*, where molecular entities are formed out of the above defined interaction sites.

3. the *footer–chapter*, which can be used for quite different things for analysing purposes. With respect to plain simulation the most prominent representative of a *footer–chapter* will be the *build–section*.

A typical SYSTEM–file containing all three mentioned chapters is shown in figure 4.2. The following text deals with a detailed explanation of what can be done.

4.2.1 Header–chapter

The header–chapter has always to be the first chapter in a SYSTEM–file. It contains the definition of interaction sites

```
#----------------------------------------------------------------------
#    MOSCITO  SYSTEM-file for SPC/E water
#    source: H.J.C. Berendsen, J.R. Grigera and T.P. Straatsma,
#            J. Phys. Chem. 91, 6269 (1987).
#----------------------------------------------------------------------

#------------------- header chapter -----------------------------------

begin{sites}                      #  charge (e)   mass (g/mol)
   OW     -0.8476      16.0
   HW      0.4238       1.0
end{sites}

begin{lorentz_lj}                 #  sigma (A)    epsilon (K)
   OW      3.1656      78.2
   HW      0.0          0.0
end{lorentz_lj}

#------------------- molecule chapter ---------------------------------

begin{molecule}

   label     spc-e

   begin{configuration}                #  site     x  y   z  (A)
      HW   0.0       0.0       0.0     #   1
      OW   1.0       0.0       0.0     #   2
      HW   1.3333    0.9428    0.0     #   3
   end{configuration}

   begin{constraints}      #     Distance constraints   (A)
      1  2     1.0
      2  3     1.0
      1  3     1.6313
   end{constraints}

   begin{exclude}
      all
   end{exclude}

end{molecule}

#-------------------- footer chapter ----------------------------------

begin{build}
   ecell         3.10635  3.10635  3.10635
   duplicate     5 5 5
   frac          spc-e   0.0   0.0   0.0
end{build}
```

Figure 4.2: Example SYSTEM–file to define the empirical SPC/E–model for water.

The site–environment

The `site`–environment defines interaction site–types. Mass and charge are assigned to a certain site–type. The outline of this section has always to be the following:

```
begin{sites}
     $1     $2     $3
            ⋮
end{sites}
```

Variable $1 defines the site–type by a string which can be up to 10 characters long. $2 specifies the site–charge (in |e|) and $2 defines the site–mass (in $\mathrm{g\,mol^{-1}}$). The mass of a site–type can be set to zero if *virtual* sites shall be considered (see the molecule–section for details). But, keep in mind that this feature should be treated with care.

Defining Lennard-Jones interactions

The Coulomb interaction between site–types has already been determined in the sites–section. The nonbonded Lennard-Jones type interaction has to be specified in a separate section, because different combination rules may be applied.

The most general approach is to define any pair interaction explicitly. This can be done by the `lj`–environment

```
begin{lj}
     $1     $2     $3     $4
                ⋮
end{lj} ,
```

where $1 and $2 define the explicit pair of interaction sites, $3 defines the Lennard-Jones σ_{ij} (in Å) and $4 specifies the Lennard-Jones ϵ_{ij} (in K). Due to the fact, that the number of terms can become quite large if the number of site–types increases, it is often useful to let the simulation program calculate the cross–terms directly. Therefore, *MOSCITO 4* provides the following two schemes to determine cross terms:

Lorentz-Berthelot mixing rules: The widely applied Lorentz–Berthelot rules use a geometrical mean for the ϵ_{ij} and a arithmetic mean for the σ_{ij}. This scheme can be activated by using the `lorentz_lj`–environment

```
begin{lorentz_lj}
     $1     $2     $3
            ⋮
end{lorentz_lj} ,
```

where only the Lennard-Jones parameters for identical pairs of type $1 have to be specified. $2 defines the Lennard-Jones σ_{ii} (in Å) and $3 specifies the Lennard-Jones ϵ_{ii} (in K).

OPLS mixing rules: Alternatively, some forcefield–concepts (like the classical OPLS–forcefield) are based on estimating a geometric mean for both, σ_{ij} and ϵ_{ij}. This can be realized by the `opls_lj`–environment

```
begin{opls_lj}
     $1     $2     $3
            ⋮
end{opls_lj} ,
```

where there parameters $1–$3 are defined the same way as in the `lorentz_lj`–environment.

Sometimes a molecular pair potential is generally based on one of the two mentioned concepts, but a few specific terms have to be modified. This can be easily realized in *MOSCITO*

4 by first using the `opls_lj`– or `lorentz_lj`–environment in combination with a subsequent `lj`–section, where these pairs are defined explictly.

Often (for example in SPC/E–water in figure 4.2), specific sites shall not exhibit a Lennard-Jones interaction at all. This can be realized by setting *both* parameters σ_{ii} and ϵ_{ii} to zero. All corresponding cross–terms will also be set to zero.

4.2.2 Molecule–chapter

In the molecule chapter a certain molecule-type is defined by a `molecule`–environment. In principle, there is no limitation of different molecule–types in a simulation. Of course, these molecules have to consist of sites which have been defined before in the *header–chapter*. The molecular properties are defined by a number of sub–environments (or "subsections") and commands, which will be discussed in the following sections

The label–command

Like the mentioned atom–types, any molecule–type has to be specified by a label. The `label`–command has to be used to identify molecular type by a string which can be up to 10 characters long.

```
    label     $1
```

The `label`–command can be placed anywhere in a `molecule`–environment. (But of course, not in one of the following mentioned sub–environments.)

The freedom–command

The `freedom`–command can be used to specify the degrees of freedom of a molecular type *by hand*.

```
    freedom     $1
```

Normally, the degrees of freedom are calculated automatically as $3N - N_{con}$, where N is the number of atoms in a molecule and N_{con} is the number of constraints applied. However, in some (actually, very rare) cases it is useful to define more constraints than would be necessary, to achieve a better convergence behaviour of the SHAKE procedure (an example for this might be a *rigid* CCl_4 model, where all C–Cl bonds and Cl–Cl distances have to be constrained). In such cases the number of degrees of freedom can be set explicitly. The `freedom`–command can be placed anywhere in a `molecule`–environment as well .

The configuration–environment

The `configuration`–environment is perhaps the most important of the discussed sub–environments

```
    begin{configuration}
         $1     $2     $3     $4
                   ⋮
    end{configuration} .
```

Note that any `molecule`–section *has* to contain a `configuration`–environment! It *always* has to be the first sub–environment in a `molecule`–section because all other environments make use of the order of sites which is declared in this section. So, all atoms in a molecule–type will be identified by their position in the `configuration`–environment. The first parameter $1 specifies the atom–type and the following three parameters ($2–$4) define Cartesian coordinates (in Å) of this site.

Applying constraints: The constraints–environment

The definition of distance–constraints within a molecule can be done by the `constraints`–environment

```
begin{constraints}
     $1     $2     $3
            ⋮
end{constraints} .
```

All distances between the atom pairs ($1,$2) will be kept fixed during a simulation run by application of SHAKE. As mentioned above, the parameters $1 and $2 refer to the position of a site in the `configuration`–environment (e.g. a "3" will refer to the third atom,...). The parameter $3 defines the desired constraint–distance in Å. If no parameter $3 is given or it is specified as zero, the distance will be calculated from the coordinates defined in the `configuration`–environment.

Switch off explicit interaction pairs: The exclude–environment

In specific cases it is necessary to switch off intramolecular nonbonded interactions explicitly. This has to be done normally for atoms involved in bonds (1–2 interactions) and bond–bending interactions (1–3 interactions). *MOSCITO 4* allows switching off *any* intramolecular interaction pair explcitily without significant loss of performance. For this purpose these pairs have to be declared in an `exclude`–environment

```
begin{exclude}
     $1     $2
            ⋮
end{exclude} .
```

The specified pairs do not contribute to the nonbonded intermolecular potential any longer. However, it is possible to add again a *modified* nonbonded interaction for the pair ($1, $1), using the `nonbonded`–environment discussed below. A special shorthand form of the `exclude`–environment can be activated by applying the `all`–keyword

```
begin{exclude}
     all
end{exclude} .
```

This key will switch off all intramolecular interactions in the given molecule. This feature has already been used in the SPC/E SYSTEM–file shown in figure 4.2.

Defining virtual sites: The virtual–environment

MOSCITO 4 allows treatment of potentials exhibiting massless, so-called *virtual* sites. Many simple models for water and ammonia use point charges defined at positions off from atoms. *MOSCITO 4* uses a scheme which distributes the forces while conserving the total force and torque on a rigid framework of real sites. Therefore the virtual site has to be given as a linear combination of the position of these sites

$$\mathbf{r}_{virt} = \sum_i c_i \, \mathbf{r}_i \qquad \text{with} \qquad \sum_i c_i = 1.$$

Note, that the coefficients c_i may also be negative. The forces are shifted the to the reference sites i according to

$$\mathbf{f}_i' = \mathbf{f}_i + c_i \mathbf{f}_{virt}.$$

This distribution–procedure is necessary, since the *MOSCITO 4* –code is entirely atomistic and therefore zero masses would result in infinite accelerations. In principle, there is no limitation of the number of framework–sites. Don't forget to check carefully that the defined virtual sites do exhibit zero masses. Otherwise, the forces will not be distributed correctly. A virtual site can be defined in a `virtual`–environment given by

```
begin{virtual}
     $1     ($2      $3)...
                  ⋮
end{virtual} .
```

The parameter $1 identifies the position of the virtual site in the `configuration`–environment. $2 specifies a real site i and $3 denotes the corresponding coefficient c_i. The bracket indicates that there can be, of course, more than one (at least, the minimum is two) reference–sites, which have to be added subsequently. Keep in mind, that any line in a `virtual`–environment is used to define a *new* virtual site!

In addition it is now possible to add as well out-of-plane site-definitions. This feature has been recently included since there are some models such as the TIP5P model for water where this is required. The syntax is the following:

```
begin{virtual}
     $1      0   $2      $3   $4      $5   $6      $7   $8
                  ⋮
end{virtual} .
```

The reference to site "0" indicates that this a definition for an out-of-plane site. $3, $5 and $7 specify the reference sites while $4, $6 and $8 denotes the corresponding coefficients c_1, c_2 and c_3 with $c_1 + c_2 + c_3 = 1$. Thevirtual site is defined according to

$$\mathbf{r}_{virt} = c_0 \left[(\mathbf{r}_2 - \mathbf{r}_1) \times (\mathbf{r}_3 - \mathbf{r}_1) \right] + \sum_{i=1}^{3} c_i \, \mathbf{r}_i \, .$$

Parameter $2 specifies c_o thus giving the distance from the plane defined by sites 1 2 and 3.

Defining bonds: The bond–environment

Within a `bond`–environment it is possible to define two distinct types of bonded pair interactions: A harmonic spring–bond and a Morse–type pair potential.

Harmonic bond: A harmonic spring–bond between two sites is defined by writing the label `harmonic` at the very beginning of each line.

```
begin{bond}
     harmonic  $1    $2    $3    $4
                 ⋮
end{bond} ,
```

The parameters $1–$4 are defined according to:

$$\mathcal{V}^{b}_{\kappa\lambda} = \frac{1}{2}\, k^{b}\, \left(r_{\kappa\lambda} - r^{0}\right)^{2}$$

$1, $2	κ, λ	
$3	k^{b}	kJ mol^{-1} Å^{-2}
$4	r^{0}	Å

k^{b} represents the force constant and r^{0} denotes the corresponding equilibrium distance for the intramolecular pair κ–λ.

Morse–type bond: An anharmonic Morse–type bond is defined in the same environment using a `morse`–keyword instead of `harmonic`.

```
begin{bond}
     morse   $1    $2    $3    $4    $5
                 ⋮
end{bond} ,
```

The parameters $1–$5 are defined according to:

$$\mathcal{V}^{bm}_{\kappa\lambda} = k^{bm}\, \Big(\left(1 - \exp\left(-\alpha^{bm}\left(r_{\kappa\lambda} - r^{0}\right)\right)\right)^{2} - 1 \Big)$$

$1, $2	κ, λ	
$3	k^{bm}	kJ mol^{-1}
$4	α^{bm}	Å^{-1}
$5	r^{0}	Å

k^{bm} denotes the dissociation–energy, r_{0} specifies the equilibrium distance and α^{bm} characterises the width of the potential well.

Defining bond–bending interactions: The angle–environment

Two types of harmonic bond–bending interactions can be considered. The first one is rather general, while the second is restricted to linear bond angles avoiding the occurrence of a singularity and therefore numerical instabilities at the 180° bond angle. Both types are defined in an `angle`–environment:

General approach: A normal harmonic bond–bending interaction is defined in an `angle`–environment according to

```
begin{angle}
     $1    $2    $3    $4    $5
                 ⋮
end{angle} ,
```

where the parameters $1–$5 are defined such that

$$\mathcal{V}^{a}_{\kappa\lambda\omega} = \tfrac{1}{2} k^{a} \left(\phi_{\kappa\lambda\omega} - \phi^{0}\right)^{2}$$

$1, $2, $3	κ, λ, ω	
$4	k^{a}	$\mathrm{kJ\,mol^{-1}\,rad^{-2}}$
$5	ϕ^{0}	degrees

k^{a} is the force constant and ϕ^{0} the equilibrium bond–angle. Note, that this setup defines a $(\kappa - \lambda - \omega)$ bond angle located at site λ.

Linear bond angles: Due to the limited numerical accuracy, the general scheme may lead to unstable trajectories when dealing with 180° bond angles. F. Müller–Plathe [8] showed that the ansatz presented below can be used to avoid the problem. To enable this option the bond–bending definition has to begin with a `linear`–keyword

```
begin{angle}
     linear   $1    $2    $3    $4
                 ⋮
end{angle} .
```

The parameters $1–$4 are defined according to:

$$\mathcal{V}^{al}_{\kappa\lambda\omega} = k^{al} \Big(1 + \cos\left(\phi_{\kappa\lambda\omega}\right)\Big)$$

$1, $2, $3	κ, λ, ω	
$4	k^{al}	$\mathrm{kJ\,mol^{-1}}$

k^{al} is the corresponding force constant. Its numerical value is identical to to value for the general representation (although the unit is different).

Defining dihedral–potentials: The dihedral–environment

For *MOSCITO 4* there are currently two forms of dihedral potentials available. Both have to be defined in a `dihedral`–section.

Proper dihedrals: using the `dihedral`–environment with a `proper`–keyword in front of each line

```
begin{dihedral}
    proper  $1    $2    $3    $4    ($5    $6    $7)
             ⋮
end{dihedral} ,
```

an anharmonic dihedral potential has to be defined in terms of a Fourier series with an, in principle, arbitrary number of cosine–terms. The number of terms specifying the interaction for the site quadruple ($\kappa - \lambda - \omega - \tau$), where ($\lambda - \omega$) represents the central bond, is therefore also technically not restricted

$$\mathcal{V}^{\mathrm{dp}}_{\kappa\lambda\omega\tau} = k^{\mathrm{dp}} \left(1 + \cos\left(m\,\psi_{\kappa\lambda\omega\tau} - \psi^0\right)\right)$$

$1, $2, $3, $4	$\kappa, \lambda, \omega, \tau$	
$5	k^{dp}	$\mathrm{kJ\,mol^{-1}}$
$6	m	integer
$7	ψ^0	degrees

k^{dp} is the force constant, m defines the multiplicity and ψ^0 the phase–angle of one term in the Fourier series. The brackets indicate that more than one set of parameters can be defined in one line (Actually, four sets are allowed). The number of terms, however, can be increased by defining more terms for the same quadruple in further lines.

Improper dihedrals: In this work we will consider *only* harmonic dihedral potentials as improper dihedrals[1] They are invoked by an `improper`–keyword according to

```
begin{dihedral}
    improper  $1    $2    $3    $4    $5    $6
             ⋮
end{dihedral} .
```

The parameters $1–$6 are defined the following way.

[1]There is some uncertainty about the term "improper dihedral". In the AMBER forcefield [5] for example improper dihedrals are defined topologically (which will simply be expressed by using a different sequence κ–λ–ω–τ). Using the AMBER forcefield it may thus happen that you have to define an improper dihedral potential as `proper` Fourier–type dihedral potential.

$$\mathcal{V}^{\mathrm{di}}_{\kappa\lambda\omega\tau} = \frac{1}{2} k^{\mathrm{di}} \left(\psi_{\kappa\lambda\omega\tau} - \psi^0\right)^2$$

\$1, \$2, \$3, \$4	κ, λ, ω, τ	
\$5	k^{di}	$\mathrm{kJ\,mol^{-1}\,rad^{-2}}$
\$6	ψ^0	degrees

k^{di} is the force constant and ψ^0 represents the equilibrium angle. The site–quadruple (κ, λ, ω, τ) is defined analogously to the previous section. But, in contrast to the `proper`–dihedral it wouldn't make any sense to define more than one term for one site–quadruple (κ, λ, ω, τ).

Modified nonbonded intramolecular interactions:
The nonbonbonded–environment

In almost all general purpose forcefields, the parameterisation of the 1–4–interactions is different from the general parameterisation[2]. However, *MOSCITO 4* enables *any* intramolecular interaction to be modified specifically without facing a significant loss of performance. In most cases, of course, 1–4 interactions will be the topic. This feature can be defined for a set of atom pairs (\$1,\$2) in a `nonbonded`–environment in the following form

```
begin{nonbonded}
     $1     $2     $3     $4     $5
                 ⋮
end{nonbonded} ,
```

where the variables are defined as

$$\mathcal{V}^{\mathrm{nb}}_{\kappa\lambda} = \frac{1}{4\pi\epsilon_0}\, f\, \frac{q_\kappa\, q_\lambda}{r_{\kappa\lambda}} + 4\,\epsilon \left\{ \left(\frac{\sigma}{r_{\kappa\lambda}}\right)^{12} - \left(\frac{\sigma}{r_{\kappa\lambda}}\right)^{6} \right\}$$

\$1, \$2	κ, λ	
\$3	σ	Å
\$4	ϵ	K
\$5	f	dimensionless

f is the Coulomb scaling factor and σ and ϵ define an explicit Lennnard-Jones interaction. Note, that the `nonbonded`–environment defines an *additional* interaction for an atom pair (κ, λ). If this interaction shall *replace* the original parameterisation (which will be desired in almost all cases), the pair (κ, λ) has also to be defined in the `exclude`–environment.

[2] In the AMBER forcefield [5], the 1–4 Coulomb interaction and the 1–4 Lennard-Jones interaction are scaled by factors of 0.8333 and 0.5, respectively.

```
begin{z-matrix}
    1
    2      1       .9600
    3      2      1.4100    1    108.5001
    4      3      1.0900    2    109.5000    1     60.0000
    5      3      1.0900    2    109.5001    1    180.0000
    6      3      1.0900    2    109.5000    1    -60.0000
zfreeze     2    2
zfreeze     3    2
zfreeze     4    2
zfreeze     5    2
zfreeze     6    2
end{z-matrix}
```

Figure 4.3: Example `z-matrix`–environment used to define the structure of methanol in internal coordinates. All bond lengths will be constrained to the given values. Such a representation can be used to perform a geometry optimisation in internal coordinates in order to refine potential parameters.

Internal coordinates: The z-matrix–environment

For potential development purposes it is often useful to express the molecular structure in internal coordinates. *MOSCITO 4* allows one to define a structure in terms of a molecular z-matrix. The outline of a `z-matrix`–environment is shown in figure 4.3. The first number identifies the centre *a* which is defined by its position in the `configuration`–environment. The numbers in the second column refers to a site *b*, which is separated from site *a* by a distance (in units of Å) which is given in the third column. The site *c* in the fourth column forms a bond angle *a*–*b*–*c* given in column five (in degrees). Column six finally specifies a dihedral angle with respect to site *d*, given in the last column (also in degrees). Note that the sites which will be referred to have to be defined above in the z-matrix. The `zfreeze`–keyword enables particular degrees of freedom to be constrained to the given values. The keyword has to be followed by two numbers: The first one refers to the line of the z–matrix and the second number identifies the degree of freedom (A ′2′ refers to the bond–distance, a ′3′ to the bond–angle and a ′4′ to the dihedral angle).

4.2.3 Footer–chapter

The purpose of this part of the SYSTEM–file is manyfold. In some sense it can be used for any purpose you can imagine, especially for the set of utility programs. However, for the *MOSCITO 4* –simulation program three different tasks can be defined through this part.

The build–environment

The `build`–environment has to be seen as a “quick and dirty” way to generate a rather simple “crystal”–structure of a molecular system. We have to emphasise that this part is by no means intended to produce real crystal structures. It is just meant to generate a structural precursor for MD–simulations of liquids.

```
begin{build}
     ecell    $1    $2    $3
     duplicate    $4    $2    $3
     frac    $1    $2    $3    $4
                 ⋮
end{build} ,
```

There are three different commands provided in a `build`–section:

- The `ecell`–command defines the size of a rectangular unit cell. The unit cell constants $1–$3 have to be given in Å.
- The `duplicate`–command defines how often this cell is replicated in each direction to form the entire system. The number of replications are specified by the integers $1–$3.
- The `frac`–command places a molecule, identified by its label $1 somewhere within the unit cell. It is moved by its centre of mass at a position in the cell defined by a set of fractional coordinates given by $1–$4. The orientation is directly taken from the corresponding `configuation`–environment.

Of course, there can be only one `ecell`– and `duplicate`–command, but an unlimited number of `frac`–commands.

Calculating electric field gradients: The efg–environment

MOSCITO 4 allows monitoring the electric field gradient (EFG) at the position of interaction sites. However, this feature should be used with care, since it is still in a more or less experimental stage. The EFG–tensor

$$\tilde{T}(\mathbf{r}_i) = \begin{pmatrix} V_{xx} & V_{xy} & V_{xz} \\ V_{yx} & V_{yy} & V_{yz} \\ V_{zx} & V_{zy} & V_{zz} \end{pmatrix} \qquad \text{with} \qquad V_{\alpha\beta} = \frac{\partial^2 \mathcal{V}^{el}(\mathbf{r}_i)}{\partial\alpha\,\partial\beta}$$

is calculated due to the surrounding point charges defined by all molecules whose centres of mass have a distance less than the cutoff radius with respect to the centre of interest. $\mathcal{V}^{el}(\mathbf{r}_i)$ denotes the electrostatic potential at the position $\mathbf{r}_i$. The electric field gradient fluctuation at the position of quadrupolar nuclei represents by far the most important contribution to the quadrupolar nuclear relaxation processes. So, simulations in combination with nuclear magnetic resonance experiments can be used to gain some insight into dynamical processes at the molecular level. Of, course this method can only be applied to centres with zero electric field gradient in the unperturbed state (like monovalent ions or noble gases).

The EFG–calculation setup is done in an `efg`–environment

```
begin{efg}
     $1    $2
                 ⋮
end{efg} ,
```

where $1 specifies the number of a molecule (not the number of the molecule type!) in the present simulation and $2 denotes the position of the site to monitored. (e.g. a combination like ’213 4’ will lead to a monitoring of site ’4’ of molecule number ’213’). It is possible, of course, to study more than one site at one time.

The six distinct tensor components are written every step to an unformatted file named `mos.EFG` in units of $\mathrm{V\,nm^{-2}}$ using the following write statement:

```
write(40) ((sngl(efg_ten(i,j)),j=1,6),i=1,n_efg)
```

The stored components per monitored site refer to $V_{xx}, V_{yy}, V_{zz}, V_{xy}, V_{xz}, V_{yz}$ in exactly the given order.

Introducing harmonic intermolecular interactions: The umbrella environment

In order to constrain the distance between selected molecules in a simulation it is sometimes useful to define a harmonic interaction between certain sites on distinct molecules. This feature can be seen as a simple way to introduce intermolecular umbrella sampling. Alternatively, one would have also the opportunity to define a "supermolecule" containing the definitions of both molecules in combination with an additional harmonic force. However, this is by far more complicated than the simple approach discussed here. The introduction of an *umbrella*–force can be achieved in an *umbrella* environment according to

```
begin{umbrella}
      $1    $2    $3    $4    $5    $6
                  ⋮
end{umbrella} ,
```

where the parameters $1–$6 are defined as

$$\mathcal{V}^{umb}_{n\kappa m\lambda} = \frac{1}{2}\,k^{umb}\,\left(r_{\kappa\lambda} - r^0\right)^2$$

$1, $2	Site κ on molecule n	
$3, $4	Site λ on molecule m	
$5	k^{umb}	$\mathrm{kJ\,mol^{-1}\,\AA^{-2}}$
$6	r^0	Å

$1 and $3 define the site-number and $2 and $4 refer to the position of the molecule in the actual simulation. The index quadruple $1–$4 defines the pair of sites between which a harmonic interaction will be established.

4.3 File–formats

This section deals with a detailed description of the file formats used to store all aspects concerning a simulation run.

4.3.1 STRUCTURE–file

The STRUCTURE–file is perhaps the most important file because it contains the complete description of the considered molecular system. It is tightly connected with the SYSTEM–file, because it refers to molecular and site definitions therein. Figure 4.4 shows the beginning of a

```
    1.972428    1.972428    1.972428
    256

spc-e
     1      3      1
hw            2
  .39353281E-01  .20456994E+01 -.26533587E-02
  .65106947E+00 -.43355057E+00 -.76593375E+00
 -.28045224E+02  .24366040E+03  .16124436E+03
ow            1
  .78211405E-01  .19564991E+01  .20441237E-01
  .59788496E+00 -.25661374E+00  .21100554E-01
 -.53825450E+03 -.47258256E+03 -.14353496E+04
hw            2
  .11928816E+00  .19609119E+01  .11150840E+00
  .11833571E+01  .70877146E+00 -.28225799E+00
  .23300606E+03  .13410653E+02  .58289987E+03

spc-e
     1      3      2
hw            2
  .87808494E-01  .30327984E+00  .27610312E+00
  .56717821E+00 -.24745483E+00  .58088539E+00
  .33250418E+03  .66080997E+03 -.62661757E+02
ow            1
  .51147165E-01  .22942869E+00  .21951730E+00
  .90989275E+00  .21248855E+00 -.25044968E+00
 -.50399382E+03 -.10836540E+04  .56457409E+03
hw            2
  .65719289E-01  .25135206E+00  .12304441E+00
  .12273978E+01  .11455106E+01  .41671700E-02
  .42281663E+03  .48262606E+03 -.30039610E+03
  .45605346E+03  .34020031E+03 -.27626695E+03
             .
             .
             .
```

Figure 4.4: Example STRUCTURE–file containing positions, velocities and forces of 256 SPC/E molecules.

STRUCTURE–file containing an ensemble of water molecules. The outline of the file is always the following: At first there is an initial section consisting of two lines: the first line contains the simulation cell dimensions (in nm), while the second specifies the number of molecules in a simulation N. What follows is a description of N molecules: The first line is empty or may contain a remark. The following line contains the molecule label (the label is only used for a better readability and is in some sense redundant). The third line in a molecule definition contains three numbers: The first number specifies the *molecule type* referring to the SYSTEM–file (A '1' will refer to the first `molecule`–environment, a '2' to the second an so forth...). The second gives the number of sites per molecule M and the third one is the actual number of the current molecule (the last term is again unimportant and is meant to make editing by hand a bit easier). Any molecule definition consist of M site definitions: The first line contains the site–label and the number of a site in the SYSTEM–file (Again, only the number is important), followed by three lines holding the Cartesian coordinates (in nm), the atomic velocities (in $\mathrm{nm\,ps^{-1}}$) and the atomic forces (in $\mathrm{kJ\,mol^{-1}\,nm^{-1}}$), respectively. Note that the file is processed list–directed, making it easier to modify a system *by hand*. However, due to the presence of tools which allow a rather flexible STRUCTURE–file handling this will only be rarely necessary.

4.3.2 LOG–file

The LOG-file `mos.log` is created every time the *MOSCITO 4* simulation program is started. It contains thoroughly documented information about what the simulation program does. The output is clearly written in a human–readable form. So, it is recommended to take a look at this file to check whether the simulation input was processed correctly. An example LOG–file resides in `./examples/SPCE_water/ref.log`. Additionally it contains rolling averages of all available thermodynamic system properties (the averaging period can be specified in `moscito.par`) and a detailed analysis of program timings.

4.3.3 DATA–file

The DATA–file contains thermodynamic data concerning a simulation run. It is a formatted file, where the parameters listed below are written periodically. Each time-step an additional line is written using `format(i9,26f16.6)` in following order:

	General:	
1	Actual time-step m	
2	Actual time-step $\Delta t\, m$	ps
3	Box-size L_x	nm
4	Box-size L_y	nm
5	Box-size L_z	nm
6	Temperature	K
7	Pressure	MPa
	Energy:	
8	Total energy	$\mathrm{kJ\,mol^{-1}}$
9	Kinetic energy	$\mathrm{kJ\,mol^{-1}}$
10	Potential energy	$\mathrm{kJ\,mol^{-1}}$
	Potential energy:	
11	Intermolecular interaction	$\mathrm{kJ\,mol^{-1}}$
12	Intramolecular interaction	$\mathrm{kJ\,mol^{-1}}$
	Intermolecular interaction:	
13	Lennard-Jones interaction	$\mathrm{kJ\,mol^{-1}}$
14	Coulomb interaction (real space)	$\mathrm{kJ\,mol^{-1}}$
15	Coulomb interaction (reciprocal lattice)	$\mathrm{kJ\,mol^{-1}}$
16	Coulomb interaction (self correction)	$\mathrm{kJ\,mol^{-1}}$
17	Coulomb interaction (molecular self correction)	$\mathrm{kJ\,mol^{-1}}$
	Intramolecular interaction:	
18	Harmonic bond	$\mathrm{kJ\,mol^{-1}}$
19	Morse bond	$\mathrm{kJ\,mol^{-1}}$
20	Harmonic bond bending	$\mathrm{kJ\,mol^{-1}}$
21	Linear bond bending	$\mathrm{kJ\,mol^{-1}}$
22	Improper dihedral	$\mathrm{kJ\,mol^{-1}}$
23	Proper dihedral	$\mathrm{kJ\,mol^{-1}}$

	Misc.:	
24	Umbrella interaction	$\mathrm{kJ\,mol^{-1}}$
25	Pressure (x-direction)	MPa
26	Pressure (y-direction)	MPa
27	Pressure (z-direction)	MPa

However, each file has an initial header-part where the given parameters are described and which is 29 lines long. Note that each energy value is given as "per molecule". So, the values have to be multiplied by the actual number of molecules to get the total energies.

4.3.4 CRD–file

The unformatted CRD–file holds all Cartesian coordinates. The first value in a CRD–file is a 4 byte integer, specifying the number of sites in the simulation (`natoms`). Each time-step, the x,y,z–coordinates and simulation box dimensions are written by

```
         write(20) (sngl(box(i)),i=1,3),
     .             (sngl(x(i)),i=1,natoms),
     .             (sngl(y(i)),i=1,natoms),
     .             (sngl(z(i)),i=1,natoms)
```

All coordinates box-lengths are given in nm. Note that the molecular trajectories are folded into the central box with respect to their centre of mass. This means, whenever a molecule's centre of mass leaves the central box on one side, the complete molecule is shifted to the opposite side. This procedure ensures that all intramolecular distances stay correct. However, the trajectories have to be unfolded again, when dynamical properties like translational diffusion coefficients are calculated.

4.3.5 XTC–file

The unformatted XTC-file format based on the XRDF-libraries written by F. Hoesel. The libraries are be used to store Cartesian coordinates more efficiently. Applying a reasonable real space resolution of about 1 pm and making use of correlations in the data this leads to substantially more compact coordinate files (About a factor of three compared to the standard CRD-format). Although a unformatted binary format, the XTC-files are machine independent. Moreover, several programs such as the VMD (Visual Molecular Dynamics) tool of the Theoretical Biophysics Group at the University of Illinois in Urbana Champaign (see `http://www.ks.uiuc.edu/Research/vmd/`) support the XTC-format. There is a conversion tool (*crd2xtc*) which interconverts both formats into each other.

4.3.6 VEL–file

The unformatted VEL–file holds all atomic velocities. The first value in a VEL–file is again the number of interaction sites. The velocities are written according to

```
         write(25) (sngl(vx(i)),i=1,natoms),
     .             (sngl(vy(i)),i=1,natoms),
     .             (sngl(vz(i)),i=1,natoms)
```

in units of $\mathrm{nm\,ps^{-1}}$.

5 MOSCITO Command Reference

This command reference gives an overview over the set of simulation and utility programs that come with the *MOSCITO 4* simulation package. Some programs require information which is not contained in the STRUCTURE-file only (e.g. the interaction parameters, bonding definitions, etc.). These programs require the additional presence of a moscito parameter-file and (specified therein) a SYSTEM-file. In any case the necessary files are always indicated. In most cases the data is read from ⟨STDIN⟩ and written to ⟨STDOUT⟩, so that a more complex operation can be performed by using a combination of commands by piping the data.

5.1 Running a MD Simulation

5.1.1 moscito

Needs: PARAMETER-file, SYSTEM-file, STRUCTURE-file or GROMOS-file
Output files: LOG-file, DATA-file, CRD-file, XTC-file, VEL-file, STRUCTURE-file GROMOS-file
Usage: *moscito -par #PARAMETER-file -sys #SYSTEM-file -in #STRUCTURE-file -out #STRUCTURE-file -gin #GROMOS-file -gout #GROMOS-file -crd #CRD-file -xtc #XTC-file -vel #VEL-file -dat #DATA-file -log #LOG-file -v #level -help*

Optional Parameters:

-v	:	*verbose, prints some info and step timings, #level=verbosity level (integer)*
-par	:	*specifies PARAMETER-file (default: moscito.par).*
-sys	:	*specifies SYSTEM-file.*
-in	:	*specifies input STRUCTURE-file (default: mosin.str) .*
-out	:	*specifies ouput STRUCTURE-file (default: mosout.str).*
-gin	:	*specifies input GROMOS-file.*
-gout	:	*specifies ouput GROMOS-file.*
-crd	:	*specifies coordinate CRD-file.*
-xtc	:	*specifies coordinate XTC-file.*
-vel	:	*specifies velocities VEL-file.*
-dat	:	*specifies DATA-file.*
-log	:	*specifies LOG-file.*
-help	:	*prints help text.*

Starts the MD simulation using a configuration in a STRUCTURE-file specified by -in or given in *mosin.str*. The forcefield is specified in the SYSTEM-file and the simulation control parameters have to be defined in a PARAMETER-file (default name is *moscito.par*). Intended for condensed phase simulations applying periodic boundary conditions.

5.1.2 moscito-net

Needs: PARAMETER-file, SYSTEM-file, STRUCTURE-file
Output files: LOG-file, DATA-file, CRD-file, VEL-file
Usage: *moscito-net -v*

Optional Parameters:

-v	:	*verbose, prints some info and step timings*

This is the parallel version of the moscito simulation program. Please note that the parallel moscito is still version 3. In addition, the executable resides in `./parallel/bin` instead of `./bin` as all other executables of the *MOSCITO4* distribution do. Since *moscito-net* is still version 3 it does not support the XTC-format nor GROMOS-in/out formats nor the extensive use of command line options. The PARAMETER-file has to be named `moscito.par` and the STRUCTURE-file has to be called `mos.structure`.

5.1.3 mosdrop

Needs: PARAMETER-file, SYSTEM-file, STRUCTURE-file or GROMOS-file
Output files: LOG-file, DATA-file, CRD-file, XTC-file, VEL-file, STRUCTURE-file GROMOS-file
Usage: *mosdrop -par #PARAMETER-file -sys #SYSTEM-file -in #STRUCTURE-file -out #STRUCTURE-file -gin #GROMOS-file -gout #GROMOS-file -crd #CRD-file -xtc #XTC-file -vel #VEL-file -dat #DATA-file -log #LOG-file -v #level -help*

Optional Parameters:

-v	:	*verbose, prints some info and step timings, #level=verbosity level (integer)*
-par	:	*specifies PARAMETER-file (default: moscito.par).*
-sys	:	*specifies SYSTEM-file.*
-in	:	*specifies input STRUCTURE-file (default: mosin.str) .*
-out	:	*specifies ouput STRUCTURE-file (default: mosout.str).*
-gin	:	*specifies input GROMOS-file.*
-gout	:	*specifies ouput GROMOS-file.*
-crd	:	*specifies coordinate CRD-file.*
-xtc	:	*specifies coordinate XTC-file.*
-vel	:	*specifies velocities VEL-file.*
-dat	:	*specifies DATA-file.*
-log	:	*specifies LOG-file.*
-help	:	*prints help text.*

This is a MOSCITO simulation program which does not apply periodic boundary conditions. It is basically intended for simulations of isolated molecules (ore *droplets* of few molecules) in the gas phase. Please note that there are some minor differences with respect to the normal MOSCITO simulation program: 1. There is, of course, no pressure scaling and definitions of box-lengths are neglected. 2. All N^2 intermolecular nonbonded interactions are considered. Consequently all Ewald/PME,cutoff, neighborlist keywords have no effect.

5.2 Basic STRUCTURE-file manipulation

5.2.1 center

Needs: STRUCTURE-file
Usage: *center bx by bz < input.structure > output.structure*

Creates a new simulation box around a molecular configuration contained in *input.structure*. The configuration is placed at the very center of a new box with dimensions bx, by and bz (in nm). The configuration is placed with respect to its geometrical center. Reads a moscito STRUCTURE-file from ⟨STDIN⟩ and writes output to ⟨STDOUT⟩.

5.2.2 changedens

Needs: STRUCTURE-file
Usage: *changedens s < input.structure > output.structure*

Creates a new configuration with a density (compared to that of *input.stucture*) scaled by a factor s. Note that only intermolecular distances (and the box-dimensions of course) are changed this way. Reads a moscito STRUCTURE-file from ⟨STDIN⟩ and writes output to ⟨STDOUT⟩.

5.2.3 delete

Needs: STRUCTURE-file
Usage: *delete #1 #2 ... #n < input.structure > output.structure*

Deletes molecules with number #1 #2 ... #n from *input.stucture*. Reads a moscito STRUCTURE-file from ⟨STDIN⟩ and writes output to ⟨STDOUT⟩.

5.2.4 duplicate

Needs: STRUCTURE-file
Usage: *duplicate nx ny nz < input.structure > output.structure*

Creates a new larger structure by duplicating the original structure. The original structure is duplicated (nx, ny, nz) times in x-, y- and z-direction. Reads a moscito STRUCTURE-file from ⟨STDIN⟩ and writes output to ⟨STDOUT⟩.

5.2.5 infostr

Needs: SYSTEM-file, STRUCTURE-file
Usage: *infostr -sys #SYSTEM-file < input.structure*

Necessary Parameters:			
	-sys	:	*specifies SYSTEM-file.*
Optional Parameters:			
	-e	:	*calculate electrostatic properties of the individual molecules.*
	-v	:	*verbose, prints some info.*
	-help	:	*prints help text.*

Reads a configuration and calculates reports some properties of the structure file such as density, number of molecule types, number of molecules, etc. When using the option -e, the program calculates the charge, dipole and quadrupole moment for any single molecule in the system. Reads a moscito STRUCTURE-file from ⟨STDIN⟩ and writes output to ⟨STDOUT⟩.

5.2.6 mirror

Needs: STRUCTURE-file
Usage: *mirror x < input.structure > output.structure*

Creates a new configuration that has been mirrored according to one of three possible mirror planes: A *x* specifies the y-z plane, a *y* specifies the x-z plane and a *z* specifies the x-y plane as mirror plane. Please note that the mirroring operation inverts the chiral center when applied to optically active molecules. Reads a moscito STRUCTURE-file from ⟨STDIN⟩ and writes data to ⟨STDOUT⟩.

5.2.7 molmove

Needs: STRUCTURE-file
Usage: *molmove n dx dy dz < input.structure > output.structure*

Translates molecule number *n* of the molecular configuration contained in *input.structure* according to vector (dx, dy, dz) (all in nm). Reads a moscito STRUCTURE-file from ⟨STDIN⟩ and writes data to ⟨STDOUT⟩.

5.2.8 rotate

Needs: STRUCTURE-file
Usage: *rotate ra rb rc < input.structure > output.structure*

Rotates a molecular configuration in three dimensions. The rotation is specified with three Euler angles given by ra, rb and rc (in degrees). Reads a moscito STRUCTURE-file from ⟨STDIN⟩ and writes output to ⟨STDOUT⟩.

5.2.9 scalevel

Needs: STRUCTURE-file
Usage: *scalevel s < input.structure > output.structure*

Scales all the velocities contained in a structurefile by a factor s. Can be used to reset the velocities or to them adjust to a different temperature. Moreover, this feature can be useful, when one has changed the mass of a particle for example. Reads a moscito STRUCTURE-file from ⟨STDIN⟩ and writes output to ⟨STDOUT⟩.

5.2.10 sortstr

Needs: STRUCTURE-file
Usage: *sortstr -sys #SYSTEM-file* < *input.structure* > *output.structure*

Necessary Parameters:

	-sys	:	*specifies SYSTEM-file.*

Optional Parameters:

	-v	:	*verbose, prints some info.*
	-help	:	*prints help text.*

Sorts molecules in a structure file according to their molecule type index (Position of its molecule-section in the system file). Reads a moscito STRUCTURE-file from ⟨STDIN⟩ and writes sorted STRUCTURE-file to ⟨STDOUT⟩.

5.2.11 struccombine

Needs: STRUCTURE-file
Usage: *struccombine x input1.structure input2.structure* > *output.structure*

Merges two configurations contained in *input1.structure* and *input2.structure* and glues them together at the y-z plane (command line option x), x-z plane (option y) or x-y plane (option z). Reads a moscito STRUCTURE-file specified at command-line and writes output to ⟨STDOUT⟩.

5.2.12 suggestk

Needs: STRUCTURE-file
Usage: *suggestk* a < *input.structure*

Extracts the box-dimensions and suggests a PME mesh spacing suitable for the fftw-based fast Fourier transform routines. Parameter a defines the mesh-width and has to be given in Å. Reads a moscito STRUCTURE-file from ⟨STDIN⟩ and writes output (three integer numbers) to ⟨STDOUT⟩.

5.2.13 sysbuild

Needs: SYSTEM-file (containing build-section)
Output files: STRUCTURE-file
Usage: *sysbuild -sys #SYSTEM-file* > *out.structure*

Necessary Parameters:

	-sys	:	*specifies SYSTEM-file.*

Optional Parameters:

	-v	:	*verbose, prints some info.*
	-help	:	*prints help text.*

sysbuild is intended to generate structure files from scratch. It reads the information contained in the build-section in SYSTEM-file and writes a moscito STRUCTURE-file to ⟨STDOUT⟩.

5.2.14 sysrandomize

Needs: SYSTEM-file, STRUCTURE-file
Output files: STRUCTURE-file
Usage: *sysrandomize -sys SYSTEM-file < input.structure > out.structure*

Necessary Parameters:			
	-sys	:	*specifies SYSTEM-file.*
Optional Parameters:			
	-v	:	*verbose, prints some info.*
	-help	:	*prints help text.*

When creating a new STRUCTURE-file (e.g. with *sysbuild*) the molecule ensemble is generated by following a clearly defined pattern. However, the correlation between the location of a molecule and its position in the STRUCTURE-file (molecule number) is sometimes not wanted. Therefore *sysrandomize* finds a new randomly chosen position in the STRUCTURE-file for each molecule and thus destroying this correlation. Reads a moscito STRUCTURE-file from ⟨STDIN⟩ and writes output to ⟨STDOUT⟩.

5.3 Advanced STRUCTURE-file manipulation

5.3.1 align_inert

Needs: SYSTEM-file, STRUCTURE-file
Usage: *align_inert -sys #SYSTEM-file -i < input.structure > output.structure*

Necessary Parameters:			
	-sys	:	*specifies SYSTEM-file.*
Optional Parameters:			
	-i	:	*Print detailed information on the moment of inertia to to ⟨STDOUT⟩instead of aligned STRUCTURE-file.*
	-v	:	*verbose, prints some general info.*
	-help	:	*prints help text.*

Reads a configuration and rotates any molecule in the configuration such that principle axis system of its moment of inertia tensor coincides with the laboratory frame. In most cases this procedure will be applied to STRUCTURE-files containing just one molecule. When option *-i* is used detailed information of the moment of inertia of the molecules is displayed and no aligned STRUCTURE-file will be produced. Reads a moscito STRUCTURE-file from ⟨STDIN⟩ and writes output to ⟨STDOUT⟩.

5.3.2 rm_moment

Needs: PARAMETER-file, SYSTEM-file, STRUCTURE-file
Usage: *rm_moment < input.structure > output.structure*

Removes a total linear and total angular momentum from a configuration contained in *input structure*. Nevertheless, when starting a simulation run the "*stop momentum*" keyword should always be set in *moscito.par* since the accuracy provided by the STRUCTURE-file is orders of magnitude lower than machine precision. Reads a moscito STRUCTURE-file from ⟨STDIN⟩ and writes output to ⟨STDOUT⟩.

5.3.3 mom_invert

Needs: STRUCTURE-file
Usage: *mom_invert < input.structure > output.structure*

Inverts the momenta of all the particles. This rather esoteric feature can be used for example to propagate a system back in time. Please notice that the the STRUCTURE-file format provides a rather low accuracy with respect to velocities. Reads a moscito STRUCTURE-file from ⟨STDIN⟩ and writes output to ⟨STDOUT⟩.

5.4 Properties of a system defined in a STRUCTURE-file

5.4.1 energy

Needs: PARAMETER-file, SYSTEM-file, STRUCTURE-file
Usage: *energy -par #PARAMETER-file -sys #SYSTEM-file -v -help < STRUCTURE-file*

Necessary Parameters:			
	-par	:	*specifies PARAMETER-file.*
	-sys	:	*specifies SYSTEM-file.*
Optional Parameters:			
	-v	:	*verbose, prints some info.*
	-help	:	*prints help text.*

Calculates potential and kinetic energy and pressure of system defined in *input structure*. The potential energy is further analysed with respect to inter- and intramolecular contributions. The components of the intermolecular electrostatic energy like real space and reciprocal lattice contribution are specified as well. *energy* can be used to check the configuration/convergence of the Ewald sum defined in *PARAMETER-file*. Reads a moscito STRUCTURE-file from ⟨STDIN⟩ and writes formatted text info output to ⟨STDOUT⟩.

5.5 Converting STRUCTURE-files into different formats

5.5.1 struc2dlp

Needs: STRUCTURE-file
Usage: *struc2dlp < input.structure > output.dl_poly*

Creates an INPUT file for the DL_POLY simulation program containing the complete structural information. Reads a moscito STRUCTURE-file from ⟨STDIN⟩ and writes the DL POLY input data to ⟨STDOUT⟩. Not a very sophisticated tool and just intended to provide a first step when setting up a DL_POLY run.

5.5.2 struc2mmf

Needs: SYSTEM-file, STRUCTURE-file
Usage: *struc2mmf -sys #SYSTEM-file < input.structure > output.mmf*

Necessary Parameters:			
	-sys	:	*specifies SYSTEM-file.*
Optional Parameters:			
	-v	:	*verbose, prints some info.*
	-help	:	*prints help text.*

Creates a MOSCITO META FILE (mmf) from information contained in the SYSTEM-file and STRUCTURE-file. The new mmf-file can be used e.g. for further forcefield development. Please note that the atom labels written to the mmf-file are identical to the atom labels contained in the system-file! Normally, these labels are *not* (!!) equivalent to the ones used in the original mmf- or forcefield files. Therefore they have to be modified by hand before one can proceed further (sorry). Reads a moscito STRUCTURE-file from ⟨STDIN⟩ and writes mmf-data to ⟨STDOUT⟩.

5.5.3 struc2pdb

Needs: STRUCTURE-file
Usage: *struc2pdb < input.structure > output.pdb*

Creates a Brookhaven crystallographic data bank format (pdb) which can be further processed by visualisation tools such as e.g. *Rasmol*. Each molecule will be represented as one "group" in the pdb-file which is labelled according to the first three characters of the molecule-label given in the STRUCTURE-file. Note that the labels are always lower case. Reads a moscito STRUCTURE-file from ⟨STDIN⟩ and writes pdb-data to ⟨STDOUT⟩.

5.5.4 struc2pov

Needs: STRUCTURE-file
Usage: *struc2pov input.structure output.pov d rx ry*

Creates a scene which can be rendered with the Persistence of Vision (POV) raytracer. The camera position has to be specified by a distance d (in nm) and polar angle rx and azimuthal angle ry (both angles have to be given in degrees). Reads a moscito STRUCTURE-file from ⟨STDIN⟩ and writes the POV scene data to ⟨STDOUT⟩.

5.5.5 struc2xyz

Needs: STRUCTURE-file
Usage: *struc2xyz < input.structure > output.xyz*

Creates a "standard" plain xyz file as used e.g. by xmakemol. The site-labels are converted into element-labels where possible. However, in some cases this procedure leads to unsatisatisfactory results, e.g. when aliphatic carbons (OPLS Type: CA) are interpreted as Calcium. Note that the labels are always lower case. Reads a moscito STRUCTURE-file from ⟨STDIN⟩ and writes xyz-data to ⟨STDOUT⟩.

5.5.6 struc2gro

Needs: STRUCTURE-file
Usage: *struc2gro* $<$ *input.structure* $>$ *output.gro*

Creates a GROMOS input file as used e.g. by the GROMACS or GROMOS. Reads a moscito STRUCTURE-file from ⟨STDIN⟩ and writes gro-data to ⟨STDOUT⟩.

5.6 Calculation of simple thermodynamic properties from simulation runs

5.6.1 average

Needs: any file
Usage: *average -e < some.dat*

Optional Parameters:

-e	:	*estimate error using block averages*

Calculates average and error estimate for any numerical quantity. The error estimate is based on the block average method by Flyvbjerg and Petersen [19]. The method is discussed in detail in the book by Frenkel and Smit [2] pp. 381. The data is read from ⟨STDIN⟩. The data-file *some.dat* should just contain one single column. The format is free. When using no option the progam calculates just the average $\langle x \rangle$ and statistical error $\sqrt{\langle x \rangle^2 - \langle x^2 \rangle}/\sqrt{N-1}$ (assuming statistically uncorrelated data) and prints it to ⟨STDOUT⟩. However, typically the data is to some exetend correlated and therefore the statistical error obtained as shown above is unrealistically small. Therefore the block averaging method can be applied: When using option -e is the block averaging is performed and the results are written to ⟨STDOUT⟩. The first column indicates the $\log_2$, of the block-length. The second column gives the statistical error according to the actual block-length and the third column species the error of column 2.

5.6.2 avdata

Needs: DATA-file
Usage: *avdata* $<$ *mos.data*

Calculates average $\langle x \rangle$ and variance $\sqrt{\langle x \rangle^2 - \langle x^2 \rangle}$ for all thermodynamic properties contained in the moscito DATA-file. Reads a moscito DATA-file from ⟨STDIN⟩ and writes formatted info output to ⟨STDOUT⟩.

5.6.3 calc_err

Needs: DATA-file
Usage: *calc_err* $<$ *mos.data*

Calculates averages and error estimates for all thermodynamic properties contained in the moscito DATA-file. The error estimation is based on the calculation of the statistical inefficiency as proposed in the textbook on "Computer simulation of Liquids" by M.P. Allen and

D.J. Tildesley (pg. 192 ff). The data order is the following: 1. Averages. 2. Fluctuation amplitudes. 3. Estimated correlation sequence lengths (corresponding to a maximum statistical inefficiency) in time steps. 4. Estimated errors. Reads a moscito DATA-file from ⟨STDIN⟩ and writes formatted info output to ⟨STDOUT⟩.

5.6.4 density_err

Needs: SYSTEM-file, STRUCTURE-file, DATA-file
Usage: *density_err -sys #SYSTEM-file -in #STRUCTURE-file < mos.dat*

Necessary Parameters:

-sys	:	*specifies SYSTEM-file.*
-in	:	*specifies STRUCTURE-file.*

Calculates averages and error estimates for the total mass density. The error estimation and output formatting is done as discussed in the previous section. Reads a moscito DATA-file from ⟨STDIN⟩ and writes formatted info output to ⟨STDOUT⟩.

5.7 Structure of liquids — calculation of pair distribution functions

5.7.1 gofr, gofr_large

Needs: SYSTEM-file, STRUCTURE-file, CRD-file/XTC-file
Output files: gr.dat, sq.dat, Nnn.dat
Usage: *gofr/gofr_large -sys SYSTEM-file -in STRUCTURE-file -crd #CRD-files -xtc #XTC-files -gap #gap -help -n #histogram entries -rmin #rmin -rmax #rmax -qmin #qmin -qmax #qmax*

Necessary Parameters:

-sys	:	*specifies SYSTEM-file.*
-in	:	*specifies STRUCTURE-file.*
-crd	:	*specifies moscito coordinate (CRD) files. (multiple files can be specified)*
-xtc	:	*specifies moscito coordinate (XTC) files. (multiple files can be specified)*

Optional Parameters:

-gap	:	*gap between two configurations*
-help	:	*prints help message*
-n	:	*number of histogram entries*
-rmin	:	*g(r) minimum r.*
-rmax	:	*g(r) maximum r.*
-qmin	:	*s(q) minimum q.*
-qmax	:	*s(q) maximum q.*

```
#........................place this section after
#                         the last molecule-section
#                         in the current {SYSTEM}-file
begin{gengofr}
   2 1    1 1       # molecule a   site a    molecule b   site b
   2 3    1 1
   2 5    1 1
   gr_range 0.1 1.0  # (range in nm)
end{gengofr}
```

Figure 5.1: Example for the required additional SYSTEM-file section for *calcgofr*.

gofr and *gofr_large* calculate pair correlation functions between site-site pairs $\alpha\beta$ in molecules according to

$$g_{\alpha\beta}(r) \quad = \quad \frac{V}{2\pi r^2 N^2} \left\langle \sum_{i=1}^{N} \sum_{j=i+1}^{N} \delta(r - r_{i\alpha j\beta}) \right\rangle \quad . \tag{5.1}$$

N represents the total number of molecules in a simulation, i and j are running indices over all molecules while α and β indicate the site-pairs. V is the volume of the simulation box. In addition the partial structure factor is calculated according to

$$s_{\alpha\beta}(q) - 1 \quad = \quad 4\pi\rho \int r^2 \left[g_{\alpha\beta}(r) - 1\right] \frac{\sin(qr)}{qr} dr \quad , \tag{5.2}$$

where ρ denotes the atomic number density of the system and q represents the momentum transfer vector.

gofr makes use of the minium image convention, hence distances larger than half of the box-length will lead to errornous data. *gofr_large* uses a $27\times$ replicated system, hence, a larger r-range is accessible (Absolute limit: box-length). However, In order to minimize the occurrence of artificial correlations at large r, distances larger than $\sqrt{3}/2$ the box-length should be avoided. It has to be stressed that *gofr_large* is several times slower than *gofr*.

The site-site pairs to be considered have to be specified explicitly in an additional SYSTEM-file section (see Figure 5.1) at the very end of the current SYSTEM-file. The syntax is the following: Line by line the site-site pairs have to be indicated explicitly by their molecule reference number (order of appearance in current SYSTEM-file) and site reference (position in in the *configuration* environment) therein. Please note that only chemically equivalent site-site pairs should be considered here since the program generates only one single pair-correlation function per run. Using the `gr_range` keyword the distance range (given in nm) can be specified. Otherwise the command line options *-rmin* and *-rmax* have to be used

The program generates three formatted ASCII files: *gr.dat* contains the pair correlation function and *Nnn.dat* contains the number of neighbors function, which is the number density weighted integral over the pair correlation function (in both cases the r-values are given in nm). *sq.dat* contains the corresponding partial structure factor where q is given in nm^{-1}. If the q-limits are not defined explicitly the program calculates them automatically according $q_{min} = 2\pi/(r_{max} - r_m in)$ and $q_{max} = 2\pi/\Delta r/5$, where Δr represents the binwidth of *gr.dat*.

5.7.2 gofrcms

Needs: SYSTEM-file, STRUCTURE-file, CRD-file/XTC-file
Output files: gr.dat, sq.dat, Nnn.dat
Usage: *gofrcms -sys SYSTEM-file -in STRUCTURE-file -crd #CRD-files -xtc #XTC-files -gap #gap -help -a #moltype a -b #moltype b -n #histogram entries -rmin #rmin -rmax #rmax -qmin #qmin -qmax #qmax*

Necessary Parameters:

-sys	:	*specifies SYSTEM-file.*
-in	:	*specifies STRUCTURE-file.*
-crd	:	*specifies moscito coordinate (CRD) files. (multiple files can be specified)*
-xtc	:	*specifies moscito coordinate (XTC) files. (multiple files can be specified)*
-a	:	*moleculetype "a"*
-b	:	*moleculetype "a"*

Optional Parameters:

-gap	:	*gap between two configurations*
-help	:	*prints help message*
-n	:	*number of histogram entries*
-rmin	:	*g(r) minimum r.*
-rmax	:	*g(r) maximum r.*
-qmin	:	*s(q) minimum q.*
-qmax	:	*s(q) maximum q.*

Calculates the center of mass pair correlation function and structure factor for a given pair of molecule types. The molecule types as well as the r-range have to be specified at the command line using the *-a*, *-b*, *-rmin* and *-rmax* options. Please note that no *gengofr*-section is used here. For more information see section 5.7.1.

5.7.3 sofq

Needs: gr.dat from ⟨STDIN⟩
Output files: ⟨STDOUT⟩
Usage: *sofq -rho #atomic density -qmin #qmin -qmax #qmax < gr.dat > sq.dat*

Optional Parameters:

-rho	:	*gap between two configurations*
-qmin	:	*s(q) minimum q.*
-qmax	:	*s(q) maximum q.*

Calculates a partial structure factor from previously calculated $g_{\alpha\beta}(r)$-data according to

$$s_{\alpha\beta}(q) - 1 \quad = \quad 4\pi\rho \int r^2 \left[g_{\alpha\beta}(r) - 1\right] \frac{\sin(qr)}{qr} dr \quad . \tag{5.3}$$

The atomic number density ρ in the system has to be specified at the command line.

5.8 Single particle (molecule) dynamics in liquids

5.8.1 msdmol

Needs: PARAMETER-file, SYSTEM-file, STRUCTURE-file, CRD-files/XTC-files
Output file: msd.dat
Usage: *msdmol -sys SYSTEM-file -in STRUCTURE-file -crd #CRD-files -xtc #XTC-files -time #time -tmax #tmax -tgap #tgap -gap #gap*

Necessary Parameters:

-sys	:	*specifies SYSTEM-file.*
-in	:	*specifies STRUCTURE-file.*
-crd	:	*specifies moscito coordinate (CRD) files. (multiple files can be specified)*
-xtc	:	*specifies moscito coordinate (XTC) files. (multiple files can be specified)*
-time	:	*time interval between two configurations in CRD-file*
-tmax	:	*length of time correlation function-array*

Optional Parameters:

-tgap	:	*smallest gap for tcf evaluation*
-gap	:	*gap for data acquisition*

Calculates the mean square displacement function

$$\mathrm{msd}(\delta t) = \langle(\vec{r}_i(t) - \vec{r}_i(t+\delta t)\rangle_{t,i}$$

for the center of mass motion of the molecules contained in a simulation. Here the brackets denote time averaging as well as averaging over all molecules belonging to a certain molecule type. Therefore a msd-function is calculated for each type and added as an additional column to *msd.dat* (in units of nm^2). The first column, of course, contains the values for δt (in units of #*time*). The (self) diffusion coefficient for each species is then given according to

$$D_{\mathrm{self}} = \frac{1}{6} \lim_{\delta t \to \infty} \frac{\partial}{\partial \delta t} \mathrm{msd}\,(\delta t)$$

as slope of the msd-function for large δt.

The structural data is read from a moscito CRD-file which has to be specified at the command line. To define the time-scale properly the program needs the exact time-spacing between two configurations (e.g. ps) using the *-time* option. This information can be taken for example from the corresponding *moscito.par* file. The array-length of the msd time correlation function has to be specified by #*tmax*. #*tmax* has to lie somewhere in the interval between one and the number of configurations contained in the CRD-file. A quite reasonable upper limit for #*tmax* is approximately about one tenth of the available total number of configurations. For larger spacings the statistics typically becomes to poor. In some cases it is perhaps useful not to use every configuration. Using #*gap*= 2 would only consider every second configuration. In contrast the *-tgap* options influences only the smallest value for δt, whereas any available configuration is still being considered for the time averaging.

5.8.2 dipolcor

Needs: PARAMETER-file, SYSTEM-file, STRUCTURE-file, CRD-files/XTC-files
Output files: dipol_p1.dat, dipol_p2.dat
Usage: *dipolcor -sys SYSTEM-file -in STRUCTURE-file -crd #CRD-files -xtc #XTC-files -time #time -tmax #tmax -tgap #tgap -gap #gap*

Necessary Parameters:

-sys	:	*specifies SYSTEM-file.*
-in	:	*specifies STRUCTURE-file.*
-crd	:	*specifies moscito coordinate (CRD) files. (multiple files can be specified)*
-xtc	:	*specifies moscito coordinate (XTC) files. (multiple files can be specified)*
-time	:	*time interval between two configurations in CRD-file*
-tmax	:	*length of time correlation function-array*

Optional Parameters:

-tgap	:	*smallest gap for tcf evaluation*
-gap	:	*gap for data acquisition*

Calculates the reorientational time correlation functions for the molecular dipole vector

$$\begin{aligned} C_1(\delta t) &= \langle \vec{u}_i(t) \cdot \vec{u}_i(t+\delta t) \rangle_{i,t} \\ C_2(\delta t) &= \left\langle \frac{3}{2} \left(\vec{u}_i(t) \cdot \vec{u}_i(t+\delta t) \right)^2 - \frac{1}{2} \right\rangle_{i,t} \end{aligned}$$

according to the definition of first and second Legendre polynome. Here $\vec{u}_i(t)$ denotes the unit vector of the molecular dipole moment of molecule i at time t. The brackets denote time averaging as well as molecule averaging. Averaging is performed over all molecules belonging to a certain molecule-type. Thus reorientational correlation functions are obtained for each molecule type and therefore for each type an additional column is added to the both output files. Two files are created: *dipol_p1.dat* contains the C_1 data and *dipol_p1.dat* holds the C_2 data. The values for δt (in units of *#time*) are always in the first column. The command line options have a similar effect as the options discussed in section 5.8.1 for the *msdmol* command.

5.8.3 vectorcor

Needs: PARAMETER-file, SYSTEM-file, STRUCTURE-file, CRD-files/XTC-files
Output file: vec_p1p2.dat
Usage: *vectorcor -sys SYSTEM-file -in STRUCTURE-file -crd #CRD-files -xtc #XTC-files -time #time -tmax #tmax -tgap #tgap -gap #gap*

```
#........................place this section after
#                        the last molecule-section
#                        in the current {SYSTEM}-file
begin{vectorcor}
    1   2   1       # molec. type   site a   site b
    1   2   3
end{vectorcor}

begin{nvectorcor}
    1   2   1   2  3  # molec. type   sites  a1  b1   a2  b2
end{nvectorcor}
```

Figure 5.2: Example for the required additional SYSTEM-file section for *vectorcor* and *nvectorcor*.

Necessary Parameters:

-sys	:	*specifies SYSTEM-file.*
-in	:	*specifies STRUCTURE-file.*
-crd	:	*specifies moscito coordinate (CRD) files. (multiple files can be specified)*
-xtc	:	*specifies moscito coordinate (XTC) files. (multiple files can be specified)*
-time	:	*time interval between two configurations in CRD-file*
-tmax	:	*length of time correlation function-array*

Optional Parameters:

-tgap	:	*smallest gap for tcf evaluation*
-gap	:	*gap for data acquisition*

Calculates the reorientational time correlation functions

$$C_1(\delta t) = \langle \vec{u}_i(t) \cdot \vec{u}_i(t+\delta t) \rangle_{i,t}$$
$$C_2(\delta t) = \left\langle \frac{3}{2} \left(\vec{u}_i(t) \cdot \vec{u}_i(t+\delta t)\right)^2 - \frac{1}{2} \right\rangle_{i,t}$$

for unit-vectors defined by pairs of sites belonging to the same molecule. The site-site pairs have to be specified explicitly in an additional SYSTEM-file section (see Figure 5.2) at the very end of the current SYSTEM-file (vectorcor-section). The syntax is the following: Line by line the site-site pairs have to be indicated explicitly by their molecule reference number (order of appearance in current SYSTEM-file) and their site references (position in in the *configuration* environment). Please notice that only chemically equivalent vectors should appear in one vectorcor-section (like the two O-H vectors in a water molecule). The two correlation functions a written to *vec_p1p2.dat* (δt: first column, C_1: second column, C_2: third column). The command line options have a similar effect as the options discussed in section 5.8.1 for the *msdmol* command.

5.8.4 nvectorcor

Needs: PARAMETER-file, SYSTEM-file, STRUCTURE-file, CRD-file
Output file: vec_p1p2.dat

Usage: *nvectorcor -sys SYSTEM-file -in STRUCTURE-file -crd #CRD-files -xtc #XTC-files -time #time -tmax #tmax -tgap #tgap -gap #gap*

Necessary Parameters:

-sys	:	*specifies SYSTEM-file.*
-in	:	*specifies STRUCTURE-file.*
-crd	:	*specifies moscito coordinate (CRD) files. (multiple files can be specified)*
-xtc	:	*specifies moscito coordinate (XTC) files. (multiple files can be specified)*
-time	:	*time interval between two configurations in CRD-file*
-tmax	:	*length of time correlation function-array*

Optional Parameters:

-tgap	:	*smallest gap for tcf evaluation*
-gap	:	*gap for data acquisition*

Calculates the reorientational time correlation functions

$$\begin{aligned} C_1(\delta t) &= \langle \vec{u}_i(t) \cdot \vec{u}_i(t+\delta t) \rangle_{i,t} \\ C_2(\delta t) &= \left\langle \frac{3}{2} \left(\vec{u}_i(t) \cdot \vec{u}_i(t+\delta t)\right)^2 - \frac{1}{2} \right\rangle_{i,t} \end{aligned}$$

for unit-vectors defined as normal vectors according to $\vec{u} = \vec{u}_1 \times \vec{u}_2$ where $\vec{u}_1$ and $\vec{u}_2$ are defined by pairs of sites belonging to the same molecule. Both vectors have to be specified explicitly in an additional SYSTEM-file section (see Figure 5.2) at the very end of the current SYSTEM-file (nvectorcor-section). The two correlation functions a written to *vec_p1p2.dat* (δt: first column, C_1: second column, C_2: third column). The command line options have a similar effect as the options discussed in section 5.8.1 for the *vectorcor* and *msdmol* command.

5.9 Geometry Optimization of individual molecules

5.9.1 zminit

Needs: SYSTEM-file
Output file: STRUCTURE-file
Usage: *zminit -sys SYSTEM-file -out STRUCTURE-file*

Necessary Parameters:

-sys	:	*specifies SYSTEM-file.*

Optional Parameters:

-out	:	*specifies ouput STRUCTURE-file (default: mosout.str).*

Performs a geometry optimization for one single molecule using the internal coordinates defined in a *zmatrix*-environment (See section 10 for details). Only intramolecular interactions are taken into account. Please note, that the SYSTEM-file shall not contain more than one molecule section. The optimization starts, of course, with the configuration defined by the initial (given) z-matrix. The parameters are successively optimized following a self converging line search procedure. When the convergence criteria is fulfilled, the converged potential intramolecular energy as well as the final z-matrix are written to ⟨STDOUT⟩and the optimized configuration is written to a output STRUCTURE-file. Please keep in mind that there is no guarantee that the observed minimum energy structure represents the global minimum. In order to confirm the a global energy minimum it is perhaps advised to start from several different trial configurations.

5.9.2 addzmat.pl

Needs: SYSTEM-file, z-matrix-file
Output files: SYSTEM-file
Usage: *addzmat.pl SYSTEM-file < z-matrix-file > SYSTEM-file (containing z-matrix)*

Necessary Parameters:

#1	:	*specifies SYSTEM-file.*

Adds the definition of a z-matrix (z-matrix-section) to the molecule-sections in a SYSTEM file. Please apply *addzmat.pl* only to SYSTEM-files containing NOT more than ONE molecule section. The program works that way: The text contained in the z-matrix-file is pasted into a z-matrix subsection of the molecule section. Please note that no checks are performed whether the definition of the z-matrix is reasonable, nor whether it is a z-matrix at all.

5.10 Creating an aqueous solution

5.10.1 solve

Needs: SYSTEM-file (solute), STRUCTURE-file (solute), STRUCTURE-file (water)
Output files: STRUCTURE-file
Usage: *solve -sys SYSTEM-file -i STRUCTURE-file (solute) -w STRUCTURE-file (water) -o STRUCTURE-file (output) -n #water model sites*

Necessary Parameters:

-sys	:	*specifies SYSTEM-file.*
-i	:	*STRUCTURE-file (solute).*
-w	:	*STRUCTURE-file (water).*
-o	:	*STRUCTURE-file (output).*
-n	:	*number of water-sites (default=3).*

Optional Parameters:

-v	:	*verbose, prints some info.*
-help	:	*prints help text.*

Creates an aqueous solution by placing an arbitrary molecular solute configuration (which may contain several different molecule types) into an aqueous surrounding given in terms of a water configuration. Both configurations are merged and overlapping water molecules are discarded. The final configuration is written to a new STRUCTURE-file.

1. *solve* needs the SYSTEM-file for the system that shall be put into the water box. The name of the SYSTEM-file is specified at the command line. Keep in mind that this file does not include the definition of the solvent water molecule.

In order to be able to run the solute/solvent simulation a new SYSTEM-file has to be to be created that includes the definition of the solvent (water) molecule. This solute-plus-water-SYSTEM-file can be easily created using the Perl scripts *addspc.pl*, *addspce.pl* and *addtip3p.pl* described in the sections below.

2. *solve* needs two structure files: the first STRUCTURE-file has to contain the geometry of the (solute) system which shall be a added to a water configuration. The second STRUCTURE-file has to contain a configuration of water molecules with the atoms ordered as O, H, H. . .. *solve* assumes three-site water models such as SPC, SPCE or TIP3P as a default. If four or five center models, such as TIP4P or TIP5P, are used, the virtual sites have be placed at the fourth (and) fifth position and the *-n* option has to be used to tell *solve* of how many interaction sites the actual water model consists.

   The solute system is then added to the water configuration. All overlapping water molecules are discarded and the combined configuration is stored in an output file defined at the command line.

   The solute molecules are stored in the beginning part of the configuration.

3. The SYSTEM-file for the new combined configuration has to follow some conventions as outlined in Figure 5.3: 1. The water sites (OW,HW) definitions have to be placed above the solute-sites definitions. 2. The water molecule definition *must* be the first molecule to be defined.

   The Perl-scripts *addspc.pl*, *addspce.pl*, *addtip3p.pl*, *tip4p.pl* and *addtip5p.pl* automatically account for these necessities. It is therefore highly recommended to use them whenever it is possible.

5.10.2 addspce.pl

Needs: SYSTEM-file (solute)
Output files: SYSTEM-file (solute+water)
Usage: *addspce.pl SYSTEM-file (solute) > SYSTEM-file (solute+water)*

Necessary Parameters:

#1	*:*	*specifies solute SYSTEM-file.*

Adds the definition of the *three* site SPC/E water model to any SYSTEM file. *addspce.pl* has been designed to work in combination with *solve* and therefore the required conventions are respected.

5.10.3 addspc.pl

Needs: SYSTEM-file (solute)
Output files: SYSTEM-file (solute+water)
Usage: *addspc.pl SYSTEM-file (solute) > SYSTEM-file (solute+water)*

Necessary Parameters:

#1	*:*	*specifies solute SYSTEM-file.*

```
#----HEADER SECTION
    begin{sites}
      OW   ....   #ADD OW SITE HERE (#1)
      HW   ....   #ADD HW SITE HERE (#2)
      ...solute molecule sites
      ...
     end{sites}

    begin{lj...}
      OW   ....
      HW   ....
      ...solute molecule sites
      ...
     end{lj...}

#----MOLECULE SECTION (ADD WATER-SECTION at #1)
    begin{molecule}
      label  h2o
      .....
    end{molecule}

#----MOLECULE SECTION (SOLUTES-SECTIONS #2...#n)
    begin{molecule}
      label solute1 # (or whatever you like)
      ....
    end{molecule}
    begin{molecule}
      label solute2 # (or whatever you like)
      ....
    end{molecule}
```

Figure 5.3: Modifications necessary to turn a solute-SYSTEM-file into a solute-plus-water-SYSTEM-file after *solve* has been used.

Adds the definition of the *three* site SPC water model to any SYSTEM file. *addspc.pl* has been designed to work in combination with *solve* and therefore the required conventions are respected.

5.10.4 addtip3p.pl

Needs: SYSTEM-file (solute)
Output files: SYSTEM-file (solute+water)
Usage: *addtip3p.pl SYSTEM-file (solute) > SYSTEM-file (solute+water)*

Necessary Parameters:

#1 : specifies solute SYSTEM-file.

Adds the definition of the *three* site TIP3P water model to any SYSTEM file. *addtip3p.pl* has been designed to work in combination with *solve* and therefore the required conventions are respected.

5.10.5 addtip4p.pl

Needs: SYSTEM-file (solute)
Output files: SYSTEM-file (solute+water)
Usage: *addtip4p.pl SYSTEM-file (solute) > SYSTEM-file (solute+water)*

Necessary Parameters:

#1	:	*specifies solute SYSTEM-file.*

Adds the definition of the *four* site TIP4P water model to any SYSTEM file. *addtip4p.pl* has been designed to work in combination with *solve* and therefore the required conventions are respected.

5.10.6 addtip5p.pl

Needs: SYSTEM-file (solute)
Output files: SYSTEM-file (solute+water)
Usage: *addtip5p.pl SYSTEM-file (solute) > SYSTEM-file (solute+water)*

Necessary Parameters:

#1	:	*specifies solute SYSTEM-file.*

Adds the definition of the *five* site TIP5P water model to any SYSTEM file. *addtip5p.pl* has been designed to work in combination with *solve* and therefore the required conventions are respected.

5.11 Building mixtures

5.11.1 mergestr

Needs: SYSTEM-file, STRUCTURE-file (system A), STRUCTURE-file (system B)
Output files: STRUCTURE-file
Usage: *mergestr -sys SYSTEM-file -a STRUCTURE-file (system A) -b STRUCTURE-file (system A) -l #limit -r #ratio > STRUCTURE-file (output)*

Necessary Parameters:

-sys	:	*specifies SYSTEM-file.*
-a	:	*STRUCTURE-file (system A).*
-b	:	*STRUCTURE-file (system B).*
-l	:	*limiting interaction energy.*
-r	:	*composition ratio [0-1]*

Optional Parameters:

-i	:	*print info.*
-v	:	*verbose, prints some info.*
-help	:	*prints help text.*

The program is used to build binary or even more complex mixtures by merging two STRUCTURE-files (A,B) and discarding overlapping molecules from systems A and B. In order to work correctly both STRUCTURE-files have to correspond to the same SYSTEM-file that has to contain all molecule definitions (in the correct order!) and which will afterwards serve as the SYSTEM-file for the mixture. The program will always take the largest box dimensions for each direction. To create a meaningful mixture configuration, overlapping molecules discarded. "Overlapping" means that the pair interaction between two molecules exceeds a certain value which has to be specified explicitly at the command line (using the -l option) in units of kJ mol^{-1}. Positive values indicate a repulsive interaction. Typically one will choose a value of a few RT, say 10.0 kJ mol^{-1} as a limit. However, the composition of the mixture has also to be determined. This is done by introducing the acceptance ratio-parameter s (-r option) which can be varied between 0 and 1. The method works as following. Given the pair interaction between two molecules exceeds the limiting value and hence the two molecules "overlap". Then the acceptance parameter s is compared with a newly created random number. If the random number is larger than s than the molecule from system B is discarded, otherwise the molecule from system A eliminated. Hence, an acceptance parameter of "0" means that all A-particles will survive while a value of "1" will result in a system containing just B-particles. Using the -i Option will just print the composition of merged system to ⟨STDOUT⟩. In order to create a certain fixed composition, one has perhaps to play around with the acceptance parameter.

5.11.2 mixsys.pl

Needs: SYSTEM-file (system A), SYSTEM-file (system B)
Output files: SYSTEM-file (merged)
Usage: *mixsys.pl SYSTEM-file (A) SYSTEM-file (B) > SYSTEM-file*

Necessary Parameters:

#1	:	*specifies SYSTEM-file (A).*
#2	:	*specifies SYSTEM-file (B).*

Merges two system files containing each a definition of one molecule in order to create a single system file with two molecule sections. Please check that there are no identical site type specifiers used in system files A and B. Otherwise *mixsys.pl* will stop without merging the two system files.

5.12 Adding hydrogens to united atom carbons

5.12.1 addh2str

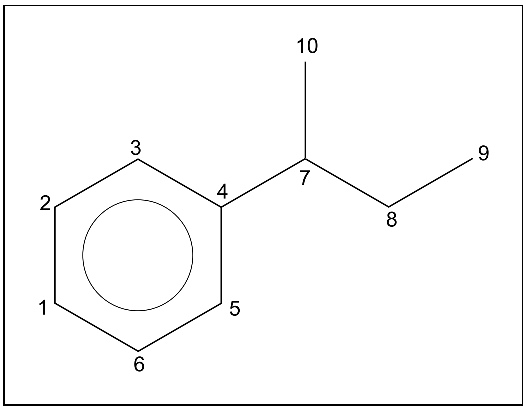
Needs: SYSTEM-file, STRUCTURE-file
Output file: STRUCTURE-file
Usage: *addh2str -sys SYSTEM-file < STRUCTURE-file > STRUCTURE-file -v #level -help*

Necessary Parameters:

-sys	:	*specifies SYSTEM-file.*

Optional Parameters:

-v	:	*verbose, prints some info and step timings, #level=verbosity level (integer)*
-help	:	*prints help text.*



```
#----MOLECULE SECTION

begin{molecule}
  ....
  begin{hydrogen}
   2  car  1  3      1.045  # Add one aromatic hydrogen
   3  car  2  4      1.045  # Add one aromatic hydrogen
   5  car  4  6      1.045  # Add one aromatic hydrogen
   6  car  1  5      1.045  # Add one aromatic hydrogen

   7  ch1  4  8  10  1.09   # Add one ternary aliphatic hydrogen

   8  ch2  7  9      1.09   # Add two secondary aliphatic hydrogens

   9  ch3  8  7      1.09   # Add three primary aliphatic hydrogens
  10  ch3  7  4      1.09   # Add three primary aliphatic hydrogens
  end{hydrogen}
  ....
end{molecule}
```

Figure 5.4: hydrogen-section defining explicit hydrogens to be added to united atom carbon centers. The section is used by *addh2crd* and *addh2str*. The numbering of the example corresponds 1-methyl-propyl-benzene as given in the insert.

Creates explicit hydrogen atoms which are added to an united atom carbon center. The stereochemical information has to be defined in a separate "hydrogen" sub-section within the corresponding molecule section (see Figure 5.4). The outline of this section is as following: The first number indicates the site to which the hydrogen atoms is attached. The second column species the type: aromatic carbon (*car*) and primary (*ch1*), secondary (*ch2*) and tertiary (*ch3*) aliphatic carbons. Two following numbers indicate the direct neighbors of the central C-atom which. Only in case of the ternary C-atom three direct neighbors have to be specified. The final column defines the CH-bond length in Å. Please note that in case of a *ch2*-definition, two hydrogen atoms and and in case of *ch3* three atoms will be created.

5.12.2 addh2crd

Needs: SYSTEM-file, STRUCTURE-file, CRD-/XTC-files
Output file: CRD-file
Usage: *addh2crd -sys SYSTEM-file -in STRUCTURE-file -v #level -crd #CRD-files -xtc #XTC-files -help*

Necessary Parameters:

-sys	:	*specifies SYSTEM-file.*
-in	:	*specifies STRUCTURE-file.*
-crd	:	*specifies moscito coordinate (CRD) files.* *(multiple files can be specified)*
-xtc	:	*specifies moscito coordinate (XTC) files.* *(multiple files can be specified)*

Optional Parameters:

-v	:	*verbose, prints some info and step timings,* *#level=verbosity level (integer)*
-help	:	*prints help text.*

Works similar as *addh2str*, but reads a trajectory file (or a series of trajectory files) and creates a new CRD-file called `mos.crd_addH` containing just the positions of the added hydrogen atoms.

5.13 Miscellaneous

5.13.1 config_mos

Needs: SYSTEM-file, (STRUCTURE-file)
Output file: PARAMETER-file
Usage: *config_mos*

Configures a complete MOSCITO simulation run in an interactive session. The program runs at the command-line by successively asking for definitions (and in most cases providing reasonable suggestions). All necessary parameters are defined in order to create a reasonable simulation run setup so that a simply typing *moscito* at the command-line should be enough to start the simulation afterwards.

5.13.2 b2l

Needs: CRD-file or VEL-file
Output file: CRD-file or VEL-file
Usage: *b2l <infile> <outfile>*

Optional Parameters:

-help	:	*prints help text.*

Interconverts binary data files such as CRD- or VEL-files from big-endian format to little-endian (and vice versa). Rerunning *b2l* on the <outfile> will create a file identical to the initial

<infile>. The program helps to analyze data produced on eg. IBM RS6000 machines (big-endian) on Intel-architectures (little-endian). Please don't apply the *b2l* program to XTC-files since these are stored already in a machine independent format.

5.13.3 crd2xtc

Needs: CRD-file or XTC-file
Output file: CRD-file or XTC-file
Usage: *crd2xtc -c CRD-file -x XTC-file -p #precision -r*

Necessary Parameters:

-c	:	*specifies moscito coordinate (CRD) file.*
-x	:	*specifies moscito coordinate (XTC) file.*

Optional Parameters:

-p	:	*defining the precision. Default: 1000*
-r	:	*(reverse: converting XTC-file to CRD-file.*
-help	:	*prints help text.*

Converts a CRD-file into the substantially more compact XTC-file and vice versa (when applying the -r Option). The -p option defines the precision which is used for storing the coordinates in the XTC-format. Here a higher number indicates a higher precision. The default value (when no -p option and value is specified) is "1000" and defines a resolution of about 1 pm which is sufficiently accurate for most cases. A value of "10000" would result in accuracy of 0.1 pm and so forth.

5.13.4 addsection.pl

Needs: SYSTEM-file, text-file
Output files: SYSTEM-file
Usage: *addsection.pl SYSTEM-file < text-file > SYSTEM-file*

Necessary Parameters:

#1	:	*specifies SYSTEM-file.*

Inserts an additional section read from an ASCII text-file into the molecule-sections of a SYSTEM file. The text is pasted after the configuration part subsection of the molecule section.

Bibliography

[1] M. P. Allen and D. J. Tildesley. *Computer Simulation of Liquids*. Oxford Science Publications, (1989). 1, 30

[2] D. Frenkel and B. Smit. *Understanding Molecular Simulation — From Algorithms to Applications*. Academic Press Inc., San Diego, (1996). 1, 31, 58

[3] S. F. R. Haberlandt, G. Peinel and K. Heinzinger. *Molekulardynamik — Grundlagen und Anwendungen*. Vieweg, (1995). 1

[4] A. R. Leach. *Molecular Modelling — Principles and Applications*. Addison Wesley Longman Limited, (1996). 1

[5] W. D. Cornell, P. Cieplak, C. I. Bayly, I. R. Gouls, K. M. M. Jr., D. M. Fergueson, D. C. Spellmeyer, T. Fox, J. W. Caldwell and P. A. Kollman, "A second generation force field for the simulation of proteins, nucleic acids and organic molecules", *J. Am. Chem. Soc.*, **117**, 5179–5197, (1995). 1, 11, 42, 43

[6] M. Frigo and S. G. Johnson. "FFTW: An Adaptive Software Architecture for the FFT". In *ICASSP conference proceedings*, Vol. 3, 1381–1384, (1998). 5

[7] J. P. Ryckaert, G. Ciccotti and H. J. C. Berendsen, "Numerical integration of the cartesian equations of motions of a system with constraints: Molecular dynamics of n–alkanes", *J. Comp. Phys.*, **23**, 327–341, (1977). 8

[8] F. Müller–Plathe, "Singularity free treatment of linear bond angles", *CCP5 Newsletter*, **44**, , July 1995. 13, 41

[9] P. P. Ewald, "Die Berechnung optischer und elektrostatischer Gitterpotentiale", *Ann. Phys.*, **64**, 253–287, (1921). 15

[10] R. W. Hockney and J. W. Eastwood. *Computer Simulation Using Particles*. IOP Publishing, Bristol and Philadelphia, (1988). 21

[11] T. A. Darden, D. York and L. Pedersen, "Particle mesh Ewald: An $N \log(N)$ method for Ewald sums in large systems", *J. Chem. Phys.*, **98**, 10089–10092, (1993). 21

[12] U. Essmann, L. Perera, M. L. Berkowitz, T. A. Darden, H. Lee and L. G. Pedersen, "A smooth particle mesh Ewald method", *J. Chem. Phys.*, **103**, 8577–8593, (1995). 21, 23, 32

[13] S. Nosé and M. L. Klein, "Constant pressure molecular dynamics for molecular systems", *Mol. Phys.*, **50**, 1055–1076, (1983). 25

[14] J. Alejandre, D. J. Tildesley and G. Chapela, "Molecular dynamics simulation of the orthobaric densities and surface tension of water", *J. Chem. Phys.*, **102**, 4574–4583, (1995). 25

[15] H. J. C. Berendsen, J. P. M. Postma, W. F. van Gunsteren, A. DiNola and J. R. Haak, "Molecular dynamics with coupling to an external bath", *J. Chem. Phys.*, **81**, 3684–3690, (1984). 25

[16] N. Anastasiou and D. Fincham. *Programs for the dynamic simulation of liquids and solids — MDIONS: rigid ions using the Ewald sum*. CCP5, Dept. Chem., Royal Holloway College, Egham, Surrey, TW20 OEX, U.K., (1981). 30

[17] W. F. van Gunsteren and H. J. C. Berendsen, "Moleküldynamik Computersimulationen: Methodik, Anwendungen und Perspektiven in der Chemie", *Angew. Chem.*, **102**, 1020–1055, (1990). 32

[18] G. S. Kell, *J. Chem. Engeneer. Dat.*, **12**, 66–69, (1967). 33

[19] H. Flyvbjerg and H. Petersen, "Error estimates on averages of correlated data", *J. Chem. Phys.*, **91**, 461–466, (1989). 58