

ORAC Manual and Guide

Piero Procacci and Massimo Marchi[†]

A Molecular Dynamics Program to Simulate Complex Molecular Systems at the atomistic level

Dipartimento di Chimica, Polo Scientifico dell'Università di Firenze
Via della Lastruccia 3, Sesto Fiorentino I-50019 (Italy)
July 13, 2009

Send comments and bug reports to procacci@chim.unifi.it

[†] Piero Procacci and Massimo Marchi are the main contributors and the license holders. Other contributors are **Simone Marsili** who implemented the Replica Exchange Method and the Metadynamics in the parallel version and **Marc Souaille** who wrote the linked cell neighbor listing routines.

Contents

1	Atomistic simulations: an introduction	2
1.1	Multiple time steps integration schemes and electrostatic interactions in complex biomolecular systems	2
1.2	Enhanced sampling in atomistic simulations	4
2	Symplectic and Reversible Integrators	7
2.1	Canonical Transformation and Symplectic Conditions	7
2.2	Liouville Formalism: a Tool for Building Symplectic and Reversible Integrators	9
2.3	Potential Subdivision and Multiple Time Steps Integrators for NVE Simulations	10
2.4	Constraints and r-RESPA	13
2.5	Applications	13
3	Multiple Time Steps Algorithms for the Isothermal-Isobaric Ensemble	17
3.1	The Parrinello-Rahman-Nosé Extended Lagrangian	17
3.2	The Parrinello-Rahman-Nosé Hamiltonian and the Equations of Motion	18
3.3	Equivalence of Atomic and Molecular Pressure	21
3.4	Liouvillean Split and Multiple Time Step Algorithm for the <i>NPT</i> Ensemble	23
3.5	Group Scaling and Molecular Scaling	25
3.6	Switching to Other Ensembles	26
4	Multiple Time Steps Algorithms For Large Size Flexible Systems with Strong Electrostatic Interactions	
4.1	Subdivision of the “Bonded” Potential	28
4.2	The smooth particle mesh Ewald method	31
4.3	Subdivision the Non Bonded Potential	35
4.4	Electrostatic Corrections for the Multiple Time Step Simulation	36
5	The Hamiltonian Replica Exchange Method	40
5.1	Temperature REM	40
5.2	Hamiltonian REM	42
5.3	Analysis of the REM data	45
6	Metadynamics Simulation: history-dependent algorithms in Non-Boltzmann sampling	47
6.1	Implementation in ORAC	48
7	Steered Molecular Dynamics	51
7.1	The Crooks theorem	51
7.2	Determination of the potential of mean force via bidirectional non equilibrium techniques	54
7.3	Implementation in ORAC	55
8	Input to ORAC	58
8.1	General Features	58
8.2	Environments, Commands and Sub-commands	58
8.2.1	&ANALYSIS	60
8.2.2	&INOUT	61
8.2.3	&INTEGRATOR	66

8.2.4	&META	69
8.2.5	&PARAMETERS	73
8.2.6	&POTENTIAL	79
8.2.7	&PROPERTIES	90
8.2.8	&REM	97
8.2.9	&RUN	100
8.2.10	&SETUP	105
8.2.11	&SIMULATION	110
8.2.12	&SOLUTE	117
8.2.13	&SOLVENT	121
8.3	Input to ORAC : Force Field and Topology Files	124
8.3.1	Force Field Parameters	124
8.3.2	Topology	128
9	Compiling and Running ORAC	135
9.1	Compiling the Program	135
9.1.1	Serial version	135
9.1.2	Parallel version	136
9.2	How to set dimensions in ORAC : The <code>config.h</code> file	136

Preface

This manual is for version 5.0 of the program ORAC.¹

In this new release many improvements have been included. Here, we only mention the most important new features.

- Now ORAC may be run in parallel using the standard message passing interface libraries (OpenMPI, mpich2). The parallelism allows to run (Hamiltonian) replica exchange simulations and multiple walkers metadynamics simulations. The REM algorithm may be implemented in a “solute tempering” fashion allowing potential scaling only for limited (user selected) part of the simulated system,.
- ORAC can run steered molecular dynamics non equilibrium trajectories with on the fly work evaluation. The driven coordinate can be any combination of intramolecular coordinates (stretching bending and torsions). This features allows one to compute along the selected “reaction coordinate” the free energy profile (PMF) using non-equilibrium Jarzinsky and Crooks theorem. Steered molecular dynamics can be done as well by varying the temperature of the Nosé bath doing on the system an adimensional work computed according to the generalized Crooks theorem.
- Minimization routines has been improved by providing the possibility of minimizing only part of the solute by keeping frozen all other degrees of freedom.
- several ancillary programs are included in this distribution for post analysis of steered molecular dynamics and replica exchange simulation data.

The present manual is organized as follows: The first seven chapters constitute the ORAC theoretical background. Chapter 1) contains general and introductory remarks. Chapter 2) deals with symplectic and reversible integrators and introduces to the Liouvillean formalism, Chapter 3) extends the Liouvillean formalism to the extended Lagrangian methods and Chapter 4) describes how to deal with long range electrostatic interactions and how to combine the SPME method with the multilevel integration of the equations of motion in order to obtain efficient simulation algorithms. Chapter for 5 to 7 have been added in the present release. Chapter 5) contains an introduction to replica exchange techniques and a description on how such a technique has been implemented in the ORAC program. Chapter 6) deals with metadynamics simulations. Chapter 7) treats steered molecular dynamics simulation and the theory of non equilibrium processes. Chapter 8) is the command reference of the ORAC program. Chapter 9) contains instructions on how to compile and run ORAC in a serial and parallel environment.

¹ The ORAC program has been copyrighted (C) by Massimo Marchi and Piero Procacci 1995-2008. This program is free software; you can redistribute it and/or modify it under the terms of the **GNU General Public License** as published by the Free Software Foundation; either version 2 of the License, or (at your option) any later version.

This program is distributed in the hope that it will be useful, but **WITHOUT ANY WARRANTY**; without even the implied warranty of MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. See the GNU for more details. A general version of the GPL may be requested at: Free Software Foundation, Inc., 59 Temple Place - Suite 330, Boston, MA 02111-1307, USA.

Chapter 1

Atomistic simulations: an introduction

In this manual we describe ORAC , a program for the molecular dynamics (MD) simulation of atomistic models of complex molecular systems. In atomistic models the coordinates of all atomic nuclei including hydrogen are treated explicitly and interactions between distant atoms are represented by a pairwise additive dispersive-repulsive potential and a Coulomb contribution due to the atomic charges. Furthermore, nearby atoms interact through special two body, three body and four body functions representing the valence bonds, bending and torsional interaction energies surfaces.

The validity of such an approach as well as the reliability of the various potential models proposed in the literature [1, 2, 3, 4] is not the object of the present book. For reading on this topic, we refer to the extensive and ever growing literature [2, 5, 6, 7]. Here, we want only to stress the general concept that atomistic simulations usually have more predictive power than simplified models, but are also very expensive with respect to the latter from a computational standpoint. This predictive power stems from the fact that, in principle, simulations at the atomistic level do not introduce any uncontrolled approximation besides the obvious assumptions inherent in the definition of the potential model and do not assume any *a priori* knowledge of the system, except of course its chemical composition and topology. Therefore, the failure in predicting specific properties of the system for an atomistic simulation is due only to the inadequacy of the adopted interaction potential. We may define this statement as the *brute force postulate*. In practice, however, in order to reduce the computational burden, severe and essentially uncontrolled approximations such as neglect of long range interactions, suppression of degrees of freedom, dubious thermostating or constant pressure schemes are often undertaken. These approximations, however, lessen the predictive power of the atomistic approach and incorrect results may follow due to the inadequacy in the potential model, baseless approximations or combinations of the two. Also, due to their cost, the predictive capability of atomistic level simulations might often only be on paper, since in practice only a very limited phase space region can be accessed in an affordable way, thereby providing only biased and unreliable statistics for determining the macroscopic and microscopic behavior of the system. It is therefore of paramount importance in atomistic simulations to use computational techniques that do not introduced uncontrolled approximations and at the same time are efficient in the sampling of the complex and rugged conformational space of biological systems. Regarding this last issue, many progresses has been done recently by devising new both non-Boltzmann and Boltzmann techniques for extended sampling of complex systems. Chapter 6 and Chapter 7 are devoted to these aspects of atomistic molecular simulations.

1.1 Multiple time steps integration schemes and electrostatic interactions in complex biomolecular systems

As stated above, simulations of complex systems at the atomistic level, unlike simplified models, have the advantage of representing with realistic detail the full flexibility of the system and the potential energy surface according to which the atoms move. Both these physically significant ingredients of the atomistic approach unfortunately pose severe computational problems: on one hand the inclusion of full flexibility

necessarily implies the selection of small step size thereby reducing in a MD simulation the sampling power of the phase space. On the other hand, especially the evaluation of inter-particle long range interactions is an extremely expensive task using conventional methods, its computational cost scaling typically like N^2 (with N being the number of particles) quickly exceeding any reasonable limit. In this book we shall devote our attention to the methods, within the framework of classical MD simulations, that partially overcome the difficulties related to the presence of flexibility and long range interactions when simulating complex systems at the atomistic level.

Complex systems experience different kind of motions with different time scales: Intramolecular vibrations have a period not exceeding few hundreds of femtoseconds while reorientational motions and conformational changes have much longer time scales ranging from few picoseconds to hundreds of nanoseconds. In the intra-molecular dynamics one may also distinguish between fast stretching involving hydrogen with period smaller than 10 fs, stretching between heavy atoms and bending involving hydrogen with double period or more, slow bending and torsional movements and so on. In a similar manner in the diffusional regime many different contributions can be identified.

In a standard integration of Newtonian equations, all these motions, irrespectively of their time scales, are advanced using *the same* time step whose size is inversely proportional to the frequency of the *fastest* degree of freedom present in the system, therefore on the order of the femtosecond or even less. This constraint on the step size severely limits the accessible simulation time. One common way to alleviate the problem of the small step size is to freeze some supposedly irrelevant and fast degrees of freedom in the system. This procedure relies on the so-called SHAKE algorithm [8, 9, 10] that implements holonomic constraints while advancing the *Cartesian* coordinates. Typically, bonds and/or bending are kept rigid thus removing most of the high frequency density of states and allowing a moderate increase of the step size. The SHAKE procedure changes the input potential and therefore the output density of the states. Freezing degrees of freedom, therefore, requires in principle an *a priori* knowledge of the dynamical behavior of the system. SHAKE is in fact fully justified when the suppressed degrees of freedom do not mix with the “relevant” degrees of freedom. This might be “almost” true for fast stretching involving hydrogen which approximately defines an independent subspace of internal coordinates in almost all complex molecular systems [11] but may turn to be wrong in certain cases even for fast stretching between heavy atoms. In any case the degree of mixing of the various degrees of freedom of a complex system is not known *a priori* and should be on the contrary considered one of the targets of atomistic simulations. The SHAKE algorithm allows only a moderate increase of the step size while introducing, if used without caution, essentially uncontrolled approximations. In other words indiscriminate use of constraints violates the brute-force postulate. A more fruitful approach to the multiple time scale problem is to devise a more efficient “multiple time step” integration of the equation of motion. Multiple time step integration in MD is a relatively old idea [12, 13, 14, 15, 16, 17] but only in recent times, due to the work of Tuckerman, Martyna and Berne and coworkers [18, 19, 20, 21, 22, 23] is finding widespread application. These authors introduced a very effective formalism for devising multilevel integrators based on the symmetric factorization of the Liouvillean classical time propagator. The multiple time step approach allows integration of all degrees of freedom at an affordable computational cost. In the simulation of complex systems, for a well tuned multilevel integrator, the speed up can be sensibly larger than that obtained imposing bond constraints. Besides its efficiency, the multiple time steps approach has the advantage of not introducing any *a priori* assumption that may modify part of the density of the state of the system.

The Liouvillean approach to multiple time steps integrator lends itself to the straightforward, albeit tedious, application to extended Lagrangian systems for the simulation in the canonical and isobaric ensembles: once the equations of motions are known, the Liouvillean and hence the scheme, is *de facto* available. Many efficient multilevel schemes for constant pressure or constant temperature simulation are available in the literature [24, 23, 25].

As already outlined, long range interactions are the other stumbling block in the atomistic MD simulation of complex systems. The problem is particularly acute in biopolymers where the presence of distributed net charges makes the local potential oscillate wildly while summing, e.g. onto spherical non neutral shells. The conditionally convergent nature of the electrostatic energy series for a periodic system such as the MD box in periodic boundary conditions (PBC) makes any straightforward truncation method essentially unreliable [26, 10, 27].

The reaction field [28, 29] is in principle a physically appealing method that correctly accounts for long range effects and requires only limited computational effort. The technique assumes explicit electrostatic

interactions within a cutoff sphere surrounded by a dielectric medium which exerts back in the sphere a “polarization” or reaction field. The dielectric medium has a dielectric constant that matches that of the inner sphere. The technique has been proved to give results identical to those obtained with the *exact* Ewald method in Monte Carlo simulation of dipolar spherocylinders where the dielectric constant that enters in the reaction field is updated periodically according to the value found in the sphere. The reaction field method does however suffer of two major disadvantages that strongly limits its use in MD simulations of complex systems at the atomistic level: during time integration the system may experience instabilities related to the circulating dielectric constant of the reaction field and to the jumps into the dielectric of molecules in the sphere with explicit interactions. The other problem, maybe more serious, is that again the method requires an *a priori* knowledge of the system, that is the dielectric constant. In pure phases this might not be a great problem but in inhomogeneous systems such as solvated protein, the knowledge of the dielectric constant might be not easily available. Even with the circulating technique, an initial unreliable guess of the unknown dielectric constant, can strongly affect the dielectric behavior of the system and in turn its dynamical and structural state.

The electrostatic series can be computed in principle *exactly* using the Ewald re-summation technique [30, 31]. The Ewald method rewrites the electrostatic sum for the periodic system in terms of two absolutely convergent series, one in the direct lattice and the other in reciprocal lattice. This method, in its standard implementation, is extremely CPU demanding and scales like N^2 with N being the number of charges with the unfortunate consequence that even moderately large size simulations of inhomogeneous biological systems are not within its reach. The rigorous Ewald method, which does not suffers of none of the inconveniences experienced by the reaction field approach, has however regained resurgent interest very recently after publication by Darden, York and Pedersen [32] of the Particle Mesh technique and later on by Essmann, Darden *et al.* [33] of the variant termed Smooth Particle Mesh Ewald (SPME). SPME is based on older idea of Hockney [34] and is essentially an interpolation technique with a charge smearing onto a regular grid and evaluation via fast Fourier Transform (FFT) of the interpolated reciprocal lattice energy sums. The performances of this techniques, both in accuracy and efficiency, are astonishing. Most important, the computational cost scales like $N \log N$, that is essentially *linearly* for any practical application. Other algorithm like the Fast Multipole Method (FMM) [35, 36, 37, 38] scales exactly like N , even better than SPME. However FMM has a very large prefactor and the break even point with SPME is on the order of several hundred thousand of particles, that is, as up to now, beyond any reasonable practical limit.

The combination of the multiple time step algorithm and of the SPME [11] makes the simulation of large size complex molecular systems such as biopolymers, polar mesogenic molecules, organic molecules in solution etc., extremely efficient and therefore affordable even for long time spans. Further, this technique do not involve any uncontrolled approximation¹ and is perfectly consistent with standard PBC.

1.2 Enhanced sampling in atomistic simulations

Standard *equilibrium* simulations of complex biosystems are usually done on a single solvated biomolecule in PBC due to computational bounds.²

In these conditions the only mean to measure, e.g., the free energy differences between two given protein conformations, is to record the number of times the protein molecule in the MD cell is observed in each of the two conformations. Swaps between these conformers can take, in (time) average, as long as 0.1-1 microseconds [39] even for small proteins. One then needs to do extremely long equilibrium simulations in order to have a sufficient number of swaps between conformational states so as to determine with good accuracy a stationary (equilibrium) ratio of the conformational probability and hence the free energy

¹Of course SPME *is* itself an approximation of the true electrostatic energy. This approximation is however totally under control since the energy can be determined to any given accuracy and the effect of finite accuracy can be easily controlled on any computed property of the system. The approximation *is not* uncontrolled.

²The explicit (i.e. atomistic) solvent introduced in the MD cell is in fact the minimum amount required such that the distance between any two portion of *different* solute replicas is sufficiently large so as to assume negligible interprotein interactions. Also the shape of the MD cell is usually chosen so as to minimize the amount of explicit solvent whose sole role, at an extremely demanding computational cost, is to provide the correct dielectric medium for the biomolecule (including microsolvation effects). For example, globular (i.e. quasi-spherical) proteins are usually simulated in a dodecahedral box. Such a system, single solvated protein in PBC, is thus representative of *dilute* solution of biomolecules since the solute molecules in the periodic systems can never come close to each other, thereby *interacting*

difference. To give just a faint idea of the computational cost involved, for the simple system of decaalanine *in vacuo*, 1.0 microseconds of *serial* simulation takes about 10 days on 2.5 MH processor. Due to this computational bounds, standard molecular dynamics simulation of even small biological systems are in general not ergodic in the accessible simulation time. Typically, the system remains trapped during the whole simulation time span in a local minimum and the rare event of escaping the trap surmounting a free energy barrier never happens.

In order to overcome such severe sampling problem, many recent MD techniques has been devised. The Replica Exchange Method (REM)[40, 41, 42, 43] provides an elegant and simple solution to quasi-ergodic sampling. In REM, several independent trajectories, called replicas, are simultaneously generated in different thermodynamic conditions. The production of these simultaneous trajectories usually occurs on an array of parallel processors. The thermodynamics conditions of these replicas are chosen so as to span homogeneously the thermodynamic space from the ensemble of interest to a different ensemble with enhanced transition rates, where the sampling is ergodic. During the simulation, neighbouring replicas are allowed to exchange their configurations, subject to specific acceptance criteria. In this fashion, a trajectory is no longer bound to an unique given equilibrium ensemble but can randomly walk in a thermodynamic space of different equilibrium conditions, visiting ensembles where an ergodic sampling is possible, and then going back to the quasi-ergodic ensemble of interest. Therefore, REM is an algorithm which employs an extended ensemble formalism in order to overcome slow relaxation. The gain in sampling efficiency with respect to a series of uncoupled parallel trajectories comes from the exchange of information between trajectories, and the replica exchange process is the tool by which “information” (e.g. a particular configuration) is carried, for example, from an high to a low temperature. The REM algorithm can be used in principle without prior knowledge of the “important” reaction coordinates of the system, i.e., in the case of biological systems, those that defines the accessible conformational space in the target thermodynamics conditions. The identification of these *a priori* unknown coordinates, along with their underlying free energy surface, are actually one of the outputs of the method. The REM algorithm is described in detail in Chapter 4.4.

Once these important coordinates are known, one can use less expensive techniques to study the associated *essential* free energy surface. Canonical reweighting or Umbrella Sampling methods[44], for example, modify (bias) the interaction Hamiltonian of the system in such a way to facilitate barrier crossing between conformational basins. The canonical average of the unperturbed systems are then reconstructed by appropriately reweighting the biased averaged.

Quasi-equilibrium techniques[45, 46, 47, 48] builds such biasing potential that favours barrier crossing by periodically adding a small perturbation to the system Hamiltonian so as to progressively flatten the free energy surfaces along selected reaction coordinates. For example, in the so-called standard “metadynamics” simulation method[45], a history-dependent potential, made of accumulated Gaussian functions deposited continuously at the instantaneous values of the given reaction coordinates, is imposed to the system. The history-dependent potential disfavors configurations in the space of the reaction coordinates that have already been visited, and it has been shown, by appropriately adjusting system dependent parameters, to numerically converge to the inverse of the free energy surface.[49] In the present version of ORAC the metadynamics technique has been implemented in the parallel version whereby multiple metadynamics simulation (walkers) are run in parallel cooperatively building a common history dependent potential which is passed among all processes. The history dependent potential is generally defined over a multidimensional domain involving several reaction coordinates. Metadynamics can be used, e.g., to identify the minimum free energy path between two metastable protein states defining the reactants and products of an elementary chemical reaction. Quasi-equilibrium techniques in biological systems converge rather slowly since the convergence rate depends crucially on the inherent slow diffusion along the conformational coordinates. So even if the potential is relatively flattened, the diffusion along a nearly free reaction coordinates can still be slow due to the friction of the orthogonal coordinates. The metadynamics algorithm is described in detail in Chapter 5.3

Non equilibrium techniques[50, 51, 52, 53] uses an additional driving potential acting on an appropriate reaction coordinates to *fast* steer the system from a given equilibrium initial state to a given final state, and *viceversa* producing a series of forward and reverse non equilibrium trajectories. The driven coordinate is strictly mono-dimensional but can be defined as a trajectory in a multidimensional reaction coordinate space. The free energy differences between the initial and final states (the reactants and the products) is connected, through the Crooks fluctuation theorem[51], to the ratio of distribution functions of the work spent in these trajectories. Free energy reconstruction, using non equilibrium steered molecular dynamics,

of the potential of mean force[54] along one arbitrary reaction coordinate is described in detail in Chapter 6.1.

Chapter 2

Symplectic and Reversible Integrators

In an Hamiltonian problem, the symplectic condition and microscopic reversibility are inherent properties of the true time trajectories which, in turn, are the exact solution of Hamilton's equation. A stepwise integration defines a t -flow mapping which may or may not retain these properties. Non symplectic and/or non reversible integrators are generally believed [55, 56, 57, 58] to be less stable in the long-time integration of Hamiltonian systems. In this section we shall illustrate the concept of reversible and symplectic mapping in relation to the numerical integration of the equations of motion.

2.1 Canonical Transformation and Symplectic Conditions

Given a system with n generalized coordinates q , n conjugated momenta p and Hamiltonian H , the corresponding Hamilton's equations of motion are

$$\begin{aligned}\dot{q}_i &= \frac{\partial H}{\partial p_i} \\ \dot{p}_i &= -\frac{\partial H}{\partial q_i} \quad i = 1, 2, \dots, n\end{aligned}\tag{2.1}$$

These equations can be written in a more compact form by defining a column matrix with $2n$ elements such that

$$\mathbf{x} = \begin{pmatrix} \mathbf{q} \\ \mathbf{p} \end{pmatrix}.\tag{2.2}$$

In this notation the Hamilton's equations (2.1) can be compactly written as

$$\dot{\mathbf{x}} = \mathbf{J} \frac{\partial H}{\partial \mathbf{x}} \quad \mathbf{J} = \begin{pmatrix} \mathbf{0} & \mathbf{1} \\ -\mathbf{1} & \mathbf{0} \end{pmatrix}\tag{2.3}$$

where \mathbf{J} is a $2n \times 2n$ matrix, $\mathbf{1}$ is an $n \times n$ identity matrix and $\mathbf{0}$ is a $n \times n$ matrix of zeroes. Eq. (2.3) is the so-called symplectic notation for the Hamilton's equations.¹

Using the same notation we now may define a transformation of variables from $\mathbf{x} \equiv \{q, p\}$ to $\mathbf{y} \equiv \{Q, P\}$ as

$$\mathbf{y} = \mathbf{y}(\mathbf{x})\tag{2.4}$$

For a restricted canonical transformation [59, 60] we know that the function $H(\mathbf{x})$ expressed in the new coordinates \mathbf{y} serves as the Hamiltonian function for the new coordinates \mathbf{y} , that is the Hamilton's equations

¹Symplectic means "intertwined" in Greek and refers to the interlaced role of coordinate and momenta in Hamilton's equations.

of motion in the \mathbf{y} basis have exactly the same form as in Eq. (2.3):

$$\dot{\mathbf{y}} = \mathbf{J} \frac{\partial H}{\partial \mathbf{y}} \quad (2.5)$$

If we now take the time derivative of Eq. (2.4), use the chain rule relating \mathbf{x} and \mathbf{y} derivatives and use Eq. (2.5), we arrive at

$$\dot{\mathbf{y}} = \mathbf{M} \mathbf{J} \mathbf{M}^t \frac{\partial H}{\partial \mathbf{y}}. \quad (2.6)$$

Here \mathbf{M} is the Jacobian matrix with elements

$$M_{ij} = \partial y_i / \partial x_j, \quad (2.7)$$

and \mathbf{M}^t is its transpose. By comparing Eqs. (2.5) and (2.6), we arrive at the conclusion that a transformation is canonical if, and only if, the Jacobian matrix \mathbf{M} of the transformation Eq. 2.4 satisfies the condition

$$\mathbf{M} \mathbf{J} \mathbf{M}^t = \mathbf{J}. \quad (2.8)$$

Eq. (2.8) is known as the symplectic condition for canonical transformations and represents an effective tool to test whether a generic transformation is canonical. Canonical transformations play a key role in Hamiltonian dynamics. For example, consider transformation ϕ

$$\mathbf{z}(t) = \phi(t, \mathbf{z}(0)) \quad (2.9)$$

where $\{p_0 q_0\} \equiv \mathbf{z}(0)$ and $\{P, Q\} \equiv \mathbf{z}(t)$, i.e. one writes the coordinates and momenta at time t , obtained from the solution of the Hamiltonian equation of motion, *as a function* of the coordinates and momenta at the initial time zero. This transformation, which depends on the scalar parameter t , is trivially canonical since both $\{p_0 q_0\}$ and $\{P, Q\}$ satisfies the Hamilton equations of motion. Hence the above transformation defines the t -flow mapping of the systems and, being canonical, its Jacobian matrix obeys the symplectic condition (2.8). An important consequence of the symplectic condition, is the invariance under canonical (or symplectic) transformations of many properties of the phase space. These invariant properties are known as ‘‘Poincare invariants’’ or canonical invariants. For example transformations or t -flow’s mapping obeying Eq. (2.8) preserve the phase space volume. This is easy to see, since the infinitesimal volume elements in the \mathbf{y} and \mathbf{x} bases are related by

$$d\mathbf{y} = |\det \mathbf{M}| d\mathbf{x} \quad (2.10)$$

where $|\det \mathbf{M}|$ is the Jacobian of the transformation. Taking the determinant of the symplectic condition Eq. (2.8) we see that $|\det \mathbf{M}| = 1$ and therefore

$$d\mathbf{y} = d\mathbf{x}. \quad (2.11)$$

For a canonical or symplectic t -flow mapping this means that the phase total space volume is invariant and therefore Liouville theorem is automatically satisfied.

A stepwise numerical integration scheme defines a Δt -flow mapping or equivalently a coordinates transformation, that is

$$\begin{aligned} Q(\Delta t) &= Q(q(0), p(0), \Delta t) \\ P(\Delta t) &= P(q(0), p(0), \Delta t) \end{aligned} \quad \mathbf{y}(\Delta t) = \mathbf{y}(\mathbf{x}(0)). \quad (2.12)$$

We have seen that exact solution of the Hamilton equations has t -flow mapping satisfying the symplectic conditions (2.8). If the Jacobian matrix of the transformation (2.12) satisfies the symplectic condition then the integrator is termed to be symplectic. The resulting integrator, therefore, exhibits properties identical to those of the exact solution, in particular it satisfies Eq. (2.11). Symplectic algorithms have also been proved to be robust, i.e resistant to time step increase, and generate stable long time trajectory, i.e. they do not show drifts of the total energy. Popular MD algorithms like Verlet, leap frog and velocity Verlet are all symplectic and their robustness is now understood to be due in part to this property. [57, 18, 21, 58]

2.2 Liouville Formalism: a Tool for Building Symplectic and Reversible Integrators

In the previous paragraphs we have seen that it is highly beneficial for an integrator to be symplectic. We may now wonder if there exists a general way for obtaining symplectic and possibly, reversible integrators from “first principles”. To this end, we start by noting that for any property which depends on time implicitly through $p, q \equiv \mathbf{x}$ we have

$$\begin{aligned} \frac{dA(p, q)}{dt} &= \sum_{q, p} \left(\dot{q} \frac{\partial A}{\partial q} + \dot{p} \frac{\partial A}{\partial p} \right) = \sum_{q, p} \left(\frac{\partial H}{\partial p} \frac{\partial A}{\partial q} - \frac{\partial H}{\partial q} \frac{\partial A}{\partial p} \right) \\ &= iLA \end{aligned} \quad (2.13)$$

where the sum is extended to all n degrees of freedom in the system. L is the Liouvillean operator defined by

$$iL \equiv \sum_{q, p} \left(\dot{q} \frac{\partial}{\partial q} + \dot{p} \frac{\partial}{\partial p} \right) = \sum_{q, p} \left(\frac{\partial H}{\partial p} \frac{\partial}{\partial q} - \frac{\partial H}{\partial q} \frac{\partial}{\partial p} \right). \quad (2.14)$$

Eq. (2.13) can be integrated to yield

$$A(t) = e^{iLt} A(0). \quad (2.15)$$

If A is the state vector itself we can use Eq. (2.15) to integrate Hamilton’s equations:

$$\begin{bmatrix} q(t) \\ p(t) \end{bmatrix} = e^{iLt} \begin{bmatrix} q(0) \\ p(0) \end{bmatrix}. \quad (2.16)$$

The above equation is a formal solution of Hamilton’s equations of motion. The exponential operator e^{iLt} times the state vector defines the t -flow of the Hamiltonian system which brings the system phase space point from the initial state q_0, p_0 to the state $p(t), q(t)$ at a later time t . We already know that this transformation obeys Eq. (2.8). We may also note that the adjoint of the exponential operator corresponds to the inverse, that is e^{iLt} is unitary. This implies that the trajectory is exactly time reversible. In order to build our integrator, we now define the *discrete time propagator* $e^{iL\Delta t}$ as

$$e^{iLt} = \left[e^{iLt/n} \right]^n; \quad \Delta t = t/n \quad (2.17)$$

$$e^{iL\Delta t} = e^{\sum_{q, p} (\dot{q} \frac{\partial}{\partial q} + \dot{p} \frac{\partial}{\partial p}) \Delta t}. \quad (2.18)$$

In principle, to evaluate the action of $e^{iL\Delta t}$ on the state vector p, q one should know the derivatives of all orders of the potential V . This can be easily seen by Taylor expanding the discrete time propagator $e^{iL\Delta t}$ and noting that the operator $\dot{q} \partial / \partial q$ does not commute with $-\partial V / \partial q (\partial / \partial p)$ when the coordinates and momenta refer to *same* degree of freedom. We seek therefore approximate expressions of the discrete time propagator that retain both the symplectic and the reversibility property. For any two linear operators A, B the Trotter formula [61] holds:

$$e^{(A+B)t} = \lim_{n \rightarrow \infty} (e^{At/n} e^{Bt/n})^n \quad (2.19)$$

We recognize that the propagator Eq. (2.18) has the same structure as the left hand side of Eq. (2.19); hence, using Eq. (2.19), we may write for Δt sufficiently small

$$e^{iL\Delta t} = e^{(\dot{q} \frac{\partial}{\partial q} + \dot{p} \frac{\partial}{\partial p}) \Delta t} \simeq e^{\dot{q} \frac{\partial}{\partial q} \Delta t} e^{\dot{p} \frac{\partial}{\partial p} \Delta t} + O(\Delta t^2) \quad (2.20)$$

Where, for simplicity of discussion, we have omitted the sum over q and p in the exponential. Eq. (2.20) is exact in the limit that $\Delta t \rightarrow 0$ and is first order for finite step size. Using Eq. (2.8) it is easy to show that the t -flow defined in Eq. (2.20) is symplectic, being the product of two successive symplectic transformations. Unfortunately, the propagator Eq. (2.20) is not unitary and therefore the corresponding algorithm is not

time reversible. Again the non unitarity is due to the fact that the two factorized exponential operators are non commuting. We can overcome this problem by halving the time step and using the approximant:

$$e^{(A+B)t} \simeq e^{At/2} e^{Bt/2} e^{Bt/2} e^{At/2} = e^{At/2} e^{Bt} e^{At/2}. \quad (2.21)$$

The resulting propagator is clearly unitary, therefore time reversible, and is also correct to the second order [62]. Thus, requiring that the product of the exponential operator be unitary, automatically leads to more accurate approximations of the true discrete time propagator [63, 62]. Applying the same argument to the propagator (2.18) we have

$$e^{iL\Delta t} = e^{(\dot{q}\frac{\partial}{\partial q} + \dot{p}\frac{\partial}{\partial p})\Delta t} \simeq e^{\dot{p}\frac{\partial}{\partial p}\Delta t/2} e^{\dot{q}\frac{\partial}{\partial q}\Delta t} e^{\dot{p}\frac{\partial}{\partial p}\Delta t/2} + O(\Delta t^3). \quad (2.22)$$

The action of an exponential operator $e^{(a\partial/\partial x)}$ on a generic function $f(x)$ trivially corresponds to the Taylor expansion of $f(x)$ around the point x at the point $x + a$, that is

$$e^{a\partial/\partial x} f(x) = f(x + a). \quad (2.23)$$

Using Eq. (2.23), the time reversible and symplectic integration algorithm can now be derived by acting with our Hermitian operator Eq. (2.22) onto the state vector at $t = 0$ to produce updated coordinate and momenta at a later time Δt . The resulting algorithm is completely equivalent to the well known velocity Verlet:

$$\begin{aligned} p(\Delta t/2) &= p(0) + F(0)\Delta t/2 \\ q(\Delta t) &= q(0) + \left(\frac{p(\Delta t/2)}{m}\right)\Delta t \\ p(\Delta t) &= p(\Delta t/2) + F(\Delta t)\Delta t/2. \end{aligned} \quad (2.24)$$

We first notice that each of the three transformations obeys the symplectic condition Eq. (2.8) and has a Jacobian determinant equal to one. The product of the three transformation is also symplectic and, thus, phase volume preserving. Finally, since the discrete time propagator (2.22) is unitary, the algorithm is time reversible.

One may wonder what it is obtained if the operators $\dot{q}\partial/\partial q$ and $-\partial V/\partial q(\partial/\partial p)$ are exchanged in the definition of the discrete time propagator (2.22). If we do so, the new integrator is

$$\begin{aligned} q(\Delta t/2) &= q(0) + \frac{p(0)}{m}\Delta t/2 \\ p(\Delta t) &= p(0) + F[q(\Delta t/2)]\Delta t \\ q(\Delta t) &= q(\Delta t/2) + \frac{p(\Delta t)}{m}\Delta t/2. \end{aligned} \quad (2.25)$$

This algorithm has been proved to be equivalent to the so-called Leap-frog algorithm [64]. Tuckerman *et al.* [18] called this algorithm *position Verlet* which is certainly a more appropriate name in the light of the exchanged role of positions and velocities with respect to the velocity Verlet. Also, Eq. (2.21) clearly shows that the position Verlet is essentially identical to the Velocity Verlet. A shift of a time origin by $\Delta t/2$ of either Eq. (2.25) or Eq. (2.24) would actually make both integrator perfectly equivalent. However, as pointed out in Ref. [19], half time steps are not formally defined, being the right hand side of Eq. (2.21) an approximation of the discrete time propagator for the *full* step Δt . Velocity Verlet and Position Verlet, therefore, do not generate numerically identical trajectories although of course the trajectories are similar.

We conclude this section by saying that is indeed noticeable that using the same Liouville formalism different long-time known schemes can be derived. The Liouville approach represent therefore a unifying treatment for understanding the properties and relationships between stepwise integrators.

2.3 Potential Subdivision and Multiple Time Steps Integrators for NVE Simulations

The ideas developed in the preceding sections can be used to build multiple time step integrators. Multiple time step integration is based on the concept of reference system. Let us now assume that the system

potential V be subdivided in n terms such that

$$V = V_0 + V_1 + \dots + V_n. \quad (2.26)$$

Additionally, we suppose that the corresponding average values of the square modulus of the forces $F_k = |\partial V_k / \partial \mathbf{x}|$ and of their time derivatives $\dot{F}_k = |d/dt(\partial V_k / \partial \mathbf{x})|$ satisfy the following condition:

$$\begin{aligned} F_0^2 &>> F_1^2 >> \dots >> F_n^2 \\ \dot{F}_0^2 &>> \dot{F}_1^2 >> \dots >> \dot{F}_n^2. \end{aligned} \quad (2.27)$$

These equations express the situation where different time scales of the system correspond to different pieces of the potential. Thus, the Hamiltonian of the k -th *reference system* is defined as

$$H = T + V_0 + \dots + V_k, \quad (2.28)$$

with a perturbation given by:

$$P = V_{k+1} + V_{k+2} + \dots + V_n. \quad (2.29)$$

For a general subdivision of the kind given in Eq. (2.26) there exist n reference *nested* systems. In the general case of a flexible molecular systems, we have fast degrees of freedom which are governed by the stretching, bending and torsional potentials and by slow intermolecular motions driven by the intermolecular potential. As we shall discuss with greater detail in section 4, in real systems there is no clear cut condition between intra and intermolecular motions since their time scales may well overlap in many cases. The conditions Eq. (2.27) are, hence, never fully met for any of all possible potential subdivisions.

Given a potential subdivision Eq. (2.26), we now show how a multi-step scheme can be built with the methods described in section 2.2. For the sake of simplicity, we subdivide the interaction potential of a molecular system into two components only: One intra molecular, V_0 , generating mostly “fast” motions and the other intermolecular, V_1 , driving slower degrees of freedom. Generalization of the forthcoming discussion to a n -fold subdivision, Eq. (2.26), is then straightforward.

For the 2-fold inter/intra subdivision, the system with Hamiltonian $H = T + V_0$ is called the intra-molecular reference system whereas V_1 is the intermolecular perturbation to the reference system. Correspondingly, the Liouvillean may be split as

$$\begin{aligned} iL_0 &= \dot{q} \frac{\partial}{\partial q} - \frac{\partial V_0}{\partial q} \frac{\partial}{\partial p} \\ iL_1 &= - \frac{\partial V_1}{\partial q} \frac{\partial}{\partial p}. \end{aligned} \quad (2.30)$$

Here L_0 is the Liouvillean of the 0-th reference system with Hamiltonian $T + V_0$, while L_1 is a perturbation Liouvillean. Let us now suppose now that Δt_1 is a good time discretization for the time evolution of the perturbation, that is for the slowly varying intermolecular potential. The discrete $e^{iL\Delta t_1} \equiv e^{(iL_0 + iL_1)\Delta t_1}$ time propagator can be factorized as

$$\begin{aligned} e^{iL\Delta t_1} &= e^{iL_1\Delta t_1/2} (e^{iL_0\Delta t_1/n})^n e^{iL_1\Delta t_1/2} \\ &= e^{iL_1\Delta t_1/2} (e^{iL_0\Delta t_0})^n e^{iL_1\Delta t_1/2}, \end{aligned} \quad (2.31)$$

where we have used Eq. (2.21) and we have defined

$$\Delta t_0 = \frac{\Delta t_1}{n} \quad (2.32)$$

as the time step for the “fast” reference system with Hamiltonian $T + V_0$. The propagator (2.31) is unitary and hence time reversible. The external propagators depending on the Liouvillean L_1 acting on the state vectors define a symplectic mapping, as it can be easily proved by using Eq. (2.8). The full factorized propagator is therefore symplectic as long as the inner propagator is symplectic. The Liouvillean $iL_0 \equiv \dot{q} \partial / \partial q - \partial V_0 / \partial q \partial / \partial p$ can be factorized according to the Verlet symplectic and reversible breakup

described in the preceding section, but with an Hamiltonian $T + V_0$. Inserting the result into Eq. (2.31) and using the definition (2.30), the resulting double time step propagator is then

$$e^{iL\Delta t_1} = e^{\frac{-\partial V_1}{\partial q} \frac{\partial}{\partial p} \Delta t_1/2} \left(e^{\frac{-\partial V_0}{\partial q} \frac{\partial}{\partial p} \Delta t_0/2} e^{i\frac{\partial}{\partial q} \Delta t_0} e^{\frac{-\partial V_0}{\partial q} \frac{\partial}{\partial p} \Delta t_0/2} \right)^n e^{\frac{-\partial V_1}{\partial q} \frac{\partial}{\partial p} \Delta t_1/2} \quad (2.33)$$

This propagator is unfolded straightforwardly using the rule (2.23) generating the following symplectic and reversible integrator from step $t = 0$ to $t = \Delta t_1$:

$$\begin{aligned} p \left(\frac{\Delta t_1}{2} \right) &= p(0) + F_1(0) \frac{\Delta t_1}{2} \\ \text{DO} \quad i=1, n & \\ p \left(\frac{\Delta t_1}{2} + i \frac{\Delta t_0}{2} \right) &= p \left(\frac{\Delta t_1}{2} + [i-1] \frac{\Delta t_0}{2} \right) + F_0 \left([i-1] \frac{\Delta t_0}{2} \right) \frac{\Delta t_0}{2} \\ q(i\Delta t_0) &= q([i-1]\Delta t_0) + p \left(\frac{\Delta t_1}{2} + i \frac{\Delta t_0}{2} \right) \frac{\Delta t_0}{m} \\ p \left(\frac{\Delta t_1}{2} + i\Delta t_0 \right) &= p \left(\frac{\Delta t_1}{2} + i \frac{\Delta t_0}{2} \right) + F_0(i\Delta t_0) \frac{\Delta t_0}{2} \\ \text{ENDDO} & \\ p(\Delta t_1) &= p' \left(\frac{\Delta t_1}{2} \right) + F_1(n\Delta t_0) \frac{\Delta t_1}{2} \end{aligned} \quad (2.34)$$

Note that the slowly varying forces F_1 are felt only at the beginning and the end of the macro-step² Δt_1 . In the inner n steps loop the system moves only according to the Hamiltonian of the reference system $H = T + V_0$. When using the potential breakup, the inner reference system is rigorously conservative and the total energy of the reference system (i.e. $T + V_0 + \dots + V_k$) is conserved during the P micro-steps.³

The integration algorithm given an arbitrary subdivision of the interaction potential is now straightforward. For the general subdivision (2.26) the corresponding Liouvillean split is

$$iL_0 = i\dot{q} \frac{\partial}{\partial q} - \frac{\partial V_i}{\partial q} \frac{\partial}{\partial p}; \quad iL_1 = -\frac{\partial V_1}{\partial q} \frac{\partial}{\partial p} \quad \dots; \quad iL_n = -\frac{\partial V_k}{\partial q} \frac{\partial}{\partial p}. \quad (2.35)$$

We write the discrete time operator for the Liouville operator $iL = L_0 + \dots L_n$ and use repeatedly the Hermitian approximant and Trotter formula to get a hierarchy of *nested* reference systems propagator, viz.

$$e^{i(\sum_{i=0}^n L_i)\Delta t_n} = e^{iL_n \frac{\Delta t_n}{2}} \left(e^{i(\sum_{i=0}^{n-1} L_i)\Delta t_{n-1}} \right)^{P_{n-1}} e^{iL_n \frac{\Delta t_n}{2}} \quad (2.36)$$

$$\begin{aligned} \Delta t_n &= \Delta t_{n-1} P_{n-1} \\ e^{i(\sum_{i=0}^{n-1} L_i)\Delta t_{n-1}} &= e^{iL_{n-1} \frac{\Delta t_{n-1}}{2}} \left(e^{i(\sum_{i=0}^{n-2} L_i)\Delta t_{n-2}} \right)^{P_{n-2}} e^{iL_{n-1} \frac{\Delta t_{n-1}}{2}} \end{aligned} \quad (2.37)$$

$$\begin{aligned} \Delta t_{n-1} &= \Delta t_{n-2} P_{n-2} \\ &\dots \\ e^{i(L_0+L_1+L_2)\Delta t_2} &= e^{iL_2 \frac{\Delta t_2}{2}} \left(e^{i(L_1+L_0)\Delta t_1} \right)^{P_1} e^{iL_2 \frac{\Delta t_2}{2}} \end{aligned} \quad (2.38)$$

$$\begin{aligned} \Delta t_2 &= \Delta t_1 P_1 \\ e^{i(L_0+L_1)\Delta t_1} &= e^{iL_1 \frac{\Delta t_1}{2}} \left(e^{iL_0\Delta t_0} \right)^{P_0} e^{iL_1 \frac{\Delta t_1}{2}} \\ \Delta t_1 &= \Delta t_0 P_0 \end{aligned} \quad (2.39)$$

where Δt_i is the generic integration time steps selected according to the time scale of the i -th force F_i . We now substitute Eq. (2.39) into Eq. (2.38) and so on climbing the whole hierarchy until Eq. (2.36). The resulting multiple time steps symplectic and reversible propagator is then

$$\begin{aligned} e^{iL\Delta t_n} &= e^{F_n \frac{\partial}{\partial p} \Delta t_n} \left(e^{F_{n-1} \frac{\partial}{\partial p} \frac{\Delta t_{n-1}}{2}} \dots \right. \\ &\dots \left(e^{F_0 \frac{\partial}{\partial p} \frac{\Delta t_0}{2}} e^{i\frac{\partial}{\partial q} \Delta t_0} e^{F_0 \frac{\partial}{\partial p} \frac{\Delta t_0}{2}} \right)^{P_0} \dots e^{F_{n-1} \frac{\partial}{\partial p} \frac{\Delta t_{n-1}}{2}} \left. \right)^{P_{n-1}} e^{F_n \frac{\partial}{\partial p} \frac{\Delta t_n}{2}} \end{aligned} \quad (2.40)$$

²When the large step size at which the intermittent impulses are computed matches the period of natural oscillations in the system, one can detect instabilities of the numerical integration due to resonance effects. Resonances occurs for pathological systems such as fast harmonic oscillators in presence of strong, albeit slowly varying, forces [57] and can be cured easily by tuning the time steps in the multilevel integration. However, for large and complex molecules it is unlikely that an artificial resonance could sustain for any length of time [57]

³In the original *force* breakup [17, 18], the energy is not generally conserved during the unperturbed motion of the inner reference systems but only at the end of the full macro-step. Force breakup and potential breakup have been proved to produce identical trajectories [21]. With respect to the force the breakup, implementation of the potential breakup is slightly more complicated when dealing with intermolecular potential separation, but the energy conservation requirement in any defined reference system makes the debugging process easier.

The integration algorithm that can be derived from the above propagator was first proposed by Tuckerman, Martyna and Berne and called r-RESPA, reversible reference system propagation algorithm [18]

2.4 Constraints and r-RESPA

The r-RESPA approach makes unnecessary to resort to the SHAKE procedure [8, 9] to freeze some fast degrees of freedom. However the SHAKE and r-RESPA algorithms are not mutually exclusive and sometimes it might be convenient to freeze some degrees of freedom while simultaneously using a multi-step integration for all other freely evolving degrees of freedom. Since r-RESPA consists in a series of nested velocity Verlet like algorithms, the constraint technique RATTLE [65] used in the past for single time step velocity Verlet integrator can be straightforwardly applied. In RATTLE both the constraint conditions on the coordinates and their time derivatives must be satisfied. The resulting coordinate constraints is upheld by a SHAKE iterative procedure which corrects the positions exactly as in a standard Verlet integration, while a similar iterative procedure is applied to the velocities at the half time step.

In a multi time step integration, whenever velocities are updated, using part of the overall forces (e.g. the intermolecular forces), they must also be corrected for the corresponding constraints forces with a call to RATTLE. This combined RATTLE-r-RESPA procedure has been described for the first time by Tuckerman and Parrinello [66] in the framework of the Car-Parrinello simulation method. To illustrate the combined RATTLE-r-RESPA technique in a multi-step integration, we assume a separation of the potential into two components deriving from intramolecular and intermolecular interactions. In addition, some of the covalent bonds are supposed rigid, i.e.

$$d_a = d_a^{(0)} \quad (2.41)$$

$$\dot{d}_a = 0 \quad (2.42)$$

equation where a runs over all constrained bonds and $d_a^{(0)}$ are constants. In the double time integration (2.34), velocities are updated four times, i.e. two times in the inner loop and two times in the outer loop. To satisfy (2.41), SHAKE must be called to correct the position in the inner loop. To satisfy (2.42), RATTLE must be called twice, once in the inner loop and the second time in the outer loop according to the following scheme

$$\begin{aligned}
 p' \left(\frac{\Delta t_1}{2} \right) &= p(0) + F_1(0) \frac{\Delta t_1}{2} \\
 p \left(\Delta t_1 \right) &= RATTLE_p \{ p' \left(\frac{\Delta t_1}{2} \right) \} \\
 \text{DO} \quad i=1, n \\
 & p' \left(\frac{\Delta t_1}{2} + i \frac{\Delta t_0}{2} \right) = p \left(\frac{\Delta t_1}{2} + [i-1] \frac{\Delta t_0}{2} \right) + F_0 \left([i-1] \frac{\Delta t_0}{2} \right) \frac{\Delta t_0}{2} \\
 & p \left(\frac{\Delta t_1}{2} + i \frac{\Delta t_0}{2} \right) = RATTLE_p \{ p' \left(\frac{\Delta t_1}{2} + i \frac{\Delta t_0}{2} \right) \} \\
 & q' (i \Delta t_0) = q \left([i-1] \Delta t_0 \right) + p \left(\frac{\Delta t_1}{2} + i \frac{\Delta t_0}{2} \right) \frac{\Delta t_0}{m} \\
 & q (i \Delta t_0) = RATTLE_q \{ q' (i \Delta t_0) \} \\
 & p \left(\frac{\Delta t_1}{2} + i \Delta t_0 \right) = p \left(\frac{\Delta t_1}{2} + i \frac{\Delta t_0}{2} \right) + F_0 (i \Delta t_0) \frac{\Delta t_0}{2} \\
 \text{ENDDO} \\
 p \left(\Delta t_1 \right) &= p' \left(\frac{\Delta t_1}{2} \right) + F_1(n \Delta t_0) \frac{\Delta t_1}{2}.
 \end{aligned} \quad (2.43)$$

Where $RATTLE_p$ and $RATTLE_q$ represent the constraint procedure on velocity and coordinates, respectively.

2.5 Applications

As a first simple example we apply the double time integrator (2.34) to the NVE simulation of flexible nitrogen at 100 K.

The overall interaction potential is given by

$$V = V_{intra} + V_{inter}$$

Where V_{inter} is the intermolecular potential described by a Lennard-Jones model between all nitrogen atoms on different molecules [67]. V_{intra} is instead the intramolecular stretching potential holding together

Table 2.1: Energy conservation ratio R for various integrators (see text). The last three entries refer to a velocity Verlet with bond constraints. $\langle V_i \rangle$ and $\langle V_m \rangle$ are the average value of the intra-molecular and intermolecular energies (in KJ/mole), respectively. CPU is given in seconds per picoseconds of simulation and Δt in fs . Single time step velocity Verlet with $\Delta t = 4.5 fs$ is unstable.

Δt	n	R	CPU	$\langle V_i \rangle$	$\langle V_m \rangle$
0.3	1	0.005	119	0.1912	-4.75
0.6	1	0.018	62	0.1937	-4.75
1.5	1	0.121	26	0.2142	-4.75
4.5	1	-	-	-	-
0.6	2	0.004	59	0.1912	-4.75
1.5	5	0.004	28	0.1912	-4.75
3.0	10	0.005	18	0.1912	-4.75
4.5	15	0.006	15	0.1912	-4.75
6.0	20	0.008	12	0.1912	-4.74
9.0	30	0.012	10	0.1911	-4.74
3.0	-	0.001	14	-	-4.74
6.0	-	0.004	8	-	-4.75
9.0	-	0.008	6	-	-4.74

the two nitrogen atoms of each given molecule. We use here a simple harmonic spring depending on the molecular bond length r_m , namely:

$$V_i = \frac{1}{2} \sum_m k(r - r_0)^2,$$

with r_0 and r the equilibrium and instantaneous distance between the nitrogen atoms, and k the force constant tuned to reproduce the experimental gas-phase stretching frequency [68].

As a measure of the accuracy of the numerical integration we use the adimensional energy conservation ratio [21, 69, 22, 11]

$$R = \frac{\langle E^2 \rangle - \langle E \rangle^2}{\langle K^2 \rangle - \langle K \rangle^2} \quad (2.44)$$

where E and K are the total and kinetic energy of the system, respectively. In table 1 we show the energy conservation ratio R and CPU timings on a IBM-43P/160MH/RS6000 obtained for flexible nitrogen at 100 K with the r-RESPA integrator as a function of n and Δt_1 in Eq. (2.34) and also for single time step integrators. Results of integrators for rigid nitrogen using SHAKE are also shown for comparison. The data in Table 1 refer to a 3.0 ps run without velocity rescaling. They were obtained starting all runs from coordinates corresponding to the experimental $Pa3$ structure [70, 71] of solid nitrogen and from velocities taken randomly according to the Boltzmann distribution at 100 K.

The entry in bold refers to the “exact” result, obtained with a single time step integrator with a very small step size of 0.3 fs. Note that R increases quadratically with the time step for single time step integrators whereas r-RESPA is remarkably resistant to outer time step size increase. For example r-RESPA with $\Delta t_1 = 9.0 fs$ and $P = 30$ (i.e. $\Delta t_0 = 0.3 fs$) yields better accuracy on energy conservation than single time step Velocity Verlet with $\Delta t = 0.6 fs$ does, while being more than six times faster. Moreover, r-RESPA integrates all degrees of freedom of the systems and is almost as efficient as Velocity Verlet with constraints on bonds. It is also worth pointing out that energy averages for all r-RESPA integrators is equal to the exact value, while at single time step even a moderate step size increase results in sensibly different averages intra-molecular energies. As a more complex example we now study a cluster of eight single chain alkanes $C_{24}H_{50}$. In this case the potential contains stretching, bending and torsional contributions plus the intermolecular Van-der-Waals interactions between non bonded atoms. The parameter are chosen according to the AMBER protocol [2] by assigning the carbon and hydrogen atoms to the AMBER types **ct** and **hc**, respectively. For various dynamical and structural properties we compare three integrators, namely a triple time step r-RESPA (R3) a single time step integrator with bond constraints on $X-H$ (S1) and a single time step integrator with all bonds kept rigid (S). These three integrators are tested, starting

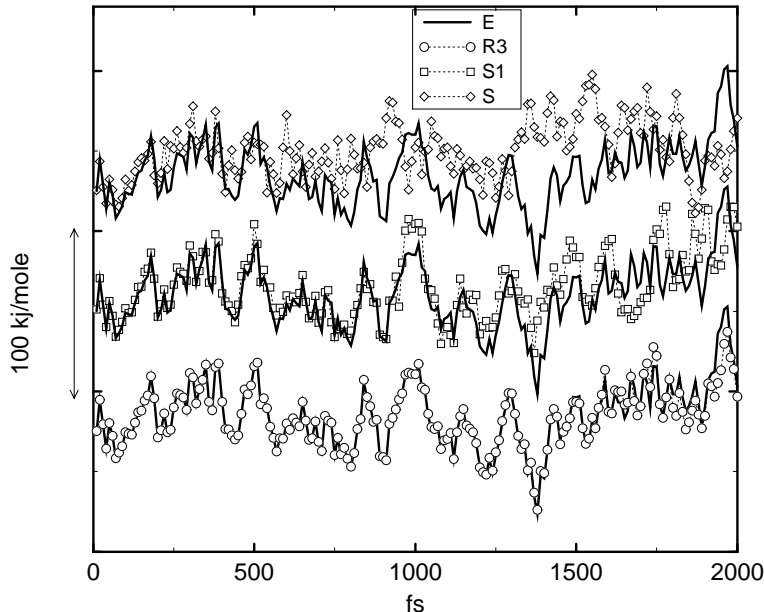


Figure 2.1: Time record of the torsional potential energy at about 300 K for a cluster of eight molecules of $C_{24}H_{50}$ obtained using three integrators: solid line integrator E; circles integrator R3; squares integrator S1; diamonds integrator S (see text)

from the same phase space point, against a single time step integrator (E) with a very small time step generating the “exact” trajectory. In Fig. 2.1 we show the time record of the torsional potential energy. The R3 integrator generates a trajectory practically coincident with the “exact” trajectory for as long as 1.5 ps. The single time step with rigid $X-H$ bonds also produces a fairly accurate trajectory, whereas the trajectory generated by S quickly drifts away from the exact time record. In Fig. 2.2 we show the power spectrum of the velocity auto-correlation function obtained with R3, S1 and S. The spectra are compared to the exact spectrum computed using the trajectories generated by the accurate integrator E. We see that R3 and S1 generates the same spectral profile within statistical error. In contrast, especially in the region above 800 wavenumbers, S generates a spectrum which differs appreciably from the exact one. This does not mean, of course, that S is unreliable for the “relevant” torsional degrees of freedom. Simply, we cannot *a priori* exclude that keeping all bonds rigid will not have an impact on the equilibrium structure of the alkanes molecules and on torsional dynamics. Actually, in the present case, as long as torsional motions are concerned all three integrators produce essentially identical results. In 20 picoseconds of simulation, R3 S1 and S predicted 60, 61, 60 torsional jumps, respectively, against the 59 jumps obtained with the exact integrator E. According to prescription of Ref. [72], in order to avoid period doubling, we compute the power spectrum of torsional motion from the auto-correlation function of the vector product of two normalized vector perpendicular to the dihedral planes. Rare events such as torsional jumps produce large amplitudes long time scale oscillations in the time auto-correlation function and therefore their contribution overwhelms the spectrum which appears as a single broaden peak around zero frequency. For this reason all torsions that did undergo a barrier crossing were discarded in the computation of the power spectrum. The power spectrum of the torsional motions is identical for all integrators within statistical error when evaluated over 20 ps of simulations.

From these results it can be concluded that S1 and R3 are very likely to produce essentially the same dynamics for all “relevant” degrees of freedom. We are forced to state that also the integrator S appears to accurately predict the structure and overall dynamics of the torsional degrees of freedom at least for the 20 ps time span of this specific system.⁴ Since torsions *are not* normal coordinates and couple to higher

⁴For example, our conclusions on the effect of SHAKE onto torsional motions for highly flexible systems differs from the

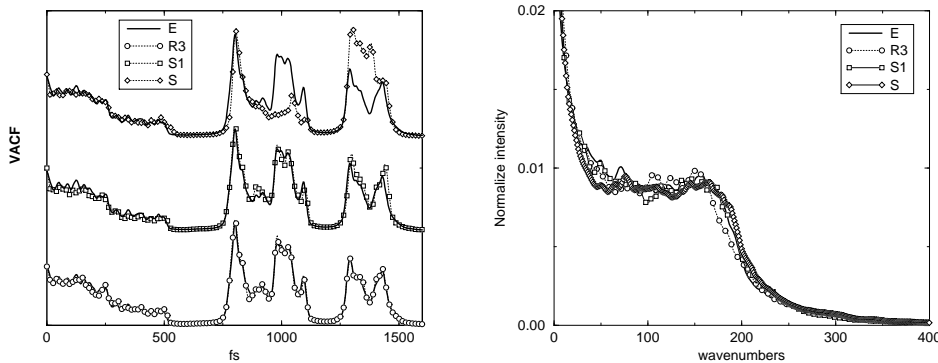


Figure 2.2: Power spectra of the velocity auto-correlation function (left) and of the torsional internal coordinates (right) at 300 K for a cluster of 8 $C_{24}H_{50}$ molecules calculated with integrators E, R3, S1 and S (see text) starting from the same phase space point

frequency internal coordinates such as bending and stretching, the ability of the efficient S integrator of correctly predicting low frequency dynamics and structural properties *cannot* be assumed *a priori* and must be, in principle, verified for each specific case. We also do not know how the individual eigenvectors are affected by the integrators and, although the overall density for S and S1 appears to be the same, there might be considerable changes in the torsional dynamics. R3 does not require any assumption, is accurate everywhere in the spectrum (see Fig. 2.2) and is as efficient as S. For these reasons R3, or a multi-step version of the equally accurate S1, must be the natural choice for the simulation of complex systems using *all-atoms* models

results published by Watanabe and Karplus [69] for another flexible system. i.e. met-enkephalin *in vacuo*. They compared SHAKE on $X-H$ against full flexibility and found that the power spectrum of torsional degrees of freedom differs significantly. For met-enkephalin their spectrum, evaluated on a 10 ps time span, shows a single strong peak at 10 or 40 wavenumbers, with and without constraints, respectively. The different behavior of the constrained and totally flexible system might be ascribed in their case to the the specificity of the system and/or the potential, although this seems unlikely [22]. In their study, on the other hand, we must remark the unusual shape of the spectral torsional profile with virtually no frequencies above 100 wavenumbers and with strong peaks suspiciously close the minimum detectable frequency according to their spectral resolution.

Chapter 3

Multiple Time Steps Algorithms for the Isothermal-Isobaric Ensemble

The integrators developed in the previous section generates dynamics in the microcanonical ensemble where total energy, number of particles and volume are conserved. The derivation based on the Liouvillean and the corresponding propagator, however lends itself to a straightforward generalization to non microcanonical ensembles. Simulations of this kind are based on the concept of extended system and generate trajectories that sample the phase space according to a target distribution function. The extended system method is reviewed in many excellent textbooks and papers [10, 73, 74, 75, 76, 77, 78, 23] to which we refer for a complete and detailed description. Here it suffices to say that the technique relies on the clever definition of a modified or *extended* Lagrangian which includes extra degrees of freedom related to the intensive properties (e.g. pressure or temperature) one wishes to sample with a well defined distribution function. The dynamics of the extended system is generated in the *microcanonical* ensemble with the true n degrees of freedom and, additionally, the extra degrees of freedom related to the macroscopic thermodynamic variables. With an appropriate choice, the equations of motion of the extended system will produce trajectories in the *extended* phase space generating the desired equilibrium distribution function upon integration over the extra (extended) variables. There are several extended system techniques corresponding to various ensembles, e.g. constant pressure in the NPH ensemble simulation with isotropic [79] and anisotropic [80] stress, constant temperature simulation [81] in the NVT ensemble and isothermal–isobaric simulation [82] in the NPT ensemble. As we shall see, the dynamic of the *real* system generated by the extended system method is never Hamiltonian. Hence, symplecticness is no longer an inherent property of the equations of motion. Nonetheless, the Liouvillean formalism developed in the preceding section, turns out to be very useful for the derivation of multiple time step reversible integrators for a general isothermal–isobaric ensemble with anisotropic stress, or **NPT**¹. This extended system is the most general among all non microcanonical simulations: The NPT, NPH the NVT and even NVE ensemble may be derived from this Lagrangian by imposing special constraints and/or choosing appropriate parameters [23, 25]

3.1 The Parrinello-Rahman-Nosé Extended Lagrangian

The starting point of our derivation of the multilevel integrator for the **NPT** ensemble is the Parrinello-Rahman-Nosé Lagrangian for a molecular system with N molecules or groups ² each containing n_i atoms and subject to a potential V . In order to construct the Lagrangian we define a coordinate scaling and a velocity scaling, i.e.

$$r_{ik\alpha} = R_{i\alpha} + l_{ik\alpha} = \sum_{\beta} h_{\alpha\beta} S_{i\beta} + l_{ik\alpha} \quad (3.1)$$

$$\dot{R}'_{i\alpha} = \dot{R}_{i\alpha} s \quad (3.2)$$

¹When **P** is not in boldface, we imply that the stress is *isotropic*

²For large molecules it may be convenient to further subdivide the molecule into groups. A group, therefore encompasses a conveniently chosen subset of the atoms of the molecule

$$\dot{l}'_{ik\alpha} = \dot{l}_{ik\alpha} s$$

Here, the indices i and k refer to molecules and atoms, respectively, while Greek letters are used to label the Cartesian components. $r_{ik\alpha}$ is the α component of the coordinates of the k -th atom belonging to the i -th molecule; $R_{i\alpha}$ is the center of mass coordinates; $S_{i\beta}$ is the scaled coordinate of the i -th molecular center of mass. $l_{ik\alpha}$ is the coordinate of the k -th atom belonging to the i -th molecule expressed in a frame parallel at any instant to the fixed laboratory frame, but with origin on the instantaneous molecular center of mass. The set of $l_{ik\alpha}$ coordinates satisfies $3N$ constraints of the type $\sum_{k=1}^{n_i} l_{ik\alpha} = 0$.

The matrix \mathbf{h} and the variable s control the pressure and temperature of the extended system, respectively. The columns of the matrix \mathbf{h} are the Cartesian components of the cell edges with respect to a fixed frame. The elements of this matrix allow the simulation cell to change shape and size and are sometimes called the “barostat” coordinates. The volume of the MD cell is related to \mathbf{h} through the relation

$$\Omega = \det(\mathbf{h}). \quad (3.3)$$

s is the coordinates of the so-called “Nosé thermostat” and is coupled to the intramolecular and center of mass velocities,

We define the “potentials” depending on the thermodynamic variables P and T

$$\begin{aligned} V_P &= P \det(\mathbf{h}) \\ V_T &= \frac{g}{\beta} \ln s. \end{aligned} \quad (3.4)$$

Where P is the *external* pressure of the system, $\beta = k_B T$, and g is a constant related to total the number of degrees of freedom in the system. This constant is chosen to correctly sample the *NPT* distribution function.

The extended *NPT* Lagrangian is then defined as

$$\mathcal{L} = \frac{1}{2} \sum_i^N M_i s^2 \dot{\mathbf{S}}_i^t \mathbf{h}^t \mathbf{h} \dot{\mathbf{S}}_i + \frac{1}{2} \sum_{ik} m_{ik} s^2 \dot{\mathbf{l}}_{ik}^t \mathbf{l}_{ik} + \frac{1}{2} W s^2 \text{tr}(\dot{\mathbf{h}}^t \dot{\mathbf{h}}) \quad (3.5)$$

$$+ \frac{1}{2} Q \dot{s}^2 - V - P_{ext} \Omega - \frac{g}{\beta} \ln s \quad (3.6)$$

The arbitrary parameters W and Q are the “masses” of the barostat and of the thermostats, respectively³. They do not affect the sampled distribution function but only the sampling efficiency [24, 81, 82]. For a detailed discussion of the sampling properties of this Lagrangian the reader is referred to Refs. [78, 25].

3.2 The Parrinello-Rahman-Nosé Hamiltonian and the Equations of Motion

In order to derive the multiple time step integration algorithm using the Liouville formalism described in the preceding sections we must switch to the Hamiltonian formalism. Thus, we evaluate the conjugate momenta of the coordinates $S_{i\alpha}$, $l_{ik\alpha}$, $h_{\alpha\beta}$ and s by taking the derivatives of the Lagrangian in Eq. (3.6) with respect to corresponding velocities, i.e.

$$\mathbf{T}_i = M_i \mathbf{G} s^2 \dot{\mathbf{S}}_i \quad (3.7)$$

$$\mathbf{p}_{ik} = m_{ik} s^2 \dot{\mathbf{l}}_{ik} \quad (3.8)$$

$$\mathbf{P}_h = s^2 W \dot{\mathbf{h}} \quad (3.9)$$

$$p_s = Q \dot{s}. \quad (3.10)$$

Where we have defined the symmetric matrix

$$\mathbf{G} = \mathbf{h}^t \mathbf{h} \quad (3.11)$$

³ W has actually the dimension of a mass, while Q has the dimension of a mass time a length squared

The Hamiltonian of the system is obtained using the usual Legendre transformation [59]

$$H(p, q) = \sum \dot{q}p - \mathcal{L}(q, \dot{q}). \quad (3.12)$$

One obtains

$$\begin{aligned} H = & \frac{1}{2} \sum_i^N \frac{\dot{\mathbf{T}}_i \mathbf{G}^{-1} \dot{\mathbf{T}}_i}{Ms^2} + \frac{1}{2} \sum_{ik} \frac{\mathbf{p}_{ik}^t \mathbf{p}_{ik}}{m_{ik}s^2} + \frac{1}{2} \frac{\text{tr}(\mathbf{P}_h^t \mathbf{P}_h)}{s^2 W} + \frac{p_s^2}{2Q} \\ & + V + P\Omega + \frac{g \ln s}{\beta} \end{aligned} \quad (3.13)$$

In the extended systems formulation we always deal with real and *virtual* variables. The virtual variables in the Hamiltonian (3.13) are the scaled coordinates and momenta while the unscaled variables (e.g $\mathbf{R}_i = \mathbf{h}\mathbf{S}_i$ or $p'_{ik\alpha} = p_{ik\alpha}/s$ are the real counterpart. The variable s in the Nosé formulation plays the role of a time scaling [73, 67, 81]. The above Hamiltonian is given in terms of *virtual variables* and in term of a virtual time and is indeed a *true* Hamiltonian function and has corresponding equation of motions that can be obtained applying Eq. (2.3) with $\mathbf{x} \equiv S_{i\alpha}, l_{ik\alpha}, h_{\alpha\beta}, s, T_{i\alpha}, p_{ik\alpha}, \pi_{\alpha\beta}, p_s$ in a standard fashion. Nonetheless, the equations of motions in terms of these virtual variable are inadequate for several reasons since for example one would deal with a fluctuating time step [73, 81]. It is therefore convenient to work in terms of real momenta and real time. The real momenta are related to the virtual counterpart through the relations

$$T_{i\alpha} \rightarrow T_{i\alpha}/s \quad (3.14)$$

$$p_{ik\alpha} \rightarrow p_{ik\alpha}/s \quad (3.15)$$

$$(\mathbf{P}_h)_{\alpha\beta} \rightarrow (\mathbf{P}_h)_{\alpha\beta}/s \quad (3.16)$$

$$p_s \rightarrow p_s/s \quad (3.17)$$

$$(3.18)$$

It is also convenient [24] to introduce new center of mass momenta as

$$\mathbf{P}_i \equiv \mathbf{G}^{-1} \mathbf{T}_i. \quad (3.19)$$

such that the corresponding velocities may be obtained directly without the knowledge of the “coordinates” \mathbf{h} in \mathbf{G}^4 , namely

$$\dot{\mathbf{S}}_i = \frac{\mathbf{P}_i}{M}. \quad (3.20)$$

Finally, a real time formulation and a new dynamical variable η are adopted:

$$t \rightarrow t/s \quad (3.21)$$

$$\eta \equiv \ln s \quad (3.22)$$

The equations of motions for the newly adopted set of dynamical variables are easily obtained from the true Hamiltonian in Eq. (3.13) and then using Eqs. (3.14-3.22) to rewrite the resulting equations in terms of the new momenta. In so doing, we obtain:

$$\dot{\mathbf{l}}_{ik} = \frac{\mathbf{p}_{ik}}{m_{ik}}, \quad \dot{\mathbf{S}}_i = \frac{\mathbf{P}_i}{M_i}, \quad \dot{\mathbf{h}} = \frac{\mathbf{P}_h}{W}, \quad \dot{\eta} = \frac{p_\eta}{Q} \quad (3.23)$$

$$\dot{\mathbf{p}}_{ik} = \mathbf{f}_{ik}^c - \frac{p_\eta}{Q} \mathbf{p}_{ik}, \quad (3.24)$$

$$\dot{\mathbf{P}}_i = \mathbf{h}^{-1} \mathbf{F}_i - \mathbf{G}^{-1} \dot{\mathbf{G}} \mathbf{P}_i - \frac{p_\eta}{Q} \mathbf{P}_i, \quad (3.25)$$

$$\dot{\mathbf{P}}_h = (\mathcal{V} + \mathcal{K} - \mathbf{h}^{-1} P_{ext} \det \mathbf{h}) - \frac{p_\eta}{Q} \mathbf{P}_h, \quad (3.26)$$

$$\dot{p}_\eta = \mathcal{F}_\eta, \quad (3.27)$$

⁴This allows to maintain Verlet-like breakup while integrating the equation of motions [24].

It can be verified that the conserved quantity \mathcal{H} is associated with the above equations of motion, namely

$$\begin{aligned}\mathcal{H} &= \frac{1}{2} \sum_{i=1}^N \frac{\mathbf{P}_i^t \mathbf{G} \mathbf{P}_i}{M_i} + \frac{1}{2} \sum_{i=1}^N \sum_{k=1}^{n_i} \frac{\mathbf{p}_{ik}^t \mathbf{p}_{ik}}{m_{ik}} + \frac{1}{2} \frac{\text{tr}(\mathbf{P}_h^t \mathbf{P}_h)}{W} + \\ &+ \frac{1}{2} \frac{p_\eta p_\eta}{Q} + V + P_{ext} \det \mathbf{h} + g k_B T \eta.\end{aligned}\quad (3.28)$$

The atomic force $\mathbf{f}_{ik}^c = \frac{\partial V}{\partial \mathbf{r}_{ik\alpha}} - \frac{m_{ik}}{M_i} \mathbf{F}_i$ includes a constraint force contribution which guarantees that the center of mass in the intramolecular frame of the $l_{ik\alpha}$ coordinates remains at the origin. \mathcal{V} and \mathcal{K} are the virial and ideal gas contribution to the internal pressure tensor $\mathbf{P}_{int} = \mathcal{V} + \mathcal{K}$ and they are defined as⁵

$$\begin{aligned}\mathcal{V} &= \sum_{i=1}^N \mathbf{F}_i \mathbf{S}_i^t \\ \mathcal{K} &= \sum_{i=1}^N M_i \left(\mathbf{h} \dot{\mathbf{S}}_i \right) \dot{\mathbf{S}}_i^t.\end{aligned}\quad (3.29)$$

Finally \mathcal{F}_η is the force driving the Nosé thermostat

$$\mathcal{F}_\eta = \frac{1}{2} \sum_{i=1}^N M_i \dot{\mathbf{S}}_i^t \mathbf{G} \dot{\mathbf{S}}_i + \frac{1}{2} \sum_{i=1}^N \sum_{k=1}^{n_i} \frac{\mathbf{p}_{ik}^t \mathbf{p}_{ik}}{m_{ik}} - g k_B T, \quad (3.30)$$

with g equal to the number of all degrees of freedom N_f including those of the barostat⁶.

Eqs. (3.14-3.21) define a generalized coordinates transformation of the kind of Eq. (2.4). This transformation is non canonical, i.e. the Jacobian matrix of the transformation from the virtual coordinates does not obey Eq. (2.8). This means that \mathcal{H} in terms of the new coordinates Eq. (3.28) is “only” a constant of motion, but is no longer a true Hamiltonian: application of Eq. (2.1) does not lead to Eqs. (3.23-3.27). Simulations using the real variables are not Hamiltonian in nature in the sense that the phase space of the real variables is compressible [83] and that Liouville theorem is not satisfied [78]. This “strangeness” in the dynamics of the real variables in the extended systems does not of course imply that the sampling of the configurational *real* space is incorrect. To show this, it suffices to evaluate the partition function for a microcanonical distribution of the kind $\delta(\mathcal{H} - E)$, with \mathcal{H} being given by Eq. (3.28). The Jacobian of the transformation of Eqs. (3.14-3.22) must be included in the integration with respect to the real coordinates when evaluating the partition function for the extended system. If the equations of motion in terms of the transformed coordinates are known, this Jacobian, \mathcal{J} , can be readily computed from the relation [60]:

$$\frac{d\mathcal{J}}{dt} = -\mathcal{J} \left(\frac{\partial}{\partial \mathbf{y}} \cdot \dot{\mathbf{y}} \right). \quad (3.31)$$

Where \mathbf{y} has the usual meaning of phase space vector containing all independent coordinates and momenta of the systems. Inserting the equations of motion of Eq. (3.27) into Eq. (3.31) and integrating by separation of variables yields

$$\mathcal{J} = e^{N_f \eta} [\det \mathbf{h}]^{6N}. \quad (3.32)$$

Using (3.32) and integrating out the thermostat degrees of freedom, the partition function can be easily shown [78, 84] to be equivalent to that that of *NPT* ensemble, i.e.

$$\Delta_{NPT} \propto \int d\mathbf{h} e^{-\beta P_{ext} \det(\mathbf{h})} Q(\mathbf{h}) \quad (3.33)$$

with $Q(\mathbf{h})$ being the canonical distribution of a system with cell of shape and size define by the columns of \mathbf{h} .⁷

⁵ In presence of bond constraints and if the scaling is group-based instead of molecular based, these expression should contain a contribution from the constraints forces. Complications due to the constraints can be avoided altogether by defining groups so that no two groups are connected through a constrained bond [25]. In that case \mathcal{V} does not include any constraint contribution.

⁶ The thermostat degree of freedom must be included [73, 78] in the count when working in *virtual* coordinates. Indeed in Eq. (3.13) we have $g = N_f + 1$

⁷ Actually in ref. [78, 25] is pointed out that the virial theorem implied by the distribution (3.33) is slightly different from

3.3 Equivalence of Atomic and Molecular Pressure

The volume scaling defined in Eq. (3.1) is not unique. Note that only the equation of motion for the center of mass momentum, Eq. (3.25), has a velocity dependent term that depends on the coordinates of the barostat through the matrix \mathbf{G} defined in Eq. (3.11). The atomic momenta, Eq. (3.24), on the contrary, are not coupled to the barostat. This fact is also reflected in the equations of motion for the barostat momenta, Eq. (3.26), which is driven by the internal pressure due only to the molecular or group center of masses. In defining the extended Lagrangian one could as well have defined an *atomic* scaling of the form

$$r_{ik\alpha} = \sum_{\beta} h_{\alpha\beta} s_{i\alpha k}. \quad (3.34)$$

Atomic scaling might be trivially implemented by eliminating the kinetic energy, which depends on the $\dot{l}_{ik\alpha}$ velocities, from the starting Lagrangian (3.6) and replacing the term $\frac{1}{2} \sum_i^N M_i s_i^2 \dot{\mathbf{S}}_i^t \mathbf{h}^t \mathbf{h} \dot{\mathbf{S}}_i$ with $\frac{1}{2} \sum_{ik} m_{ik} s_{ik}^2 \dot{\mathbf{s}}_{ik}^t \mathbf{h}^t \mathbf{h} \dot{\mathbf{s}}_{ik}$. The corresponding equations of motions for atomic scaling are then

$$\dot{\mathbf{r}}_{ik} = \frac{\mathbf{p}_{ik}}{m_{ik}}, \quad \dot{\mathbf{h}} = \frac{\mathbf{P}_h}{W}, \quad \dot{\eta} = \frac{p_\eta}{Q} \quad (3.35)$$

$$\dot{\mathbf{p}}_{ik} = \mathbf{h}^{-1} \dot{\mathbf{p}}_{ik} - \mathbf{G}^{-1} \dot{\mathbf{G}} \dot{\mathbf{p}}_{ik} - \frac{p_\eta}{Q} \dot{\mathbf{p}}_{ik}, \quad (3.36)$$

$$\dot{\mathbf{P}}_h = (\mathcal{V} + \mathcal{K} - \mathbf{h}^{-1} P_{ext} \det \mathbf{h}) - \frac{p_\eta}{Q} \mathbf{P}_h, \quad (3.37)$$

$$\dot{p}_\eta = \mathcal{F}_\eta \quad (3.38)$$

where the quantities $\mathcal{V}, \mathcal{K}, \mathcal{F}_\eta$ depend now on the *atomic* coordinates

$$\begin{aligned} \mathcal{V} &= \sum_{i=1}^N \mathbf{f}_{ik} \mathbf{s}_{ik}^t \\ \mathcal{K} &= \sum_{i=1}^N M_i (\mathbf{h} \mathbf{s}_{ik}) \mathbf{s}_{ik}^t \end{aligned} \quad (3.39)$$

$$\mathcal{F}_\eta = \frac{1}{2} \sum_{i=1}^N \sum_{k=1}^{n_i} \frac{\mathbf{p}_{ik}^t \mathbf{p}_{ik}}{m_{ik}} - g k_B T. \quad (3.40)$$

In case of atomic, Eq. (3.34), or molecular scaling, Eq. (3.1), the internal pressure entering in Eqs. (3.26, 3.37) is then

$$P_{int} = \langle P_{atom} \rangle = \left\langle \frac{1}{3V} \sum_i \sum_k \left(\frac{\mathbf{p}_{ik}^2}{m_{ik}} + \mathbf{r}_{ik} \cdot \mathbf{f}_{ik} \right) \right\rangle \quad (3.41)$$

$$P_{int} = \langle P_{mol} \rangle = \left\langle \frac{1}{3V} \sum_i \left(\frac{\mathbf{P}_i^2}{M_i} + \mathbf{R}_i \cdot \mathbf{F}_i \right) \right\rangle \quad (3.42)$$

respectively. Where the molecular quantities can be written in term of the atomic counterpart according to:

$$\mathbf{R}_i = \frac{1}{M_i} \sum_k m_{ik} \mathbf{r}_{ik} \quad (3.43)$$

$$\mathbf{P}_i = \sum_k \mathbf{p}_{ik} \quad (3.44)$$

$$\mathbf{F}_i = \sum_k \mathbf{f}_{ik} \quad (3.45)$$

the exact virial in the NPT ensemble. Martyna *et al.* [78] proposed an improved set of equations of motion that generates a distribution satisfying exactly the virial theorem.

The equation of motion for the barostat in the two cases, Eqs.(3.37, 3.26), has the same form whether atomic or molecular scaling is adopted. The internal pressure in the former case is given by Eq. (3.41) and in the latter is given by Eq. (3.42). The two pressures, Eqs. (3.41,3.42), differ instantaneously. Should the difference persist after averaging, then it would be obvious that the equilibrium thermodynamic state in the *NPT* ensemble depends on the scaling method. The two formulas (3.41,3.42) are fortunately equivalent. To prove this statement, we closely follow the route proposed by H. Berendsen and reported by Ciccotti and Ryckaert [85] and use Eqs. (3.43-3.45) to rearrange Eq. (3.42). We obtain

$$\sum_i \langle \mathbf{R}_i \bullet \mathbf{F}_i \rangle = \sum_i \frac{1}{M_i} \sum_{kl} \langle m_{ik} \mathbf{r}_{ik} \bullet \mathbf{f}_{il} \rangle. \quad (3.46)$$

Adding and subtracting $m_{ik} \mathbf{r}_{il} \bullet \mathbf{f}_{il}$, we get

$$= \sum_i \frac{1}{M_i} \sum_{kl} \langle m_{ik} (\mathbf{r}_{ik} - \mathbf{r}_{il}) \bullet \mathbf{f}_{il} + m_{ik} \mathbf{r}_{il} \bullet \mathbf{f}_{il} \rangle \quad (3.47)$$

which can be rearranged as

$$= \sum_i \frac{1}{M_i} \left\{ \sum_{kl} \left[\frac{1}{2} \langle (\mathbf{r}_{ik} - \mathbf{r}_{il}) \bullet (m_i \mathbf{f}_{il} - m_j \mathbf{f}_{ik}) \rangle \right] + \sum_l \langle \mathbf{r}_{il} \bullet \mathbf{f}_{il} \rangle \right\} \quad (3.48)$$

using the newton law $\mathbf{f}_{ik} = m_{ik} \mathbf{a}_{ik}$, where \mathbf{a}_{ik} is the acceleration, we obtain

$$= \sum_i \frac{1}{M_i} \left\{ \sum_{kl} \left[\frac{1}{2} \langle m_j m_i (\mathbf{r}_{ik} - \mathbf{r}_{il}) \bullet (\mathbf{a}_{il} - \mathbf{a}_{ik}) \rangle \right] + \sum_l \langle \mathbf{r}_{il} \bullet \mathbf{f}_{il} \rangle \right\}. \quad (3.49)$$

The first term in the above equation can be decomposed according to:

$$(\mathbf{r}_{ik} - \mathbf{r}_{il}) \bullet (\mathbf{a}_{il} - \mathbf{a}_{ik}) = \frac{d}{dt} [(\mathbf{r}_{ik} - \mathbf{r}_{il}) \bullet (\mathbf{v}_{il} - \mathbf{v}_{ik})] + (\mathbf{v}_{il} - \mathbf{v}_{ik})^2 \quad (3.50)$$

The first derivative term on the right hand side is zero rigorously for rigid molecules or rigid group and is zero on average for flexible molecules or groups, assuming that the flexible molecules or groups do not dissociate. This can be readily seen in case of ergodic systems, by evaluating directly the average of this derivatives as

$$\left\langle \frac{d}{dt} [(\mathbf{r}_{ik} - \mathbf{r}_{il}) \bullet (\mathbf{v}_{il} - \mathbf{v}_{ik})] \right\rangle = \lim_{\tau \rightarrow \infty} \frac{1}{\tau} \int_0^\tau \frac{d}{dt} [(\mathbf{r}_{ik} - \mathbf{r}_{il}) \bullet (\mathbf{v}_{il} - \mathbf{v}_{ik})] dt \quad (3.51)$$

$$= \lim_{\tau \rightarrow \infty} \frac{1}{\tau} [(\mathbf{r}_{ik}(\tau) - \mathbf{r}_{il}(\tau)) \bullet (\mathbf{v}_{il}(\tau) - \mathbf{v}_{ik}(\tau)) + C] \quad (3.52)$$

So if the quantity $\mathbf{r}_{ikl}(\tau) \bullet \mathbf{v}_{ilk}(\tau)$ remains bounded (which is true if the potential is not dissociative, since k, l refers to the same molecule i), the average in Eq. (3.52) is zero⁸. Thus, we can rewrite the average of Eq. (3.49) as

$$\left\langle \sum_i \mathbf{R}_i \bullet \mathbf{F}_i \right\rangle = \sum_i \frac{1}{2M_i} \sum_{kl} m_{ik} m_{il} \langle (\mathbf{v}_{il} - \mathbf{v}_{ik})^2 \rangle + \sum_{ik} \langle \mathbf{r}_{ik} \bullet \mathbf{f}_{ik} \rangle. \quad (3.53)$$

The first term on the right hand side of the above equation can be further developed obtaining the trivial identity:

$$\begin{aligned} \sum_{kl} m_{ik} m_{il} \langle (\mathbf{v}_{il} - \mathbf{v}_{ik})^2 \rangle &= \sum_{kl} m_{ik} m_{il} \langle \mathbf{v}_{ik}^2 \rangle + \sum_{kl} m_{ik} m_{il} \langle \mathbf{v}_{il}^2 \rangle - \\ &- \sum_{kl} 2m_{ik} m_{il} \langle \mathbf{v}_{il} \bullet \mathbf{v}_{ik} \rangle \end{aligned} \quad (3.54)$$

$$= 2M_i \sum_k m_{ik} \langle \mathbf{v}_{ik}^2 \rangle - 2\langle \mathbf{P}_i^2 \rangle \quad (3.55)$$

⁸The statement *the molecule of group does not dissociate* is even too restrictive. It is enough to say that the quantity (3.52) remains bound.

Substituting Eq. (3.55) in Eq. (3.53) we get

$$\langle \sum_i \mathbf{R}_i \bullet \mathbf{F}_i \rangle = \sum_{ik} m_{ik} \langle \mathbf{v}_{ik}^2 \rangle - \frac{1}{M_i} \langle \mathbf{P}_i^2 \rangle + \sum_{ik} \langle \mathbf{r}_{ik} \bullet \mathbf{f}_{ik} \rangle \quad (3.56)$$

Substituting Eq. (3.56) into Eq. (3.42) leads speedily to (3.41) which completes the proof. As a consequence of the above discussion, it seems likely that both the equilibrium and non equilibrium properties of the MD system are not affected by coordinate scaling. We shall see later that this is actually the case.

3.4 Liouvillean Split and Multiple Time Step Algorithm for the *NPT* Ensemble

We have seen in section 2.2 that the knowledge of the Liouvillean allows us to straightforwardly derive a multi-step integration algorithm. Thus, for simulation in the *NPT* ensemble, the Liouvillean $iL = \dot{\mathbf{y}} \nabla_{\mathbf{y}}$ is readily available from the equations of motion in (3.23-3.27). For sake of simplicity, to build our *NPT* multiple time step integrator we assume that the system potential contains only a fast *intramolecular* V_0 term and a slow *intermolecular* term V_1 , as discussed in Sec. 2.3. Generalization to multiple intra and inter-molecular components is straightforward.

We define the following components of the *NPT* Liouvillean

$$iL_x = - \sum_i \mathbf{P}_i \frac{p_\eta}{Q} \nabla_{\mathbf{P}_i} - \sum_{ik} \mathbf{p}_{ik} \frac{p_\eta}{Q} \nabla_{\mathbf{p}_{ik}} - \sum_{\alpha\beta} (\mathbf{P}_h)_{\alpha\beta} \frac{p_\eta}{Q} (\nabla_{\mathbf{P}_h})_{\alpha\beta} \quad (3.57)$$

$$iL_y = \mathcal{F}_\eta \nabla_{p_\eta} \quad (3.58)$$

$$iL_z = \sum_i \mathbf{G}^{-1} \dot{\mathbf{G}} \mathbf{P}_i \nabla_{\mathbf{P}_i} \quad (3.59)$$

$$iL_u = \sum_{\alpha\beta} (\mathcal{K} - \mathbf{h}^{-1} P_{ext} \det \mathbf{h})_{\alpha\beta} (\nabla_{\mathbf{P}_h})_{\alpha\beta} \quad (3.60)$$

$$iL_s = \sum_i \mathbf{J}_i \nabla_{\mathbf{P}_i} + \sum_{ik} \mathbf{f}_{ik}^c \nabla_{\mathbf{p}_{ik}} + \sum_{\alpha\beta} (\mathcal{V})_{\alpha\beta} (\nabla_{\mathbf{P}_h})_{\alpha\beta} \quad (3.61)$$

$$\begin{aligned} iG_0 &= \sum_i \frac{\mathbf{P}_i}{M_i} \nabla_{\mathbf{S}_i} + \sum_{ik} \frac{\mathbf{p}_{ik}}{m_{ik}} \nabla_{\mathbf{l}_{ik}} + \sum_{\alpha\beta} \frac{(\mathbf{P}_h)_{\alpha\beta}}{W} (\nabla_{\mathbf{h}})_{\alpha\beta} + \\ &+ \frac{p_\eta}{Q} \nabla_\eta - \nabla_{\mathbf{l}_{ik}} V_0 \nabla_{\mathbf{p}_{ik}}, \end{aligned} \quad (3.62)$$

where in Eq. (3.61) the scaled forces \mathbf{F}_i have been replaced by its real space counterparts, i.e. $\mathbf{J}_i = \mathbf{h}^{-1} \mathbf{F}_i$.

The atomic scaling version of this Liouvillean breakup is derived on the basis of Eqs. (3.38). One obtains

$$iL_x = - \sum_{ik} \mathbf{p}_{ik} \frac{p_\eta}{Q} \nabla_{\mathbf{p}_{ik}} - \sum_{\alpha\beta} (\mathbf{P}_h)_{\alpha\beta} \frac{p_\eta}{Q} (\nabla_{\mathbf{P}_h})_{\alpha\beta} \quad (3.63)$$

$$iL_y = \mathcal{F}_\eta \nabla_{p_\eta} \quad (3.64)$$

$$iL_z = \sum_{ik} \mathbf{G}^{-1} \dot{\mathbf{G}} \mathbf{p}_{ik} \nabla_{\mathbf{p}_{ik}} \quad (3.65)$$

$$iL_u = \sum_{\alpha\beta} (\mathcal{K} - \mathbf{h}^{-1} P_{ext} \det \mathbf{h})_{\alpha\beta} (\nabla_{\mathbf{P}_h})_{\alpha\beta} \quad (3.66)$$

$$iL_s = \sum_{ik} \mathbf{j}_{ik} \nabla_{\mathbf{p}_{ik}} + \sum_{\alpha\beta} (\mathcal{V})_{\alpha\beta} (\nabla_{\mathbf{P}_h})_{\alpha\beta} \quad (3.67)$$

$$\begin{aligned} iG_0 &= \sum_{ik} \frac{\mathbf{p}_{ik}}{m_{ik}} \nabla_{\mathbf{l}_{ik}} + \sum_{\alpha\beta} \frac{(\mathbf{P}_h)_{\alpha\beta}}{W} (\nabla_{\mathbf{h}})_{\alpha\beta} + \\ &+ \frac{p_\eta}{Q} \nabla_\eta - \nabla_{\mathbf{l}_{ik}} V_0 \nabla_{\mathbf{p}_{ik}}, \end{aligned} \quad (3.68)$$

where $\mathbf{j}_{ik} = \mathbf{h}^{-1}\mathbf{f}_{ik}$ and $\mathcal{V}, \mathcal{K}, \mathcal{F}_\eta$ are given in Eqs. (3.39,3.40). For the time scale breakup in the *NPT* ensemble we have the complication of the extra degrees of freedom whose time scale dynamics can be controlled by varying the parameter Q and W . Large values of Q and W slow down the time dynamics of the barostat and thermostat coordinates. The potential V determines the time scale of the iG_0 term (the fast component) and of the iL_s contribution (the slow component). All other sub-Liouvillians either handle the coupling of the true coordinates to the extra degrees of freedom (iL_x expresses the coupling of all momenta (including barostat momenta) to the thermostat momentum, while iL_z is a coupling term between the center of mass momenta and the barostat momentum), or drive the evolution of the extra coordinates of the barostat and thermostat (iL_y and iL_u). The time scale dynamics of these terms depends not only on the potential subdivision and on the parameters W and Q , but also on the type of scaling [25]. When the molecular scaling is adopted the dynamics of the virial term \mathcal{V} contains contributions only from the intermolecular potential since the barostat is coupled only to the center of mass coordinates (see Eq. (3.29)). Indeed, the net force acting on the molecular center of mass is independent on the intramolecular potential, since the latter is invariant under rigid translation of the molecules. When atomic scaling or group (i.e. sub-molecular) scaling is adopted, the virial \mathcal{V} (see Eq. (3.39)) depends also on the fast intramolecular such as stretching motions. In this case the time scale of the barostat coordinate is no longer slow, unless the parameter W is changed. For standard values of W , selected to obtain an efficient sampling of the *NPT* phase space [86, 67], the barostat dependent Liouvillians, Eqs. (3.60,3.59), have time scale dynamics comparable to that of the intramolecular Liouvillean iG_0 and therefore must be associated with this term⁹.

Thus, the molecular split of the Liouvillean is hence given by

$$\begin{aligned} iL_1 &= iL_x + iL_y + iL_z + iL_u + iL_s \\ iL_0 &= iG_0 \end{aligned} \quad (3.69)$$

whereas the atomic split is

$$\begin{aligned} iL_1 &= iL_x + iL_y + iL_s \\ iL_0 &= iG_0 + iL_z + iL_u \end{aligned} \quad (3.70)$$

For both scaling, a simple Hermitian factorization of the total time propagator e^{iLt} yields the double time discrete propagator

$$e^{iL_1+iL_0} = e^{iL_1\Delta t_1/2} (e^{iL_0\Delta t_0})^n e^{iL_1\Delta t_1/2} \quad (3.71)$$

where Δt_0 , the small time step, must be selected according to the intramolecular time scale whereas Δt_1 , the large time step, must be selected according to the time scale of the intermolecular motions. We already know that the propagator (3.71) cannot generate a symplectic. The alert reader may also have noticed that in this case the symmetric form of the multiple time step propagator Eq. (3.71) does not imply necessarily time reversibility. Some operators appearing in the definition of L_1 (e.g. iL_z and iL_s) for the molecular scaling and in the definitions of iL_1 and iL_0 for the atomic scaling are in fact non commuting. We have seen in section 2.2 that first order approximation of non commuting propagators yields time irreversible algorithms. We can render the propagator in Eq. (3.71) time reversible by using second order symmetric approximant (i.e. Trotter approximation) for any two non commuting operators. For example in the case of the molecular scaling, when we propagate in Eq. (3.71) the slow propagator $e^{iL_1\Delta t/2}$ for half time step, we may use the following second order $O(\Delta t^3)$ split

$$e^{iL_1\frac{\Delta t_1}{2}} \simeq e^{iL_y\frac{\Delta t_1}{4}} e^{iL_z\frac{\Delta t_1}{2}} e^{iL_y\frac{\Delta t_1}{4}} e^{iL_x\frac{\Delta t_1}{4}} e^{i(L_s+L_u)\frac{\Delta t_1}{2}} e^{iL_x\frac{\Delta t_1}{4}} \quad (3.72)$$

An alternative simpler and equally accurate approach when dealing with non commuting operators is simply to preserve the unitarity by reversing the order of the operators in the first order factorization of the right and left operators of Eq. (3.71) without resorting to locally second order $O(\Delta t^3)$ approximation like in Eq. (3.72). Again for the molecular scaling, this is easily done by using the approximant

$$\left(e^{iL_1\frac{\Delta t_1}{2}} \right)_{\text{left}} = e^{iL_x\frac{\Delta t_1}{2}} e^{iL_y\frac{\Delta t_1}{2}} e^{iL_z\frac{\Delta t_1}{2}} e^{iL_u\frac{\Delta t_1}{2}} e^{iL_s\frac{\Delta t_1}{2}} \quad (3.73)$$

⁹Similar considerations hold for the thermostat coordinate which in principle depends on the kinetic energy of all degrees of freedom, modulated hence by the fast motion also. In this case, however, the value of the thermostat inertia parameter Q can be chosen to slow down the time scale of the η coordinates without reducing considerably the sampling efficiency.

for the left propagator, and

$$\left(e^{iL_1 \frac{\Delta t_1}{2}}\right)_{\text{right}} = e^{iL_s \frac{\Delta t_1}{2}} e^{iL_u \frac{\Delta t_1}{2}} e^{iL_z \frac{\Delta t_1}{2}} e^{iL_y \frac{\Delta t_1}{2}} e^{iL_x \frac{\Delta t_1}{2}} \quad (3.74)$$

for the rightmost propagator. Note that

$$\left(e^{iL_1 \frac{\Delta t_1}{2}}\right)_{\text{left}}^{-1} = \left(e^{iL_1 \frac{\Delta t_1}{2}}\right)_{\text{right}} \quad (3.75)$$

Inserting these approximations into (3.71) the overall integrator is found to be time reversible and second order. Time reversible integrators are in fact always even order and hence *at least* second order [58, 55]. Therefore the overall molecular and atomic (or group) discrete time propagators are given by

$$\begin{aligned} e^{iL_{\text{mol}} \Delta t_1} &= e^{iL_s \frac{\Delta t_1}{2}} e^{iL_u \frac{\Delta t_1}{2}} e^{iL_z \frac{\Delta t_1}{2}} e^{iL_y \frac{\Delta t_1}{2}} e^{iL_x \frac{\Delta t_1}{2}} (e^{iG_0 \Delta t_0})^n \times \\ &\times e^{iL_x \frac{\Delta t_1}{2}} e^{iL_y \frac{\Delta t_1}{2}} e^{iL_z \frac{\Delta t_1}{2}} e^{iL_u \frac{\Delta t_1}{2}} e^{iL_s \frac{\Delta t_1}{2}} \end{aligned} \quad (3.76)$$

$$\begin{aligned} e^{iL_{\text{atom}} \Delta t_1} &= e^{iL_s \frac{\Delta t_1}{2}} e^{iL_y \frac{\Delta t_1}{2}} e^{iL_x \frac{\Delta t_1}{2}} \times \\ &\times (e^{iL_u \Delta t_0/2} e^{iL_z \Delta t_0/2} e^{iG_0 \Delta t_0} e^{iL_z \Delta t_0/2} e^{iL_u \Delta t_0/2})^n \times \\ &\times e^{iL_x \frac{\Delta t_1}{2}} e^{iL_y \frac{\Delta t_1}{2}} e^{iL_s \frac{\Delta t_1}{2}} \end{aligned} \quad (3.77)$$

The propagator $e^{iG_0 \Delta t_0}$, defined in Eq. (3.62), is further split according to the usual velocity Verlet breakup of Eq. (2.22). Note that in case of molecular scaling the “slow” coordinates (\mathbf{S} , \mathbf{h} , η) move with constant velocity during the n small times steps since there is no “fast” force acting on them in the inner integration. The explicit integration algorithm may be easily derived for the two propagators in Eqs. (3.76) and (3.77) using the rule in Eq. (2.23) and its generalization:

$$\begin{aligned} e^{a\mathbf{y}\nabla_{\mathbf{y}}} f(\mathbf{y}) &= f(\mathbf{y}e^a) \\ e^{\mathbf{a}\mathbf{y}\nabla_{\mathbf{y}}} f(\mathbf{y}) &= f(e^{\mathbf{a}}\mathbf{y}) \end{aligned} \quad (3.78)$$

Where a and \mathbf{a} are a scalar and a matrix, respectively. The exponential matrix $e^{\mathbf{a}}$ on the right hand side of Eq. (3.78) is obtained by diagonalization of \mathbf{a} .

As stated before the dynamics generated by Eqs. (3.23-3.27) or (3.35-3.38) in the *NPT* ensemble is not Hamiltonian and hence we cannot speak of symplectic integrators [87] for the t -flow's defined by Eqs. (3.76, 3.77). The symplectic condition Eq. (2.8) is violated at the level of the transformation (3.14-3.22) which is not canonical. However, the algorithms generated by Eqs. (3.76, 3.77) are time reversible and second order like the velocity Verlet. Several recent studies have shown [24, 23, 25] that these integrators for the non microcanonical ensembles are also stable for long time trajectories, as in case of the symplectic integrators for the NVE ensemble.

3.5 Group Scaling and Molecular Scaling

We have seen in section 3.3 that the center of mass or *molecular* pressure is equivalent to the atomic pressure. The atomic pressure is the natural quantity that enters in the virial theorem [10] irrespectively of the form of the interaction potential among the particles. So, in principle it is safer to adopt atomic scaling in the extended system constant pressure simulation. For systems in confined regions, the equivalence between atomic or true pressure and molecular pressure (see sec. 3.3) holds for any definition of the *molecular* subsystem irrespectively of the interaction potentials. In other words we could have defined virtual molecules made up of atoms selected on different *real* molecules. We may expect that, as long as the system, no matter how its unities or *particles* are defined, contains a sufficiently large number of particles, generates a distribution function identical to that generated by using the “correct” atomic scaling. From a computational standpoint molecular scaling is superior to atomic scaling. The fast varying Liouvillean in Eq. (3.70) for the atomic scaling contains the two terms iL_z, iL_u . These terms are slowly varying when molecular scaling is adopted and are assigned to the slow part of the Liouvillean in Eq. (3.69). The inner part of the time propagation is therefore expected to be more expensive for the multiple time step integration with atomic scaling rather than with molecular scaling. Generally speaking, given the

equivalence between the molecular and atomic pressure, molecular scaling should be the preferred choice for maximum efficiency in the multiple time step integration.¹⁰

For large size molecules, such as proteins, molecular scaling might be inappropriate. The size of the molecule clearly restricts the number of *particles* in the MD simulation box, thereby reducing the statistics on the instantaneous calculated molecular pressure which may show nonphysical large fluctuations. Group scaling [25] is particularly convenient for handling the simulation of macromolecules. A statistically significant number of groups can be selected in order to avoid all problems related to the poor statistics on molecular pressure calculation for samples containing a small number of large size particles. Notwithstanding, for *solvated* biomolecules and provided that enough solvent molecules are included, molecular scaling again yields reliable results. In Ref. [25] Marchi and Procacci showed that the scaling method in the *NPT* ensemble does not affect neither the equilibrium structural and dynamical properties nor the kinetic of non equilibrium MD. For group-based and molecular-based scaling methods in a system of one single molecule of BPTI embedded in a box of about a 1000 water molecules, they obtained identical results for the system volume, the Voronoi volumes of the proteins and for the mean square displacement of both solvent and protein atoms under normal and high pressure.

3.6 Switching to Other Ensembles

The *NPT* extended system is the most general among all possible extended Lagrangians. All other ensemble can be in fact obtained within the same computational framework. We must stress [25] that the computational overhead of the extended system formulation, due to the introduction and handling of the extra degrees of freedom of the barostat and thermostat variables, is rather modest and is negligible with respect to a *NVE* simulation for large samples ($N_f > 2000$) [24, 23, 25]. Therefore, a practical, albeit inelegant way of switching among ensembles is simply to set the inertia of the barostat and/or thermostat to a very large number. This must be of course equivalent to decouple the barostat and/or the thermostat from the true degrees of freedom. In fact, by setting W to infinity¹¹ in Eqs. (3.23-3.27) we recover the *NVT* canonical ensemble equations of motion. Putting instead Q to infinity the *NPH* equations of motion are obtained. Finally, setting both W and Q to infinity the *NVE* equations of motion are recovered.

Switching to the *NPT* isotropic stress ensemble is less obvious. One may define the kinetic term associated to barostat in the extended Lagrangian as

$$K = \frac{1}{2} \sum_{\alpha\beta} W_{\alpha\beta} s^2 \dot{h}_{\alpha\beta}^2 \quad (3.79)$$

such that a different inertia may in principle be assigned to each of 9 extra degrees of freedom of the barostat. Setting for example

$$W_{\alpha\beta} = W \quad \text{for } \alpha \leq \beta \quad (3.80)$$

$$W_{\alpha\beta} = \infty \quad \text{for } \alpha > \beta \quad (3.81)$$

$$(3.82)$$

one inhibits cell rotations [25].

This trick does not work, unfortunately, to change to isotropic stress tensor. In this case, there is only one independent barostat degrees of freedom, namely the volume of the system. In order to simulate isotropic cell fluctuations a set of five constraints on the \mathbf{h} matrix are introduced which correspond to the

¹⁰There are also other less material reasons to prefer molecular scaling: atomic scaling and molecular scaling yield different dynamical properties because the equations of motions are different. Dynamical data computed via extended system simulations should always be taken with caution. With respect to pure Newtonian dynamics, however, the *NPT* dynamical evolution is slightly modified by a barostat coupled to the molecular center of mass [24] but is brutally damaged when the barostat is coupled to the fast degrees of freedom. For example in liquid flexible nitrogen at normal pressure and 100 K, atomic scaling changes the internal frequency by 20 cm^{-1} while no changes are detected when the barostat is coupled to the centers of mass.

¹¹The value of W which works as “infinity” depends on the “force” that is acting on barostat coordinate expressed by the Eq. (3.25), i.e. on how far the system is from the thermodynamic equilibrium. For a system near the thermodynamic equilibrium with $N_f \simeq 10000$ a value of $W = 10^{20}$ a.m.u. is sufficient to prevent cell fluctuations.

conditions:

$$\begin{aligned} \frac{h_{\alpha\beta}}{h_{11}} - \frac{h_{\alpha\beta}^0}{h_{11}^0} &= 0 \\ \dot{h}_{\alpha\beta} - \frac{h_{\alpha\beta}^0}{h_{11}^0} \dot{h}_{11} &= 0 \quad \text{for } \alpha \leq \beta \end{aligned} \tag{3.83}$$

with \mathbf{h}^0 being some reference \mathbf{h} matrix. These constraints are implemented naturally in the framework of the multi time step velocity Verlet using the RATTLE algorithm which evaluates iteratively the constraints force to satisfy the constraints on both coordinates \mathbf{h} and velocities $\dot{\mathbf{h}}$ [25]. In Ref. [25] it is proved that the phase space sampled by the *NPT* equations with the addition of the constraints Eq. (3.83) correspond to that given by *NPT* distribution function.

Chapter 4

Multiple Time Steps Algorithms For Large Size Flexible Systems with Strong Electrostatic Interactions

In the previous sections we have described how to obtain multiple time step integrators *given* a certain potential subdivision and have provided simple examples of potential subdivision based on the inter/intra molecular separation. Here, we focus on the time scale separation of model potentials of complex molecular systems. Additionally, we provide a general potential subdivision applying to biological systems, as well as to many other interesting chemical systems including liquid crystals. This type of systems are typically characterized by high flexibility and strong Coulombic intermolecular interactions. Schematically, we can then write the potential V as due to two contributions:

$$V = V_{\text{bnd}} + V_{\text{nbnd}}. \quad (4.1)$$

Here, the “bonded” or intramolecular part V_{bnd} is fast and is responsible for the flexibility of the system. The “non bonded” or intermolecular (or intergroup) term V_{nbnd} is dominated by Coulombic interactions. The aim of the following sections is to describe a general protocol for the subdivision of such forms of the interaction potential and to show how to obtain reasonably efficient and transferable multiple time step integrators valid for any complex molecular system.

4.1 Subdivision of the “Bonded” Potential

As we have seen in Sec. 2.3 the idea behind the multiple time step scheme is that of the reference system which propagates for a certain amount of time under the influence of some *unperturbed* reference Hamiltonian, and then undergoes an impulsive *correction* brought by the remainder of the potential. The exact trajectory spanned by the complete Hamiltonian is recovered by applying this impulsive correction onto the “reference” trajectory. We have also seen in the same section that, by subdividing the interaction potential, we can determine as many “nested” reference systems as we wish. The first step in defining a general protocol for the subdivision of the bonded potential for complex molecular systems consists in identifying the various time scales and their connection to the potential. The interaction bonded potential in almost all popular force fields is given as a function of the stretching, bending and torsion internal coordinates and has the general form

$$V_{\text{bnd}} = V_{\text{stretch}} + V_{\text{bend}} + V_{\text{tors}}, \quad (4.2)$$

where

$$\begin{aligned} V_{\text{stretch}} &= \sum_{\text{Bonds}} K_r (r - r_0)^2 \\ V_{\text{bend}} &= \sum_{\text{Angles}} K_\theta (\theta - \theta_0)^2. \end{aligned}$$

$$V_{\text{tors}} = \sum_{\text{Dihedrals}} V_{\phi} [1 + \cos(n\phi - \gamma)]. \quad (4.3)$$

Here, K_r and K_{θ} are the bonded force constants associated with bond stretching and angles bending respectively, while r_0 and θ_0 are their respective equilibrium values. In the torsional potential, V_{tors} , ϕ is the dihedral angle, while K_{ϕ} , n and γ are constants.

The characteristic time scale of a particular internal degrees of freedom can be estimated assuming that this coordinate behaves like a harmonic oscillator, uncoupled from the rest the other internal degrees of freedom. Thus, the criterion for guiding the subdivision of the potential in Eq. (4.2) is given by the characteristic frequency of this uncoupled oscillator. We now give, for each type of degree of freedom, practical formula to evaluate the harmonic frequency from the force field constants given in Eq. (4.3).

Stretching: The stretching frequencies are given by the well known expression

$$\nu_s = \frac{1}{2\pi} \left(\frac{K_r}{\mu} \right)^{(1/2)}, \quad (4.4)$$

where μ is reduced mass.

Bending: We shall assume for the sake of simplicity that the *uncoupled* bending frequencies depends on the masses of the atom 1 and 3 (see Fig. 4.1), that is mass 2 is assumed to be infinity. This turns out to be in general an excellent approximation for bending involving hydrogen and a good approximation for external bendings in large molecules involving masses of comparable magnitude. The frequency is obtained by writing the Lagrangian in polar coordinates for the mechanical system depicted in Fig. 4.1. The Cartesian coordinates are expressed in terms of the polar coordinates as

$$x_1 = -r_{12} \sin(\alpha/2) \quad y_1 = r_{12} \cos(\alpha/2) \quad (4.5)$$

$$x_3 = r_{32} \sin(\alpha/2) \quad y_3 = r_{32} \cos(\alpha/2) \quad (4.6)$$

where the distance r_{32} and r_{12} are constrained to the equilibrium values. The velocities are then

$$\dot{x}_1 = -r_{12} \cos(\alpha/2) \frac{\dot{\alpha}}{2} \quad \dot{y}_1 = -r_{12} \sin(\alpha/2) \frac{\dot{\alpha}}{2} \quad (4.7)$$

$$\dot{x}_3 = r_{32} \cos(\alpha/2) \frac{\dot{\alpha}}{2} \quad \dot{y}_3 = -r_{32} \sin(\alpha/2) \frac{\dot{\alpha}}{2} \quad (4.8)$$

The Lagrangian for the uncoupled bending is then

$$\mathcal{L} = \frac{1}{2} (m_1 \dot{x}_1^2 + m_1 \dot{y}_1^2 + m_3 \dot{x}_3^2 + m_3 \dot{y}_3^2) - V_{\text{bend}} \quad (4.9)$$

$$= \frac{1}{8} (m_1 r_{12}^2 + m_3 r_{32}^2) \dot{\alpha}^2 - \frac{1}{2} k_{\theta} (\alpha - \alpha_0)^2. \quad (4.10)$$

The equation of motion $\frac{d}{dt} \frac{\partial \mathcal{L}}{\partial \dot{\alpha}} - \frac{\partial \mathcal{L}}{\partial \alpha} = 0$ for the α coordinate is given by

$$\ddot{\alpha} + \frac{4K_{\theta}}{I_b} (\alpha - \alpha_0)^2 = 0. \quad (4.11)$$

Where, $I_b = m_1 r_{12}^2 + m_3 r_{32}^2$ is the moment of inertia about an axis passing by atom 3 and perpendicular to the bending plane. Finally, the *uncoupled* bending frequency is given by

$$\nu_b = \frac{1}{2\pi} \left(\frac{4K_{\theta}}{m_1 r_{12}^2 + m_3 r_{32}^2} \right)^{(1/2)} \quad (4.12)$$

Torsion: We limit our analysis to a purely torsional system (see Fig. 4.1b) where atoms 2 and 3 are held fixed, and all bond distances and the angle θ are constrained to their equilibrium values. The system has only one degree of freedom, the dihedral angle ϕ driven by the torsional potential V_{ϕ} . Again we rewrite the kinetic energy in terms of the bond distances, the dihedral angle and the constant bend angle θ . For the kinetic energy, the only relevant coordinates are now those of atoms 1 and 4:

$$\begin{aligned} x_1 &= -d_{12} \cos \theta + \frac{d_{23}}{2} & x_4 &= d_{34} \cos \theta + \frac{d_{23}}{2} \\ y_1 &= d_{12} \sin \theta \cos(\phi/2) & y_4 &= d_{34} \sin \theta \cos(\phi/2) \\ z_1 &= -d_{12} \sin \theta \sin(\phi/2) & z_4 &= d_{34} \sin \theta \sin(\phi/2). \end{aligned} \quad (4.13)$$

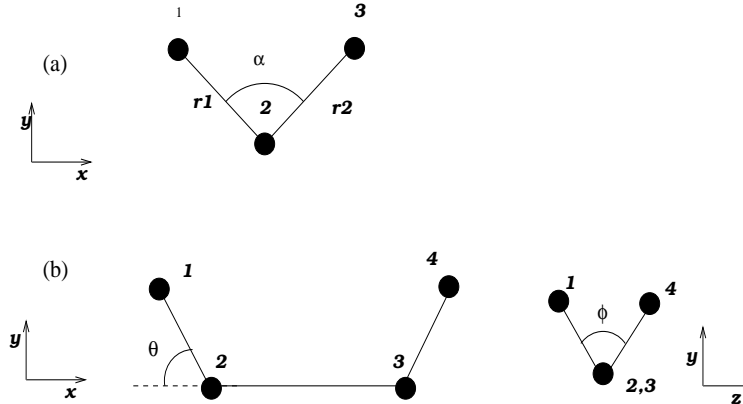


Figure 4.1: Bending and dihedral angles

The Lagrangian in terms of the dihedral angle coordinate is then

$$\mathcal{L} = \frac{1}{8} I_t \dot{\phi}^2 - V_\phi [1 + \cos(n\phi - \gamma)], \quad (4.14)$$

where

$$I_t = \sin^2 \theta (m_1 d_{12}^2 + m_4 d_{34}^2). \quad (4.15)$$

Assuming small oscillations, the potential may be approximated by a second order expansion around the corresponding equilibrium dihedral angle ϕ_0

$$V_{tors} = \frac{1}{2} \left(\frac{\partial^2 V_{tors}}{\partial \phi^2} \right)_{\phi=\phi_0} (\phi - \phi_0)^2 = \frac{1}{2} V_\phi n^2 (\phi - \phi_0)^2 \quad (4.16)$$

Substituting (4.16) into Eq. (4.14) and then writing the Lagrange equation of motion for the coordinate ϕ , one obtains again a differential equation of a harmonic oscillator, namely

$$\ddot{\phi} + \frac{4V_\phi n^2}{I_t} (\phi - \phi_0) = 0. \quad (4.17)$$

Thus, the *uncoupled* torsional frequency is given by

$$\nu_t = \frac{n}{2\pi} \left(4 \frac{V_\phi}{\sin^2 \theta (m_1 d_{12}^2 + m_4 d_{34}^2)} \right)^{(1/2)}. \quad (4.18)$$

For many all-atom force fields, improper torsions [1, 4] are modeled using a potential identical to that of the proper torsion in Eq. (4.3) and hence in these cases Eq. (4.18) applies also to the improper torsion uncoupled frequency, provided that indices 1 and 4 refer to the *lighter* atoms. In figure (4.2) we report the distribution of frequencies for the hydrated protein Bovine Pancreatin Trypsin Inhibitor (BPTI) using the AMBER [2] force field. The distributions might be thought as a density of the uncoupled intramolecular states of the system. As we can see in the figure there is a relevant degree of overlap for the various internal degrees of freedom. For example, “slow” degrees of freedom such as torsions may be found up to 600 wavenumber, well inside the “bending” region; these are usually improper or proper torsions involving hydrogen. It is then inappropriate to assign such “fast” torsions involving hydrogen to a slow reference system. We recall that in a multiple time simulation the integration of a supposedly slow degree of freedom with an excessively large time step is enough to undermine the entire simulation. In Fig. 4.2 we also notice that almost all the proper torsions fall below 350 cm^{-1} . An efficient and simple separation of the intramolecular AMBER potential [25] assigns all bendings stretching and the improper or proper torsions involving hydrogen to a “fast” reference system labeled $n0$ and all proper torsions to a slower reference

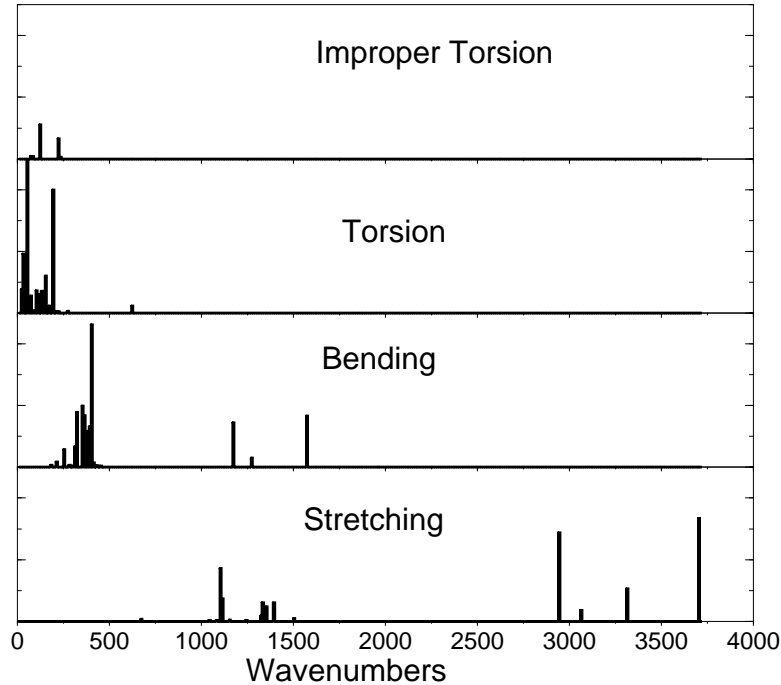


Figure 4.2: density of the uncoupled (see text) states for stretching, bending, proper and improper torsion obtained with the AMBER force field on the protein bovine pancreatic trypsin inhibitor (BPTI). Frequencies were calculated according to Eqs. (4.4,4.12,4.18)

system labeled $n1$. The subdivision is then

$$\begin{aligned} V_{n0} &= V_{\text{stretch}} + V_{\text{bend}} + V_{\text{i-tors}} + V_{\text{p-tors}}^{(h)} \\ V_{n1} &= V_{\text{p-tors}} \end{aligned} \quad (4.19)$$

Where with $V_{\text{p-tors}}^{(h)}$ we indicate proper torsions involving hydrogen. For the reference system V_{n0} , the hydrogen stretching frequencies are the fastest motions and the Δt_{n0} time step must be set to 0.2–0.3 fs. The computational burden of this part of the potential is very limited, since it involves mostly two or three body forces. For the reference system V_{n1} , the fastest motion is around 300 cm^{-1} and the time step Δt_{n1} should be set to 1–1.5 fs. The computational effort for the reference system potential V_{n1} is more important because of the numerous proper torsions of complex molecular systems which involve more expensive four body forces calculations. One may also notice that some of the bendings which were assigned to the $n0$ reference system fall in the torsion frequency region and could be therefore integrated with a time step much larger than $\Delta t_{n0} \simeq 0.2\text{--}0.3$. However, in a multiple time step integration, this overlap is just inefficient, but certainly not dangerous. Indeed, no instability may derive for integrating slow degrees of freedom with exceedingly small time steps.

4.2 The smooth particle mesh Ewald method

Before we discuss the non bonded multiple time step separation it is useful to describe in some details one of the most advanced techniques to handle long range forces. Indeed, this type of non bonded forces are the most cumbersome to handle and deserve closer scrutiny.

In the recent literature, a variety of techniques are available to handle the problem of long range interactions in computer simulations of charged particles at different level of approximation [28, 29, 10]. In this section, we shall focus on the Ewald summation method for the treatment of long range interactions in periodic systems [30, 31, 88]. The Ewald method gives the *exact* result for the electrostatic energy of a

periodic system consisting of an infinitely replicated neutral box of charged particles. The method is the natural choice in MD simulations of complex molecular system with PBC.

The Ewald potential [31] is given by

$$V'_{\text{qd}} = \frac{1}{2} \sum_{ij}^N q_i q_j \sum_n \frac{1}{|\mathbf{r}_{ij} + \mathbf{r}_n|} \text{erfc}(\alpha |\mathbf{r}_{ij} + \mathbf{r}_n|) \quad (4.20)$$

$$V_{\text{qr}} = \left[\frac{1}{2\pi V} \sum_{\mathbf{m} \neq 0}^{\infty} \frac{\exp(-\pi^2 |\mathbf{m}|^2 / \alpha^2)}{|\mathbf{m}|^2} S(\mathbf{m}) S(-\mathbf{m}) - \frac{\alpha}{\pi^{1/2}} \sum_i q_i^2 \right] - V_{\text{intra}}. \quad (4.21)$$

with

$$S(\mathbf{m}) = \sum_i^N q_i \exp(2\pi i \mathbf{m} \cdot \mathbf{r}_i) \quad (4.22)$$

$$V_{\text{intra}} = \sum_{ij-\text{excl.}} q_i q_j \frac{\text{erf}(\alpha r_{ij})}{r_{ij}}, \quad (4.23)$$

where, \mathbf{r}_i is the vector position of the atomic charge q_i , $\mathbf{r}_{ij} = \mathbf{r}_i - \mathbf{r}_j$, \mathbf{r}_n is a vector of the direct lattice, $\text{erfc}(x) = \pi^{-1/2} \int_x^\infty e^{-t^2} dt$ is the complementary error function, $\text{erf}(x) = 1 - \text{erfc}(x)$, V the unit cell volume, \mathbf{m} a reciprocal lattice vector and α is the Ewald convergence parameter. In the direct lattice part, Eq. (4.20), the prime indicates that intramolecular excluded contacts¹ are omitted. In addition, in Eq. (4.21) the term V_{intra} subtracts, in direct space, the intra-molecular energy between bonded pairs, which is automatically included in the right hand side of that equation. Consequently, the summation on i and j in Eq. (4.23) goes over all the excluded intra-molecular contacts. We must point out that in the Ewald potential given above, we have implicitly assumed the so-called “tin-foil” boundary conditions: the Ewald sphere is immersed in a perfectly conducting medium and hence the dipole term on the surface of the Ewald sphere is zero [31]. For increasingly large systems the computational cost of standard Ewald summation, which scales with N^2 , becomes too large for practical applications. Alternative algorithms which scale with a smaller power of N than standard Ewald have been proposed in the past. Among the fastest algorithms designed for periodic systems is the particle mesh Ewald algorithm (PME)[32, 33], inspired by the particle mesh method of Hockney and Eastwood [34]. Here, a multidimensional piecewise interpolation approach is used to compute the reciprocal lattice energy, V_{qr} , of Eq. 4.21, while the direct part, V_{qd} , is computed straightforwardly. The low computational cost of the PME method allows the choice of large values of the Ewald convergence parameter α , as compared to those used in conventional Ewald. Correspondingly, shorter cutoffs in the direct space Ewald sum V_{qd} may be adopted. If \mathbf{u}_j is the scaled fractional coordinate of the i -th particle, the charge weighted structure factor, $S(\mathbf{m})$ in Eq. (4.22), can be rewritten as:

$$S(\mathbf{m}) = \sum_{j=1}^N q_j \exp \left[2\pi i \left(\frac{m_1 u_{j1}}{K_1} + \frac{m_2 u_{j2}}{K_2} + \frac{m_3 u_{j3}}{K_3} \right) \right]. \quad (4.24)$$

Where, N is the number of particles, K_1, K_2, K_3 and m_1, m_2, m_3 are integers. The α component of the scaled fractional coordinate for the i -th atom can be written as:²

$$u_{i\alpha} = K_\alpha \mathbf{k}_\alpha \cdot \mathbf{r}_i, \quad (4.25)$$

where \mathbf{k}_α , $\alpha = 1, 2, 3$ are the reciprocal lattice basic vectors.

$S(\mathbf{m})$ in Eq. (4.24) can be looked at as a discrete Fourier transform (FT) of a set of charges placed irregularly within the unit cell. Techniques have been devised in the past to approximate $S(\mathbf{m})$ with expressions involving Fourier transforms on a regular grid of points. Such approximations of the weighted structure factor are computationally advantageous because they can be evaluated by fast Fourier transforms (FFT). All these FFT-based approaches involve, in some sense, a smearing of the charges over nearby grid

¹By excluded contacts we mean interactions between charges on atoms connected by bonds or two bonds apart.

²The scaled fractional coordinate is related to the scaled coordinates in Eqs (3.1,3.34) by the relation $s_{i\alpha} = 2u_{i\alpha}/K_\alpha$.

points to produce a regularly gridded charge distribution. The PME method accomplishes this task by interpolation. Thus, the complex exponential $\exp(2\pi i m_\alpha u_{i\alpha}/K_\alpha)$, computed at the position of the i -th charge in Eq. (4.24), are rewritten as a sum of interpolation coefficients multiplied by their values at the nearby grid points. In the smooth version of PME (SPME) [33], which uses cardinal B-splines in place of the Lagrangian coefficients adopted by PME, the sum is further multiplied by an appropriate factor, namely:

$$\exp(2\pi i m_\alpha u_{i\alpha}/K_\alpha) = b(m_\alpha) \sum_k M_n(u_{i\alpha} - k) \exp(2\pi i m_\alpha k/K_\alpha), \quad (4.26)$$

where n is the order of the spline interpolation, $M_n(u_{i\alpha} - k)$ defines the coefficients of the cardinal B-spline interpolation at the scaled coordinate $u_{i\alpha}$. In Eq. (4.26) the sum over k , representing the grid points, is only over a finite range of integers, since the functions $M_n(u)$ are zero outside the interval $0 \leq u \leq n$. It must be stressed that the complex coefficients $b(m_i)$ are independent of the charge coordinates \mathbf{u}_i and need be computed only at the very beginning of a simulation. A detailed derivation of the $M_n(u)$ functions and of the b_α coefficients is given in Ref. [33]. By inserting Eq. (4.26) into Eq. (4.24), $S(\mathbf{m})$ can be rewritten as:

$$S(\mathbf{m}) = b_1(m_1)b_2(m_2)b_3(m_3)\mathcal{F}[Q](m_1, m_2, m_3), \quad (4.27)$$

where $\mathcal{F}[Q](m_1, m_2, m_3)$ stands for the discrete FT at the grid point m_1, m_2, m_3 of the array $Q(k_1, k_2, k_3)$ with $1 \leq k_i \leq K_i$, $i = 1, 2, 3$. The gridded charge array, $Q(k_1, k_2, k_3)$, is defined as:

$$Q(k_1, k_2, k_3) = \sum_{i=1, N} q_i M_n(u_{i1} - k_1) M_n(u_{i2} - k_2) M_n(u_{i3} - k_3) \quad (4.28)$$

Inserting the approximated structure factor of Eq. (4.27) into Eq. (4.21) and using the fact that $\mathcal{F}[Q](-m_1, -m_2, -m_3) = K_1 K_2 K_3 \mathcal{F}^{-1}[Q](m_1, m_2, m_3)$, the SPME reciprocal lattice energy can be then written as

$$\begin{aligned} V_{\text{qr}} &= \frac{1}{2} \sum_{m_1=1}^{K_1} \sum_{m_2=1}^{K_2} \sum_{m_3=1}^{K_3} B(m_1, m_2, m_3) C(m_1, m_2, m_3) \times \\ &\times \mathcal{F}[Q](m_1, m_2, m_3) \mathcal{F}[Q](-m_1, -m_2, -m_3) \end{aligned} \quad (4.29)$$

$$\begin{aligned} &= \frac{1}{2} \sum_{m_1=1}^{K_1} \sum_{m_2=1}^{K_2} \sum_{m_3=1}^{K_3} \mathcal{F}^{-1}[\Theta_{\text{rec}}](m_1, m_2, m_3) \mathcal{F}[Q](m_1, m_2, m_3) \times \\ &\times K_1 K_2 K_3 \mathcal{F}^{-1}[Q](m_1, m_2, m_3), \end{aligned} \quad (4.30)$$

with

$$B(m_1, m_2, m_3) = |b_1(m_1)|^2 |b_2(m_2)|^2 |b_3(m_3)|^2 \quad (4.31)$$

$$C(m_1, m_2, m_3) = (1/\pi V) \exp(-\pi^2 \mathbf{m}^2/\alpha^2)/\mathbf{m}^2 \quad (4.32)$$

$$\Theta_{\text{rec}} = \mathcal{F}[BC]. \quad (4.33)$$

Using the convolution theorem for FFT the energy (4.30) can be rewritten as

$$V_{\text{qr}} = \frac{1}{2} \sum_{m_1=1}^{K_1} \sum_{m_2=1}^{K_2} \sum_{m_3=1}^{K_3} \mathcal{F}^{-1}[\Theta_{\text{rec}} \star Q](m_1, m_2, m_3) \mathcal{F}[Q](m_1, m_2, m_3) \quad (4.34)$$

We now use the identity $\sum_{\mathbf{m}} F(A)(\mathbf{m}) B(\mathbf{m}) = \sum_{\mathbf{m}} A(\mathbf{m}) F(B)(\mathbf{m})$ to arrive at

$$V_{\text{qr}} = \frac{1}{2} \sum_{m_1=1}^{K_1} \sum_{m_2=1}^{K_2} \sum_{m_3=1}^{K_3} (\Theta_{\text{rec}} \star Q)(m_1, m_2, m_3) Q(m_1, m_2, m_3) \quad (4.35)$$

We first notice that Θ_{rec} does not depend on the charge positions and that $M_n(u_{i\alpha} - k)$ is differentiable for $n > 2$ (which is always the case in practical applications). Thus the force on each charge can be obtained

by taking the derivative of Eq. (4.35), namely

$$F_{i\alpha}^{(\text{qr})} = -\frac{\partial V_{\text{qr}}}{\partial r_{i\alpha}} = \sum_{m_1=1}^{K_1} \sum_{m_2=1}^{K_2} \sum_{m_3=1}^{K_3} \frac{\partial Q(m_1, m_2, m_3)}{\partial r_{i\alpha}} (\Theta_{\text{rec}} \star Q)(m_1, m_2, m_3). \quad (4.36)$$

In practice, the calculation is carried out according to the following scheme: i) At each simulation step one computes the grid scaled fractional coordinates $u_{i\alpha}$ and fills an array with Q according to Eq. (4.28). At this stage, the derivative of the M_n functions are also computed and stored in memory. ii) The array containing Q is then overwritten by $\mathcal{F}[Q]$, i.e. Q 's 3-D Fourier transform. iii) Subsequently, the electrostatic energy is computed via Eq. (4.30). At the same time, the array containing $\mathcal{F}[Q]$ is overwritten by the product of itself with the array containing BC (computed at the very beginning of the run). iv) The resulting array is then Fourier transformed to obtain the convolution $\Theta_{\text{rec}} \star Q$. v) Finally, the forces are computed via Eq. (4.36) using the previously stored derivatives of the M_n functions to recast $\partial Q/\partial r_{i\alpha}$.

The memory requirements of the SPME method are limited. $2K_1K_2K_3$ double precision real numbers are needed for the grid charge array Q , while the calculation of the functions $M_n(u_{i\alpha} - j)$ and their derivatives requires only $6 \times n \times N$ double precision real numbers. The K_α integers determines the fineness of the grid along the α -th lattice vector of the unit cell. The output accuracy of the energy and forces depends on the SPME parameters: The α convergence parameter, the grid spacing and the order n of the B-spline interpolation. For a typical $\alpha \simeq 0.4 \text{ \AA}^{-1}$ a relative accuracies between $10^{-4} - 10^{-5}$ for the electrostatic energy are obtained when the grid spacing is around 1 \AA along each axis, and the order n of the B-spline interpolation is 4 or 5. A rigorous error analysis and a comparison with standard Ewald summation can be found in Refs. [33] and [89]. For further readings on the PME and SPME techniques we refer to the original papers [32, 89, 27, 33].

PME tests on 5CB

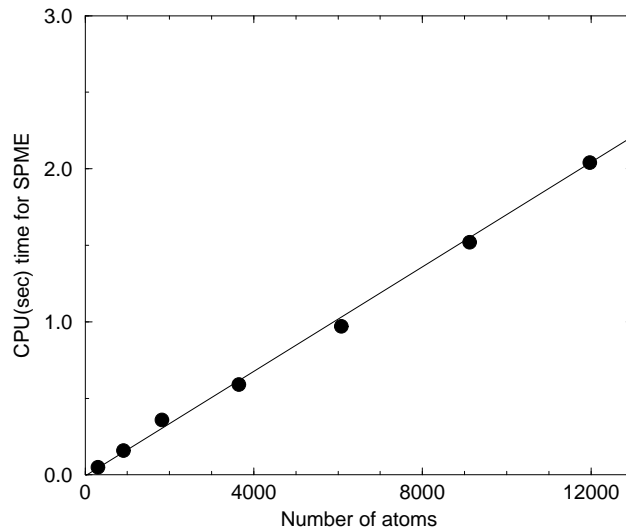


Figure 4.3: CPU time versus number of particles for the SPME algorithm as measured on a 43P/160MH IBM workstation

The power of the SPME algorithm, compared to the straightforward implementation of the standard Ewald method, is indeed astonishing. In Fig. (4.3) we report CPU timing obtained on a low end 43P/160MH IBM workstation for the evaluation of the reciprocal lattice energy and forces via SPME for cyanobiphenil as a function of the number of atoms in the system. Public domain 3-D FFT routines were used. The algorithm is practically linear and for 12,000 particles SPME takes only 2 CPU seconds to perform the calculation. A standard Ewald simulation for a box $64 \times 64 \times 64 \text{ \AA}^3$ (i.e. with a grid spacing in k space of $k = 2\pi/64 \simeq 0.01 \text{ \AA}^{-1}$) for the same sample and at the same level of accuracy would have taken several minutes.

4.3 Subdivision the Non Bonded Potential

In addition to the long range electrostatic contributions, V_{qr} and V_{qd} , given in Eqs. (4.20,4.21), more short range forces play a significant role in the total non bonded potential energy. The latter can be written as:

$$V_{nbn} = V_{vdw} + V_{qr} + V_{qd} + V_{14}. \quad (4.37)$$

Where, V_{vdw} is the Lennard-Jones potential, namely

$$V_{vdw} = \sum_{i < j}^{N'} 4\epsilon_{ij} \left[\left(\frac{\sigma_{ij}}{r_{ij}} \right)^{12} - \left(\frac{\sigma_{ij}}{r_{ij}} \right)^6 \right]. \quad (4.38)$$

Here, the prime on the sum indicates that interactions between atoms separated by less than three consecutive bonds must be omitted. The term V_{14} is typical for force fields of complex molecular systems [1, 3]. While non bonded forces between atoms involved in the same covalent bond or angle bending interaction are generally excluded, the potential between atoms separated by three covalent bonds is retained and readjusted in various ways. In all cases, the V_{14} term remains in general a very stiff and, hence, a fast varying term. The computational cost of the V_{14} contribution is very small compared to other non bonded interactions. Thus, it is safer to assigns this potential term to the slowest intramolecular reference system potential V_{n1} of Eq. (4.19).

The V_{qr} reciprocal lattice term, *including* the correction due to the excluded or partially excluded (i.e. the electrostatic part of V_{14}) interactions cannot be split when using SPME and must be assigned altogether to only one reference system. The time scale of the potential V_{qr} depends on the convergence parameter α . Indeed, this constant controls the relative weights of the reciprocal lattice energy V_{qr} , and of the direct lattice energy V_{qd} . By increasing α , one increases the weight of the reciprocal lattice contribution V_{qr} to the total Coulomb energy. When using SPME the cost of the reciprocal lattice sums is cut down dramatically and, therefore, the use of *large* α 's becomes helpful to reduce the computational burden of the direct lattice calculation. For a value of α increased beyond a certain limit, there is no longer a computational gain, since the pair distances must always be evaluated in direct space until convergence of the Lennard-Jones energy (usually occurring at a 10 Å cutoff). Furthermore, the larger is α , the more short-ranged and fast varying becomes the potential V_{qr} , thus requiring short time steps to integrate correctly the equations of motion. A good compromise for α , valid for cell of any shape and size, is $\alpha = 0.4-0.5$.

The direct space potential is separated [11, 25] in three contributions according to the interaction distance. The overall non bonded potential breakup is therefore

$$\begin{aligned} V_{n1} &= V_{14} \\ V_m &= V_{vdw}^{(1)} + V_{qd}^{(1)} \\ V_l &= V_{vdw}^{(2)} + V_{qd}^{(2)} + V_{qr} \\ V_h &= V_{vdw}^{(3)} + V_{qd}^{(3)} \end{aligned} \quad (4.39)$$

where the superscripts m, l, h of the direct space term V_{vdw} and V_{qd} refer to the short, medium and long range non-bonded interactions, respectively. The m -th reference system includes non-bonded direct space interactions at short range, typically between 0 to 4.3-5.3 Å. V_l contains both the medium range direct space potential, with a typical range of 4.3-5.3 to 7.3-8.5 Å, and the reciprocal space term, V_{qr} . Finally, the h -th reference system, which is the most slowly varying contains, the remaining direct space interactions from 7.3-8.3 Å to cutoff distance. As the simulations proceeds the particles seen by a target particle may cross from one region to another, while the number of two body contacts in one distance class [17] or reference system potential must be continuously updated. Instabilities caused by this flow across potential shell boundaries are generally handled by multiplying the pair potential by a group-based switching function [22]³. Thus, at any distance r the direct space potential V can be written schematically as:

$$V = V_1 + V_2 + V_3 \quad (4.40)$$

³Here, the word group has a different meaning than in Sec. 3 and stands for sub ensemble of contiguous atoms defined as having a total charge of approximately zero.

with

$$V_1 = VS_1 \quad (4.41)$$

$$V_2 = V(S_2 - S_1) \quad (4.42)$$

$$V_3 = V(S_3 - S_2) \quad (4.43)$$

$$(4.44)$$

where S_j is the switching function for the three shells, $j = m, l, h$ defined as:

$$S_j(R) = \begin{cases} 1 & R_{j-1} \leq R < R_j \\ S_{3p}^{(j)} & R_j \leq R < R_j + \lambda_j \\ 0 & R_j + \lambda_j < R. \end{cases} \quad (4.45)$$

Here, R is the intergroup distance and λ_j is the healing interval for the j -th shell. While R_0 is zero,

Table 4.1: Potential breakup and relative time steps for complex systems with interactions modeled by the AMBER[2] force field and electrostatic computed using the SPME method

Component	Contributions	Spherical Shells	Time step
V_{n0}	$V_{\text{stretch}} + V_{\text{bend}} +$ $+V_{\text{i-tors}} + V_{\text{p-tors}}^{(h)}$	-	$\Delta t_{n0} = 0.33 \text{ fs}$
V_{n1}	$V_{\text{p-tors}} + V_{14}$	-	$\Delta t_{n1} = 1.0 \text{ fs}$
V_m	$V_{\text{LJ}}^{(1)} + V_{\text{qd}, \alpha=0.43}^{(1)}$	$0 < r < 4.5 \text{ \AA}$	$\Delta t_m = 2.0 \text{ fs}$
V_l	$V_{\text{LJ}}^{(2)} + V_{\text{qd}, \alpha=0.43}^{(2)} +$ $+V_{\text{qr}, \alpha=0.43}$	$4.5 \leq r < 7.5 \text{ \AA}$	$\Delta t_l = 4.0 \text{ fs}$
V_h	$V_{\text{LJ}}^{(3)} + V_{\text{qd}, \alpha=0.43}^{(3)}$	$7.5 \leq r < 10.0 \text{ \AA}$	$\Delta t_h = 12.0 \text{ fs}$

$R_1 = R_m$, $R_2 = R_l$, and $R_3 = R_h$ are the short, medium, long range shell radius, respectively. The switching $S_{3p}^{(j)}(R)$ is 1 at R_j and goes monotonically to 0 at $R_j + \lambda_j$. Provided that $S_{3p}^{(j)}$ and its derivatives are continuous at R_j and $R_j + \lambda_j$, the analytical form of S_{3p} in the healing interval is arbitrary [14, 18, 22, 11]. The full breakup for an AMBER type force field along with the integration time steps, valid for any complex molecular system with strong electrostatic interactions, is summarize in table II. The corresponding five time steps integration algorithm for the NVE ensemble is given by

$$e^{iL\Delta t_h} = \exp \left[-\frac{\partial V_h}{\partial \mathbf{r}_i} \frac{\partial}{\partial \mathbf{p}_i} \frac{\Delta t_h}{2} \right] \cdot \left\{ \exp \left[-\frac{\partial V_l}{\partial \mathbf{r}_i} \frac{\partial}{\partial \mathbf{p}_i} \frac{\Delta t_l}{2} \right] \left\{ \exp \left[-\frac{\partial V_m}{\partial \mathbf{r}_i} \frac{\partial}{\partial \mathbf{p}_i} \frac{\Delta t_m}{2} \right] \right. \right. \\ \left. \left\{ \exp \left[-\frac{\partial V_{n1}}{\partial \mathbf{r}_i} \frac{\partial}{\partial \mathbf{p}_i} \frac{\Delta t_{n1}}{2} \right] \right\} \exp \left[\dot{\mathbf{r}}_i \frac{\partial}{\partial \mathbf{r}_i} \Delta t_0 \right] \right. \\ \left. \exp \left[-\frac{\partial V_{n0}}{\partial \mathbf{r}_i} \frac{\partial}{\partial \mathbf{p}_i} \frac{\Delta t_{n0}}{2} \right] \right\}^{n_{n0}} \exp \left[-\frac{\partial V_{n1}}{\partial \mathbf{r}_i} \frac{\partial}{\partial \mathbf{p}_i} \frac{\Delta t_{n1}}{2} \right] \right\}^{n_{n1}} \\ \exp \left[-\frac{\partial V_m}{\partial \mathbf{r}_i} \frac{\partial}{\partial \mathbf{p}_i} \frac{\Delta t_m}{2} \right] \right\}^{n_m} \exp \left[-\frac{\partial V_l}{\partial \mathbf{r}_i} \frac{\partial}{\partial \mathbf{p}_i} \frac{\Delta t_l}{2} \right] \right\}^{n_l} \exp \left[-\frac{\partial V_h}{\partial \mathbf{r}_i} \frac{\partial}{\partial \mathbf{p}_i} \frac{\Delta t_h}{2} \right], \quad (4.46)$$

where $n_l = \Delta t_h / \Delta t_l$, $n_m = \Delta t_l / \Delta t_m$, $n_{n1} = \Delta t_m / \Delta t_{n1}$, $n_{n0} = \Delta t_{n1} / \Delta t_{n0}$. The explicit integration algorithm can be easily derived applying the five-fold discrete time propagator (4.46) to the state vector $\{\mathbf{p}, \mathbf{q}\}$ at time 0 using the rule Eq. (2.23). The efficiency and accuracy for energy conservation of this r-RESPA symplectic and reversible integrator have been discussed extensively in Refs. [11, 90]. Extension of this subdivision to non NVE simulation is described in Ref. [25].

4.4 Electrostatic Corrections for the Multiple Time Step Simulation

In *flexible* molecular systems of large size, the Ewald summation presents computational problems which are crucial to constructing efficient and *stable* multiple time step integrators [91, 90]. We have seen that

intra-molecular Coulomb interactions between bonded atoms or between atoms bonded to a common atom are excluded in most of the standard force fields for protein simulation. In any practical implementation of the Ewald method, the intra-molecular energy V_{intra} is automatically included in the reciprocal space summation and is subtracted in *direct space* (see Eqs. (4.23,4.21)). In actual simulations the reciprocal space sum is computed with a finite accuracy whereas the intra-molecular term V_{intra} , due to the excluded Coulombic interactions, is computed exactly. This clearly prevents the cancellation of the intra-molecular forces and energies. When the stretching and bending forces are integrated explicitly, the intra-molecular term due to the excluded contacts varies rapidly with time and so does the cancellation error. Consequently, instability may be observed when integrating the reciprocal lattice forces in reference systems with large time steps. The correction due to the truncation can be evaluated by approximating the reciprocal lattice sum for the excluded contacts in Eq. (4.21) to an integral in the 3-dimensional k space and evaluating this integral from the cutoff $k_{\text{cut}} \equiv 2\pi|\mathbf{m}|_{\text{max}}$ to infinity in polar coordinates. The neglected reciprocal lattice intra-molecular energy is then [92]

$$V_{\text{corr}} = \frac{\alpha}{\pi^{1/2}} \text{erfc}(k_{\text{cut}}/2\alpha) \sum_i q_i^2 + \sum_{ij-\text{excl.}} q_i q_j \chi(r_{ij}, k_{\text{cut}}, \alpha) \quad (4.47)$$

with

$$\chi(r, k_{\text{cut}}, \alpha) = \frac{2}{\pi} \int_{k_{\text{cut}}}^{\infty} e^{-k^2/4\alpha^2} \frac{\sin(kr)}{kr} dk. \quad (4.48)$$

The first constant term in (4.47) refers to the self energy, while the second accounts for the intra-molecular excluded interactions⁴. This correction must be included in the same reference systems to which V_{qr} is assigned, e.g. V_l in our potential separation (see Table 2).

In principle the correction in Eq. (4.47) applies only to standard Ewald and not to the reciprocal lattice energy computed via SPME. We can still, however, use the correction Eq. (4.47), if a spherical cutoff k_{cut} is applied to SPME. This can be done easily by setting $\exp(-\pi^2 \mathbf{m}^2 / \alpha^2) / \mathbf{m}^2 = 0$ for $2\pi \mathbf{m} > k_{\text{cut}} \equiv f_f \pi N_f / L$ where L is the side length of the cubic box and N_f is the number of grid points in each direction. The factor f_f must be chosen slightly less than unity. This simple device decreases the effective cutoff in reciprocal space while maintaining the same grid spacing, thus reducing the B-spline interpolation error (the error in the B-spline interpolation of the complex exponential is, indeed, maximum precisely at the tail of the reciprocal sums [33]). In Ref. [92] the effect of including or not such correction in electrostatic systems using multiple time step algorithms is studied and discussed thoroughly.

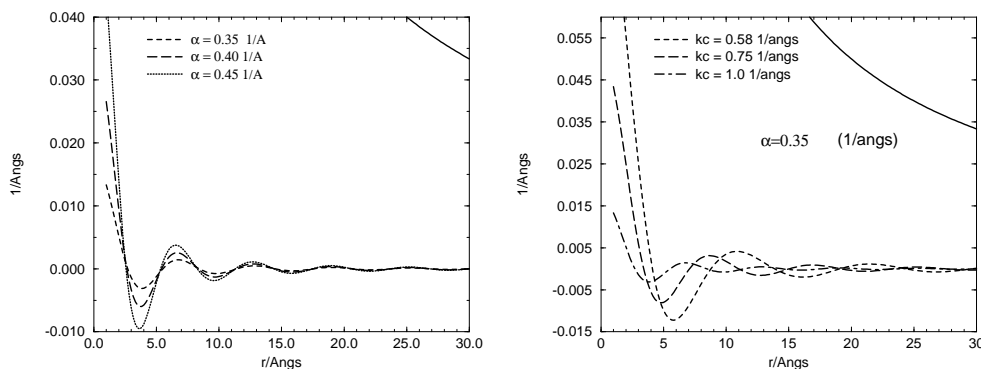


Figure 4.4: The correction potential $\chi(r, k_c, \alpha)$ as a function of the distance for different values of the parameters α (left) and k_c (right). The solid line on the top right corner is the bare Coulomb potential $1/r$

The potential $\chi(r, k_{\text{cut}}, \alpha)$ yields, in *direct space*, the neglected reciprocal energy due to the truncation of the reciprocal lattice sums, and must, in principle, be included for each atom pair distance in direct

⁴Note that $\lim_{k \rightarrow 0} \chi(r, k, \alpha) = \text{erf}(\alpha r)/r$.

space. Thus, the corrected direct space potential is then

$$V'_{\text{qd}} = \frac{1}{2} \sum_{ij}^N q_i q_j \left[\sum_n \frac{\text{erfc}\alpha |\mathbf{r}_{ij} + \mathbf{r}_n|}{|\mathbf{r}_{ij} + \mathbf{r}_n|} + \chi(|\mathbf{r}_{ij} + \mathbf{r}_n|, k_{\text{cut}}, \alpha) \right] \quad (4.49)$$

which is then split as usual in short-medium-long range according to (4.39). The correction is certainly more crucial for the excluded intramolecular contacts because V_{corr} is essentially a short-ranged potential which is non negligible only for intramolecular short distances. For systems with hydrogen bonds, however, the correction is also important for intermolecular interactions.

In Fig. 4.4 the correction potential is compared to the Coulomb potential (solid line in the top right corner) for different value of the reciprocal space cutoff k_c and of the convergence parameter α . For practical values of α and k_c , the potential is short ranged and small compared to the bare $1/r$ Coulomb interaction. In the asymptotic limit V_{corr} goes to zero as $\sin(ar)/r^2$ where a is a constant. This oscillatory long range behavior of the correction potential V_{corr} is somewhat nasty: In Fig. 4.5 we show the integral

$$I(r, k_c, \alpha) = \int_0^r \chi(x, k_c, \alpha) x^2 dx \quad (4.50)$$

as a function of the distance. If this integral converges then the $\chi(r, k)$ is absolutely convergent in 3D. We

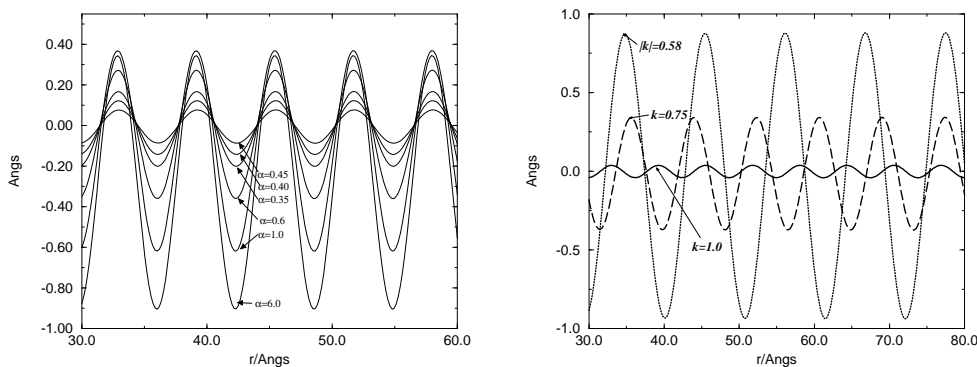


Figure 4.5: The integral $I(r)$ of Eq. (4.50) as a function of the distance for different values of the parameters α (left) and k_c (right)

see that the period of the oscillations in $I(r)$ increases with k_c while α affects only the amplitude. The total energy is hence again conditionally convergent, since the limit $\lim_{r \rightarrow \infty} I(r)$ does not exist. However, unlike for the $1/r$ bare potential, the energy integral remains in this case bounded. Due to this, a cutoff on the small potential V_{corr} is certainly far less dangerous than a cutoff on the bare $1/r$ term. In order to verify this, we have calculated some properties of liquid water using the SPC model[93] from a 200 ps MD simulation in the NPT ensemble at temperature of 300 K and pressure of 0.1 MPa with i) a very accurate Ewald sum (column EWALD in Table 4.2), ii) with inaccurate Ewald but corrected in direct space using Eq. (4.49) (CORRECTED) and iii) with simple cutoff truncation of the bare Coulomb potential and no Ewald (CUTOFF). Results are reported in Table 4.2. We notice that almost all the computed properties of water are essentially independent, within statistical error, of the truncation method. The dielectric properties, on the contrary, appear very sensitive to the method for dealing with long range tails: Accurate and inaccurate Ewald (corrected in direct space through 4.49) yields, within statistical error, comparable results whereas the dielectric constant predicted by the spherical cutoff method is more than order of magnitude smaller. We should remark that method ii) (CORRECTED) is almost twice as efficient as the “exact” method i).

	EWALD	CORRECTED	CUTOFF
Coulomb energy (KJ/mole)	-55.2 \pm 0.1	-55.1 \pm 0.1	-56.4 \pm 0.1
Potential energy (KJ/mole)	-46.2 \pm 0.1	-46.1 \pm 0.1	-47.3 \pm 0.1
Heat Capacity (KJ/mole/K)	74 \pm 24.5	94 \pm 22.0	87 \pm 23.2
Volume (cm ³)	18.2 \pm 0.1	18.3 \pm 0.1	18.1 \pm 0.1
Volume Fluctuation (\AA^3)	136.9 \pm 3.5	147.0 \pm 3.5	138.7 \pm 3.5
R_{0-0} (\AA)	2.81 \pm 0.01	2.81 \pm 0.01	2.81 \pm 0.01
Dielectric constant	59 \pm 25.8	47 \pm 27.3	3 \pm 2

Table 4.2: Properties of liquid water computed from a 200 ps simulation at 300 K and 0.1 Mpa on a sample of 343 molecules in PBC with accurate Ewald ($\alpha = 0.35 \text{ \AA}^{-1}$; $k_c = 2.8 \text{ \AA}^{-1}$) and no correction Eq. (4.47) (column EWALD), with inaccurate Ewald ($\alpha = 0.35 \text{ \AA}^{-1}$; $k_c = 0.9 \text{ \AA}^{-1}$) but including the correction Eq. (4.47) and with no Ewald and cutoff at 10.0 \AA . R_{0-0} is the distance corresponding to the first peak in the Oxygen-Oxygen pair distribution function.

Chapter 5

The Hamiltonian Replica Exchange Method

5.1 Temperature REM

The Replica Exchange Method is based on multiple concurrent (parallel) canonical simulation that are allowed to occasionally exchange their configurations. For a system made of N atoms, by “configuration” we mean a state defined by a $3N$ dimensional *coordinate* vector, independent of the momenta. Thus, in a replica exchange, only coordinates and not momenta are exchanged. In the standard implementation of the methodology, each replica, bearing a common interaction potential, is characterized by a given temperature and configurations between couple of replicas are tentatively exchanged at prescribed time intervals using a probabilistic criterion. The *target* temperature, i.e. the temperature corresponding to the thermodynamic state of interest, is usually the lowest among all replicas. In this manner, “hot” configurations from hot replicas, i.e. configurations where energy barrier are easily crossed, may be occasionally accepted at the target temperature. The canonical probability of a coordinate configuration X for m -th replica is given by

$$P_m(X) = \frac{1}{Z_m} e^{-\beta_m V(X)} \quad (5.1)$$

where m is the replica index, $\beta^{-1} = k_B T$, $V(x)$ is the potential of the system, and $Z_m = \int e^{-\beta_m V(X)} dX$ is the configurational partition function for m -th replica. Being the M replicas independent, the probability distribution for a generic configuration of the M -fold extended system $\mathbf{X} = (X_1, \dots, X_M)$ is

$$P_{\mathbf{X}} = \prod_m^M P_m(X_m) \quad (5.2)$$

As stated above, the global state \mathbf{X} of the extended system may evolve in two ways: i) by evolving each replica independently (i.e. via MC or MD simulation protocols) and ii) by exchanging the configurations of two replicas. Regarding the second mechanism, we introduce the transition probability $W(X, \beta_m; X', \beta_n)$ for the exchange between the configuration X of replica at T_m and the configuration X' for the replica T_n . The probability for the inverse exchange is clearly given by $W(X', \beta_m; X, \beta_n)$. The detailed balance condition on the extended system for this kind of moves is given by

$$P_{\mathbf{X}}(\dots, X, \beta_m, X', \beta_n, \dots) W(X, \beta_m; X', \beta_n) = \quad (5.3)$$

$$P_{\mathbf{X}'}(\dots, X', \beta_m; X, \beta_n, \dots) W(X', \beta_m; X, \beta_n) \quad (5.4)$$

which, using the expressions 5.2 and 5.1 for the global probability, is satisfied if the transition probability satisfies the equation

$$\frac{W(X, \beta_m, X', \beta_n)}{W(X', \beta_m; X, \beta_n)} = e^{-(\beta_m - \beta_n)(E(X') - E(X))}. \quad (5.5)$$

The exchange of configurations of replicas obeying the detailed balance condition 5.5 can be as usual implemented by using the Metropolis algorithm

$$P_{\text{acc}} = \min(1, e^{-\Delta}) \quad (5.6)$$

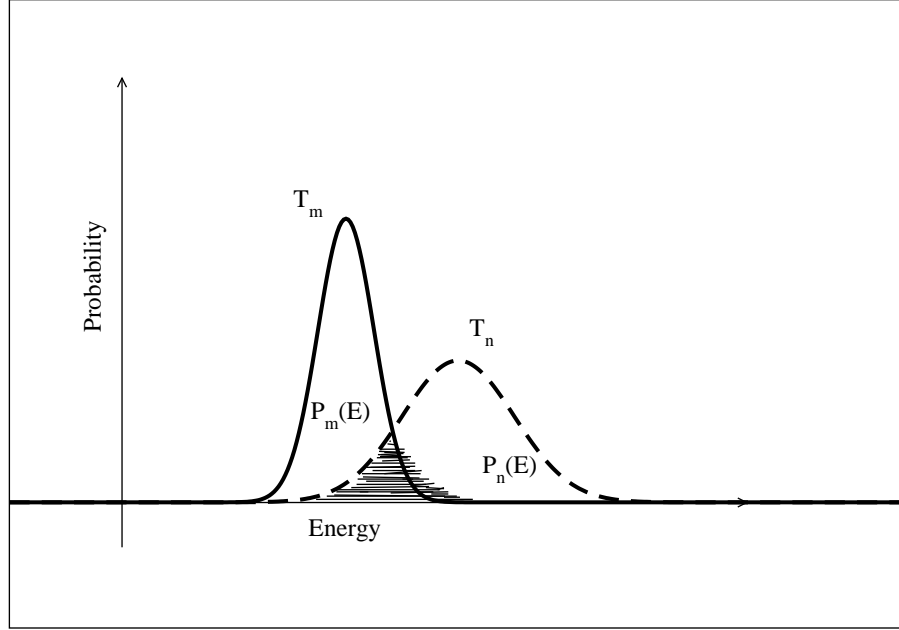


Figure 5.1: Overlapping configurational energy distribution for two replicas. The shaded area is the acceptance probability for the configuration exchange. The overlap between the two distribution is a lower bound for the acceptance probability

with $\Delta = (\beta_m - \beta_n)(E(X') - E(X))$. Like in a standard MC technique, because of the detailed balance condition for the extended system, the sampling in the \mathbf{X} multi-configuration space in REM evolves towards a global equilibrium defined by the multi-canonical probability distribution of the extended system Eq. 5.2.

In principle Eq. 5.6 refers to the probability of an exchange between any two replicas. In practice the exchanges are attempted between replicas that are contiguous in temperature. Let's see why. For any two replicas m, n , the total number of accepted exchanges between them is given by

$$N^{\text{acc}} = N^{\text{acc}}(\Delta E < 0) + N^{\text{acc}}(\Delta E > 0) \quad (5.7)$$

where $\Delta E = (E(X') - E(X))$ and $N^{\text{acc}}(\Delta E < 0)$, $N^{\text{acc}}(\Delta E > 0)$ are the number of accepted exchanges for which $\Delta E < 0$ and $\Delta E > 0$, respectively. When the extended system is at equilibrium, we clearly must have that

$$N^{\text{acc}}(\Delta E < 0) = N^{\text{acc}}(\Delta E > 0) \quad (5.8)$$

Inserting the above equation into Eq. 5.7, we obtain

$$N^{\text{acc}} = 2N^{\text{acc}}(\Delta E < 0) \quad (5.9)$$

Since, according to the prescription 5.6, the probability for accepting the move when $\Delta E < 0$ is unitary, we may write that

$$\frac{N^{\text{acc}}}{N^{\text{tot}}} = 2P(\Delta E < 0) \quad (5.10)$$

where N^{tot} is the total number of attempted exchanges and $P(\Delta E < 0)$ is the cumulative probability that a $E(X') < E(X)$. Eq. 5.10 states that if the two normalized (configurational) energy distribution $P_m(E)$ of replica m and $P_n(E)$ of replica n are identical, then the probability for a successful exchange between the two replica is equal to the area of the overlap of the two distribution (i.e. the shaded area in Fig. 5.1). If $P_m(E)$ and $P_n(E)$ are not identical, we have in general that the overlap of the two distribution is a lower bound for the acceptance probability (the standard deviation δE generally increases with the mean energy \bar{E}). Based on the above, and assuming that M , the total number of replicas, is even, one can then set up

an exchange protocol periodically attempting $M/2$ simultaneous contiguous replica exchanges $m \leftrightarrow m+1$ with m odd, or $M/2 - 1$ simultaneous contiguous replica exchanges $m \leftrightarrow m+1$ with m even, accepting *each* of them with probability given by 5.6.

Given the above scheme, what is the optimal spacing in temperatures for enhanced sampling of the configuration space at the target temperature? First of all, the hottest temperature T_M , defining the full temperature range $\Delta T = T_M - T_1$ of the extended system, must be clearly selected such that $k_B T_M$ is of the order of the maximum height of the free energy barriers that must be overcome at the target temperature T_1 . Concerning the temperature spacing, we have seen that acceptance probability for an exchange is larger, the larger is the overlap of the two energy distributions referring to the two contiguous replica, i.e. the closer are the temperatures. Of course, the closer are the temperatures and the larger is the number of replicas to be simulated, i.e. the heavier is the CPU cost of the simulation. For an optimal choice, we thus set

$$\bar{E}_{m+1} - \bar{E}_m = \sigma_{E_m} \quad (5.11)$$

where \bar{E}_m and σ_{E_m} are the mean energy and the standard deviation of energy distribution for the m -th replica. Assuming then that the system can be described by an ensemble of N harmonic oscillators, we have that $\bar{E}_m = NkT_m$ and $\sigma_{E_m} = cN^{1/2}k^{1/2}T_m$.¹ Substituting these values in Eq. 5.11, we obtain the temperature spacing for optimal superposition:

$$\bar{T}_{m+1} - \bar{T}_m = \left(\frac{c^2}{Nk_b} \right)^{1/2} T_m \quad (5.12)$$

In the parallel implementation of the temperature REM, in order to keep the communication overhead at the lowest possible level, we standardly exchange the temperatures and not the configurations. So the m -th slave process may explore *the entire range* of temperatures. When the m -th slave process periodically writes out the coordinates of the configuration (Typically in `pdb` or `xyz` format), one must also keep track of the current temperature (the program does this automatically) in order to be able to reconstruct *a posteriori* the true m -th temperature configurational space of the m -th replica. In Fig. 5.2, we show a typical parallel REM simulation for a general system with 8 processes. In the x-axis we report the simulation time, in the left y-axis the process index and in the right y-axis the replica index which is bound to the actual temperature. Each color represents a process running in parallel with other processes with different colors. As it can be seen, on each process the temperature (i.e. the replica index) changes continuously. So, for example the configurational sampling of the replica at the lowest temperature in the given time interval must be reconstructed combining the data for the slave processes 1,2,3,4,6. If the algorithm is working properly, (i.e. if the temperature spacing is chosen correctly and if there are no phase transition between T_1 and T_M), the temperature in each parallel process must perform a random walk in the temperature domain $[T_1, T_M]$.

Going back to equation 5.12, two important issues must be stressed: i) the temperature spacing for optimal overlap between contiguous replicas while keeping the total number of replicas not too high, is not uniform but grows with the replica temperature; ii) the temperature spacing between contiguous replicas must be decreased with increasing number of degrees of freedom. The latter is indeed a severe limitation of the standard REM technology, since, as the size of the system grows, a larger number of replicas must be employed for preserving a significant exchange acceptance probability. This is due to the inescapable fact that the energy fluctuations grow with $N^{1/2}$ while the energy grows with N . Moreover, in many important cases, one has to effectively sample reaction coordinates that are rather localized in the protein, like e.g. in the case of substrate-active site interactions. In the standard temperature REM, the extra heat in the hot replicas is clearly distributed among *all the degrees of freedom* of the system and therefore *most* of this heat is uselessly used for exchanging uninteresting configurations (e.g. solvent configurations).

5.2 Hamiltonian REM

In this program we adopt a variant of the replica exchange called Hamiltonian REM, that is far more flexible than the standard temperature REM technique illustrated above. In the Hamiltonian REM, each replica is

¹In the latter equation c is a constant that depends on the density of states of the N harmonic oscillators and $c^2 N = C_v$ with C_v being the constant volume heat capacity of the system.

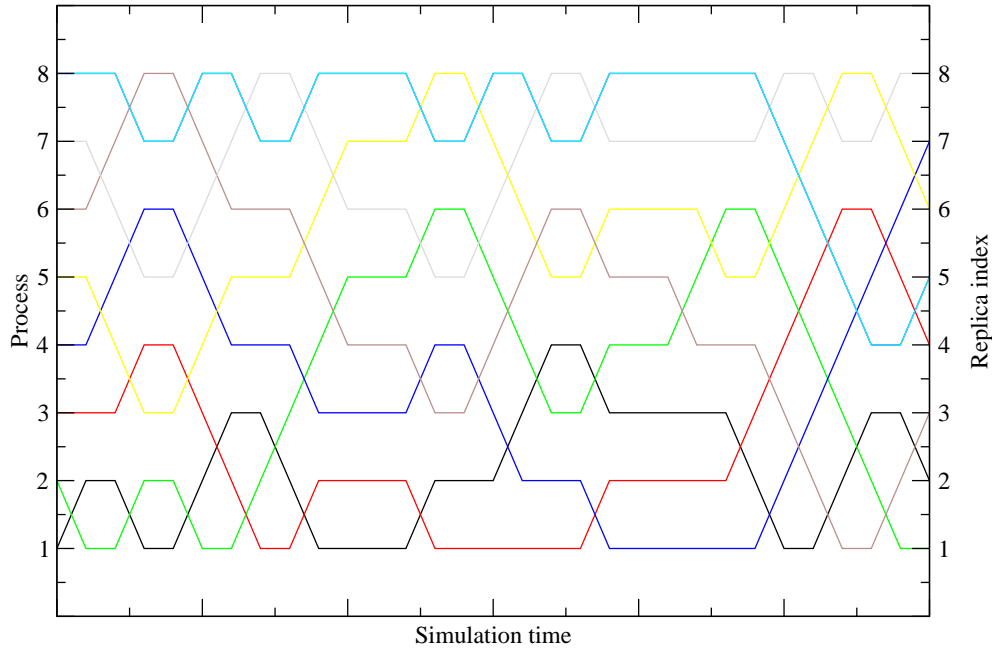


Figure 5.2: Typical REM Simulation with 8 replicas. Each process bear a particular color and the color follows the right scale, i.e. the replica index which is connected to the temperature. To reconstruct a trajectory at a given temperatures, one must combine the data form several processes.

characterized by a different potential energy rather than by a temperature. In its simplest implementation, the potential energies of the replicas differ by a scaling factor c_m , with $c_1 = 1$ for the target replica. Clearly, as long as the exchanged states differ only in the coordinates (i.e. momenta are not exchanged), the scaling of the potential energy of a canonical system (NVT) is equivalent to an inverse temperature scaling. Thus, Hamiltonian REM with *full* potential energy scaling and temperature REM are perfectly equivalent in an extended Monte Carlo simulation. When the replica simulations are done by numerically, integrating the Nosé-Hoover equations of motion at constant volume, Hamiltonian REM with full *potential* energy scaling and temperature REM are clearly no longer equivalent. Since momenta are not exchanged, Eq. 5.6 is valid for both full Hamiltonian REM and temperature REM, but in the latter technique *both* the kinetic and the potential energy are scaled, while in the former as implemented in ORAC one scales *only* the potential energy.

The advantage of using the Hamiltonian REM is two-fold: i) as all the replica have the same operating temperature, one does not have, like in temperature REM, to reinitialize the velocities after one successful configuration exchange and ii) since the mean atomic velocities are the same throughout the extended system, one does not have to adapt the time step size for preserving the quality of r-RESPA integrator as it should be done in temperature REM.

Hamiltonian REM can also be applied to *a specific part* of the potential, thereby localizing the effect of the configurational exchanges to specific part of the systems. Given a potential made up of a sum of various $i = 1, \dots, k$ contributions (e.g. stretching, bending, torsional, solute-solvent solute-solute solvent-solvent non bonded etc.), then one can define in a general way the m -th replica of the extended system as

$$V_m(X) = \sum_{i=1}^k c_i^{(m)} v_i(X) \quad (5.13)$$

where $c_i^{(m)}$ is the scaling factor for the i -th contribution, $v_i(X)$ of the potential of the m -th replica. So each replica is characterized by a k -dimensional scaling vector $\mathbf{c}_m = (c_1^{(m)}, \dots, c_i^{(m)}, \dots, c_k^{(m)})$ whose component are the scaling factors of k contributions of the interaction potential for that replica. The target replica,

replica 1, is such that $V_1(x) = V(x)$, the unscaled potential corresponding to the target system for which $\mathbf{c}_m = (1, 1, 1, \dots)$. In vector notation we may compactly write Eq. 5.13 as

$$V_m(X) = \mathbf{c}_m \cdot \mathbf{v}(X) \quad (5.14)$$

Using this formalism, the probability of a configuration X in the m -th replica may be written as

$$P_m(X) = \frac{1}{Z_m} e^{-\beta \mathbf{c}_m \cdot \mathbf{v}(X)}. \quad (5.15)$$

with $Z_m = \int e^{-\beta \mathbf{c}_m \cdot \mathbf{v}(X)} dX$. The detailed balance condition for the exchange of configurations between replica m (characterized by the scaling vector \mathbf{c}_m) and replica n (characterized by the scaling vector \mathbf{c}_n) is then given by

$$\frac{W(X, \mathbf{c}_m, X', \mathbf{c}_n, \beta)}{W(X', \mathbf{c}_m; X, \mathbf{c}_n, \beta)} = e^{-\beta(\mathbf{c}_m - \mathbf{c}_n) \cdot [\mathbf{v}(X') - \mathbf{v}(X)]} \quad (5.16)$$

Again, the detailed balance is implemented through Eq. 5.6 with

$$\begin{aligned} \Delta &= \beta(\mathbf{c}_m - \mathbf{c}_n) \cdot [\mathbf{v}(X') - \mathbf{v}(X)] \\ &= \beta \sum_{i=1}^k (c_i^{(m)} - c_i^{(n)}) [V_i(X') - V_i(X)] \end{aligned} \quad (5.17)$$

There is considerable freedom in the splitting of the potential (Eq. 5.13) and in the selection of the corresponding scaling factors. These factors are always positive and can be either smaller or greater than one, meaning that the corresponding potential contributions, for $m > 1$, imply a heating and a cooling, respectively, of the involved degrees of freedom. For example we could use $c < 1$ for torsions and $c > 1$ for bending, so that, with increasing m , torsional degrees of freedom are heated up while bending are frozen down.

Global scaling: In the present implementation of ORAC, one can do a *global* subdivision (i.e. ignoring the distinction between solvent and solute) of the overall atomistic interaction potential for biomolecular system according to the following:

$$V_m(X) = c_{\text{ba}}^m (V_{\text{Bonds}} + V_{\text{Angle}} + V_{\text{i-tors}}) + c_t^{(m)} (V_{\text{tors}} + V_{14}) + \quad (5.18)$$

$$+ c_{\text{nb}}^{(m)} (V_{\text{vdw}} + V_{\text{qr}} + V_{\text{qd}}) \quad (5.19)$$

where the meaning of the subscripts is given in Sec. 4.1. Typically one then sets $c_{\text{ba}}^{(m)} = 1 \ \forall m$, as there is little advantage for conformational sampling in exchanging configurations involving stiff degrees of freedom such as bending, stretching and improper torsion. On the other hand, conformational transitions in proteins are mainly driven by torsional and intraprotein and protein-solvent non bonded interactions. It is thus convenient to heat up these degrees of freedom by scaling with $c_t^{(m)} < 1$ and $c_{\text{nb}}^{(m)} < 1$ the corresponding potential functions. With this choice the quantity Δ in Eq. 5.17 is given by

$$\begin{aligned} \Delta &= \beta(c_t^{(m)} - c_t^{(n)}) [V_{\text{tors}}(X') + V_{14}(X') - V_{\text{tors}}(X) - V_{14}(X)] + \\ &+ \beta(c_{\text{nb}}^{(m)} - c_{\text{nb}}^{(n)}) [V_{\text{vdw}}(X') + V_{\text{qr}}(X') + V_{\text{qd}}(X') - (V_{\text{vdw}}(X) + V_{\text{qr}}(X) + V_{\text{qd}}(X))] \end{aligned} \quad (5.20)$$

Local scaling: Hamiltonian REM in ORAC can work also by tempering only a user defined “solute”. Unlike standard implementation of the solute tempering techniques[94], the “solute” in the present version can be any portion of the system including solvent molecules. Once the solute has been defined, the complementary “non solute” portion of the system is by definition the “solvent”. In this manner, the scaling (i.e. the heating or freezing) can be localized in specific part of the system with the remainder (the “solvent”) of the system behaving normally (i.e. with the target interaction potential). In order to clarify how local scaling work, we illustrate the technique with a working general example. Suppose to choose a subset of atoms n in the system that define the “solute”. This subset can be chosen arbitrarily and may include disconnected portions of the protein, as well as selected solvent molecules. The solvent is then made up of the remaining $N - n$ atoms. According to this subdivision, the global potential of the system may be written as

$$V(X) = V_{\text{bd}}(X_N) + V^{(\text{Slv})}(X_n) + V^{(\text{Slv-Slv})}(X_n, X_{N-n}) + V^{(\text{Slv-Slv})}(X_{N-n}) \quad (5.21)$$

where

$$\begin{aligned}
V_{\text{bd}}(X_N) &= (V_{\text{Bonds}} + V_{\text{Angle}} + V_{\text{i-tors}}) \\
V^{(\text{Slt})}(X_n) &= V_{\text{tors}}(X_n) + V_{14}(X_n) + (V_{\text{vdw}}(X_n) + V_{\text{qd}}(X_n)) \\
V^{(\text{Slt-Slv})} &= (V_{\text{vdw}}(X_n, X_{N-n}) + V_{\text{qd}}(X_n, X_{N-n})) \\
V^{(\text{Slv-Slv})}(X_{N-n}) &= (V_{\text{vdw}}(X_n, X_{N-n}) + V_{\text{qr}}(X_n, X_{N-n}) + V_{\text{qr}} + V_{\text{bd}}(X_N))
\end{aligned} \tag{5.22}$$

The solute potential $V^{(\text{Slt})}(X_n)$ includes all the proper torsions and the 14 non bonded interactions involving the n atoms of the solute.² The solute-solvent interaction involve all non-bonded interactions between the N solvent atoms and the n solute atoms. The solvent-solvent interaction involve all non-bonded interactions among the $N - n$ solvent atoms. It should also be remarked that the reciprocal lattice contribution V_{qr} , i.e. the long range electrostatics, is in any case assigned to the solvent-solvent term even if it includes all kinds of non bonded interaction (solute-solute, solvent-solute and solvent-solvent). The reason why V_{qr} is not split in the solute-solute, solute-solvent and solvent-solvent components is both physical and practical. Firstly, the long-range potential associated to each of three component of this term is expected, in general, to be rather insensitive along arbitrary reaction coordinates, such that a scaling of V_{qr} do not correspondingly produce a significant heating of any conformational coordinate. Secondly, in the Particle Mesh Approach approach the solute-solute, solvent-solute and solvent-solvent contribution to V_{qr} can no longer be easily separated, and this term must be thus arbitrarily assigned to one of the three components. Given the subdivision Eq. 5.22, the local scaling for replica m in ORAC is implemented as

$$V_m(X) = c_{\text{Slt}}^{(m)} V^{(\text{Slt})}(X_n) + c_{(\text{Slt-Slv})}^{(m)} V^{(\text{Slt-Slv})}(X_n, X_{N-n}) + V^{(\text{Slv-Slv})}(X_{N-n}) + V_{\text{bd}}(X_N) \tag{5.23}$$

The solvent-solvent interactions are not scaled in the local approach. As one can see, also the *global* fast bonded potential is not scaled thus being virtually assimilated to a solvent-solvent contribution that remains cold throughout the extended system.

Solute-solute interactions and solute-solvent interactions as defined in Eq. 5.22 are scaled independently, thereby generalizing the so-called solute-tempering approach recently proposed by Liu *et al.*[94]. This generality allows a complete freedom in the choice of the scaling protocol. For example, one can choose to set $c_{(\text{Slt-Slv})}^{(m)} > 1$, i.e. to progressively “freeze” the solute-solvent interaction as the replica index m grows, while at the same time setting $c_{(\text{Slt-Slt})}^{(m)} = 1$ for all replicas, thereby favouring, at large $c_{(\text{Slt-Slv})}^{(m)}$ the “solvation” of the solute, i.e., for example, favouring the unfolding.

The global REM algorithm (i.e. uniform scaling of the full interaction potential) as implemented in ORAC works also for constant pressure simulation (see `ISOSTRESS` directive). In that case, the selected external pressure refers to that of the *target* replica ($m = 1$). Since the PV is a configurational term and *is not* scaled in the current implementation, the non target replicas sample coordinate configurations according to a *higher* external pressure, i.e. $P_m = P_1/c_m$ where c_m is the scaling factor of replica m . This choice is done in order to avoid, through an increase of the external pressure, a catastrophic expansion of the simulation box for low scaling factors (or high temperatures).

5.3 Analysis of the REM data

As recently shown by Shirts and Chodera[95], all the configurations produced by a REM simulation of M replicas, each characterized by a distribution function $P_m(X)$, can be effectively used to obtain equilibrium averages for any target distribution $P_n(X)$. In the ORAC REM implementation, the most general distribution function for replica m is given by Eq. 5.15. Given that for each replica m one has saved N_m configuration of the kind $\{x_1^m, \dots, x_k^m, \dots\}$, it can be easily shown that

$$Z_n = \frac{\sum_{m=1}^M Z_m N_m^{-1} \sum_{k=1}^{N_m} \alpha_{nm}(x_k^m) P_n(x_k^m)}{N_n^{-1} \sum_{m=1}^M \sum_{k=1}^{N_n} \alpha_{nm}(x_k^n) P_m(x_k^n)} \tag{5.24}$$

²With this definition, $V_{(\text{Slt})}(X_n)$ may also depend on the coordinate of few solvent atoms. Being the definition of the solute atom based rather than potential based, it may be necessary to include in $V_{\text{slv}}^{bd}(X_n)$, e.g., torsional terms that involve boundary solvent atoms.

Eq. 5.24 holds for any arbitrary *bridge function* $\alpha_{nm}(X)$, In particular, choosing[95]

$$\alpha_{nm}(X) = \frac{N_m Z_m^{-1}}{\sum_l^M N_l Z_l^{-1} P_l(X)} \quad (5.25)$$

Eq. 5.24 transforms as

$$Z_n = \sum_{k=1}^N \frac{P_n(x_k)}{\sum_{l=1}^M \frac{Z_l^{-1} N_l P_l(x_k)}{Z_n}} \quad (5.26)$$

where we have collapsed the two indices k , for the configurations, and m , for the replicas in one single index k *running on all the configurations* $N = \sum_m N_m$ produced by the REMD. Except for an arbitrary multiplicative factor, Eq. 5.26 can be solved iteratively for the partition function Z_n . At the beginning of the process, one sets $Z_i = 1$ for all i . . At the iteration $i + 1$ we have that

$$Z_n(i+1) = \sum_k^N W_n(x_k, i) \quad (5.27)$$

where the weights depends on the Z_l calculated at the previous iteration.

$$W_n(x_k, i) = \frac{e^{-\beta \mathbf{c}_n \cdot \mathbf{v}(x_k)}}{\sum_l N_l e^{-\ln Z_l(i) - \beta \mathbf{c}_l \cdot \mathbf{v}(x_k)}} \quad (5.28)$$

and we have used the definition of 5.15 for the replica distribution in ORAC . Once the Z_i have been determined, Z_* for an arbitrary distribution $P_*(X)$ can be calculated using the configurations sampled in the REMD simulation:

$$Z_* = \sum_k^N \frac{P_*(x_k)}{\sum_l Z_l^{-1} N_l P_l(x_k)} = \sum_k^N W_*(x_k) \quad (5.29)$$

Setting for example $P_* = P_1(X) * A(X)$, where $P_1(X)$ is the *target* distribution $e^{-\beta V(x)}/Z$ (i.e. that of the replica 1) and $A(x)$ is an arbitrary configurational property, we obtain

$$\langle A \rangle_1 = \frac{\int P_1(X) A(X) dX}{Z_1} = \frac{\int P_*(X) dX}{Z_1} = \frac{Z_*}{Z_1} \quad (5.30)$$

From Eq. 5.29, we have that $Z_* = \sum_k^N W_1(x_k) A(x_k)$ and that $Z_1 = \sum_k^N W_1(x_k)$. Substituting these results into Eq. 5.30, we obtain

$$\langle A \rangle_1 = \frac{\sum_k^N W_1(x_k) A(x_k)}{\sum_k^N W_1(x_k)} \quad (5.31)$$

where the weights W_1 for *all sampled points* in the REMD simulation are given by

$$W_1(x_k, i) = \frac{e^{-\beta V(x_k)}}{\sum_l N_l e^{-\ln Z_l(i) - \beta \mathbf{c}_l \cdot \mathbf{v}(x_k)}} \quad (5.32)$$

Summarizing, using the configurational energies from all M replicas, we first solve iteratively the system 5.27 for all Z_n (except for a multiplicative factor), with $1 \leq n \leq M$. In doing this, the weights W_n (including $n = 1$) are also determined. Finally configurational averages at, e.g., the target distribution can be determined using *all the REMD configurations* by means of Eq. 5.31.

Chapter 6

Metadynamics Simulation: history-dependent algorithms in Non-Boltzmann sampling

If we are studying a prototypical elementary reaction, in which two stable states are separated by an high free energy barrier $\Delta A^* \gg k_B T$ along the reaction coordinate s , configurations corresponding to the free energy maximum (the transition state s^*) can be sampled by adding a restraining potential to the original Hamiltonian of the system, so as to obtain a frequency histogram for the value of the reaction coordinate s centered around the transition state itself. If we were good enough in locating the transition state and matching the curvature of the potential, this distribution will overlap with the two distributions obtained starting two different simulations from the two metastable states. The free energy difference between the metastable states, as well as the height of the free energy barrier at the transition state, can then be computed using the sampling from this “bridging” distribution. This solution is known as Umbrella Sampling[44]. More generally, if the transition state can be identified and located at some value of the reaction coordinate, the procedure of modifying the energetics of a system in order to balance the activation barrier and flatten the free energy profile is known with the name of Non-Boltzmann sampling. The original free energy can be computed from the free energy of the modified ensemble through the formula

$$A(s) \sim A'(s) - V(s) \tag{6.1}$$

where $A'(s)$ denotes the free energy computed by simulating the modified ergodic system. As in the Umbrella Sampling algorithm, the hardest part of the Non-Boltzmann sampling approach is the construction of a good biasing potential, since this task can be performed only iteratively. Given a rough (because of some free energy barrier) estimate of $A(s)$ from an old simulation, the simplest way to know how good this estimate is consists in performing a new simulation using this estimate, inverted in sign, as a bias potential. If the free energy profile of the modified system is flat, $A' = \text{constant}$, then $A(s) \sim -V(s)$ is the free energy inverted in sign. Otherwise, from this simulation we can compute an improved estimate for $A(s)$ through Eq. 6.1. The effectiveness of this tedious approach is due to the fact that each correction to the biasing potential makes the system more ergodic, and therefore each successive simulation is statistically more accurate than the former.

This iterative approach to the problem[96, 97] led to the development of adaptive biasing potential methods that improve the potential “on the fly” [98, 47, 45, 99], *i.e.*, while the simulation is performed. All these methods share all the common basic idea, namely, “to introduce the concept of memory”[98] during a simulation by changing the potential of mean force perceived by the system, in order to penalize conformations that have been already sampled before. The potential becomes history-dependent since it is now a functional of the past trajectory along the reaction coordinate. Among these algorithms, the Wang-Landau [47] and the metadynamics[45] algorithms have received most attention in the fields of the Monte Carlo (MC) and Molecular Dynamics (MD) simulations, respectively. This success is mainly due to the clearness and the ease of implementation of the algorithm, that is basically the same for the two methods. The Wang-Landau algorithm was initially proposed as a method to compute the density

of states $g(E)$, and therefore the entropy $S(E) = \ln g(E)$, of a simulated discrete system. During a Wang-Landau MC simulation, $S(E)$ is estimated as an histogram, incrementing by a fixed quantity the frequency of the visited energy levels, while moves are generated randomly and accepted with a Metropolis probability $\text{acc}(E \rightarrow E') = \min\{1, \exp(-\Delta S)\}$, where $\Delta S = S(E') - S(E)$ is the current estimate of the entropy change after the move. While for a random walk in energy the system would have been trapped in entropy maxima, the algorithm, that can be easily extended to the computation of any entropy-related thermodynamic potential along a generic collective variable, helps the system in escaping from these maxima and reconstructs the entropy $S(E)$. The metadynamics algorithm extends this approach to off-lattice systems and to Molecular Dynamics. Metadynamics has been successfully applied in the computation of free energy profiles in disparate fields, ranging from chemical physics to biophysics and material sciences. For a system in the canonical ensemble, metadynamics reconstructs the free energy along some reaction coordinate s as a sum of Gaussian functions deposited along the trajectory of the system. This sum inverted in sign is used during the simulation as a biasing potential $V(s, t)$ that depends explicitly on time s :

$$V(s, t) = \sum_{t'=\tau, 2\tau, \dots, t} \mathcal{G}(s; s_{t'}, h, \sigma) \quad (6.2)$$

where $\mathcal{G}(s; s_t, h, \sigma) = h \exp(-(s - s_t)^2/2\sigma^2)$ is a Gaussian function centered in s_t with height h and variance σ^2 . During a metadynamics simulation, the potential $V(s, t)$ will grow faster for states with an higher probability, pushing out the system from minima in the free energy landscape. If the rate of deposition $\omega = h/\tau$ is sufficiently slow, the system can be considered in equilibrium with the biased Hamiltonian $H'(x, t) = H(x) + V(s, t)$, and therefore the probability of visiting state s at time t is the equilibrium canonical distribution $p(s, t) \propto \exp[-\beta(A(s) + V(s, t))]$. Once all the free energy minima have been “filled” by the biasing potential, and therefore $V(s, t) = -A(s)$, such a probability is uniform along s and the potential will grow uniformly.

The thermodynamical work spent in changing the potential from the original Hamiltonian $H(x)$ to $H'(x, t) = H(x) + V(s, t)$ can be computed through the relation $W = \int_0^t d\tau \left(\frac{\partial H}{\partial t} \right)_\tau$. In the limit of an adiabatic transformation, this quantity is equal to the Helmholtz free energy difference $\Delta A = A' - A_0$ between two systems with energy functions H' and H , where $A' = \int dx \exp(-\beta H')$ and $A_0 = \int dx \exp(-\beta H)$ [100]. However, if the process is too fast with respect to the ergodic time scale, a part of the work spent during the switching will be dissipated in the system, resulting in an non-equilibrium, non-canonical distribution, and in a systematic error in the free energy estimate. In particular, it is assumed that during a metadynamics simulation all the microscopic variables different from the macroscopic reaction coordinate s are always in the equilibrium state corresponding to the value of s [101]. This property is known with the name of Markov property, and it summarizes the main assumption of the algorithm: all the slow modes of the system coupled to the reaction under study have to be known *a priori* and they have to be included in the number of the reaction coordinates. Therefore, at variance with the methods presented in the previous chapters, metadynamics should be considered a quasi-equilibrium method, in which the knowledge about the variables that capture the mechanism of a reaction is exploited to gain insight on the transition states and more generally to compute the free energy landscape along the relevant reaction coordinates.

6.1 Implementation in ORAC

From the practical point of view, a metadynamics simulation consists in two steps. In the first one, a set of reaction coordinates is chosen whose dynamics describes the process under study. As we said, such a procedure requires an high degree of chemical and physical intuition for its application to complex molecular system, since these variables are not obviously determined from a molecular structure.

The second step is the metadynamics simulation itself, during which an history-dependent potential is constructed by summing, at regular time intervals, repulsive potential terms centered in the current position of the system in the space of the reaction coordinates. In its standard implementation, the history-dependent potential of metadynamics is given by a sum of small repulsive Gaussian, Eq.6.2. Some variants have been introduced, with the intent of improving the accuracy or the efficiency of the method[102, 103]. In the ORAC program we have used Lucy’s function[104] as a very efficient alternative to the use of Gaussians.

It is defined as ¹

$$\mathcal{L}(s; s_0, h, w) = h \left(1 + 2 \frac{|s - s_0|}{w} \right) \left(1 - \frac{|s - s_0|}{w} \right)^2; 0 \text{ if } |s - s_0| > w \quad (6.3)$$

with the origin at s_0 . The symbols h and w denote the height and the width, respectively. Such a function is normalizable, $\int_{-\infty}^{\infty} ds \mathcal{L}(s; s_0, w) = hw$, has a finite range w , has a maximum at the origin and it is differentiable everywhere. A Lucy's function can be compared with a Gaussian function with the same value at the origin and at $|s| = s_0 + w/2$, such that

$$2\sigma = w/(2 \ln 2)^{1/2} \quad (6.4)$$

A Lucy's function can be regarded as a Gaussian function with σ in Eq.6.4, but without the long tails of the Gaussian, as can be seen in Fig.6.1 where a Lucy's function with $h = w = 1$ and a Gaussian function with the same height and $\sigma = w/2(2 \ln 2)^{1/2}$ are shown. The parameters h , w and τ affects the accuracy of the free energy reconstruction in a similar manner to the height and the width of Gaussian functions and a comprehensive review on the analysis of the error during a metadynamics run can be found in [49].

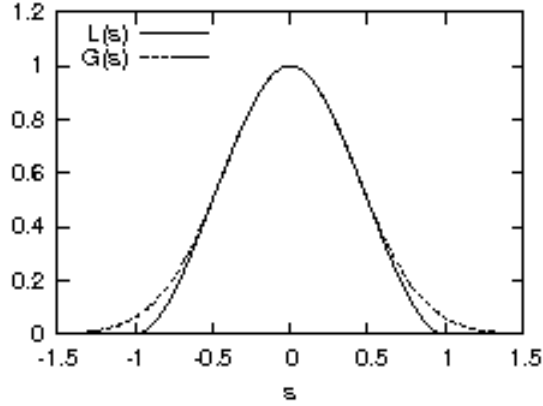


Figure 6.1: Lucy's function L with $h = w = 1$, along with a Gaussian function G with the same height and $2\sigma = w/(2 \ln 2)^{1/2}$.

The history dependent potential used during an ORAC simulation can therefore be written as

$$V(s, t) = \sum_{t'=\tau, 2\tau, \dots} \mathcal{L}(s; s_{t'}, h, w) \quad (6.5)$$

During a simulation, forces from this biasing potential are computed in the shell $n1$ as a sum of derivatives of \mathcal{L} functions:

$$\frac{\partial \mathcal{L}(s; s_0, h, w)}{\partial s} = \frac{6h}{w^3} (s - s_0) (|s - s_0| - w); 0 \text{ if } |s - s_0| > w \quad (6.6)$$

Such a derivative is computationally attractive, since it does not require the evaluation of an exponential function as in the case of the derivative of a Gaussian function. Moreover, since \mathcal{L} has a finite range by

¹ Lucy's function can be defined for a generic order n such that it has $n - 1$ continuous derivative everywhere[105]. The original definition[104] was given for $n = 3$; here it is employed with $n = 2$.

definition, it does not need to be smoothly truncated[103], as there are no contribution to the forces from hills farther than the width w .

Using the standard metadynamics approach, during a simulation the algorithm keeps on adding terms to the history-dependent potential (the sum in Eq.6.5) with the same constant rate $\omega = h/\tau$. However, the optimal solution would be to use a faster rate at the beginning of the simulation, so as to produce a rough estimate of the free energy, and then to reduce ω to refine this estimate[106]. This problem corresponds to finding an optimal protocol for the evolution of the modification factor in the original Wang-Landau algorithm. Various solutions have been proposed[107, 99, 108, 109] in which the energy h in 6.3 is time-dependent. We propose instead to add a term to the biasing potential with a given probability $P_t(\text{add})$, depending parametrically on time. For example, for $P_t(\text{add}) \propto 1/t$, the evolution of the rate would be given by $\omega(t) = P_t(\text{add})\omega_0 \propto \omega_0/t$. This procedure can be seen on average as an increasing deposition interval $\tau(t)$, such that $\omega(t) = h/\tau(t)$ decreases in time. In the present implementation of ORAC, three different choices are available for the probability $P(\text{add})$: the default one is simply $P(\text{add}) = 1$ and corresponds to the standard metadynamics algorithm. The second one is given by

$$P_t(\text{add}) = e^{-V_{\max}(t)/k_B T'} \quad (6.7)$$

where $V_{\max}(t)$ is the maximum value of the potential $V(s, t)$ at time t . During the simulation, the effective rate $\omega(t)$ decreases as $V_{\max}(t)$ increases. As $V_{\max} \gg k_B T'$, the deposition rate $\omega(t)$ is so slow that the transformation can be considered adiabatic, and the biasing potential converges to the free energy inverted in sign, $A(s) = -V(s, t)$. The slowdown of ω can be tuned through the parameter T' . Finally, following the well-tempered metadynamics approach[109], the third choice is given by

$$P_{s,t}(\text{add}) = e^{-V(s,t)/k_B T'} \quad (6.8)$$

where the probability depends parametrically both on time t and on position s of the system along the reaction coordinate through the biasing potential V . In this case, the biasing potential does not converge to the free energy inverted in sign as in the previous case, since in general ω turns out to be coordinate-dependent even when the potential has flattened the free energy profile. However, as shown in [109], the relation

$$A(s) = -\frac{T + T'}{T} V(s, t) \quad (6.9)$$

can be used to recover the original free energy from the biasing potential.

The multiple walkers version of metadynamics algorithm[110] was implemented in the parallel version of the code through the MPI library. This approach is based on running simultaneously multiple replicas of the system, contributing equally to the same history-dependent potential, and therefore to the same free energy surface reconstruction. For N replicas, $V(s, t)$ can be written as a double sum

$$V(s, t) = \sum_{t'=\tau, 2\tau, \dots, t} \sum_{i=1, N} \mathcal{L}(s; s_{i,t'}, h, \sigma) \quad (6.10)$$

where $s_{i,t'}$ is the position at time t' of the i -th replica along s . In particular, the enhanced efficiency of this algorithm with respect to uncoupled simulations contributes to make the calculation of FESs in high dimensions more accessible.

In the ORAC distribution at <http://www.chim.unifi.it/orac> we provide some example of metadynamics simulations using Lucy's functions on multi-dimensional surfaces of simple molecules in the gas phase along with some ancillary codes for the analysis of the program output.

Chapter 7

Steered Molecular Dynamics

Steered molecular dynamics simulation (SMD) is a technique mimicking the principle of the atomic force microscopy (AFM). In practice, one applies a time dependent mechanical external potential that obliges the system to perform some prescribed motion in a prescribed simulation time. SMD has been widely used to explore the mechanical functions of biomolecules such as ligand receptor binding/unbinding and elasticity of muscle proteins during stretching at the atomic level[111]. The SMD has also been used in the past to approximately estimate the potential of mean force (PMF)¹ along a given mechanical coordinate (for example a distance or an angle). The model upon which this technique for estimating the PMF relies was based on the assumption that the driven motion along the reaction coordinate z could be described by an over-damped one-dimensional Langevin equation of the kind

$$\gamma\dot{z} = -\frac{dW}{dz} + F_{\text{ext}}(z, t) + \xi(t) \quad (7.1)$$

where γ is the friction coefficient, W is the underlying potential of mean force, $F_{\text{ext}}(z, t)$ is the external force due the driving potential and $\xi(t)$ is a stochastic force related to the friction through the second fluctuation dissipation theorem. The PMF $W(z)$ can then be determined only if one knows (or can somehow figure it out) the friction coefficient, so as to evaluate the frictional force that discounts the irreversible work done in the driven process. The method also relies on the strong assumption that the friction along z is local in time, i.e. the underlying equilibrium process is *Markovian*.

7.1 The Crooks theorem

Recent development on non equilibrium thermodynamics have clarified that the PMF along the given reaction coordinate z can actually be reconstructed *exactly* using an ensemble of steered molecular dynamics simulations without resorting to any assumption on, or having any knowledge of the frictional behaviour of the system along the reaction coordinate. These developments date back to a paper by Evans, Searls[112] where the first example of *transient fluctuation theorem* for a system driven out of equilibrium was formulated, demonstrating the connection between the time integral of the phase compression factor in Liouville space along an arbitrary time interval and the probability ratio of producing the entropy A and $-A$ along a deterministic trajectory of a many particles non equilibrium steady state system. Gavin Crooks in his *phd* thesis proposed[51], in the context of Monte Carlo simulations in the canonical ensemble (NVT), a *transient*[112] fluctuation formula (from now on indicated with CT) involving the dissipative work for systems driven out of equilibrium by varying some arbitrary mechanical parameter. The CT is actually even more general than the Evans and Searls fluctuation theorem[112] since in the latter the driven z coordinate has an underlying zero PMF (i.e. only entropy is produced in the non equilibrium process) while in the former the system can also cross different thermodynamics states (i.e. the underlying PMF can also be non zero such that thermodynamic work can also be done). The Crooks theorem (CT) reads

$$\frac{p(\Gamma(z_0) \rightarrow \Gamma(z_\tau))}{p(\Gamma^*(z_0) \leftarrow \Gamma^*(z_\tau))} = \exp[\beta(W_{\Gamma(z_0) \rightarrow \Gamma(z_\tau)} - \Delta F)], \quad (7.2)$$

¹The potential of mean force is defined as $W(z) = -k_B T \ln P(z)$, where $P(z) = \langle \delta(z - z(\mathbf{r})) \rangle$ is the probability to find the system at the value of the reaction coordinate $z(\mathbf{r}) = z$ independently on all the other coordinates.

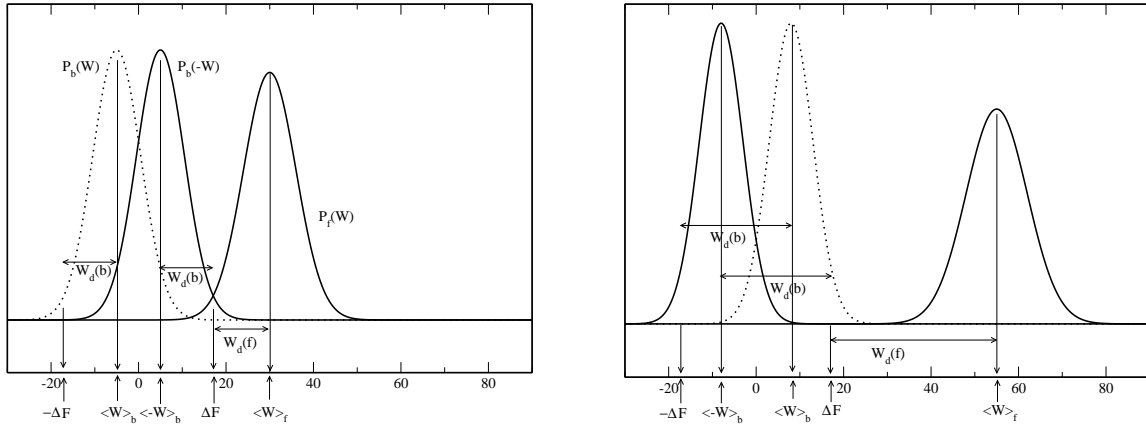


Figure 7.1: Physical significance of the Crooks theorem for a general driven process: for nearly reversible processes (left) the forward $P_f(W)$ and backward $P_b(-W)$ work distributions overlap significantly. The dotted line is the the backward work distribution for the inverse process, without changing the sign of the work. The crossing of the two solid distribution occurs at the free energy value for the *forward* process $\Delta F = 18$. When the process is done faster (right panel), the dissipation W_d both in the forward and in the backward process is larger, the overlap is negligible and the crossing point of the two solid distribution can no longer easily identified.

where τ is the duration time of the driven non equilibrium process, $W_{\Gamma(z_0) \rightarrow \Gamma(z_\tau)}$ is the work done on the system during the driven trajectory $\Gamma(z_0) \rightarrow \Gamma(z_\tau)$; $p(\Gamma(z_0) \rightarrow \Gamma(z_\tau))$ is the joint probability of taking the microstate $\Gamma(z_0)$ from a canonical distribution with a given initial Hamiltonian $H(z = z_0)$ and of performing the forward transformation to the microstate $\Gamma(z_\tau)$ corresponding to a different Hamiltonian $H(z = z_\tau)$; $p(\Gamma^*(z_0) \leftarrow \Gamma^*(z_\tau))$ is the analogous joint probability for the *time reversal* path, producing the work $W_{\Gamma(z_\tau) \rightarrow \Gamma(z_0)} = -W_{\Gamma(z_0) \rightarrow \Gamma(z_\tau)}$. $\Delta F = F(z = z_\tau) - F(z = z_0)$ is the free energy difference between the thermodynamic states associated to the Hamiltonians $H(z = z_\tau)$ and $H(z = z_0)$. Although the CT can be stated in a more general formulation (see Gavin Crooks, *phd* thesis), here the essential assumptions are that i) the system is deterministic and satisfies the time reversal symmetry and ii) the reverse trajectory is done following a *reversed* time schedule such that $W_{\Gamma(z_\tau) \rightarrow \Gamma(z_0)} = -W_{\Gamma(z_0) \rightarrow \Gamma(z_\tau)}$. The first assumption is satisfied by any kind of standard MD equation of motion (Newtonian, Nosé-Hoover, Parrinello-Rahman) while the second condition can be easily imposed in a SMD experiment. A very simple proof of Eq. 7.2 goes as follows: suppose the z_0 is drawn from a canonical distribution, and that the driven trajectory that brings the system to z_τ is done adiabatically, i.e. removing the thermal bath. For the reverse trajectory, drawing z_τ from a canonical distribution, due to the time reversal symmetry of the Hamilton equations, one ends up adiabatically in z_0 . Under these assumptions, the ratio of the two probabilities on the left hand side of Eq. 7.2 can be written as

$$\begin{aligned}
 \frac{p(\Gamma(z_0) \rightarrow \Gamma(z_\tau))}{p(\Gamma^*(z_0) \leftarrow \Gamma^*(z_\tau))} &= \frac{e^{-\beta H(z=z_0)}}{Z_0} \frac{Z_\tau}{e^{-\beta H(z=z_\tau)}} \\
 &= e^{\beta(H(z=z_\tau) - H(z=z_0) - \Delta F)} \\
 &= \exp \beta(W_{\Gamma(z_0) \rightarrow \Gamma(z_\tau)} - \Delta F)
 \end{aligned} \tag{7.3}$$

equation where we have used the facts that $\beta F(z = z_0) = \ln Z_0$, $\beta F(z = z_\tau) = \ln Z_\tau$ and that the energy difference $H_B - H_Z$ in the forward adiabatic trajectory equals to the external work done on the systems. Equation 7.2 refers to the probability of a *single* forward or backward trajectory. Suppose now to perform a large number of forward trajectories all with a give time schedule, but each started from a different initial phase point sampled according to the canonical equilibrium distribution characterized by the Hamiltonian $H(z = z_0)$ and a large and not necessarily equal number of backward trajectories with reverse time schedule

and starting from initial phase points this time sampled according to the canonical equilibrium distribution characterized by the Hamiltonian $H(z = z_t)$.² By collecting all trajectories yielding the work W in (7.2), the CT may compactly be written as:

$$\frac{P_F(W)}{P_R(-W)} = \exp[\beta(W - \Delta F)], \quad (7.4)$$

where $P_F(W)$ and $P_R(W)$ are the normalized forward and backward distribution functions (note that, due to the time reversal symmetry, for the backward distribution the work is taken with the minus sign, i.e. $P_R(-W)$ is the mirror symmetric with respect to $P_R(W)$). According to Eq. 7.4, the ΔF may be thus evaluated constructing the two work distribution function: ΔF is the work value where the two distribution cross, i.e. $P_F(W = \Delta) = P_R(-W = \Delta F)$. We point out in passing that, the famous Jarzynski identity[50] (JI),

$$\langle e^{-\beta W} \rangle = e^{-\beta \Delta F}, \quad (7.5)$$

is actually a trivial consequence of the CT, being derived from the latter by integrating out the work variable and using the fact that the work distribution function $P_F(W)$ and $P_R(-W)$ are normalized.

The physical meaning of the Crooks equation sounds indeed very reasonable and can be even be considered as a probabilistic restatement of the second law or of a generalization of the H-Boltzmann theorem: Given a forward *deterministic* non equilibrium trajectory starting from equilibrium and producing a work W , the probability to observe a trajectory for the reverse process again starting from equilibrium and producing the work $-W$ is $e^{\beta W_d}$ small than the former, where $W_d = W - \Delta F$ is the dissipated work in the forward process. When the dissipated work is zero, i.e. when the driven process is *quasi-static* and is done always at equilibrium, then the two probabilities are identical. With this regard, one important point to stress is that the CT and the JI hold *for all systems* and for any kind of arbitrary non equilibrium process, *no matter how fast is performed*. In particular, if the non equilibrium process is *instantaneous*, i.e. if it is done at infinite speed, then the work done on the system is simply equal to $W = (H_1 - H_0)$, with H_0 and H_1 being the Hamiltonian of the initial and final state, respectively. The JI reduces in this case to the famous free energy perturbation Zwanzig[115] formula $\langle e^{-\beta(H_1 - H_0)} \rangle_0 = e^{-\beta \Delta F}$ with the subscript 0 indicating that the canonical average must be taken according to the equilibrium distribution of the system with Hamiltonian H_0 .

For fast non equilibrium experiments, a large amount of the work, rather than in advancing the reaction coordinate, is dissipated in heat that is in turn (only partly) assimilated by the thermal bath³ A consequence of this is that the maxima of two work distributions $P_F(W)$ and $P_R(W)$ tend to get further apart from each other so that the determination of ΔF becomes less accurate. The faster are performed the non equilibrium experiments, the large is the average dissipation and the smaller is the overlap between the two work distributions (see Fig. 7.1) The reason why CT and JI can be so useful in evaluating the free energies along given reaction paths in the molecular dynamics simulation of complex biological system lies on the fact that this methodologies are inherently more accurate the smaller is the sample. Let's see why. As one can see from Fig. 7.1, ΔF can be determined with accuracy if the two work distributions overlap appreciably, or stated in other terms, if there are sufficient trajectories that in both directions transiently violate the second law, i.e trajectories for which $W < \Delta F$. This is clearly not in contrast with the second law which states that $\bar{W} \leq \Delta F$ where $\bar{W} = \int P(W)WdW$ is the *mean* irreversible work. In general, the probability of an overlap of the two work distributions (i.e. the probability of transiently violating the second law) is clearly larger the smaller is the system. Suppose to simultaneously and irreversibly unfold N identical proteins in a dilute solution starting from their native states. In the assumption that the

² The Hamiltonian $H(z = z_t)$ may be imposed practically in steered molecular dynamics using constraints or adding a stiff harmonic potential that keeps the system at $z = z_t$. Both these methods requires small corrections when reconstructing the PMF. In particular, the use of constraints on z sets also $\dot{z} = 0$, a condition that is not present in the definition of the PMF (see previous footnote). The correction to the PMF due this extra artificial condition imposed through a generic constraint is discussed in Ref. [113]. Stiff harmonic potentials, in the sense that the associated stretching motion is decoupled from the degrees of freedom of the system, behaves essentially like constraints.[114] The depuration of the the PMF from the non stiff harmonic driving potential in AFM experiments has been proposed by Hummer and Szabo.[52]

³ During the non equilibrium experiment, the instantaneous "temperature" of the system as measured by the kinetic energy may well exceed that of the thermal bath. Actually the "temperature" cannot even be defined for a system that is not at equilibrium as part of it, near the reaction path, can be warmer than other parts that are far from the reaction coordinate. This has clearly no consequences whatsoever on the CT, since the temperature in Eq. 7.2 that of the system at the initial points which are drawn by hypothesis at equilibrium

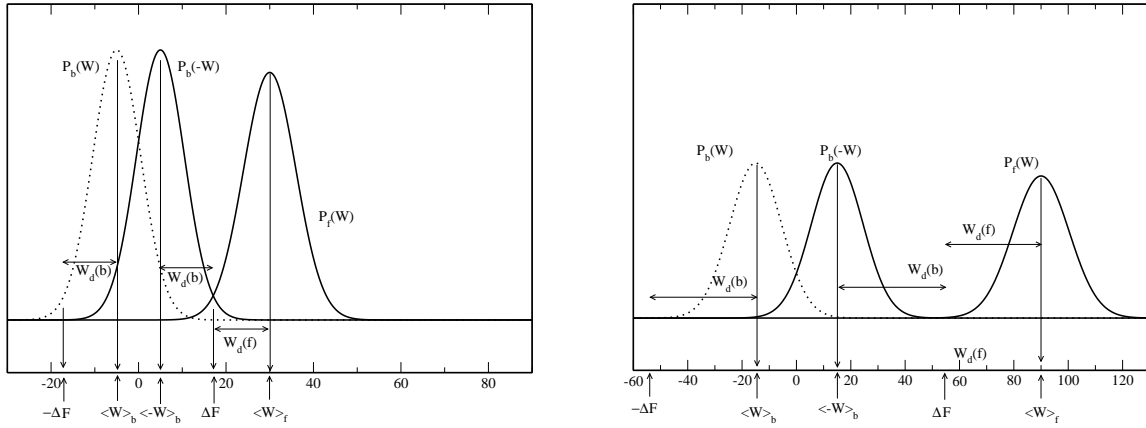


Figure 7.2: Effect of the size of the system on the overlap of the forward and backward work distributions. In the left panel the non equilibrium processes are done in a given time τ on a single molecule. In the right panel the processes, as in left panel of duration τ , are done independently on three identical molecules. This implies a factor 3 on energies and a factor $3^{1/2}$ on widths. As a result of the increased size, the overlap between $P_f(W)$ and $P_b(-W)$ decreases significantly.

intraprotein interaction are negligible, the mean work for this system will be simply N times the mean work done on a *single* molecule, while the width of the work distribution for the N molecule systems will be only $N^{1/2}$ larger than that of the single molecule system. This effect is illustrated in Fig. 7.2. Now, biomolecular simulation of biosystems are usually done, for computational reasons, on a *single* solvated biomolecule, i.e. in the conditions where the non equilibrium techniques, for the reason explained above, are deemed to be more successful.

7.2 Determination of the potential of mean force via bidirectional non equilibrium techniques

The Jarzynski identity is seemingly a better route than the CT to evaluate the full potential of mean force along $F(z)$ in the interval $[z_0, z_t]$, with $0 < t < \tau$. However the exponential averages in Eq. 7.5 is known to be strongly biased, i.e. it contains a systematic error[116] that grows with decreasing number of non equilibrium experiments. This can be qualitatively explained with the fact that, for dissipative fast non equilibrium experiments, the forward work distribution $P(W)$ has its maximum where the exponential factor $e^{-\beta W}$ is negligibly small, so that the size of the integrand $P(W)e^{-\beta W}$ is *de facto* controlled by the left tail of the $P(W)$ distribution.[52] An unfortunate consequence of this, is that the PMF calculated through the JI becomes more and more biased as the reaction z coordinate is advanced, since the accumulated dissipation work shift the maximum of the $P(W)$ distribution

The CT is far more precise than the JI to evaluate free energy differences. Shirts and Pande[53] have restated the CT theorem showing that the maximum likelihood estimate (MLE) of the free energy difference exactly correspond to the so-called Bennett acceptance ratio[117]⁴. The MLE restatement of the CT is the following

$$\sum_{i=1}^{n_F} \frac{1}{1 + \frac{n_F}{n_R} e^{\beta(W[\mathbf{F}_i] - \Delta F)}} - \sum_{i=1}^{n_R} \frac{1}{1 + \frac{n_R}{n_F} e^{\beta(W[\mathbf{R}_i] + \Delta F)}} \quad (7.6)$$

⁴Bennett was the first researcher to clearly recognize and formalize through the BAR the superiority of bidirectional methods in the computation of free energy differences. We cite *verbatim* from his paper[117]: “The best estimate of the free energy difference is usually obtained by dividing the available computer time approximately equally between the two ensembles; its efficiency (variance x computer time) is never less, and may be several orders of magnitude greater, than that obtained by sampling only one ensemble, as is done in perturbation theory.”

where the n_F , n_R are the number of forward and backward non equilibrium experiments and $W[\mathbf{F}_i]$ $W[\mathbf{R}_i]$ indicate the outcome of i -th forward and backward work measurement. This equation has only one solution for ΔF , i.e. the MLE. As such, however, the Crooks theorem allows, through the MLE estimate based on bidirectional work measurements, to compute the free energy difference ΔF between the *end points* (i.e. between thermodynamic states at fixed and given reaction coordinates $z = z_0$ and $z = z_t$). In principle, to reconstruct the full PMF along the reaction coordinate z , in the spirit of thermodynamics integration, One should provide a series of equilibrium ensembles of configurations at *intermediate* values of z_t . Here, we briefly sketch out a methodology for reconstructing the full PMF in the segment $[z_0, z_\tau]$ doing only the two work measurements from z_0 to z_τ and back. We first rewrite the Crooks equation, Eq. 7.2, as follows

$$\rho_F(\Gamma) = \rho_R(\hat{\Gamma})e^{\beta(W-\Delta F)}, \quad (7.7)$$

where ρ_F , ρ_b are the probability to observe a particular trajectory Γ in the forward and reverse process, respectively and $\hat{\Gamma}$ indicate the time trajectory taken with inverted time schedule. Eq. 7.7 trivially implies that

$$\langle \mathcal{F} \rangle_F = \langle \hat{\mathcal{F}} e^{\beta(W-\Delta F)} \rangle_R \quad (7.8)$$

$$\langle \mathcal{F} \rangle_R = \langle \hat{\mathcal{F}} e^{-\beta(W-\Delta F)} \rangle_F \quad (7.9)$$

where $\mathcal{F} = \mathcal{F}(\Gamma)$, $\hat{\mathcal{F}} = \mathcal{F}(\hat{\Gamma})$ is an arbitrary functional of the trajectory Γ and of its inverted time schedule counterpart $\hat{\Gamma}$. Using Eq. 7.7, we thus can combine the direct estimate of $\rho_F(\Gamma)$ with the indirect estimate of the same quantity obtained from $\rho_R(\hat{\Gamma})$. This latter, according to Eq. 7.7, must be unbiased with the weight factor corresponding to the exponential of the dissipated work in the forward measurement. If the direct and indirect (Eq. 7.7) estimates are done with n_F forward measurements and n_R reverse measurements, respectively, the optimal (minimum variance) combination of these two estimates of $\rho_F(\Gamma)$ is done according to the WHAM formula[118]

$$\rho_F(\Gamma) = \frac{n_F \rho_F(\Gamma) + n_R \rho_R(\hat{\Gamma})}{n_F + n_R e^{-\beta(W-\Delta F)}}. \quad (7.10)$$

Here W is the work done in the full Γ path from the end point at $t = 0$ to the end point at $t = \tau$. We now calculate the average of the trajectory functional $e^{-\beta W_0^t}$ at intermediate times $0 < t < \tau$, using the optimized above density. Taking the average of this functional over forward (Γ) and reverse ($\hat{\Gamma}$) work measurements, exploiting the Jarzynski identity 7.5 in the form $\langle e^{-\beta W_0^t} \rangle = e^{-\beta(F(z=z_t)-F(z=z_0))}$, using the fact that W is odd under time reversal and that $W_0^t[\hat{\Gamma}] = -W_{(\tau-t)}^\tau[\Gamma]$, we obtain the following estimate for the free energy at intermediate t , with $0 < t < \tau$:

$$e^{-\beta(F_t-F_0)} = \left\langle \frac{n_F e^{-\beta W_0^t}}{n_F + n_R e^{-\beta(W-\Delta F)}} \right\rangle_F + \left\langle \frac{n_R e^{\beta W_{(\tau-t)}^\tau}}{n_F + n_R e^{\beta(W+\Delta F)}} \right\rangle_R \quad (7.11)$$

This equation, due to Minh and Adib[119], allows to reconstruct the entire potential of mean force $F_t - F_0$ along the reaction coordinate spanned during the bidirectional non equilibrium experiments of duration τ , *no matter how fast the driven processes are done*. Note that $\Delta W = F_\tau - F_0$ and W in Eq. 7.11 are the forward free energy difference and work *relative the end points*, respectively.

For fast pulling experiments, i.e. when the dissipated work is large, it can be shown[120], that Eq. 7.11 reduce to

$$e^{-\beta(F_t-F_0)} = \langle e^{-\beta W_0^t} \rangle_F + e^{-\beta \Delta F} \langle e^{-\beta W_\tau^{(t)}} \rangle_R \quad (7.12)$$

In both Eq. 7.12 and Eq. 7.11 one need to know the free energy difference between the end points ΔF . An unbiased estimate of ΔF is easily available through the Bennett acceptance ratio, Eq. 7.6.

7.3 Implementation in ORAC

Steered molecular dynamics in ORAC is implemented by adding an external driving potential depending on user defined internal coordinates in the form of stretching, bending, torsions. The general form of the

time dependent external potential that bring the system from an initial state at $t = 0$ to a different final state $t = \tau$ is given by

$$V_{ext}(t) = \frac{1}{2} \left[\sum_{i=1}^{N_r} K_i (r_i - r_{i0}(t))^2 + \sum_{i=1}^{N_\alpha} K_i (\alpha_i - \alpha_{i0}(t))^2 + \sum_{i=1}^{N_\theta} K_i (\theta_i - \theta_{i0}(t))^2 \right] \quad (7.13)$$

where r_i , α_i and θ_i represents the actual i -th stretching, bending and torsional driven coordinate defined by arbitrarily selecting in the corresponding input definition the involved atoms. So a driven torsion or a stretching may be defined using arbitrarily chosen atoms of the solute that are not connected by any real bond. $r_{i0}(t)$, $\alpha_{i0}(t)$ and $\theta_{i0}(t)$ are time dependent parameters that defines the non equilibrium trajectory in the space of the coordinates. In ORAC, each of these parameters, given the duration τ of the non equilibrium experiment, is varied *at constant speed* from an initial value at time $t = 0$ defining the reactants, to a final value at time $t = \tau$ defining the products :

$$\begin{aligned} r_i(t) &= r_{i0} + \frac{r_{i\tau} - r_{i0}}{\tau} t = r_{i0} + v_{ir} t \\ \alpha_i(t) &= \alpha_{i0} + \frac{\alpha_{i\tau} - \alpha_{i0}}{\tau} t = \alpha_{i0} + v_{i\alpha} t \\ \theta_i(t) &= \theta_{i0} + \frac{\theta_{i\tau} - \theta_{i0}}{\tau} t = \theta_{i0} + v_{i\theta} t \end{aligned} \quad (7.14)$$

As all the steering velocities are constant during the experiments, the above equations define a *line*

$$\mathbf{z}(t) = \{(r_1(t), r_2(t), \dots, \alpha_{1(t)}, \alpha_2(t), \dots, \theta_1(t), \theta_2(t), \dots)\} \quad (7.15)$$

in a reaction coordinate space at $N_r + N_\alpha + N_\theta$ dimensions

The work done by the external potential, Eq. 7.13, in the time τ of the non equilibrium driven process along the coordinate \mathbf{z} is calculated as

$$W_0^\tau = \int_0^\tau \left[\sum_{i=1}^{N_r} K_i (r_i - r_{i0}(t)) v_{ir} + \sum_{i=1}^{N_\alpha} K_i (\alpha_i - \alpha_{i0}(t)) v_{i\alpha} + \sum_{i=1}^{N_\theta} K_i (\theta_i - \theta_{i0}(t)) v_{i\theta} \right] dt \quad (7.16)$$

The equilibrium distribution of the starting points for independent work measurements can be determined (either by a standard equilibrium molecular dynamics simulation or by some enhanced simulation technique) by constraining the system with the harmonic constraint

$$V_{ext}(0) = \frac{1}{2} \left[\sum_{i=1}^{N_r} K_i (r_i - r_{i0})^2 + \sum_{i=1}^{N_\alpha} K_i (\alpha_i - \alpha_{i0})^2 + \sum_{i=1}^{N_\theta} K_i (\theta_i - \theta_{i0})^2 \right] \quad (7.17)$$

for the reactants' state and

$$V_{ext}(\tau) = \frac{1}{2} \left[\sum_{i=1}^{N_r} K_i (r_i - r_{i\tau})^2 + \sum_{i=1}^{N_\alpha} K_i (\alpha_i - \alpha_{i\tau})^2 + \sum_{i=1}^{N_\theta} K_i (\theta_i - \theta_{i\tau})^2 \right] \quad (7.18)$$

for the products' state. Having produced the work in a series of bidirectional experiments, one can then either apply the Bennett formula, Eq. 7.6, to compute the free energy differences between the reactants and the products states, or, using the intermediate work values W_0^t , apply Eq. 7.11 or Eq. 7.12 to reconstruct the entire potential of mean force along the the mono-dimensional driven trajectory in a multidimensional reaction coordinate space defined in Eq. 7.14. In order to define a non necessarily linear trajectory in a multidimensional reaction coordinate space (e.g. a putative minimum free energy path), one must be able to assign to a each steered coordinate a different steering time protocol. This can be done in ORAC by providing an auxiliary file defining the path in coordinate space. The file has the general form shown in Table 7.3. The free energy or potential of mean force obtained with the described protocols are not depurated by the jacobian terms arising from the definition of the reaction coordinates. For example, the potential of mean force, calculated with Eq. 7.11 or Eq. 7.12 along a driven *distance* for a freely rotating object includes the *additional* contribution $J(t) = 2k_b T \ln(r_t/r_0)$ arising from the fact that the

t_1	$r_1(t_1)$...	$r_{N_r}(t_1)$	$\alpha_1(t_1)$...	$\alpha_{N_\alpha}(t_1)$	$\theta_1(t_1)$...	$\theta_{N_\alpha}(t_1)$
t_2	$r_1(t_2)$...	$r_{N_r}(t_2)$	$\alpha_1(t_2)$...	$\alpha_{N_\alpha}(t_2)$	$\theta_1(t_2)$...	$\theta_{N_\alpha}(t_2)$
						
t_n	$r_1(t_n)$...	$r_{N_r}(t_n)$	$\alpha_1(t_n)$...	$\alpha_{N_\alpha}(t_n)$	$\theta_1(t_n)$...	$\theta_{N_\alpha}(t_n)$

Table 7.1: General format of the file defining of an arbitrary time protocol for a curvilinear path in a reaction coordinates space at $N_r + N_\alpha + N_\theta$ dimensions in ORAC . For a generic coordinate $\zeta = r, \alpha, \theta$, the steering velocity between times t_k and t_{k+1} is constant and equal to $v_\zeta(t_k) = (\zeta(t_{k+1}) - \zeta(t_k))/(t_{k+1} - t_k)$

configurational probability $P(r)$, for two non interacting particles grows with the square of the distance. Moreover the PMF calculated using the driving potential given in Eq. 7.13 are in principle affected by the so-called stiff spring approximation,[114] i.e. if the constant K_r, K_α, K_θ in Eq. 7.13 are not large enough, then one actually computes the free energy associated to the Hamiltonian $\mathcal{H} = H + V_{ext}(\mathbf{z} - \mathbf{z}_t)$ rather than that associated to the Hamiltonian $H(\mathbf{z} = \mathbf{z}_t)$. However the impact of the strength of the force constant on the computed non equilibrium average, especially if the reaction coordinate is characterized by inherently slow dynamics and/or the underlying unbiased potential of mean force is much less stiffer than the harmonic driving potential, is generally rather small even at relatively low values of force constant. With this respect, it has been shown that[114]

$$\phi(\mathbf{z}) = F(\mathbf{z}) + \frac{1}{2k}F'(\mathbf{z}) - \frac{1}{2\beta k}F''(\mathbf{z}) + O(1/k^2) \quad (7.19)$$

where $\phi(\mathbf{z})$ is PMF of the unbiased system with the Hamiltonian $H(\mathbf{z})$, while $F(\mathbf{z})$ is the PMF that is actually measured in the SMD experiments, i.e. that corresponding to the *biased* Hamiltonian $\mathcal{H} = H(\mathbf{z}) + V_{ext}(\mathbf{z})$. From Eq. 7.19, one sees that if the derivatives of F are not too high or k is chosen large enough, then one can safely assume that $\phi(\mathbf{z}) \simeq F(\mathbf{z})$.

Chapter 8

Input to ORAC

8.1 General Features

Input File. At execution time ORAC reads an input file in free format from the standard input. Each line of the input is read as a 80 character string and interpreted. If the first character of the input line is a # the line is ignored. In order to be interpreted the input line is parsed in the composing words which are sequences of characters separated by blanks or commas. Each word represents an instruction which must be interpreted by the program.

Instructions Set. The instruction set of ORAC includes *environments*, *commands* and *sub-commands*. An input file is made out of a series of environments. Each environment allows a series of commands which might use a few sub-commands. Environments resembles Fortran NAMELIST, but have not been programmed as NAMELIST. The environment name is a string which always starts with a & followed by capital letters. Each environment ends with the instruction &END. Command names are characters strings all in capital letters. Each command reads a variable set of parameters which can be characters and/or numbers (real or integer). There are also commands (*structured commands*) which are composed of more than one input line. A structured command end with the instruction END and allows a series of sub-commands in its inside. Sub-commands are in lower case and can read sub-string of characters and/or real or integer numbers. In the following section we will describe in details all the supported instruction allow by ORAC .

Handling External Files. Many ORAC commands provide instructions to open external files. No unit number needs to be provided as ORAC open sequentially the required files assigning at each file a unit number according to their order of occurrence in the input file. The file units begin at unit 10 and are augmented of one unit for each new file.

8.2 Environments, Commands and Sub-commands

The following 10 environments are available:

- 1 &ANALYSIS retrieves the history file.
- 2 &INOUT contains commands concerning input/output operations which can be carried out during run time. The commands allowed within the INOUT environment write history files in different formats and dump the restart files.
- 3 &INTEGRATION includes commands defining the integration algorithms to be used during the simulation run.
- 4 &META includes commands defining the metadynamics simulation.
- 5 &POTENTIAL includes commands which define the general features of the system interacting potentials. These features are common to both solute and solvent and concern only the non-bonded interactions.

- 6 **&PROPERTIES** includes commands which make ORAC compute run time observables. The commands allowed within the **&PROPERTIES** environment can compute on the fly: pair correlation functions, static structure factors and velocity auto-correlation function. This environment is not supported.
- 6 **&REM** setup the Replica Exchange simulation (work only with the parallel version).
- 8 **&RUN** defines run time parameters which concern output printing and run averages.
- 9 **&SETUP** includes commands concerned with the simulation box setup. In this environment, the simulation cell parameters, dimensions and symmetry can be initialized. Moreover, files containing the system coordinates in appropriate format can be provided.
- 10 **&SIMULATION** includes commands which define the type of simulation that is to be carried out. In particular, commands are available to run steepest descent energy minimization, and molecular dynamics simulations in various ensembles.
- 11 **&SOLUTE** includes commands which are concerned with specific aspects of the solute force field and structure.
- 12 **&SOLVENT** includes commands which are concerned with specific aspects of the solvent force field and structure.
- 13 **&PARAMETERS** includes commands which read the topology and force field parameter files of the solute. These files contain sufficient information to define the solute topology and to assign potential parameters to the solute molecules.

Commands' supporting policy. Since ORAC is free of charge no professional support is provided. Bugs are fixed, upon request to the E-mail address `oracmaster@cecddec.cecarn.fr`, at the authors earliest convenience and support may be requested only for environments, commands or sub-commands which are not marked **unsupported** in this manual. ORAC is simultaneously an ancient code and a new code which is still in the developing stage. ORAC has indeed a stable "core" i.e. the part which is officially maintained, but it also has some obsolete options, some features used for diagnostic or debugging purposes and some other experimental features not yet fully tested, i.e. the unsupported material. Unsupported features are by no means essential to the ORAC functioning and may belong to three categories:

- **Experimental:**
These features have been generally tested on only one or two unix platform (usually OSF1 or HP-UX), some of them can be used only while running in single time mode, some other can be used only while running with r-RESPA. These features are documented normally and in the WARNINGS section they are referred as
Experimental - **Unsupported**.
- **Diagnostic:**
These feature were introduced in the developing stage for diagnostics and debugging purposes. In the current version the diagnostic features are kept since they may turn to be useful to the programmer when modifying the code. These features are documented normally and in the WARNINGS section they are referred as
Diagnostic - **Unsupported**.
- **Obsolete features:**
These features are no longer used and will be eliminated in the next ORAC release. They are poorly documented and they are referred as
Obsolete - **Unsupported**.

8.2.1 &ANALYSIS

This environment includes commands which define the starting and ending record for reading the trajectory file (see also `TRAJECTORY`, `DUMP(&INOUT)`). The following are allowed commands:

START, STOP, UPDATE

START

NAME

START

SYNOPSIS

START *nconf*

DESCRIPTION

The trajectory file specified with the command `TRAJECTORY(&INOUT)` is read in starting from configuration *nconf*.

EXAMPLES

START 1

STOP

NAME

STOP

SYNOPSIS

STOP *nconf*

DESCRIPTION

The trajectory file specified with the command `TRAJECTORY(&INOUT)` is read stopping at configuration *nconf*.

EXAMPLES

STOP 1000

UPDATE

NAME

UPDATE – Update neighbor list for analysis

SYNOPSIS

UPDATE *nconf* *rcut*

DESCRIPTION

Update the neighbor lists for, e.g., radial distribution function calculations, every *nconf* configurations using a cut-off of *rcut* Å.

EXAMPLES

UPDATE 2 10.0 Å.

WARNINGS

Diagnostics - **Unsupported**

8.2.2 &INOUT

The environment &INOUT contains commands concerning input/output operations which can be carried out during run time. The commands within the INOUT environment allow to write history files in different formats and to dump restart files. The following commands are available:

ASCII, ASCII_OUTBOX, DYNAMIC, DUMP, PLOT, RESTART, SAVE, TRAJECTORY

ASCII

NAME

ASCII – Write solute and solvent coordinates to a history PDB file

SYNOPSIS

ASCII *fplot* OPEN *filename*

DESCRIPTION

This command is active both for solute and solvent molecules. It writes a history PDB file containing the system coordinates. The centers of mass of all molecules are always *inside* the simulations cell. The dumping frequency, in fs, is *fplot*. At each writing the system coordinates in PDB format are appended to the history file *filename*.

EXAMPLES

ASCII 10.0 OPEN test.pdb

Write system coordinates to the history pdb file test.pdb every 10 fs.

WARNINGS

Work only during the acquisition phase (see TIME in environment &RUN).

ASCII_OUTBOX

NAME

ASCII – Write solute and solvent coordinates to a history PDB file

SYNOPSIS

ASCII *fplot* OPEN *filename*

DESCRIPTION

This command is active both for solute and solvent molecules. It writes a history PDB file containing the system coordinates. The dumping frequency, in fs, is *fplot*. The centers of mass the molecules are at the position given by the simulation and may be hence also *outside* the simulation box. At each writing the system coordinates in PDB format are appended to the history file *filename*.

EXAMPLES

ASCII_OUTBOX 10.0 OPEN test.pdb

Write system coordinates to the history pdb file test.pdb every 10 fs.

WARNINGS

Work only during the acquisition phase (see TIME in environment &RUN).

DYNAMIC

NAME

DYNAMIC – Write force field parameters in extended format (see also DEBUG(&RUN))

SYNOPSIS

DYNAMIC OPEN *filename*

DESCRIPTION

This command prints out to the file *filename* the parameters of the force field *for the solute only* in a verbose format.

EXAMPLES

DYNAMIC OPEN ff.out

DUMP

NAME

DUMP – Write coordinates to a direct access unformatted file with a given frequency. The file is written in a particular format such that it can be easily retrieved at analysis time “by time” and “by atoms”.

SYNOPSIS

DUMP

....

END

DESCRIPTION

The DUMP structured command stores the coordinates of the system during a simulation run with a selected frequency. The coordinates are stored in single precision to save disk space. The following subcommands may be specified within DUMP:

atom_record, occupy, write

- **atom_record** *natom_rec*
Defines the number of atoms per record. Atomic coordinates are dumped to disk as REAL*4. RecordLength is defined as $lrec1 = natom * 3 * 4$
- **occupy**
Allocates disk storage for history file *before* the simulation is started. **occupy** fills with zeroes the *entire* direct access history file(s) whose dimensions are controlled by the command MAXRUN(&RUN) and by the number of atoms in the systems. If **occupy** is not specified the history file is expanded at each write request *during* the simulation. This command is useful when sharing disk resources with others, preventing the simulation to die because of sudden lack of disk space.
- **write** *ftime* OPEN *file_name*
Defines the dumping frequency and the trajectory auxiliary *file_name*. Coordinates are dumped to disk every *ftime* femtoseconds. The auxiliary file *file_name* contains the names for parameters and trajectory files and must be user supported. At execution time this file is rewritten by the program which supports extra information, computed according to input specifications, needed when retrieving the file (see &ANALYSIS). The file *file_name* looks like

```
system_file_0
traject_file_1
...
traject_file_n
```

`system_file` is the parameters file where the time steps and the CO matrix are specified. All other files are reserved for the trajectory. Partitioning a very long trajectories in many files allows to overcome, e.g., OS set file size limits or filesystem limits.

EXAMPLES

```
DUMP
  write 30.0 OPEN alk-1.aux
  occupy
  atom_record 30
END
```

Writes history file and parameters file as specified in auxiliary file `alk-1.aux` every 30.0 fs. After execution the file `alk-1.aux` is rewritten by the program and looks like

Rewritten by Program

<code>system_file</code>	66	0	0
<code>traject_file_1</code>	1320	20	30

The numbers in columns 1 are the length of the file in records. The numbers in the second columns and second row are the number of records per point (calculated by the program) and the number of atoms per record (given in input: see subcommand `atom_record`) in the trajectory file `traject_file_1`. In the above example the record length is $30 \times 4 \times 3 = 360$ bytes, the total size of the file of bytes, allocated at simulation start, is given by $30 \times 3 \times 4 \times 1230 = 442800$, and the total number of bytes dumped per phase space point is given by $20 \times 30 \times 3 \times 4 = 7200$.

WARNINGS

Work only during the acquisition phase (see `TIME` in environment `&RUN`).

PLOT

NAME

PLOT – Write solute coordinates and connection table to a history file in Protein Data Bank Format (PDB).

SYNOPSIS

```
PLOT fplot OPEN filename
PLOT FRAGMENT fplot OPEN filename
PLOT CENTER fplot OPEN filename
PLOT STEER fplot OPEN filename
PLOT STEER_ANALYTIC fplot OPEN filename
PLOT STEER_TEMPERATURE fplot OPEN filename
```

DESCRIPTION

It writes a history formatted file containing the coordinates of selected part of the solute (and) the solvent coordinates. The dumping frequency in fs is *fplot*.

EXAMPLES

```
PLOT 10.0 OPEN test.pdb
```

Write coordinates of the backbone atoms of the solute in PDB format every 10 fs to file *test.pdb*

```
PLOT CENTER 10.0 OPEN test.pdb
```

Write coordinates of all atoms of the system in PDB format every 10 fs to file *test.pdb*. Identical to `ASCII_OUTBOX(&INOUT)`

```
PLOT FRAGMENT 10.0 OPEN test.xyz
```

Write coordinates of a fragment of the solute (in xyz format) selected according the DEF_FRAGMENT (&PROPERTIES) directive every 10 fs to file *test.xyz*. The fragment is defined as follows:

```
&PROPERTIES
```

```
....
```

```
DEF_FRAGMENT 1 38
```

```
...
```

```
&END
```

The file *test.xyz* can be animated using the XMOL public domain molecular graphics program.

This defines a fragment consisting of the first 38 atoms of the solute. The numeral order of the atoms corresponds to that specified in the topology file (Sec. 8.3).

```
PLOT STEER 50.0 OPEN wrk.out
```

write the accumulated work (see Eq. 7.16) to the file *wrk.out* every 50 fs. The accumulated work at time t is calculated as $\mathcal{W} = \mathcal{H}(t) - \mathcal{H}(0)$, where $\mathcal{H}(t)$ is the total energy of the microcanonical *extended* system, i.e. it includes the energy of the thermostat and/or of the barostat. If the integration time steps are too large and the simulation shows a energy drift, then the accumulated work includes the dissipation due to the energy drift of the integrator.

```
PLOT STEER_ANALYTIC 50.0 OPEN WRK.out
```

Write the accumulated work (see Eq. 7.16) to the file *wrk.out* every 50 fs. The accumulated work at time t is calculated analitically according to Eq. 7.16. This option is slightly more computationally demanding than the previous one, but in this case, the accumulated work is not affected by the energy drift. The last two commands are to be used in conjunction with the STEER(&RUN) command and with the commands ADD_STR_BONDS, ADD_STR_BENDS, ADD_STR_TORS (namelist &POTENTIAL) for defining an external steering potential for SMD.

```
PLOT STEER_TEMPERATURE 50.0 OPEN WRKTEMP.out
```

In a steered temperature sumulation[120], write the accumulated (adimensional) therml work every 50 fs to the file *WRKTEMP.out*. This command must be used in conjunction with the STEER (&RUN) command for steered molecular dyanmics simulations and with the THERMOS(&SIMULATION) command for running NVT simulations.

RESTART

NAME

RESTART – Write or read an unformatted file from which a simulation might be restarted

SYNOPSIS

```
RESTART
```

```
...
```

```
END
```

DESCRIPTION

The RESTART may include two subcommands:

- **read** *filename*
read restart configuration from file *filename*. When this subcommand is active CONTROL(&RUN) must be non zero and the command READ_TPGPRM(&PARAMETERS) must have been entered.
- **read_multiple_restart** *filename_prefix* *num*
This command works only if the code is compiled using the MPI libraries and is not recognized when running in serial. Each of the *nprocs* processor will read a restart file named *filename_prefix* // *iproc+num* // .rst. So if *filename_prefix* is /u/foo/restarts/ala and *num* is 0, then process 0 will read the file /u/foo/restarts/ala0000.rst , process 1 will

read the file `/u/foo/restarts/ala0001.rst` and so on. This command is useful when running in parallel multiple steered molecular dynamics trajectories (see also commands `ADD_STR_BONDS`, `ADD_STR_BENDS`, `ADD_STR_TORS`, of namelist `&POTENTIAL`.)

- `rmr` *filename_prefix* *num*
Same as above.
- `write` *fprint* `OPEN` *filename*
write restart configuration to file *filename* every *fprint* fs.
- `write` *fprint* `SAVE_ALL_FILES` *filename*
write restart configuration to files *filename//i//''*.`rst''` every *fprint* fs. (see also command `read_multiple_restart`).

EXAMPLES

```
RESTART
  read file1.rst
  write 1000.0 OPEN file2.rst
END
```

```
RESTART
  rmr ../RESTARTS/ala 0
END
```

NB: In the last example valid for parallel runs, the relative path is specified with respect to the **actual** `pwd` of the parallel processes.

TRAJECTORY

NAME
TRAJECTORY – Read history file

SYNOPSIS
TRAJECTORY *filename*

DESCRIPTION
The `TRAJECTORY` command instructs the program to read the history file produced at an earlier time (see command `DUMP` in this environment). The auxiliary file *filename* contains the names of the parameters' and history files(s). See also environment `&ANALYSIS` for retrieving the history file and environment `&PROPERTIES` for computing properties from history files.

EXAMPLES
TRAJECTORY `file.aux`

8.2.3 &INTEGRATOR

This environment includes commands defining the integration algorithms to be used during the simulation run. The following commands are allowed:

MTS_RESPA, Timestep.

MTS_RESPA

NAME

MTS_RESPA – Use a multiple time step integrator

SYNOPSIS

MTS_RESPA

....

END

DESCRIPTION

The MTS_RESPA structured command opens an environment which includes several subcommands used to define a multiple time step integrator. The MTS_RESPA directive can be specified for NVE simulations and extended system simulations NHP, NPT and NVT. MTS_RESPA is also compatible with constraints. The following subcommands may be specified within MTS_RESPA:

step dirty very_cold_start energy_then_die k-ewald test-times p_test s_test

- **step** *type* *n* [*r* [*hl* [*dr*]]] [*reciprocal*]

The command **step** is used to define the potential subdivision and the corresponding time steps. The string *type* can be either “intra” or “nonbond”: in the former case the command defines an intramolecular shell, whereas in the latter a nonbonded shell is defined. If “intra” is specified only one keyword is expected, i.e. the integer *n*. When two subcommand of the type - **step** intra *n* - are entered, the first is assumed to refer to the faster intramolecular subsystem (the V_{n0} subsystem as defined in eq. 4.3 with $n = n0$) and the second is assumed to define the slower intramolecular subsystem (the V_{n1} subsystem as defined in eq. 4.3 with $n = n1$). If only one subcommand - **step** intra *n* - is entered then $n0$ is set to 1 and $n1 = n$. If no - **step** intra *n* subcommand is given then $n1 = n0 = 1$.

If the first argument of the **step** subcommand is the string “nonbond” then at least an integer and a real are expected. The integer *n* is the time step dividing factor of the nonbonded shell while the real argument equals the shell upper radius. Two more optional real arguments can be defined, i.e. the healing length at the upper shell radius and the corresponding neighbor list offset. The default values of the healing length are As for the intra shell, the more rapidly varying nonbonded shells are entered first. If three - **step nonbond** - subcommands are entered, then the first refers to the V_m , the second to the V_l and the third to the V_h subsystems, with *n* being *m*, *l*, 1 such that $\Delta t_h = \Delta t$, $\Delta t_l = \Delta t_h/l$, $\Delta t_m = \Delta t_l/m$, (see Table 4.3). *n* for the last nonbonded shell is set automatically to 1 disregarding its actual value. If two shells are entered then only two intermolecular time steps are used, i.e. $n = m$ and $l = 1$. If one shell is entered only one time step is defined and $m = l = 1$. When using Ewald, the V_{qr} term (Eq. 4.21) in the reciprocal lattice is assigned by entering the string **reciprocal** as the last argument of a - **step nonbond** directive.

- **k-ewald** *kl* *lambdakl* *km* *lambdakm* – Obsolete - **Unsupported**

kl and *km* define the shells in reciprocal space. Wave vectors $k = |\mathbf{k}|$ such that $rkcut \geq k > kl$, $kl \geq k > km$, and $km \geq k > 0$ are assigned to the *h*-shell *l*-shell and *m*-shell, respectively. *lambdakm*, *lambdakl* are the upper healing lengths for the reciprocal space *m* and *l* shells and the lower healing length for the reciprocal space *h* and *l* shells, respectively.

Warning: To be used only when `on` is specified in the directive `EWALD` (environment `&POTENTIAL`); `rkcut` must be defined in the directive `EWALD`). The reciprocal lattice assignment is best done via the keyword `reciprocal` of the command `step nonbond`.

- **test-times** `OPEN filename` – Diagnostic - **Unsupported**

Produce the time record of the potential and kinetic energies at the end of the propagation step (i.e. at intervals of Δt_h fs). The following is the format used for dumping the energies:

```
WRITE(ktest,300) tim,utot,ustot,uptot,upstot,ektot,pottot
300  FORMAT(' TotalEnergy',f12.3,6f15.3)
```

Where `tim,utot,ustot,uptot,upstot,ektot,pottot` are the values of the time, total energy, solvent potential energy, solute potential energy, solvent-solute potential energy, total kinetic energy, total potential energy. Time is given in fs and all energies in *KJ/mole*. The energy conservation ratio $R \equiv \Delta E / \Delta K$ and the drift $D = \frac{(E - \langle E \rangle)t}{t(t - \langle t \rangle)}$ are printed periodically (every $1000 * \Delta t_h$) and at the end of the simulation onto the file *filename*.

- **dirty** – Obsolete - **Unsupported**

Scales velocities to the initial total energy $E(0)$ during production stage. The scaling is done randomly with a Monte Carlo algorithm.

- **p_test** *n1 n2 n3 n4 n5* – Diagnostic - **Unsupported**

To be used in conjunction with subcommand `test-times`: print out time record of the subsystems potential and forces for the protein for atoms *n1 n2 n3 n4 n5*.

- **s_test** *n1 n2 n3* – Diagnostic - **Unsupported**

To be used in conjunction with subcommand `test-times`: print out time record of the subsystems potential and forces for the solvent for atoms *n1 n2 n3*.

- **very_cold_start** *rmax*

This option is useful when minimizing a protein in a highly unfavorable configuration. The real argument *rmax* is the maximum allowed displacement (in Å) for any atom when integrating the equations of motion irrespectively of the intensity of the force on that atom. This constraint avoid blowing up of the simulation.

- **energy_then_die**

Print out energies and then stops.

EXAMPLES

```
step intra 2
step intra 2
step nonbond 4 4.2
step nonbond 4 7.3  reciprocal
step nonbond 1 9.7
```

Here five time steps are defined, three for nonbonded potentials and two for intramolecular potential. The largest timestep Δt_h is defined by the command `TIMESTEP` in this environment (see above) and refers to the nonbonded subsystem with shell in the range 7.3 – 9.7 Å. We then have $\Delta t_l = \Delta t_h / 4$ referring to the 4.2 – 7.3 Å shell and $\Delta t_i = \Delta t_h / 4 / 4$ referring to the 0 – 4.2 Å shell. The reciprocal potential is assigned to the intermediate 4.2 – 7.3 Å shell. The two intramolecular shells have time steps $\Delta t_{n1} = \Delta t_h / 4 / 4 / 2$ and $\Delta t_{n0} = \Delta t_h / 4 / 4 / 2 / 2$.

```
step intra 2
step nonbond 3 6.5  reciprocal
step nonbond 1 9.5
test-times OPEN file-tests
```

Here only one intramolecular and two intermolecular time steps are defined. The reciprocal (PME or standard) contribution is assigned to the fastest intermolecular shell. Energy records are printed onto the file *file-tests* each Δt_h femtoseconds.

DEFAULTS

```
step intra 1
step intra 1
step nonbond 1 4.1 0.3 0.35
step nonbond 1 7.3 0.3 0.45 reciprocal
step nonbond 1 9.7 0.3 1.5
```

WARNINGS

- 1 When standard Ewald is used and the reciprocal space contribution is subdivided in **k**-shells, the intramolecular term of Eq. 4.21 is always assigned to the fastest **k**-shell. This may cause instability of the integration. Subdivision of the reciprocal lattice contribution with standard Ewald, although technically possible, is not recommended.
- 2 The directive **dirty** makes fast integrators stable but may severely affect dynamical properties.

TIMESTEP

NAME

TIMESTEP – Define the simulation time step

SYNOPSIS

TIMESTEP *time*

DESCRIPTION

The argument *time* represents the integration time step used during the run. As integration of the equations of motion is always done with the r-RESPA algorithm, *time* is the outer most time step. *time* must be given in units of femtoseconds.

EXAMPLES

TIMESTEP 9.0

8.2.4 &META

Define run time parameters concerning Metadynamics Simulation. The following commands are available:

ADD_BOND, ADD_BEND, ADD_TORS, RATE, READ, SAVE

ADD_BOND

NAME

ADD_BOND – Add the distance between two atoms to the list of reaction coordinates.

SYNOPSIS

ADD_BOND *iat1* *iat2* *w*

DESCRIPTION

This command adds to the list of the reaction coordinates of a metadynamics simulation the distance between atom *iat1* and *iat2*. The numeric order of the atom indices *iat1*, *iat2* is that specified in the topology file (see 8.3). The repulsive potential terms deposited in the space of the reaction coordinates during the simulation (see 5.3) will have a width *w* (in Å) in the direction of this distance.

EXAMPLES

ADD_BOND 1 12 0.2

Add the distance between atom 1 and atom 12 to the list of the reaction coordinates.

ADD_BEND

NAME

ADD_BEND – Add the bending angle between three atoms to the list of the reaction coordinates.

SYNOPSIS

ADD_BEND *iat1* *iat2* *iat3* *w*

DESCRIPTION

This command adds to the list of the reaction coordinates of a metadynamics simulation the bending angle between atom *iat1*, *iat2* and *iat3*. The central atom of the bending is *iat2*. The numeric order of the atom indices *iat1*, *iat2*, *iat3* is that specified in the topology file (see 8.3). The repulsive potential terms deposited in the space of the reaction coordinates during the simulation (see 5.3) will have a width *w* (in arc degrees) in the direction of this angle.

EXAMPLES

ADD_BEND 1 7 12 4.0

Add the bending angle between atom 1, atom 7 and atom 12 to the list of the reaction coordinates.

ADD_TORS

NAME

ADD_TORS – Add the torsional angle between four atoms to the list of the reaction coordinates.

SYNOPSIS

ADD_TORS *iat1* *iat2* *iat3* *iat4* *w*

DESCRIPTION

This command adds to the list of the reaction coordinates of a metadynamics simulation the torsional angle between atom *iat1*, *iat2*, *iat3* and *iat4*. The axis of the torsion is defined by the atoms *iat2* and *iat3*. The numeric order of the atom indices *iat1*, *iat2*, *iat3*, *iat4* is that specified in the topology file (see 8.3). The repulsive potential terms deposited in the space of the reaction coordinates during the simulation (see 5.3) will have a width *w* (in arc degrees) in the direction of this angle.

EXAMPLES

ADD_TORS 1 5 8 11 4.0

Add the torsional angle between atom 1, atom 5, atom 8 and atom 11 to the list of the reaction coordinates.

RATE

NAME

RATE – Define the deposition rate of a metadynamics run.

SYNOPSIS

RATE *mtime* [*mheight*]

DESCRIPTION

This command defines the deposition frequency *mtime* (in fs) and the height *mheight* (in kJ mol⁻¹) of the repulsive potential terms deposited during a metadynamics run. If *mheight* is not specified, then a zero height is assumed.

EXAMPLES

RATE 100.0 0.05

Depose an hill of height 0.05 kJ mol⁻¹ every 100 fs.

READ

NAME

READ – Read a trajectory from a previous metadynamics run.

SYNOPSIS

READ *filename*

DESCRIPTION

When present, the program reads a trajectory *filename* from a previous metadynamics run.

EXAMPLES

READ old_traj.out

Read trajectory from file old_traj.out.

WARNINGS

The metadynamics parameters of the previous and of the new simulation must be the same in order to obtain meaningful results.

TEMPERED

NAME

TEMPERED – During a metadynamics simulation, adds an hill to the biasing potential with a decreasing probability.

SYNOPSIS

TEMPERED T'

DESCRIPTION

When present, the program adds an hill to the current biasing potential with a probability given by $P(\text{acc}) = \exp(-V_{\text{max}}(t))/k_B T'$, where $V_{\text{max}}(t)$ is the maximum value of the potential $V(s, t)$ at time t and T' is a user-defined temperature.

EXAMPLES

TEMPERED 1000.0

Run a tempered metadynamics simulation adding new potential terms with a probability that depends on the ratio $V_{\text{max}}(t)/k_B * 1000.0$.

DEFAULTS

T' 0.0

WTEMPERED

NAME

WTEMPERED – During a metadynamics simulation, adds an hill to the biasing potential with a decreasing probability, following the well-tempered metadynamics algorithm.

SYNOPSIS

WTEMPERED T'

DESCRIPTION

When present, the program adds an hill to the current biasing potential with a probability given by $P(\text{acc}) = \exp(-V(s, t))/k_B T'$, where $V(s, t)$ is the value of the biasing potential and T' is a user-defined temperature.

EXAMPLES

WTEMPERED 1000.0

Run a tempered metadynamics simulation adding new potential terms with a probability that depends on the ratio $V(s, t)/k_B * 1000.0$.

DEFAULTS

T' 0.0

SAVE

NAME

SAVE – Save periodically a trajectory file during a metadynamics run.

SYNOPSIS

SAVE *fprint* [*filename*]

DESCRIPTION

When present, the program writes the trajectory in the space of the reaction coordinates, sampled with the frequency defined through the command **RATE**, to file *filename* every *fprint* fs. The first line of the file contains the number of hills deposited when the file was dumped and the height and the

width along each reaction coordinate. If at the beginning of the run a trajectory file from a previous metadynamics simulation was read through the command **READ**, then the program prints the *whole* trajectory.

EXAMPLES

SAVE 10000.0 traj.out

Print trajectory in file traj.out every 10 ps.

8.2.5 &PARAMETERS

This environment includes commands which read the topology and force field parameter files of the solute. These files described in Sec. 8.3 contain sufficient information to define the solute topology and to assign potential parameters to the solute molecules. The following commands are allowed:

```
ADD_TPG, JOIN, PRINT_TOPOLOGY, READ_TPGPRM READ_PRM_ASCII, READ_TPG_ASCII, REPL_RESIDUE,
WRITE_TPGPRM_BIN
```

ADD_TPG SOLUTE

NAME

ADD_TPG SOLUTE – Add topology components to the current solute molecule

SYNOPSIS

```
ADD_TPG SOLUTE
```

```
...
```

```
...
```

```
END
```

DESCRIPTION

The structured command **ADD_TPG SOLUTE** opens an environment including commands which add extra bonds, proper and improper torsions to the topology of the current solute molecule(s). The command is closed by **END**. This command must be used to connect atoms belonging to different residues of the current molecule. For instance to connect through a sulphur bridge two cysteine residues or to bind ligands to a metal atom.

- **bond 1ata 2atb residue num1 num2**

Add a bond to the topology of the current solute molecule which connect atom *ata* of residue number *num1* with atom *atb* of residue number *num2*. The number 1 and 2 refer to residue *num1* and *num2*, respectively. The atom label *ata* and *atb* must be defined in the general formatted topology file as labels of actual atoms of residue number *num1* and *num2*. Here, residue numbers are the sequential numbers of the residues as given in input to command **JOIN**.

- **torsion 1ata 1atb 2atc 2atd residue num1 num2**

Add a proper torsion to the topology of the current solute molecule. The number 1 and 2 refer to residue *num1* and *num2*, respectively. Atom *ata* and *atb* belong to residue number *num1*, while atoms *atc* and *atd* are on residue number *num2*. Additional torsion having three atoms on one residue and one atom on the other residue are also allowed. Here, residue numbers are the sequential numbers of the residues as given in input to command **JOIN**. If the command **AUTO_DIHEDRAL** of the environment **&SOLUTE** is used, no extra torsions need to be added to the current topology.

- **i.torsion 1ata 1atb 2atc 2atd residue num1 num2**

Add an improper torsion to the topology of the current solute molecule. The number 1 and 2 refer to residue numbers *num1* and *num2*, respectively. Atom *ata* and *atb* belong to residue number *num1*, while atoms *atc* and *atd* are on residue number *num2*. Additional improper torsions having three atoms on one residue and one atom on the other residue are also allowed. Here, residue numbers are the sequential numbers of the residues as given in input to command **JOIN**.

EXAMPLES

```

ADD_TPG SOLUTE
    bond 1sg 2sg residue 6 127
    bond 1sg 2sg residue 30 115
    bond 1sg 2sg residue 64 80
    bond 1sg 2sg residue 76 94
END

```

Add extra bonds to the current topology. In this example, the four sulphur bridges of hen egg lysozyme are given.

WARNINGS

This command is inactive when used in conjunction with READ_TPGPRM. To have the desired effect, the ADD_TPG environment must be used in conjunction with READ_TPG_ASCII and READ_PRM_ASCII.

JOIN

NAME

JOIN – Provide the list of residues forming the current solute or solvent molecule(s).

SYNOPSIS

```
JOIN [SOLUTE | SOLVENT]
```

```
...
```

```
...
```

```
END
```

DESCRIPTION

The structured command JOIN reads the sequential list of labels corresponding to the residues forming the solute molecule(s). The list of residues begins at the line following JOIN. The end of this list is signaled by END on the line following the last residue label. Each residue labels must have been defined in the general formatted topology file read by READ_TPG_ASCII. See Sec. 8.3 for explanations.

EXAMPLES

JOIN SOLUTE

```

lys-h val phe gly arg cys glu leu ala ala ala met lys
arg hsd gly leu asp asn tyr arg gly tyr ser leu gly
asn trp val cys ala ala lys phe glu ser asn phe asn
thr gln ala thr asn arg asn thr asp gly ser thr asp
tyr gly ile leu gln ile asn ser arg trp trp cys asn
asp gly arg thr pro gly ser arg asn leu cys asn ile
pro cys ser ala leu leu ser ser asp ile thr ala ser
val asn cys ala lys lys ile val ser asp gly asn gly
met asn ala trp val ala trp arg asn arg cys lys gly
thr asp val gln ala trp ile arg gly cys arg leu-o

```

```
END
```

Sequence of residues for hen-egg lysozyme. All labels must have been defined in the general formatted topology file.

JOIN SOLVENT

```
hoh
```

```
END
```

Defines the topology of the solvent

WARNINGS

The command is inactive when used in conjunction with READ_TPGPRM. To have the desired effect, the JOIN environment must be used in conjunction with READ_TPG_ASCII, READ_PRM_ASCII and, optionally, the REPL_RESIDUE environment.

PRINT_TOPOLOGY

NAME

PRINT_TOPOLOGY – Print topology components of the current solute molecule.

SYNOPSIS

```
PRINT_TOPOLOGY
```

```
...
```

```
END
```

DESCRIPTION

PRINT_TOPOLOGY is a structured command to be used for printing out part of the topology and potential information for the solute molecule. The following subcommands may be specified within PRINT_TOPOLOGY:

atoms bendings bonds constraints I-torsions P-torsions sequence

- **bonds**
Print the bonds list.
- **bendings**
Print the bendings list.
- **constraints**
Print the bond constraints list.
- **I-torsions**
Print the proper torsion list.
- **P-torsions**
Print the improper torsion list.
- **sequence**
Print info on the units sequence of both solvent and solute

EXAMPLES

```
PRINT_TOPOLOGY
```

```
  bonds
```

```
  P-torsions
```

```
END
```

READ_TPGPRM

NAME

READ_TPGPRM – Read an unformatted parameter and topology file

SYNOPSIS

READ_TPGPRM *filename* [no_warning]

DESCRIPTION

The command reads the binary force field parameters and topology file *filename*. This file contains the topology and force field parameters tables. It is created with the commands WRITE_TPGPRM_BIN, READ_TPG_ASCII and READ_PRM_ASCII. The tables contained in file *filename* are associated only with the current solute molecule(s) and can only be used for that (those) molecule(s). In alternative to the command READ_TPGPRM, READ_TPG_ASCII and READ_PRM_ASCII, which read the general formatted topology and parameters files, can be used. Since the use of the latter commands implies the calculation of the topology and parameters tables for the current solute molecule, it is advisable to use them only a first time to create the unformatted file read by READ_TPGPRM. When READ_TPGPRM is entered, all the topology of the system is read in from the specified binary file and the topology commands such as JOIN_SOLUTE JOIN_SOLVENT or ADD_TPG are ignored. Also the environments &SOLUTE, &SOLVENT, &SETUP need not to be specified.

EXAMPLES

```
&PARAMETERS
  READ_TPGPRM_BIN  benz.prmtpg
&END
&SIMULATION
  ...
&END
&INOUT
  RESTART          50.0    OPEN    benz.rst
&END
&INTEGRATOR
  ...
&END
&POTENTIAL
  ..
&END
&RUN
  CONTROL          1
  ..
&END
```

In this example all topology information and the coordinates of all atoms in the system are taken in care by only three directives: READ_TPGPRM_BIN, RESTART, CONTROL. The files benz.prmtpg and benz.rst which contains the topology and the coordinates, respectively must have been produced with a previous run.

READ_PRM_ASCII

NAME

READ_PRM_ASCII – Read a general formatted parameters file

SYNOPSIS

READ_PRM_ASCII *filename*

DESCRIPTION

Here *filename* is the ASCII parameter file. The general formatted force field parameters file is described in Sec. 8.3. In this file one must define each potential energy parameter of the given force field defined in Eq. 4.3. It must be consistent with the general topology file read by READ_TPG_ASCII. The same parameters file can be used for many different solute molecules. This is the reason of the word “general”.

EXAMPLES

READ_PRM_ASCII forcefield.prm

Read the general formatted parameters file forcefield.prm.

WARNINGS

Must be used in conjunction with command READ_TPG_ASCII.

READ_TPG_ASCII

NAME

READ_TPG_ASCII – Read a general formatted topology file

SYNOPSIS

READ_TPG_ASCII *filename*

DESCRIPTION

Here *filename* is the ASCII topology file. The general topology file is described in Sec. 8.3. It must define each residue contained in the current solute molecule. The same topology file can be used for many different solute molecules. This is the reason of the word “general”.

EXAMPLES

READ_TPG_ASCII forcefield.tpg

Read the formatted topology file forcefield.tpg.

WARNINGS

Must be used in conjunction with command READ_PRM_ASCII.

REPL_RESIDUE

NAME

REPL_RESIDUE – Replace or add the topology of a certain residue

SYNOPSIS

REPL_RESIDUE

...

...

END

DESCRIPTION

The command REPL_RESIDUE opens an environment which includes the same series of commands and subcommands accepted by the general formatted topology file described in Sec. 8.3.

EXAMPLES

REPL_RESIDUE

```
RESIDUE gly ( Total Charge = 0.0 )
atoms
```



```

group
n      n      -0.41570
hn     h      0.27190
group
ca     ct     -0.02520
ha1    h1     0.06980
ha2    h1     0.06980
group
c      c      0.59730
o      o     -0.56790
end

bonds
n      hn      n      ca      o      c      c      ca
ca     ha1     ca     ha2
end

imphd
-c     ca      n      hn      ca      +n      c      o
end

termatom n c
backbone n ca c
RESIDUE_END
END

```

Replace or add the topology for residue gly.

WRITE_TPGPRM_BIN

NAME

WRITE_TPGPRM_BIN – Write an unformatted parameter and topology file

SYNOPSIS

WRITE_TPGPRM_BIN *filename*

DESCRIPTION

This command must be used in combination with READ_TPG_ASCII and READ_PRM_ASCII. It produces the binary file *filename* containing the force field and topology tables associated with the current solute molecule(s) which can be reread in subsequent runs by the command READ_TPGPRM.

EXAMPLES

WRITE_TPGPRM_BIN molecule1.prmtpg

Write the unformatted topology and parameter file for the current solute molecule that can be read by READ_TPGPRM.

WARNINGS

Must be used in conjunction with commands READ_TPG_ASCII and READ_PRM_ASCII.

8.2.6 &POTENTIAL

The environment &POTENTIAL includes commands which define the general features of the system interacting potentials. These features are common to both solute and solvent and concern both bonded and non-bonded interactions. The following are allowed commands:

ADD_STR_BONDS, ADD_STR_BENDS, ADD_STR_TORS, ADJUST_BONDS, AUTO_DIHEDRAL, BENDING, CONSTRAINT, CUTOFF, ERF_CORR, ERFC_SPLINE, EWALD, GROUP_CUTOFF, I-TORSION, JORGENSEN, KEEP_BONDS, LJ-FUDGE, LINKED_CELL, QQ_FUDGE, SELECT_DIHEDRAL, STEER_PATH, STRETCHING, UPDATE, VERLET_LIST

ADD_STR_BONDS

NAME

ADD_STR_BONDS – Add a stretching potential between two target atoms.

SYNOPSIS

ADD_STR_BONDS *iat1* *iat2* *k* *r*₀ [*r*_τ]

DESCRIPTION

This command can be used to impose an additional stretching constraint between atom *iat1* and *iat2* of the solute. The numeric order of the solute atom indices *iat1*, *iat2* is that specified in the topology file (see 8.3). The added stretching potential has force constant *k* (in Kcal/mol/Å²) and equilibrium distance *r*₀ (in Å). If *r*_τ is also specified, then the added stretching potential is time dependent and *r*_τ is the equilibrium distance after the steering time *τ* (see STEER(&RUN)) command for the definition of the steering time in a SMD simulation)

WARNINGS

If the chosen *r*₀ is very different from the *actual* value of the distance |**r**_{iat1} – **r**_{iat2}| at time 0, a very large force is experienced by the atoms involved in the added stretching and the simulation may catastrophically diverge after few steps.

EXAMPLES

– Example 1.

```
ADD_STR_BONDS  1 104  400.  31.5
```

– Example 2.

```
&PARAMETERS
  READ_TPGPRM_BIN  ala10_A.prmtpg
&END

...
&POTENTIAL
  ...
  ADD_STR_BONDS  1 104  400.  31.5  15.5
  ..
&END
....
```

```

...
&RUN
  CONTROL 2
  ...
  REJECT 0.0
  STEER 10000. 50000.
  TIME 50009.0
  ...
&END
..
&INOUT
  RESTART
    rmr ../RESTART/ala10 0
&END

```

In the first example a stretching constraint is imposed between atom 1 and atom 104 of the solute. In the second example a time-dependent driving potential is applied to the same atoms of the solute. The equilibrium distance of such harmonic driving potential moves at constant velocity in $\tau = 40$ ps (starting at $t=10$ ps) between $r_0 = 31.5$ and $r_\tau = 15.5$. Since the directive `rmr` (or `read_multiple_restart`) is issued in the `RESTART(&INOUT)` command, the example is assumed to run in parallel. Each process reads a *different* restart file named `ala10iproc.rst` in the `../RESTART` directory. Note that the path of the restart files is specified with respect to the actual value of the `pwd` command when the parallel version is executed (i.e. in the `PARxxxx` directories).

ADD_STR_BENDS

NAME

ADD_STR_BENDS – Add a bending potential between three target atoms.

SYNOPSIS

ADD_STR_BENDS *iat1* *iat2* *iat2* *k* α_0 [*alpha_τ*]

DESCRIPTION

This command can be used to impose an additional bending constraint between atom *iat1*, *iat2* and *iat3* of the solute. The numeric order of the solute atom indices *iat1*, *iat2* is that specified in the topology file (see Sec. 8.3). The central atom of the bending is *iat2*. The added bending potential has force constant *k* (in Kcal/mol/rad²) and equilibrium bending angle α_0 (in degrees). If α_τ is also specified, then the added bending potential is time dependent and α_τ is the equilibrium bending angle after the steering time τ (see `STEER(&RUN)` command for the definition of the steering time in a SMD simulation)

WARNINGS

If the chosen α_0 is very different from the *actual* value of the bending angle at time 0, a very large force is experienced by the atoms involved in the added bending and the simulation may catastrophically diverge after few steps.

EXAMPLES

– Example 1.

```
ADD_STR_BENDS 1 50 104 400. 180.0
```

– Example 2.

```

&POTENTIAL
...
  ADD_STR_BENDS  1 50 104  400.  180.0 90.0
..
&END
....
...
&RUN
...
  STEER 10000.  50000.
...
&END

```

In the first example a bending constraint is imposed between atom 1, atom 50, and atom 104 of the solute. In the second example a time-dependent driving potential is applied to the same atoms of the solute. The equilibrium bending angle of such harmonic driving potential move at constant velocity in $\tau = 40$ ps (starting at $t=10$ ps) between $\alpha_0 = 180.0$ and $\alpha_\tau = 90.0$.

ADD_STR_TORS

NAME

ADD_STR_TORS – Add a harmonic bending potential between three target atoms.

SYNOPSIS

```
ADD_STR_TORS      iat1      iat2      iat3      iat3      k       $\theta_0$       [  $\theta_\tau$  ]
```

DESCRIPTION

This command can be used to impose an additional (harmonic) torsional constraint between atoms *iat1*, *iat2*, *iat3* and *iat4* of the solute. The axis of the torsion is defined by the atoms *iat2*, *iat3*. The numeric order of the solute atom indices *iat1*, *iat2*, *iat3*, *iat4* is that specified in the topology file (see Sec. 8.3). The added torsional potential has force constant *k* (in Kcal/mol/rad²) and equilibrium dihedral angle θ_0 (in degrees). If θ_τ is also specified, then the added torsional potential is time dependent and θ_τ is the equilibrium dihedral angle after the steering time τ (see STEER(&RUN)) command for the definition of the steering time in a SMD simulation)

WARNINGS

If the chosen θ_0 is very different from the *actual* value of the dihedral angle at time 0, a very large force is experienced by the atoms involved in the added bending and the simulation may catastrophically diverge after few steps.

EXAMPLES

– Example 1.

```
ADD_STR_TORS  1 50 70 104  400.  60.0
```

– Example 2.

```

&POTENTIAL
...
  ADD_STR_TORS  1 50 70 104  400.  60.0  90.0
..
&END

```

```

    . . . .
    . . .
    &RUN
    . . .
    STEER 10000.  50000.
    . . .
    &END

```

In the first example a torsional constraint is imposed between atom 1, atom 50, atom 70 and atom 104 of the solute. In the second example a time-dependent driving potential is applied to the same atoms of the solute. The equilibrium dihedral angle of such harmonic driving potential moves at constant velocity in $\tau = 40$ ps (starting at $t=10$ ps) between $\theta_0 = 60$ and $\theta_\tau = 90$ degrees.

ADJUST_BONDS

NAME

ADJUST_BONDS – Constraints bond lengths to starting values.

SYNOPSIS

ADJUST_BONDS

DESCRIPTION

This command should be specified when bond constraints are imposed to the system (see command **STRETCHING** and **CONSTRAINT** in this environment). If specified, all bonds to be constrained are constrained to the lengths specified in the force field parameter file (see sec 8.3.1 PDB file

DEFAULTS

ADJUST_BONDS is *.TRUE.*

AUTO_DIHEDRAL

NAME

AUTO_DIHEDRAL – Include all the proper torsion angle in the interaction potential

SYNOPSIS

AUTO_DIHEDRAL

WARNINGS

Obsolete - **Unsupported**

BENDING

NAME

BENDING – Constrain bendings.

SYNOPSIS

BENDING **on**

BENDING **off**

DESCRIPTION

With the argument **on**, this command includes harmonic bending potentials in the total solute potential. Conversely, if the argument is **off** all the bending of the solute molecules are constrained.

DEFAULTS

BENDING off

WARNINGS

Obsolete - **Unsupported**

CONSTRAINT

NAME

CONSTRAINT – Constrain bendings.

SYNOPSIS

CONSTRAINT SHAKE

CONSTRAINT MIM *mimlim*

DESCRIPTION

Select procedure for fulfilling constraints. With the argument **SHAKE** ORAC uses SHAKE. With argument **MIM** ORAC uses the matrix inversion method (MIM). In the latter case the maximum physical dimension of the constraint matrix *mimlim* must be specified. MIM is best used in conjunction with **STRETCHING HEAVY**

DEFAULTS

BENDING off

CUTOFF

NAME

CUTOFF

SYNOPSIS

CUTOFF *rspoff*

WARNINGS

Obsolete - **Unsupported**

ERF_CORR

NAME

ERF_CORR – Implements intramolecular Ewald correction

SYNOPSIS

ERF_CORR *nbin rlow rup*

DESCRIPTION

Adds correction of Eq. 4.48, evaluated only for excluded *intra*-molecular contacts (stretching, bending and “fudged” part of 1-4 interactions) to account for reciprocal space cutoff error. The function $\chi(r, \alpha)$ is B-splined using *nbin* points in the range $rlow < r < rup$.

EXAMPLES

ERF_CORR 2000 0.8 4.5

WARNINGS

Choose carefully *rlow* and *rup*. If an intramolecular distance outside the range is found during execution, unpredictable results may occur.

ERFC_SPLINE

NAME

ERFC_SPLINE – Use spline to compute the complementary error function used for electrostatics in direct space

SYNOPSIS

ERFC_SPLINE *erfc_bin*

ERFC_SPLINE *erfc_bin* **corrected** *rcut*

DESCRIPTION

By default ORAC uses a 5 parameter expansion to compute the complementary error function required by the direct space electrostatic potential (V_{qd} in Eq. 4.20). With the command **ERFC_SPLINE** this expansion is replaced by a B-spline. The function $erfc(x)$ is splined from $x = 0$ to $x = 1.1\alpha r_{cut}$, where α and r_{cut} are the Ewald sum parameter and the radial cutoff, respectively. The argument *erfc_bin* is the bin size of the spline. The usage of the **ERFC_SPLINE** option is useful when running on workstations where a saving of 10-15 % in CPU time is usually obtained. **ERFC_SPLINE** may also be used to speed up the Ewald method. By specifying the directive **corrected** ORAC corrects for the reciprocal lattice cutoff for all *intermolecular* interactions in the direct lattice using the same oscillating potential of Eq. (4.48) (see Sec. 4.4) used for correcting the *intra*-molecular potential (see **ERF_CORR** in this environment). This allows the use shorter cutoffs in reciprocal space (or coarser grids in SPME). The argument *rcut* corresponds to the maximum distance for the spline table. Must be larger than the current cutoff (see examples).

EXAMPLES

ERFC_SPLINE 0.01

A B-spline is used to evaluate the direct space sum. To evaluate the B-spline the original function is computed on a grid of 0.01 bin size.

ERFC_SPLINE 0.01 **corrected** 14.0

The splined potential is now given by standard the direct lattice Ewald term *plus* the $\chi(r, \alpha)$ potential defined in Eq. (see also command **ERF_CORR** in this environment). The B-spline look up table is done for distances $0 < r < 14$.

WARNINGS

rcut is an *atomic* cutoff. Always define *rcut* large enough to assure that all atoms are included within *rcut* for any molecular pair. E. g., if r_h the largest cutoff defined in the structured command **MTS_RESPA** (&INTEGRATOR) and the molecule has a maximum extension in any possible direction of ΔR , choose $rcut = r_h + \Delta R$

EWALD

NAME

EWALD – Determine if standard Ewald or particle mesh Ewald sum must be used

SYNOPSIS

EWALD **off**

EWALD **ON** *alphal* *rkcute*

EWALD **PME** *alphal* *ftt1* *ftt2* *ftt3* *order*

EWALD **REMOVE_MOMENTUM**

DESCRIPTION

As described in Sec. 4.1 ORAC can handle the electrostatic interactions by Ewald summation. If the argument to the command is **on** followed by *alphal* and *rkcute*, standard Ewald is used with $\alpha = \text{alphal}$ in \AA^{-1} and the cutoff in k-space $k_{cut} = rkcute$ in \AA^{-1} . The output will show what degree of convergence has been reached showing the numerical value of $\text{erfc}(r_{cut}\alpha)/r_{cut}$ and of $\exp(-k_{cut}^2/2\alpha)/k_{cut}$. To chose instead PME the argument **pme** *alphal* *ftt1* *ftt2* *ftt3* *order* must be chosen. Here, **alphal** has

the same meaning as before, while **order** is the order of the spline and *fft1* *fft2* *fft3* define the three dimensional grid in direct space providing the number of bins along the *a*, *b*, *c* crystal axis, i.e. the dimensions of the 3-D FFT used in the PME method. For best efficiency *fft1*, *fft2*, and *fft3* must be multiples of 2,3 or 5. Of course, if the argument is **off** no Ewald summation is used for the electrostatic interactions. When using PME, Newton third law is not obeyed exactly but to the numerical accuracy of the interpolation. This leads to a small momentum of the MD cell which can be removed by specifying the argument **REMOVE_MOMENTUM**

EXAMPLES

1 **EWALD** on 0.4 2.0

Standard Ewald is used with parameters $\alpha = 0.4 \text{ \AA}^{-1}$ and $k_{cut} = 2.0 \text{ \AA}^{-1}$.

2 **EWALD** pme 0.4 45 32 45 6

EWALD REMOVE_MOMENTUM

The electrostatic interaction is handled by PME with $\alpha = 0.4 \text{ \AA}^{-1}$, the order of the spline is 6 and the number of bins for defining the grid are 45,32,45 along the *a*, *b*, *c* crystal axis, respectively. Typically, acceptable relative accuracy (10^{-4} - 10^{-5}) on electrostatic energies and forces is obtained with a grid spacing of about 1-1.2 \AA along each dimension. In this example the second invocation of the **EWALD** command is used in order to remove the linear momentum of the MD cell.

GROUP_CUTOFF

NAME

GROUP_CUTOFF

WARNINGS

Obsolete - **Unsupported**

H-MASS

NAME

H-MASS – Change the hydrogen mass

SYNOPSIS

H-MASS *hdmass*

DESCRIPTION

The command **H-MASS** changes the mass of all solute hydrogens to *hdmass* given in a.m.u. This allows to use larger time steps during equilibration.

EXAMPLES

H-MASS 10.0

I-TORSION

NAME

I-TORSION – Set the type of improper torsion potential

SYNOPSIS

I-TORSION *itor_type*

DESCRIPTION

This command defines which improper torsion potential must be used. If the argument string *itor_type* is **HARMONIC** a harmonic CHARMM-like potential functions is chosen. Conversely, if the argument is **COSINE** a sinusoidal AMBER-like potential function is chosen.

EXAMPLES

I-TORSION HARMONIC

JORGENSEN

NAME

JORGENSEN – Allow Jorgensen-type interaction potentials.

SYNOPSIS

JORGENSEN

DESCRIPTION

If the system is composed of solute molecules of the same type it is sometime useful to use different interaction parameters for intermolecular and intramolecular interactions. The command **JORGENSEN** is designed to handle this situation. Sec. 8.3 describes how the inter and intra-molecular Lennard-Jones parameters are read by **ORAC** .

WARNINGS

Experimental - **Unsupported**

KEEP_BONDS

NAME

KEEP_BONDS – Constraints bond lengths to starting values.

SYNOPSIS

KEEP_BONDS

DESCRIPTION

This command should be specified when bond constraints are imposed to the system (see command **STRETCHING** and **CONSTRAINT** in this environment). If specified, all bonds to be constrained are constrained to the *initial* length found in the starting PDB file

DEFAULTS

KEEP_BONDS is *.FALSE.*

LJ-FUDGE

NAME

LJ-FUDGE – Set the fudge factor of the Lennard-Jones interaction

SYNOPSIS

LJ-FUDGE *lj-fudge*

DESCRIPTION

The argument to this command, *lj-fudge*, is the multiplicative factor of the 1-4 Lennard-Jones interaction.

EXAMPLES
LJ-FUDGE 0.5

DEFAULTS
LJ-FUDGE 1.0

LINKED_CELL

NAME
LINKED_CELL - Compute linked cell neighbor lists

SYNOPSIS
LINKED_CELL *ncx ncy ncz [nupdt]*

DESCRIPTION
The LINKED_CELL command switches to linked cell neighbor lists in place of conventional Verlet lists. The command can be used also for non orthogonal MD boxes. The integers *ncx ncy ncz* define the three dimensional grid by providing the number of bins along the *a, b, c* crystal axis, respectively. The optimum fineness of the cell grid depends on the density of the sample. For normal density a grid spacing of 3.0-3.5 Å along each axis is recommended. The Verlet neighbor list computation depends on N^2 where N is the number of particle in the system. The linked cell neighbor algorithm [121] scales linearly with N but it has a large prefactor. The break even point for the two methods is at about 7000 atoms for scalar machines. The frequency of updating of the index cell list is controlled by the argument *nupdt* and by the command UPDATE in this environment. If *fupdt* is the updating time specified in the command UPDATE the updating time for the linked list is *fupdt* × *nupdt*

EXAMPLES

```
&SETUP
...
CELL 54.0 72.0 41.0 90.0 102.0 90.0
...
&END
&POTENTIAL
...
LINKED_CELL 15 20 12 1
...
&END
```

Here a grid spacing of about 3.5 Å along each crystal axis is selected.

DEFAULTS
nupdt = 1

QQ-FUDGE

NAME
QQ-FUDGE – Set the fudge factor of the electrostatic interaction

SYNOPSIS
QQ-FUDGE *qq-fudge*

DESCRIPTION
The argument to this command, *qq-fudge*, is the multiplicative factor of the 1-4 electrostatic interaction.

EXAMPLES

QQ-FUDGE 0.5

DEFAULTS

QQ-FUDGE 1.0

SELECT_DIHEDRAL

NAME

SELECT_DIHEDRAL – Include only selected torsion angles in the potential.

SYNOPSIS

SELECT_DIHEDRAL

DESCRIPTION

In old force field only selected torsion angles were included. This command handles this situation.

DEFAULTS

The action taken by the command AUTO_DIHEDRAL is the default.

WARNINGS

Diagnostic - **Unsupported**

STEER_PATH

NAME

STEER_PATH – Steer along an arbitrary curvilinear coordinate

SYNOPSIS

STEER_PATH OPEN *filename*

DESCRIPTION

This command allows to do a MD simulation by steering the system along an arbitrary curvilinear path with arbitrary time protocol in a n -dimensional coordinate space (see Sec. 7.3). This curvilinear coordinate and time protocol can be given in terms of time dependent added stretching, bending and torsions external potentials to be specified in the file *filename*. The format of this file is shown in Table 7.3. Refer to Sec. 7.3 for more details.

STRETCHING

NAME

STRETCHING – Include stretching potentials

SYNOPSIS

STRETCHING [HEAVY]

DESCRIPTION

This command assigns a harmonic stretching potential (see Sec. 8.3.1) between covalently bonded atoms in the solute. Without argument, stretching potentials are assigned to all possible covalently bonded pairs. If the argument **HEAVY** is provided, bonds involving hydrogens are maintained rigid and only stretching potentials for bonded pairs involving non-hydrogen atoms are assigned.

DEFAULT

Constraints on all bonds is the default.

UPDATE

NAME

UPDATE – Assign parameters for the computation of the Verlet neighbor list

SYNOPSIS

UPDATE *fupdt* *rscut*

DESCRIPTION

ORAC computes Verlet neighbor lists the atomic groups of both the solvent and solute. There exist three different neighbor lists: a solvent–solvent, a solute–solute and a solvent–solute list. During the run, the calculation is carried out with a frequency equal to *fupdt* fs. All the group–group interactions within a radial cutoff of $r_{cut} + r_{scut}$ are included in the neighbor lists. The dimensions of the three lists are printed at run time. In the ORAC output **nnlww**, **nnlpp** and **nnlpw** refers to the solvent–solvent, solute–solute and solute–solvent neighbor list. The current version of ORAC can also use linked cell in place of the conventional Verlet neighbor list (see command **LINKED_CELL**).

EXAMPLES

UPDATE 65.0 1.4

Update the neighbor lists every 65.0 fs and use a cutoff of $r_{cut} + 1.4$ Å.

DEFAULTS

UPDATE 100.0 1.0

WARNINGS

The neighbor list cutoff must not be chosen larger than half of the simulation box size. The calculation of the neighbor list is performed by default. Only for solvent–only simulations, if the radial cutoff is equal to half of the box size, the force calculation is carried out without the use of neighbor list. When using r-RESPA the value of *rscut* is ignored in the UPDATE directive and is taken as an argument of the last **step nonbond** command in the **MTS_RESPA** structured command.

VERLET_LIST

NAME

VERLET_LIST - Compute Verlet neighbor list

SYNOPSIS

VERLET_LIST The conventional Verlet List computation is the default.

WARNINGS

Obsolete - **Unsupported**

8.2.7 &PROPERTIES

The &PROPERTIES directive is used to compute statistical properties on the fly or *a posteriori* once the trajectory file has been produced (see command DUMP (&INOUT) and &ANALYSIS environment.) ORAC can compute radial distribution functions, structure factors (GOFR), velocity autocorrelation functions (TIME_CORRELATIONS). The &PROPERTIES environment is still in the developing stage in the current version of ORAC. Thus, **none of the &PROPERTIES features is officially supported..** Some properties can no longer be computed on the fly in the current version and have to be computed using the &ANALYSIS environment once the trajectory file has been produced.

```
DEF_FRAGMENT DIST_FRAGMENT FORCE_FIELD GOFR, HBONDS PRINT_DIPOLE, STRUCTURES,
TIME_CORRELATIONS, VORONOI, WRITE_GYR
```

DEF_FRAGMENT

NAME

DEF_FRAGMENT – Define a fragment of a solute.

SYNOPSIS

```
DEF_FRAGMENT begin end
```

DESCRIPTION

This command is used in conjunction with the command PLOT FRAGMENT in &INOUT, or in conjunction with the command DIST_FRAGMENT in this environment. The arguments indicate the ordinal numbers of the first *begin* and the last *end* atom of a solute fragment. These numbers may be deduced by inspection of the PDB file *including* the hydrogens atoms (see command ASCII for generating a PDB file with hydrogens). The command DEF_FRAGMENT can appear more than one time in the environment. The atoms of different solute molecules defined with this command can overlap.

EXAMPLES

```
DEF_FRAGMENT 1 80
DEF_FRAGMENT 81 90
DEF_FRAGMENT 1001 1256
```

DIST_FRAGMENT

NAME

DIST_FRAGMENT – Print out distances between solute fragments.

SYNOPSIS

```
DIST_FRAGMENT ffragm OPEN filename
```

DESCRIPTION

Write the distances between the centroids of the fragments defined in the command DEF_FRAGMENT to the file *filename*. This command works only while retrieving the trajectory file by specifying the &ANALYSIS environment.

EXAMPLES

```
DIST_FRAGMENT 10.0 OPEN file_dist.frg
```

WARNINGS

This command has no action while running a simulation.

FORCE_FIELD

NAME

FORCE_FIELD – Print force field parameters

SYNOPSIS

FORCE_FIELD

WARNINGS

Work only in the production/simulation stage. It has no effect when reading the trajectory file.

GOFR

NAME

GOFR – Compute solvent and/or solute pair correlation function $g(r)$ and structure factors $S(k)$.

SYNOPSIS

GOFR

...

...

END

DESCRIPTION

The command **GOFR** opens an environment which includes a series of subcommands to define the parameters used in the calculation of the radial distribution functions.

average *favg*

Average the $g(r)$'s over length *favg* given in units of femtoseconds.

compute *fcomp*

Compute the $g(r)$'s with a frequency of *fcomp* femtoseconds.

cutoff *fcut*

Cut the calculation of the $g(r)$'s at distance equal to *fcut* Å.

delta *delrg*

Set the bin size of the $g(r)$'s to *delrg* Å.

intra

Include intramolecular contacts in solvent-solvent $g(r)$'s.

print *fconf* OPEN *filename*

$g(r)$'s are printed to the file *filename* every *fconf* fs.

use_neighbor

Use the neighbor list to compute the $g(r)$'s.

Radial distribution function can be computed on the fly.

EXAMPLES

GOFR

```
print 1000.0 OPEN test.gofr
```

```
use_neighbor
```

```
average 1000.0
```

```
compute 10.0
```

```
cutoff 12.0
```

```
delta 0.02
```

END

WARNINGS

When the the subcommand `use_neighbor` is used `cutoff` cannot exceed the neighbor lists cutoffs.

HBONDS

NAME

HBONDS – Compute solute H-bonds structural properties

SYNOPSIS

HBONDS

...

...

END

DESCRIPTION

The command HBONDS opens an environment which includes a series of subcommands which allow to compute hydrogen bond related properties. The hydrogen bond donor-acceptor pairs *must* be defined in the topological file. (see section 8.3). If these definition where not included when generating the trajectory file, and if READ_TPGPRM is specified in the &PARAMETERS environment, HBONDS produces no output. These definitions may be provided at analysis time by the READ_TPG (&PARAMETERS) directive. In the following we indicate with the letters *A* and *D* the donor and acceptor pair.

`angular_cutoff` *cutoff1* [*cutoff2*]

defines two angular cutoffs (in degrees) for $A...H - D$ and $A - D...H$, respectively. If only one argument is specified, the two cutoff are equal.

`histogram` *fbin*

define the bin size (in Å) for hydrogen bonds histograms.

`print` *nprint* OPEN *filename*

print hydrogen bond output to file *filename* every *nprint* configurations. The output format depends on READ_PDB (&SETUP) directive. If this directive is specified the output contains details concerning atomic types, hydrogen bond distances and angles.

`print_histo` *nprint* OPEN *filename*

print hydrogen histogram to file *filename* every *nprint* configurations

`radial_cutoff` *cutoff*

define the radial cutoff (in Å) for the hydrogen bond.

`residue`

printout hydrogen bonds per residues.

`total`

printout the total number of hydrogen bonds (the default)

`use_neighbor` *nconf* *rcut*

compute neighbor list for hydrogen bonds. *nconf* defines how frequently the neighbor list must be computed; *rcut* defines the radius of the neighbor list sphere.

EXAMPLES

HBONDS

total

residues

radial_cutoff 2.5

angular_cutoff 200.0 200.0

print 10 OPEN test.hbnd

histogram 0.1

```

    use_neighbors  5 5.0
    print_histo 2 OPEN test.hst
END

```

WARNINGS

`residue` and `total` are ineffective when `READ_PDB` is also specified. Experimental - **Unsupported**

PRINT_DIPOLE

NAME

PRINT_DIPOLE – Print out dipole.

SYNOPSIS

```
PRINT_DIPOLE    fdipole    OPEN filename
```

DESCRIPTION

Print out the components of the *total* instantaneous dipole **M** (in debye Å) of the basic cell each *fdipole* fs and the running average of the dielectric constant (relative permittivity).

EXAMPLES

PRINT_DIPOLE 10.5 OPEN dipole.out The file dipole.out looks like the following:

```

....
399115.500    0.50455E+02    -0.29885E+02    0.46023E+01    11.330
399126.000    0.48858E+02    -0.40479E+02    0.80527E+01    12.520
399136.500    0.52146E+02    -0.35302E+02    0.70597E+01    13.023
399147.000    0.57283E+02    -0.32666E+02    0.52314E+01    13.372
399157.500    0.62705E+02    -0.36044E+02    -0.76743E+01    13.913
....

```

In the first column, the current simulation time is reported. Column 2-4 contain the instantaneous values of the *x, y, z* component of the cell dipole (in Debye). In column 5 the *running average* of the dielectric constant is reported. The dielectric constant is computed under the assumption of thin-foil boundary conditions[122, 31] (i.e. no surface dipole term at the sphere boundary) using the formula $\epsilon = 1 + 4\pi(\langle \mathbf{M}^2 \rangle - \langle \mathbf{M} \rangle^2)/(3VRT)$ [122].

WARNINGS

Diagnostic - **Unsupported**

STRUCTURES

NAME

STRUCTURES – Compute the root mean square deviations from a given solute reference structure

SYNOPSIS

STRUCTURES

```

...
...
END

```

DESCRIPTION

The command **STRUCTURES** opens an environment which includes a series of subcommands which allow to compute average and instantaneous root mean square displacements (rms) *of the solute* for various atomic type (α -carbon, heavy atoms, backbone atoms etc.). The reference structure for the solute is entered with the command **TEMPLATE(&SETUP)**

averaged ca
compute average rms of α -carbons

averaged heavy
compute average rms of non hydrogen atoms

inst_xrms *type_1 type_2 ...*
specifies which instantaneous rms's have to computed. *type_n* can be any combination of the four keywords **ca heavy backbone allatoms**. The keyword **all** stands for all the the preceding keywords simultaneously. The **inst_xrms** keyword is mandatory when **print inst_xrms** is specified.

print *type nprint* **OPEN** *filename*
print rms's calculation as specified by *type* to file *filename* every *nprint* configurations (see also command **DUMP(&INOUT)**) The keyword *type* can be any of the following:

- averaged** - the full protein (solute) coordinates in pdb format are printed to the file *filename* with a constant orientation so that atomic rms's are minimized. Feeding directly the file to **rasmol** gives an pictorial view of the atomic displacements
- avg_xrms** - The time running averages of the rms's are printed, averaged over *alpha*-carbons, backbone atoms, and all heavy atoms.
- inst_xrms** - The instantaneous values of the rms's are printed averaged over *alpha*-carbons, backbone atoms, and all heavy atoms. If this subcommand is specified, along with **inst_xrms** subcommand **orac** also produces the file *filename.atm* which contains the final values of the atomic rms's for the atom types (*alpha* carbon, backbone atoms etc.) specified in the command **inst_xrms**.

EXAMPLES

STRUCTURES

```
print averaged 2 OPEN test.str
print avg_xrms 3 OPEN test.arms
print inst_xrms 3 OPEN test.irms
inst_xrms ca backbone
averaged ca
print rms 2 OPEN test.rms
END
```

WARNINGS

STRUCTURES commands works only in conjunction with the **&ANALYSIS** environment. Experimental - **Unsupported**

TIME_CORRELATIONS

NAME

TIME_CORRELATIONS – Compute velocity autocorrelation functions and root mean displacements.

SYNOPSIS

TIME_CORRELATIONS

```
...
...
END
```

DESCRIPTION

The command **TIME_CORRELATIONS** opens an environment which includes a series of subcommands to define the parameters used in the calculation.

```
diffusion    OPEN    filename
compute the mean square displacements  $|r(t) - r(0)|^2$ 

divide_step  nspline
provide a number equal to nspline of interpolated points between data points.

vacf         OPEN    filename
Compute velocity autocorrelation functions and print out results to file filename
```

EXAMPLES

TIME_CORRELATIONS

```
vacf OPEN vacf.test2
divide_step 2
diffusion OPEN diff.test2
END
```

WARNINGS

When the the subcommand `use_neighbor` is used `cutoff` cannot exceed the neighbor lists cutoffs.
 Experimental - **Unsupported**

VORONOI

NAME

VORONOI – Compute the Voronoi polihedra of atoms, residues and molecules

SYNOPSIS

VORONOI

```
...
...
END
```

DESCRIPTION

The command VORONOI opens an environment which includes a series of subcommands which allow to compute average and instantaneous properties related to the Voronoi polihedra of the solute and of the solvent.

compute accessibility

Compute the area of the Voronoi polihedron for all residues of the solute (computed as the sum of the voronoi volumes of the individual atoms) and evaluate for each residue the fraction of the surface that is accessible to the solvent (solute and solvent as defined in the command JOIN(&PARAMETERS))

compute contact_solute *int1 int2*

Compute the contact surface among selected solute residues with residue number *int1* and *int2* as in the the PDB file.

compute neighbors

Compute the Voronoi coordination number relative to the whole solute-solute, solvent-solvent and solute-solvent contacts.

compute volume

Compute the Voronoi volumes of all residues in the solute.

cutoff *value*

values the cutoff (Å) for

heavy_atoms

Use only non hydrogen atoms for evaluating Voronoi polihedra

```
print nprint OPEN filename
Print all output as to file filename every nprint configurations
```

EXAMPLES

VORONOI

```
print 5 OPEN 6.vor
cutoff 8.0
heavy_atoms
compute contact_solute 1 2
compute contact_solute 5 6
compute volume
compute neighbors
compute accessibility
END
```

In this example we compute the voronoi volumes, areas and accessibility and neighbors for the residues of a proteins every 5 configurations. Also the contact surfaces between residues 1 and 2 and residue 5 and 6 are evaluated. All output are printed to the file 6.vor.

WARNINGS

VORONOI commands works only in conjunction with the &ANALYSIS environment. Experimental - **Unsupported**

WRITE_GYR

NAME

WRITE_GYR – Print gyration radius.

SYNOPSIS

```
WRITE_GYR ngyr OPEN filename
```

EXAMPLES

```
WRITE_GYR 10.0 OPEN test.gyr
```

WARNINGS

Work only at single time step. Experimental - **Unsupported**

8.2.8 &REM

Define run time parameters concerning Replica Exchange Simulation. Work only with the parallel version (see Chapter 9), i.e. this namelist is not recognized when the serial program is compiled. The following commands are allowed:

PRINT, REMSTEP, SEGMENT, SETUP

PRINT

NAME

PRINT – print out info on REM.

SYNOPSIS

PRINT *iprint*

DESCRIPTION

Controls intermediate printing of the acceptance ratio between adjacent replicas.

EXAMPLES

PRINT 1000

Print info on the current acceptance ratios every 1000 attempted replica exchanges.

DEFAULTS

PRINT 0

REMSTEP

NAME

REMSTEP – exchange time for REM

SYNOPSIS

REMSTEP *rtime*

DESCRIPTION

Define the time (in fs) for attempting an exchange between adjacent replicas.

EXAMPLES

REMSTEP 5.0

Attempt replica exchanges every 5 fs.

DEFAULTS

REMSTEP 0

WARNINGS

If REMSTEP is not set, *rtime* is set to the time step of the *m*-th intermolecular shell,

SEGMENT

NAME

SEGMENT – Define the “solute” in solute tempering simulations.

SYNOPSIS

```
SEGMENT
```

```
....
```

```
END
```

DESCRIPTION

This structured command is used to define the “solute” in the a solute/tempering REM simulation and to assign the scaling factors for the Hamiltonian REM simulation (see Sec. 5.2) to the intrasolute, solute-solvent and solvent-solvent interactions. The following subcommands may be specified within SEGMENT:

define, kind

- **define** *n1* *n2*

The **define** command is used to crop a piece of solute for Hamiltonian scaling in a REMD simulation. One can use up to a maximum of 10 *define* commands, cropping 10 disconnected (non overlapping) part of the solute. *n1* and *n2* are the atom indices of the selected solute parts, The numeric order of the atoms is that specified in the topology file (see Sec. 8.3).

- **kind** *inter_type*

Once the “solute” has been defined using the **define** subcommand, the subcommand **kind** is used to scale the solute-solute, solute-solvent interactions. Possible choices for the string *inter_type* are **intra** and **inter**. **intra** means that the non-bonded energy scaling (see **SETUP** command) is applied to the *intrasolute* non-bonded interactions only, i.e. solute-solvent interactions are not scaled where by “solvent” we mean the actual solvent and the solute atoms which were not selected using the **define** subcommand. *inter* scales only solute-non solute (i.e. solvent) non bonded interactions. Intrasolute interactions are NOT scaled if **inter** is specified. If the subcommand **kind** is not specified, the ORAC assumes that *both* solute-solvent and solute-solute interactions are scaled.

EXAMPLES

SEGMENT

```
define 1 10
define 1300 1325
kind inter
END
```

SETUP

NAME

SETUP – Define the scaling in a REM simulation.

SYNOPSIS

```
SETUP scale1 [ scale2    scale3 ] irest
```

DESCRIPTION

The **SETUP** command is used to define the lowest scaling factor(s) (i.e the highest temperature) of the last replica. The number of replicas in the REMD simulations are equal to the number of processors passed to the MPI routines (nprocs). The spacing between the replicas is controlled by the *irest* integer. If only the *scale1* real parameter is specified, an equal scaling is applied to all parts of the potential. If the three parameters *scale1*, *scale2*, *scale3* are specified, then *scale1* refers to the bending, stretching and improper torsional potential, *scale2* to the (proper) torsional potential and to the 14

non-bonded interactions and finally *scale3* refers to the non bonded potential. NB: when the Ewald summation is used *scale3* scales only the *direct* (short-ranged) part of the electrostatic interactions and the (long-ranged) reciprocal part has a scaling factor of 1.0 (i.e. these interaction are not scaled).

If *irest*=0, the run is restarted from a previous one. This implies that the directories PARXXXX are present and are equal in number to *nprocs* as specified in the `mpiexec/mpiexec` command.

If *irest*≠ 0 then the run refers to a cold start from scratch and

- if *irest*= 1, then the scaling factors of the intermediate replicas are derived according to a geometric progression, namely $scale_i_m = scale_i^{m/(nprocs-1)}$, where *scale_i*_{*m*} is the scaling factor for the potential *i* of the replica *m* with $0 \leq m \leq nprocs - 1$. For example, if *scale_i*= 0.6 and *nprocs*=4, then replica *m* = 0 has *scale_i*₀=1, replica *m* = 1 has *scale_i*₂= 0.843433 ,replica *m* = 2 has *scale_i*₂= 0.711379 and the replica *m* = 3 *scale₃*=*scale_i*=0.6.
- if *irest*= 2, the scaling factors are read from an auxiliary file called “REM.set” that must be present in the directory from which the program is launched using the `mpiexec/mpiexec` command. This ASCII file has as many lines as parallel processes and on each line the three (or one) scale factors must be specified.

EXAMPLES

SETUP 1.0 1.0 0.6 1

Scales only the non bonded potential (direct part) using a geometric progression.

DEFAULTS

SETUP 1.0 1.0 1.0 1

8.2.9 &RUN

Define run time parameters which concern output printing and run averages. The following commands are allowed:

CONTROL, DEBUG, OPTION, MAXRUN, PRINT, PROPERTY, REJECT, STEER TIME

CONTROL

NAME

CONTROL – Indicate initial conditions

SYNOPSIS

CONTROL *icontrol*

DESCRIPTION

ORAC can run simulations or minimization reading the system initial momenta and/or coordinates from different sources. If the integer argument *icontrol* is zero, the simulation run must commence from coordinates either stored entirely in a PDB file or generated by ORAC itself from some initial configuration (see CELL or SPACE_GROUP in &SETUP). CONTROL 0 implies that all system momenta are initialized from the Boltzmann distribution at the wanted temperature. When CONTROL 2, the run is started from the restart file defined by the command SAVE or RESTART in &INOUT. With CONTROL 2 all system averages are zeroed. The same action is taken if CONTROL 1, but the averages are **not** initialized to zero.

EXAMPLES

CONTROL 2

Run a simulation from a restart file and set all averages to zero.

DEFAULTS

CONTROL 0

WARNINGS

When restarting a run with a different integration scheme from the one used in the restart file, CONTROL should be set to 2. If not, unpredictable behavior may occur.

DEBUG

NAME

DEBUG – Print debug information

SYNOPSIS

DEBUG **all**

DEBUG *debug_type*

DESCRIPTION

Print various arrays to the standard output for debugging. Information regarding the solute topology and force field is written only if the solute topology and parameter list is actually computed and **not** read from a binary file (i.e. READ_TPGPRM_BIN in &PARAMETERS must be inactive). The *debug_type* string can be

residue_sequence

The residue sequence is printed.

bond_table

Details about bonds and corresponding stretching parameters are printed.

bend_table

Details about bending and corresponding parameters are printed.

ptors_table

Details about proper torsions and corresponding parameters are printed.

pitors_table

Details about improper torsions and corresponding parameters are printed.

EXAMPLES

DEBUG all

print out all tables.

DEBUG bond_table

DEBUG bend_table

DEBUG residue_sequence

print out the bond and bending table and the residue sequence.

MAXRUN**NAME**

MAXRUN – Provide the maximum simulation length (in fs)

SYNOPSIS

MAXRUN *fmaxrun*

DESCRIPTION

This command controls the total length of the direct access file. The number of records initialized by the **DUMP(&INOUT)** command is given by $nrec = fmaxrun * natoms / atom_rec$ where *natoms* and *atom_rec* are the total number of atoms in the system and the atoms per record, respectively. *fmaxrun* cannot be less than *ftime* (see command **TIME** in this environment).

EXAMPLES

MAXRUN 500000.0

PRINT**NAME**

PRINT – Print instantaneous results

SYNOPSIS

PRINT *fprint*

DESCRIPTION

ORAC writes the instantaneous energies of the system to standard output. The real argument *fprint* indicates the chosen printing frequency in fs.

EXAMPLES

PRINT 5.0

PROPERTY

NAME

PROPERTY – Print averages with a given frequency

SYNOPSIS

PROPERTY *fprop*

DESCRIPTION

ORAC writes to the standard output the running averages of the current run. The real argument *fprop* is the frequency of printing in femtoseconds.

EXAMPLES

PROPERTY 500.0

Write averages every 500.0 fs.

WARNINGS

- An error condition will occur if this command is not included in the input to ORAC or if the argument, *fprop* is zero.
- The command is not active only in the rejection phase (see command REJECT).

REJECT

NAME

REJECT – Provide the length of the rejection phase

SYNOPSIS

REJECT *freject*

DESCRIPTION

During the equilibration or rejection phase only instantaneous results are printed, while averages are discarded. The real argument *freject* indicates the time lag, in femtoseconds, of the rejection phase.

EXAMPLES

REJECT 1000.0

Does not accumulate averages during the initial 1000.0 fs of the run.

WARNINGS

This command is inactive during a minimization run (see command MDRUN in &SIMULATION).

STEER

NAME

STEER – Provide the starting and final time (in fs) for a steered molecular dynamics. The time dependent harmonic potential is defined in the namelist &POTENTIAL using the command ADD_STR_BONDS, ADD_STR_BENDS, ADD_STR_TORS.

SYNOPSIS

STEER *tiniz* *tfinal*

STEER **temperature** *temp0* *tempt* *tiniz* *tfinal*

DESCRIPTION

Steer the system with the time dependent mechanical potential defined in the namelist &POTENTIAL, starting the SMD at time *tiniz* and ending at time *tfinal*. This command can also be used to gradually change the temperature in conjunction with the command PLOT STEER_TEMPERATURE(&INOUT) where

the adimensional thermal work done on the thermostat is printed at regular time intervals (see the PLOT (&INOUT) command). Steered molecular dynamics can be automatically restarted. In order to do this, one sets once for all the steering time (*t_{final}*) to the desired value, updating, at each restart, *only* the simulation time given by the TIME(&RUN) directive.

EXAMPLES

```
&RUN
...
STEER 1000.0 10000.0
..
&END
&INOUT
..
PLOT STEER_ANALYTIC 100.0 OPEN WRK.out
..
&END
```

Start to apply the time dependent potential (see Eq. 7.13) at 1 ps and switch it off at 11 ps. Print out the accumulated work every 100 fs to the file WRK.out. The accumulated work is calculated according to Eq. 7.16.

```
&RUN
...
STEER temperature 300. 1500. 1000.0 11000.0
..
&END
&INOUT
..
PLOT STEER_TEMPERATURE 100.0 OPEN WRKTEMP.out
..
&END
```

rise the temperature from 300 to 1500 K starting at 1 ps and ending at 11 ps, with the constant speed of 120 K /ps. The thermal work is printed every 100 fs to the file WRKTEMP.out.

```
&RUN
CONTROL 1
STEER 0.0 18000.0
TIME 10000.0
..
&END
&INOUT
..
PLOT STEER_ANALYTIC 100.0 OPEN WRK.out
..
&END
..
&INOUT
  RESTART
    read file.rst
    write 30.0 OPEN new.rst
  END
&END
```

In this example, the simulation starts from the restart file `file.rst` and goes from the time found in that file to 10000.0 fs. The total steering time is 18000.0 fs. In the next restarted run the configuration of the system at $t = 10000$ fs is found in the file `new.rst`. The next restarted simulation could be thus of the kind:

```
&RUN
  CONTROL 1
  STEER 0.0 18000.0
  TIME 18000.0
  ..
&END
&INOUT
  ..
  PLOT STEER_ANALYTIC 100.0 OPEN WRK_10000_18000.out
  ..
&END
  ..
&INOUT
  RESTART
    read new.rst
  END
&END
```

In this example, the steering is complete and in the output file `WRK_10000_18000.out`, the work is calculated from $t = 10000$ fs to $t = 18000$ fs.

TIME

NAME

TIME – Length of the simulation not including the rejection phase

SYNOPSIS

TIME *ftime*

DESCRIPTION

This command gives the length of the acquisition run which is to be carried out after the rejection (equilibration) phase. The unit of its real argument *ftime* is femtoseconds. During the acquisition run averages are accumulated.

EXAMPLES

TIME 100000.0

8.2.10 &SETUP

The environment &SETUP includes commands concerned with the simulation box setup. In this environment, the simulation cell parameters, dimensions and symmetry can be initialized. Moreover, files containing the system coordinates in appropriate format can be provided. The following commands are incorporated in &SETUP:

CRYSTAL, READ_PDB, RECONSTRUCT, REPLICATE, TEMPLATE

CHANGE_CELL

NAME

CHANGE_CELL – Recomputes the atomic coordinates according to input.

SYNOPSIS

CHANGE_CELL

DESCRIPTION

This command has an effect only when the run is restarted (see commands RESTART(&INOUT) and CONTROL(&RUN)). This command must be specified, in case one wishes to change the MD cell parameters, with respect to those dumped in the available restart file, to those specified in the CRYSTAL directive in this environment. If CHANGE_CELL is not specified and the run is restarted, the CRYSTAL directive is ignored and the cell parameters are taken from the last configuration of the restart file. If CONTROL 0 is entered in the environment &RUN, this command has no effect.

EXAMPLES

CHANGE_CELL

CRYSTAL

NAME

CRYSTAL – Read the cell parameters defining the shape of the simulation box

SYNOPSIS

CRYSTAL *a* [*b* *c* [*α* *β* *γ*]]

DESCRIPTION

The arguments α , β and γ to this command are defined using the usual crystallographic conventions: α is the angle between the *b* and *c* axis, β is the angle between *a* and *c*, and γ is the angle between *a* and *b*.

EXAMPLES

CRYSTAL 12.3 14.5 12.3 90.0 95.0 90.0

CRYSTAL 15.0

DEFAULTS

$\alpha = \beta = \gamma = 90.0$

READ_PDB**NAME**

READ_PDB – Read input system coordinates from a PDB file

SYNOPSIS

READ_PDB *filename*

DESCRIPTION

This command indicates the name of a file in the protein data bank format which contains the solute and/or solvent coordinates. The name of this file, *filename*, must be provided. The coordinates of the solvent molecules, if present, must follow those of the solute in the PDB file. The atom labels for solute and/or solvent must correspond with those defined in the topology file (see description in Sec. 8.3). The order of the atoms, within a solute residue or a solvent molecule, specified in the PDB file is unimportant (the “ORAC order” corresponds to that specified in the topology file). If the system contains hydrogens, the PDB file ought not to include the hydrogens coordinates. If hydrogens atoms are not present in the PDB file, but they are included in the topological specification of residue or solvent, their coordinates are generated by ORAC according to geometry considerations.

EXAMPLES

READ_PDB test.pdb

WARNINGS

This command has no action if CONTROL in &RUN is different from zero, i.e. if the system coordinates are read from a restart file (see RESTART in &INOUT).

REPLICATE**NAME**

REPLICATE – Replicate the unit cell generated by SPACE_GROUP(&SOLUTE)

SYNOPSIS

REPLICATE *icl icm icn*

DESCRIPTION

The integer arguments *icl*, *icm*, *icn* indicate how many times along the three axis the unit cell must be replicated. The cell parameters of the replicated structure are input to the command CRYSTAL.

EXAMPLES

REPLICATE 4 4 5

Replicate the unit cell 4, 4 and 5 times along the *a*, *b* and *c* crystal axis, respectively.

WARNINGS

This command has no action if CONTROL in &RUN is different from zero, i.e. if the system coordinates are read from a restart file (see RESTART in &INOUT).

RESET_CM**NAME**

RESET_CM – Reset to zero the position of the center of mass of the solute atoms.

SYNOPSIS

RESET_CM

DESCRIPTION

This command is active only if the solute coordinates are read from a PDB file. Before the run starts RESET_CM set the center of mass of the solute to zero.

READ_CO

NAME

READ_CO – Read Crystal to Orthogonal (CO) matrix.

SYNOPSIS

```

READ_CO
  ax  bx  cx
  ax  by  cy
  ax  by  cz
END

```

DESCRIPTION

This command is active only the simulation is restarted and overwrites the CO matrix retrieved from the restart file.

SOLUTE

NAME

SOLUTE – assume solute

SYNOPSIS

SOLUTE [ON] SOLUTE [OFF]

DESCRIPTION

This command is active only if the solute coordinates are read from a PDB file. If ON is specified, ORAC assumes that a solute is present and its coordinates are read in from the file PDB specified by the directive READ_PDB in this environment. When SOLUTE [ON] is specified, the namelist &SOLUTE may be omitted. When SOLUTE OFF is specified, the namelist &SOLUTE *must* be omitted.

EXAMPLES

```

&SETUP
  READ_PDB solute.pdb
  SOLUTE ON
&END

```

A solute is present and the coordinates are read in from the file PDB. The “residue” sequence found in the PDB must match that given in the JOIN SOLUTE (&PARAMETERS) directive. If the environment &SOLUTE is entered, solute is assumed anyway, this command has no effect.

SOLVENT

NAME

SOLVENT – Reset to zero the position of the center of mass of the solvent atoms.

SYNOPSIS

SOLVENT [ON] SOLVENT [OFF]

DESCRIPTION

If ON is specified, ORAC assumes that a solvent is present and its coordinates are read in from the file PDB specified by the directive READ_PDB in this environment. This command is not mandatory as, if the solvent is present the environment &SOLVENT (which has the same effect of SOLVENT [ON]) must be entered anyway in order to specify how to generate the solvent or the number of solvent molecules in the PDB file. When SOLVENT OFF is specified, the namelist &SOLVENT *must* be omitted.

EXAMPLES

```

&SETUP
  READ_PDB solvent.pdb
  SOLVENT ON
&END
...
&SOLVENT
  ADD_UNITS 432
&END
...
&PARAMETERS
  ...
  JOIN SOLVENT
    hoh
  END
&END
...

```

A solvent (432 molecules) is present and the coordinates are read in from the file PDB. The “residue” sequence for the solvent found in the PDB must match that given in the JOIN SOLVENT (&PARAMETERS) directive. If a solute is also present and its coordinates are given in the PDB file specified by the READ_PDB command, then the coordinates of the solvent molecules *must* follow those of the solute in the PDB file. An example is the following

```

&SETUP
  READ_PDB solute+solvent.pdb
  SOLVENT ON
  SOLUTE ON
&END
...
&SOLVENT
  ADD_UNITS 432
&END
...
&PARAMETERS
  ...
  JOIN SOLVENT
    hoh
  END
  JOIN SOLUTE
    ala-h ala ala ala-o
  END
&END
...

```

 TEMPLATE

NAME

TEMPLATE – Define a template or reference structure

SYNOPSIS

TEMPLATE *filename*

DESCRIPTION

This command defines a template PDB file *filename* which contains reference solute coordinates used during run time analysis for computing root mean square displacements (see command **X_RMS** in **&PROPERTIES** for instance).

EXAMPLES

TEMPLATE test_template.pdb

8.2.11 &SIMULATION

The environment includes commands which define the type of simulation that is to be carried out. In particular, commands are available to run steepest descent energy minimizations, and molecular dynamics simulations in various ensembles. The environment &SIMULATION allows the following commands:

ANDERSEN ANNEALING ISEED, ISOSTRESS, MINIMIZE, MDSIM, SCALE, STRESS, TEMPERATURE,
WRITE_PRESSURE

ANDERSEN

NAME

ANDERSEN – The simulation is performed in the NVT Ensemble using the stochastic collision method by Andersen.

SYNOPSIS

ANDERSEN *time*

DESCRIPTION

Implement Andersen thermostat with a period for random collision of *time* femtoseconds.

EXAMPLES

ANDERSEN 1000.0

WARNINGS

Diagnostic - **Unsupported**

ANNEALING

NAME

ANNEALING – Velocities are multiplied by factor to speed up

SYNOPSIS

ANNEALING *scalefactor*

DESCRIPTION

Velocities are multiplied by *scalefactor*.

EXAMPLES

ANNEALING 2.0

WARNINGS

Diagnostic - **Unsupported**

ISEED

NAME

ISEED – Provide a seed for the random number generator.

SYNOPSIS

ISEED *seed*

EXAMPLES

ISEED 34567

DEFAULTS

ISEED 12345667

WARNINGS

Diagnostic - **Unsupported**

ISOSTRESS

NAME

ISOSTRESS – Run MD simulations at constant pressure with an isotropic volume variable

SYNOPSIS

ISOSTRESS [PRESS-EXT *pext*] [BARO-MASS *wpr*] [COMPR *compressibility*]

DESCRIPTION

This command allows to run simulations and minimizations at a given pressure with isotropic volume changes. If the command is used alone ORAC runs simulations in the NPH ensemble. Simulations in the NPT ensemble can instead be carried out if ISOSTRESS is used in conjunction with the command THERMOS. The external pressure in MPa is read in by the keyword PRESS-EXT. Also, the keyword BARO-MASS expects the mass of the barostat in cm^{-1} . [67] The system compressibility in MPa^{-1} is read in as input to keyword compr. According to the relation given in Ref. [67] compressibility and frequency should be consistent. If compr is not specified the default value is used.

EXAMPLES

ISOSTRESS PRESS-EXT 0.1 BARO-MASS 10.0 COMPR 5.3e-4

Run a simulation at pressure 0.1 MPA (atmospheric pressure) with a barostat mass corresponding to 10.0 cm^{-1} . The compressibility is set to $5.3 \times 10^{-4} \text{ MPa}^{-1}$.

DEFAULTS

ORAC uses the water compressibility at 300 K (i.e. $5.3 \times 10^{-4} \text{ MPa}^{-1}$) as the default compressibility.

WARNINGS

- 1 ORAC can carry out constant pressure runs with isotropic volume changes only for orthogonal cells.
- 2 Make sure that when simulations at constant pressure are run ORAC has been compiled with the appropriate PRESSURE option in the config.h file (see Chapter 9)

FREQUENCIES

NAME

FREQUENCIES – Compute harmonic frequencies of the system. All atoms (solute and solvent) are included in the dynamical computation.

SYNOPSIS

FREQUENCIES

...

END

DESCRIPTION

The following subcommands may be specified within FREQUENCIES:

dist_max, no_step, print

- **dist_max** *hdist*
The differential increment (in Å) for numerical computation of the dynamical matrix. The default is 0.03 Å, which is OK for most systems and force fields.
- **no_step** *steps*
Order of Chebyshev polynomial for numerical computation of the dynamical matrix. The default is 6 which is OK for most systems and force fields
- **print OPEN** *filename*
Write frequencies and eigenvectors to file filename. If not specified frequencies are written to the main output file.

EXAMPLES

FREQUENCIES

```
print OPEN myfreq.out
END
```

MINIMIZE

NAME

MINIMIZE – Run steepest descent-like or conjugate gradient minimization at constant volume or at a given pressure

SYNOPSIS

MINIMIZE

...

END

DESCRIPTION

Run energy minimization using a method of choice (steepest descent or conjugate gradient). After minimization is done, the dynamical matrix is computed and diagonalized and the normal frequencies are listed along with eigenvectors. The following subcommands may be specified within MINIMIZE:

CG, SD, WRITE_GRADIENT

- **GC** *eps_energy*
Use conjugate gradient with energy tolerance *eps_energy*.
- **SD** *eps_energy*
Use steepest descent with energy tolerance *eps_energy*.
- **WRITE_GRADIENT**
Write final gradient at each atom.

EXAMPLES

MINIMIZE

```
CG 0.00001
WRITE_GRADIENT
END
```

MDSIM**NAME**

MDSIM – Run molecular dynamics simulations

SYNOPSIS

MDSIM

DESCRIPTION

Use this command to run molecular dynamics simulation in any ensembles. It has no argument.

DEFAULTS

MDSIM is the default.

SCALE**NAME**

SCALE – Periodic temperature scaling

SYNOPSIS

SCALE *fscale*

DESCRIPTION

Use this command for periodically re-scale the temperature with frequency *fscale* in units of femtoseconds. Scaling stands here for random initialization of the system velocities at temperature *temp* according to a Gaussian distribution.

EXAMPLES

SCALE 100.0

Reinitialize the system velocities every 100 fs.

WARNINGS

Work only during the rejection phase (see **REJECT** in environment **&RUN**).

SCALING**NAME**

SCALING – Choose scaling methods for constant pressure simulations

SYNOPSIS

SCALING MOLECULAR

SCALING GROUP

SCALING ATOMIC

DESCRIPTION

This command allows you to switch between scaling methods when running with a barostat (see **STRESS** and **ISOSTRESS** directive in this environment. The scaling can be i) “molecular”, with the barostat coupled to the center of mass of the molecules in the system; ii) “atomic”, and the barostat is coupled to the coordinates of the atoms; iii) “group”, with the barostat coupled to the smallest groups which are not connected by a constraint. If no constraints have been imposed to system (see **STRETCHING(&POTENTIAL)**) **SCALING GROUP** and **SCALING ATOMIC** have the same behavior.

EXAMPLES

SCALING MOLECULAR

Run with molecular scaling. **SCALING GROUP**

Run with group scaling.

STRESS

NAME

STRESS – Run MD simulations at constant pressure with a non-isotropic volume changes

SYNOPSIS

STRESS [PRESS-EXT *pext*] [BARO-MASS *wpr*] [COMPR *compressibility*]

DESCRIPTION

This command allows to run simulations and minimizations at a given pressure with non-isotropic volume changes according to the Parrinello-Rahman equation of motion. If the command is used alone ORAC runs simulations in the NPH ensemble. Simulations in the NPT ensemble can instead be carried out if STRESS is used in conjunction with the command THERMOS. The external pressure in MPa is read in by the keyword PRESS-EXT. Also, the keyword BARO-MASS expects the mass of the barostat in cm^{-1} (see ISOSTRESS). The system compressibility in MPa^{-1} is read in as input to keyword COMPR (see ISOSTRESS)

EXAMPLES

&SIMULATION

MDSIM

TEMPERATURE 300.0 25.0

STRESS PRESS-EXT 0.1 BARO-MASS 10.0 COMPR 1.0e-4

&END

Run a simulation in the NHP ensemble at pressure 0.1 MPa (atmospheric pressure) with a barostat mass corresponding to 10.0 cm^{-1} . The compressibility is set to $1.0 \times 10^{-4} \text{ MPa}^{-1}$. Velocities are initialized and (optionally scaled) according to a temperature of 300 K.

&SIMULATION

MDSIM

TEMPERATURE 300.0 25.0

STRESS PRESS-EXT 0.1 BARO-MASS 10.0 COMPR 1.0e-4

THERMOS

...

END

&END

Same as before but with a Nosé thermostat. The simulation is hence in the NPT ensemble with $T=300 \text{ K}$.

DEFAULTS

ORAC uses the water compressibility at 300 K (i.e. $5.3 \times 10^{-4} \text{ MPa}^{-1}$) as the default compressibility.

WARNINGS

Make sure that when simulations at constant pressure are run ORAC has been compiled with the appropriate PRESSURE option in the `config.h` file (see Chapter 9)

TEMPERATURE

NAME

TEMPERATURE – Set the system temperature for the run

SYNOPSIS

TEMPERATURE *temp* *dt*

DESCRIPTION

The argument *temp* is the target temperature for the simulation run. *dt* is used only during the rejection phase (see command **REJECT** of environment **&RUN**) and indicates the temperature window in Kelvin outside which temperature scaling occurs. Scaling stands here for random initialization of the system velocities at temperature *temp* according to a Gaussian distribution. System scaling in rejection phase occurs also during constant temperature simulations (see command **THERMOS** in **&SIMULATION**).

EXAMPLES

TEMPERATURE 300.0 50.0

WARNINGS

Work only during the rejection phase (see **REJECT** in environment **&RUN**).

THERMOS

NAME

THERMOS – Run with Nosé thermostats for NVT or NPT simulations.

SYNOPSIS

THERMOS

...

END

DESCRIPTION

For a faster and better energy equipartition, **ORAC** uses three thermostats. The first, coupled to the center of mass momenta of all molecules in the system, the second coupled to the momenta of the atoms of the solute (if present) and the third coupled to the momenta of solvent atoms (if present). The following subcommands may be specified within **THERMOS**:

cofm, defaults, solute, solvent, temp_limit

- **cofm** *freq_mass*
Specify the mass of the barostat coupled to the centers of mass of the molecules. This mass is also assigned to the barostat coupled to the box momenta in NPT simulation, in case **STRESS** or **ISOSTRESS** have been specified. Actually, what is entered with the variable *freq_mass* is the (approximate) frequency of oscillation of the thermostat. The actual “mass” *W* (in units of mass times a length to the power of two) of the barostat may be recovered according to the relation $freq_mass = (2Nk_B T/W)^{1/2}$. [67]
- **defaults**
Use defaults value for “mass” variables. The defaults are $freq_mass_solute = freq_mass_solvent$ $freq_mass = 30.0$.
- **solute** *freq_mass_solute*
Specify mass (units of cm^{-1}) of the barostat coupled to the momenta of the solute atoms.
- **solvent** *freq_mass_solvent*
Specify mass (units of cm^{-1}) of the barostat coupled to the momenta of the solvent atoms.
- **temp_limit** *maxtemp*
Specify maximum temperature allowed for all Nosé thermostat when the argument of the command **REJECT(&RUN)** is different from zero. In principle, for a system out of equilibrium, no temperature scaling should be enforced when using Nosé thermostating. Actually, when equilibrating systems in the NVT or NPT ensembles, it is strongly recommended to specify the subcommand **temp_limit** along with a rejection time **REJECT(&RUN)** as normally done for conventional scaling in NVE dynamics. In a NV(P)T system out of equilibrium, while the temperature of the system remains close to the selected temperature, the temperature of the thermostat coordinates (which are not themselves thermostatted) may raise dramatically, if not scaled.

EXAMPLES

```
&SIMULATION
  TEMPERATURE 300.0 25.0
  MDSIM
  THERMOS
    cofm 30.0
    solute 30.0
    solvent 30.0
  END
&END
```

Run a simulation in the NVT ensemble at $T = 300$ K.

WRITE_PRESSURE**NAME**

WRITE_PRESSURE – Write the pressure of the system during a simulation

SYNOPSIS

WRITE_PRESSURE

DESCRIPTION

This command is used to print the system pressure and stress tensor to the simulation output. It has no argument.

WARNINGS

Make sure that when simulations at constant pressure are run ORAC has been compiled with the appropriate **PRESSURE** option in the `config.h` file (see Chapter 9)

8.2.12 &SOLUTE

The &SOLUTE environment includes commands which are concerned with specific aspects of the solute force field and structure. The following commands are allowed:

COORDINATES, DEF_SOLUTE, SCALE_CHARGES, SPACE_GROUP

COORDINATES

NAME

COORDINATES – Define the coordinates of a solute.

SYNOPSIS

COORDINATES *filename*

DESCRIPTION

Read the coordinates of the solute (in PDB format) from file *filename*. This command is best used when also the solvent atoms must be read in.

EXAMPLES

&SETUP

CRYSTAL 20.00 20.00 20.00 90.0 90.0 90.0

REPLICATE 2 2 2

&END

&SOLUTE

COORDINATES solute.pdb

SPACE_GROUP OPEN benz.group P 2/c

&END

&SOLVENT

CELL SC

INSERT 1.5

COORDINATES solvent.pdb

GENERATE RANDOMIZE 4 4 4

GENERATE RANDOMIZE 8 8 8

&END

In this example the coordinates of the solute are read in from the file `solute.pdb` while the coordinates of the solvent molecule (see &SOLVENT) are read in from the file `solvent.pdb`. As is now, this input would produce in a box of $20 \times 20 \times 20 \text{ \AA}^3$, 1 solute along with 64 replicas (see command GENERATE(&SOLVENT) of the solvent molecule. Of this 64 molecule, those that overlap with the solute molecule (see command INSERT(&SOLVENT)) are discarded. If the second line in the environment &SOLUTE is uncommented, the solute is assumed to be arranged in the MD box according to the space group specified by the SPACE_GROUP directive. In the present example the group contains 4 molecules per unit cell. So 4 molecules of solute are arranged in the box according to the P 2/c space group along with 64 replicas of solvent molecules. Again the overlapping solvent molecules (say n_o) will be discarded. If we comment the line GENERATE RANDOMIZE 4 4 4 and uncomment the lines # GENERATE RANDOMIZE 8 8 8 and # REPLICATE 2 2 2 we double the size of the sample: we will have 8 cell of $20 \times 20 \times 20 \text{ \AA}^3$ each with 4 molecules of solute and $8 \times 8 \times 8 = 512$ solvent molecules minus $8 \times n_o$ overlapping molecules.

DEF_SOLUTE

NAME

DEF_SOLUTE – Define a solute molecule

SYNOPSIS

DEF_SOLUTE *begin end*

DESCRIPTION

This command is used in conjunction with the command STRUCTURES in &PROPERTIES. It defines the solute atoms from which mean square displacements are to be computed. and TEMPLATE in &INOUT. The arguments indicate the ordinal numbers of the first *begin* and the last atom *end* of a solute molecule. These numbers may be deduced by inspection of the Template file. The command DEF_SOLUTE can appear more than one time in the environment. The atoms of different solute molecules defined with this command can overlap.

EXAMPLES

&SETUP

```
...
DEF SOLUTE 1 10
DEF SOLUTE 31 57
END
```

&ANALYSIS

```
UPDATE 3 2.0
START 1
STOP 199
```

&END

&PROPERTIES

```
STRUCTURES
  inst_xrms heavy
  print inst_xrms 1 OPEN isnt.xrms
END
&END
```

Computes instantaneous mean square displacements for heavy atoms for the solute chunks 1-10 and 31-57. ...

WARNINGS

This command has no action while running a simulation. It works only during analysis stage (see &ANALYSIS)

SCALE_CHARGES

NAME

SCALE_CHARGES – Scale the total charge on the solute to zero

SYNOPSIS

SCALE_CHARGES *nmol i₁ i₂ ...i_{nmol}*

DESCRIPTION

If Q is the excess charge on the solute, electro-neutrality is imposed by equally distributing $-Q$ charge over the atoms of *nmol* disconnected molecules of solute specified by the indices i_1, \dots, i_{nmol} . Disconnected molecules are ordered according to the sequence given in the structured command JOIN.

EXAMPLES

SCALE_CHARGES 4 1 5 7 11

The excess charge is distributed over 4 molecules: the 1-st, the 5-th, the 7-th and the 11-th molecule as specified in the sequence give in JOIN.

WARNINGS

This command is active only if the solute topology and parameter list is actually computed and **not** read from a binary file (i.e. READ_TPGPRM in &PARAMETERS must be inactive).

SPACE_GROUP

NAME

SPACE_GROUP – Generate a simulation box applying symmetry operations to an input asymmetric unit

SYNOPSIS

SPACE_GROUP OPEN *filename* *group*

DESCRIPTION

Read the space group *group* parameters (inequivalent molecules and corresponding interchange matrices and fractional translations) from the ASCII file *filename*. The file *filename* is a user database which may contain many entries corresponding to different space groups. The following is an example of an entry of this file:

Space Group Symmetry P 2_1

2

1.00	.00	.00
.00	1.00	.00
.00	.00	1.00
.00	.00	.00
-1.00	.00	.00
.00	1.00	.00
.00	.00	-1.00
.00	.50	.00

Space Group Symmetry P 2/c

4

1.00	.00	.00
.00	1.00	.00
.00	.00	1.00
.00	.00	.00
-1.00	.00	.00
.00	-1.00	.00
.00	.00	0.00
.50	.00	.50
-1.00	.00	.00
.00	1.00	.00
.00	.00	-1.00
.00	.50	.50
1.00	.00	.00
.00	-1.00	.00
.00	.00	-1.00
.50	.50	.00

The space group file is parsed by ORAC as usual by interpreting the composing tokens of each line string. The space group name is taken to begin after the third word **Symmetry** in the first line and

may be composed of more than one word. The number of inequivalent molecules *nmol* in the cell is read in the immediately following line. Then, for each molecule, four lines must be provide where the interchange matrix and the fractional translations are read in. No comment lines may be included. In the present example, for the first molecule the identity matrix and the zero translation are given from line 3-6, while in, e.g., the $P2_1$ group, for the second molecule a C_{2y} (line 7-9) rotation and a 0.5 fractional translation (line 10) along the same axis are given. The coordinates of the asymmetric unit must be provided in input through the command **READ_PDB**. The command **REPLICATE** is used to generate a simulation box larger than the unitary cell. Note that the cell parameters of the simulation box are input to the command **CRYSTAL**.

EXAMPLES

SPACE.GROUP sgroup.dat P 2_1

The symmetry transformations of the space group $P2_1$ are applied to the asymmetric unit in order to generate the coordinates of the other molecules contained in the unit cell.

8.2.13 &SOLVENT

The &SOLVENT environment includes commands which are concerned with specific aspects of the solvent structure. In the present version of ORAC force field and topology specifications are given in the same Force fields and topology files used for the solute. The following commands are allowed:

ADD_UNITS COORDINATES GENERATE INSERT READ_SOLVENT REDEFINE

ADD_UNITS

NAME

ADD_UNITS – Add solvent molecules

SYNOPSIS

ADD_UNITS *nmol*

DESCRIPTION

Reads *nmol* molecules from PDB file specified in the READ_PDB(&SETUP) command. This command must be entered when starting from a PDB file which includes *both* solute and solvent coordinates.

EXAMPLES

&SETUP

CRYSTAL 20.00 20.00 20.00 90.0 90.0 90.0

READ_PDB solute+342solvent.pdb

&END

&PARAMETERS

READ_TPG_ASCII ../tpg-prm/amber95.tpg

READ_PRM_ASCII ../tpg-prm/amber95.prm

JOIN SOLUTE

ala-h ala ala ala ala-o

END

JOIN SOLVENT

hoh

END

&END

&SOLVENT

ADD_UNITS 342

&END

...

The file `solute+342solvent.pdb` contains the coordinates of a penta-alanine along with 342 water molecules.

COORDINATES

NAME

COORDINATES – Define the coordinates of a solvent molecule

SYNOPSIS

COORDINATES *filename*

DESCRIPTION

Read the coordinates of the solvent molecule (in PDB format) from file *filename*.

EXAMPLES

```
&SETUP
  CRYSTAL  20.00  20.00  20.00  90.0  90.0  90.0
&END
&SOLVENT
  CELL SC
  INSERT 1.5
  COORDINATES  solvent.pdb
  GENERATE RANDOMIZE  4 4 4
&END
```

In this example the coordinates of the solvent are read in from the file `solvent.pdb` (see `&SOLVENT`). This input would produce 64 solvent molecules in a box of $20 \times 20 \times 20 \text{ \AA}^3$. For generating solvent in presence of the solute see `COORDINATES(&SOLUTE)`

GENERATE

NAME

GENERATE – Replicate solvent molecules.

SYNOPSIS

GENERATE [RANDOMIZE] *ia ib ic*

DESCRIPTION

This command is used to generate a crystal of $ia \times ib \times ic$ cells belonging to the crystalline class specified in the command `CELL`. The optional string `RANDOMIZE` is used for assigning a random rotation to each solvent molecule in the crystal.

EXAMPLES

```
&SOLVENT
  CELL SC
  GENERATE RANDOMIZE  4 4 4
  . . . .
&END
```

The elementary cell is simple cubic with one molecule per unit cell. 64 cells are generated (four in each direction).

INSERT

NAME

INSERT – Insert solute molecules in the solvent

SYNOPSIS

INSERT *radius*

DESCRIPTION

This command is designed to insert solute molecules in a simulation box containing solvent molecules. The solvent molecules which overlap with the solute are discarded. ORAC assumes that two molecules overlap if their distance is less than the sum of their respective Lennard–Jones radii multiplied by *radius*. There is no optimal value for *radius*, however reasonable values are within 0.6 and 0.8.

EXAMPLES

```
INSERT 0.6
```

WARNINGS

This command has no action if CONTROL in &RUN is different from zero, i.e. if the system coordinates are read from a restart file (see RESTART in &INOUT).

READ_SOLVENT

NAME

READ_SOLVENT – Read solvent molecules

SYNOPSIS

```
READ_SOLVENT nmol
```

DESCRIPTION

This command is a synonymous of ADD_UNITS

REDEFINE

NAME

REDEFINE – Read solvent molecules

SYNOPSIS

```
REDEFINE unit_name
```

DESCRIPTION

This command is used for deleting the unit *unit_name* from the solute list and assigning it to the solvent molecules. As long as energies and properties are concerned, the unit *unit_name* will pertain to the solvent.

EXAMPLES

&PARAMETERS

```
    READ_TPGPRM_BIN  benz.prmtpg
```

```
&END
```

```
...
```

&SOLVENT

```
    REDEFINE po4
```

```
&END
```

We redefine the *solute* unit po4 as a *solvent* unit

8.3 Input to ORAC : Force Field and Topology Files

Compared to molecular liquids, simulating any complex macromolecule, poses additional problems due to the covalent structure of the systems and to the related complexity of the potential force fields. ORAC builds the covalent topology needed to evaluate the potential energy from the structure of its constituents. In case of a protein the constituents are the amino acids. Also, ORAC tries to minimize the size and the complexity of the actual input needed to construct this topology.

In practice, the minimal information to be provided in order to describe the residue topology is the constituent atoms, the covalent bonds and, in case of polymers or biopolymers, the terminal atoms used to connect the unit to the rest of the chain. In addition, in order to assign the correct potential parameters to the bonds, bending and torsions of the residue, the type of each atom needs to be specified. Finally, to each atom type must correspond a set of non-bonded parameters.

When the bonding topology of the different residues contained in the solute molecule(s) is known, the units are linked together according to their occurrence in the sequence. In this fashion the total bonding topology is obtained. From this information, all possible bond angles are collected by searching for all possible couples of bonds which share one atom. Similarly, by selecting all couples of bonds linked among each other by a distinct bond, torsions can be obtained.

The following sections describe the format of the topology and force field parameters files read by ORAC. The reading of the two files is carried out immediately after the command `READ_TPF_ASCII` and `READ_PRM_ASCII` in the environment `&PARAMETERS` are encountered in the input file. The topology and force field parameters files are strongly dependent from each other and together fully define the molecular force field of the solute molecule(s). In the ORAC distribution archive the most recent AMBER[2] force field and topology files are provided.

8.3.1 Force Field Parameters

The force field parameters must be placed in the file defined by the command `READ_PRM_ASCII` of the environment `&PARAMETERS`. This file can contain the directives defining the stretching, angle bending, proper and improper torsion, Lennard-Jones potential parameters. Each directive is terminated by the keyword `END` subsequent to the last line of input. The allowed commands are the followings:

`BENDINGS, BOND, NONBONDED [MIXRULE, NOMIXRULE], TORSION [PROPER, IMPROPER]`

BENDING

NAME

`BENDINGS` – Read angle bending potential parameters

SYNOPSIS

`BENDINGS`

...

typ1 typ2 typ3 K_{angle} θ_0

...

`END`

DESCRIPTION

The command reads a sequence of angle bending potential parameters. *typ1*, *typ2* and *typ3* are three character strings, not to exceed 7 characters, indicating the atom types of the three atoms involved in the angle bending interaction. K_{angle} and θ_0 are the angle bending force constant and the equilibrium angle, respectively. The units used for the K_{angle} and r_0 are $\text{Kcal mol}^{-1} \text{rad}^{-2}$ and degree.

EXAMPLES

`BENDINGS`

`cb c na 70.00 111.30`

```

cb   c   o      80.00   128.80
cm   c   o      80.00   125.30
n*   c   na     70.00   115.40
END

```

BOND

NAME

BOND – Read stretching potential parameters

SYNOPSIS

BOND

...

typ1 typ2 K_{stretch} r₀

...

END

DESCRIPTION

The command reads a sequence of stretching potential parameters. *typ1* and *typ2* are two character strings, not to exceed 7 characters, indicating the atom types of the two atoms involved in the stretching interaction. *K_{stretch}* and *r₀* are the stretching force constant and the stretching equilibrium distance, respectively. The units used for the *K_{stretch}* and *r₀* are Kcal mol⁻¹ Å⁻² and Å.

EXAMPLES

BOND

```

c     ca      469.00    1.409
c     cb      447.00    1.419
c     cm      410.00    1.444
END

```

NONBONDED

NAME

NONBONDED – Read Lennard-Jones parameters

SYNOPSIS

NONBONDED [MIXRULE, NOMIXRULE]

...

END

DESCRIPTION

The command reads the Lennard-Jones parameters for the solute non-bonded interactions:

$$\frac{A}{r^{12}} - \frac{B}{r^6} = 4\epsilon \left[\left(\frac{\sigma}{r} \right)^{12} - \left(\frac{\sigma}{r} \right)^6 \right] \quad (8.1)$$

Arguments MIXRULE and NOMIXRULE to the command indicate if Lennard-Jones mixing rules are to be used by ORAC or, conversely, explicit mixed Lennard-Jones potentials are to be expected in input. The format of the nonbonded potential is different in the two alternative cases. If mixing rules are to be found the input to NONBONDED looks like:

NONBONDED MIXRULE

...


```
typ1 r_min  $\epsilon$   $\gamma$  mass
```

```
...
```

```
END
```

Here, *typ1* is a character string, not to exceed 7 characters, labeling the atom type for the atom; r_{min} is the radius corresponding to the minimum of the Lennard-Jones potential; ϵ the Lennard-Jones well depth; γ is reserved for later usage and should be set to zero; *mass* is the atom mass. The non-bonded potential format changes if different Lennard-Jones potentials must be used for the 1-4 interactions in which atom type *typ1* is involved:

```
NONBONDED MIXRULE
```

```
...
```

```
typ1 r_min  $\epsilon$   $r_{min}^{14}$   $\epsilon^{14}$  mass
```

```
...
```

```
END
```

Here, parameters r_{min}^{14} ϵ^{14} are used only for 1-4 interactions. In case the argument NOMIXRULE is used, the input to NONBONDED looks like:

```
NONBONDED NOMIXRULE
```

```
...
```

```
typ1 r_min  $\epsilon$   $\gamma$  mass
```

```
...
```

```
END
```

```
...
```

```
 $B_{ij}$   $A_{ij}$ 
```

```
...
```

First the sequence of the N_{type} force field atom types and Lennard-Jones parameters is read interrupted by the keyword **END** at the beginning of a new line. Second, a list of the $N_{type}(N_{type} + 1)/2$ interaction potential parameters B and A must be provided in input. For most of the biomolecular force fields non-bonded mixing rules are commonly used.

EXAMPLES 1

```
NONBONDED MIXRULE
```

```
h4      1.409   0.015   0.000   1.008
```

```
o       1.661   0.210   0.000  16.000
```

```
ca      1.908   0.086   0.000  12.010
```

```
END
```

EXAMPLES 2

```
NONBONDED NOMIXRULE
```

```
h       0.000   0.000   0.000   1.008
```

```
o       1.700   0.120   0.000  15.999
```

```
c       2.000   0.110   0.000  12.011
```

```
END
```

```
0.0      0.0           Interaction type [h -- h]
```

```
0.0      0.0           Interaction type [h -- o]
```

```
0.0      0.0           Interaction type [h -- c]
```

```
1200.0   600700.0      Interaction type [o -- o]
```

```
1000.0   800000.0      Interaction type [o -- c]
```

```
2000.0   500100.0      Interaction type [c -- c]
```

WARNINGS

If the 1-4 interaction parameters are not provided in input to NONBONDED MIXRULE, the regular non-bonded parameters multiplied by the 1-4 factor in input to LJ-FUDGE of environment **&SOLUTE** are used instead. For interactions involving one atom for which the 1-4 parameters are provided and another for which they are not, regular non-bonded parameters for the interaction are used multiplied by the eventual LJ-FUDGE factor.

TORSION PROPER

NAME

TORSION – Read proper torsion potential

SYNOPSIS

TORSION PROPER

...

typ1 typ2 typ3 typ4 K_{phi} n γ

...

END

DESCRIPTION

typ1, *typ2*, *typ3* and *typ4* are four character strings, not to exceed 7 characters, indicating the atom types of the four atoms involved in the torsion interaction (a *x* string is taken to be as a wild card indicating *any* atom). The torsional axis, according to the ORAC convention is the one connecting the *type2* and *type3*. The parameters K_ϕ and n and γ are defined in Eq. 4.3. K_{phi} is in unit of Kcal mol⁻¹; n is an integer indicating the number of minima(maxima) for 360 degree rotation about the torsional axis; γ is given in degrees and can be either 0.0 or 180.0.

EXAMPLE

TORSION PROPER

x	c	ca	x	3.6250	2.0	180.0
x	cw	na	x	1.5000	2.0	180.0
ct	ct	os	ct	0.3830	3.0	0.0
ct	ct	os	ct	0.1000	2.0	180.0

END

TORSION IMPROPER

NAME

TORSION IMPROPER – Read proper torsion potential

SYNOPSIS

– AMBER form [cosine]

TORSION IMPROPER

...

typ1 typ2 typ3 typ4 K_{phi} n γ [cosine]

...

END

– CHARMM form [harmonic]

TORSION IMPROPER

...

typ1 typ2 typ3 typ4 K_{phi} angle [harmonic]

...

END

DESCRIPTION

typ1, *typ2*, *typ3* and *typ4* are four character strings, not to exceed 7 characters, indicating the atom types of the four atoms involved in the torsion interaction (a *x* string is taken to be as a wild card indicating *any* atom) For improper torsions, ORAC allows both the CHARMM-like form (a simple harmonic potential) or the AMBER-like form (a torsional potential):

For the CHARMM form K_{phi} must be given in Kcal mol⁻¹ rad⁻², while *angle* is the equilibrium angle of the improper torsion in degree.

For the AMBER form the meaning of the symbol are identical to those described in the TORSION PROPER directive.

EXAMPLE

```
TORSION IMPROPER
x    x    ca  h4      1.1000  2.0  180.0 cosine
x    x    ca  h5      1.1000  2.0  180.0
ck   cb   n*  ct      1.0000  2.0  180.0 cosine
cm   c    n*  ct      1.0000  2.0  180.0 cosine
ha   cpa  cpa  cpm     29.40   0.0  harmonic
ha   cpb  c    c       20.00   0.0  harmonic
ha   ha   c    c       20.00  180.0  harmonic
END
```

8.3.2 Topology

ORAC is instructed to read the topology file by the command

```
READ_TPG_ASCII field.tpg
```

of the `&PARAMETERS` environment. File `field.tpg` contains information on the series of residues needed to define the topology of the actual solute molecules. This information is provided through a series of free format keywords and their corresponding input data as done in the main input file `sys.mddata`. In this way, ORAC reads the solute connectivity, the atomic charges, the atomic labels corresponding to those found in the PDB file, and the atomic types according to the chosen force field (i.e. AMBER, CHARMM or others). Moreover, the atomic groups and the improper torsions are also defined.

As for the mail input file, the file `field.tpg` is parsed and the composing substrings of each line are interpreted. Comment lines must have the “#” character in column 1. Each residue or unit definition starts with the keyword

```
RESIDUE residue_name
```

where *residue_name* is a character label which *must* match labels found in the command JOIN of the environment `&PARAMETERS`, and must end with the keyword `RESIDUE_END`. These residue delimiting keywords are the only one in capital letters in *field.tpg* (see the valine example later on in this section)

Atom type definitions and charges are read in between the keywords `atom` and `end`. For each atom three strings must be entered: the PDB atom label, the potential type according to the selected force field as specified in parameter file (see Sec.8.3.1) and the point charge in electron units. Groups are composed of all atoms entered between two successive `group` keywords. The PDB labels must be all *different* from each others since they are used to establish the topology and connectivity of the solute.

The bond connectivity is specified between the keywords `bond` and `end` by providing the series of bonds present in the residue. Each bond is specified by two atom labels corresponding to the atoms participating to the bond.

All possible bendings and proper torsions are computed by ORAC from bond connectivity and need not to be specified. Improper torsions must instead be provided. Improper torsion are used to impose geometrical constraints to specific quadruplets of atoms in the solute. In modern *all-atoms* force fields, improper torsions are generally used to ensure the planarity of an *sp*₂ hybridized atom. The convention in ORAC to compute the proper or improper torsion dihedral angle is the following: If **r**₁, **r**₂, **r**₃, **r**₄ are the position vectors of the four atoms identifying the torsion, the dihedral angle χ is defined as

$$\chi = \arccos \left[\frac{(\mathbf{r}_2 - \mathbf{r}_1) \times (\mathbf{r}_3 - \mathbf{r}_2)}{|\mathbf{r}_2 - \mathbf{r}_1| |\mathbf{r}_3 - \mathbf{r}_2|} \cdot \frac{(\mathbf{r}_3 - \mathbf{r}_2) \times (\mathbf{r}_4 - \mathbf{r}_3)}{|\mathbf{r}_3 - \mathbf{r}_2| |\mathbf{r}_4 - \mathbf{r}_3|} \right] \quad (8.2)$$

RESIDUE

NAME

RESIDUE – Read covalent topology of the residue

SYNOPSIS

RESIDUE *res1*

...

END

DESCRIPTION

The command **RESIDUE** read the covalent topology for the residue labeled *res1*. *res1* must be a character string not to exceed 8 characters. The environment generated by this command can accept the following keywords:

atoms, bonds, rigid, dihed, imphd, omit_angle, backbone termatom, acc, don

This are described in the following paragraphs.

EXAMPLES

Residue topology of amino acid valine.

RESIDUE val (Total Charge = 0.0)

atoms

group

n	n	-0.41570
hn	h	0.27190
ca	ct	-0.08750
ha	h1	0.09690

group

cb	ct	0.29850
hb	hc	-0.02970

group

cg1	ct	-0.31920
hg11	hc	0.07910
hg12	hc	0.07910
hg13	hc	0.07910

group

cg2	ct	-0.31920
hg21	hc	0.07910
hg22	hc	0.07910
hg23	hc	0.07910

group

c	c	0.59730
o	o	-0.56790

end

bonds

cb	ca	cg1	cb	cg2	cb	n	hn		
n	ca	o	c	c	ca	ca	ha		
cb	hb	cg1	hg11	cg1	hg12	cg1	hg13	cg2	hg21
cg2	hg22	cg2	hg23						

end

imphd

-c	ca	n	hn	ca	+n	c	o
----	----	---	----	----	----	---	---

end

```

termatom n c
backbone n ca c
END

```

atoms

NAME

atoms – Read the list of atoms forming the residue.

SYNOPSIS

```

atoms
group
...
lab1 typ1 charge
...
group
...
end

```

DESCRIPTION

The command read the list of atoms and corresponding charges *charge* in electron forming the residue. The list can (and must!) contain the keyword **group** to define atomic groups and is terminated by **end**. *lab1* and *typ1* are both character strings, not to exceed 7 characters, and correspond to the atom label and type, respectively. While, each atom type listed by **atoms** must be defined in the parameters file, each atom label defines uniquely a particular atom of the residue. ORAC expects that labels found in **atoms** be consistent with those used in the input coordinates (i.e. in the PDB file). Atoms in between two consecutive **group** (or between a **group** and the final **end**) form the atomic group.

EXAMPLES

```

atoms
group
n      n      -0.41570
hn     h       0.27190
ca     ct     -0.08750
ha     h1      0.09690
group
cb     ct      0.29850
hb     hc     -0.02970
end

```

WARNINGS

The keyword **atoms** must appear at the beginning of the **RESIDUE** environment.

rigid

NAME

rigid – Define a rigid unit

SYNOPSIS

```
rigid
```

DESCRIPTION

Not supported

bonds

NAME

bonds – Read list of bonds

SYNOPSIS

bonds

...

lab1 lab2 lab3 lab4 ...

...

end

DESCRIPTION

The keyword is used to define a list of covalent bonds among the atoms forming the residue. The list is terminated by **end**. On the lines following **bonds** a series of pairs of atom labels is expected. In the synopsis, atom *lab1* is covalently bound to atom *lab2* and *lab3* to *lab4*. The labels appearing in input to **bonds** must be defined in the atom list given with the command **atoms**.

EXAMPLES

bonds

n ca o c c ca ca ha

cg2 hg22 cg2 hg23

end

WARNINGS

The keyword **atoms** must appear before **bonds**.

omit_angles

NAME

omit_angles – Provide a list of angle bendings to omit

SYNOPSIS

omit_angles

...

lab1 lab2 lab3 lab4 lab5 lab6 ...

...

end

DESCRIPTION

Given the list of bonds for the solute molecule(s) ORAC generates all possible angle bendings. The keyword **omit_angles** allows the deletion of any angle bendings from the residue angle bendings list. Following the line with **omit_angles** a series of triplets of atom labels is expected. In the synopsis, *lab1*, *lab2* and *lab3* are the three atoms involved in one angle bending to be deleted from the residue list. Labels starting with a - or a + correspond to atoms belonging to the preceding and following residue in the solute sequence.

EXAMPLES

omit_angles

n ca c c ca ha

end

WARNINGS

The keyword **bonds** must appear before **omit_angles**.

dihed

NAME

dihed – Define proper torsions list for the residue. Obsolete **Unsupported**

SYNOPSIS

dihed

...

lab1 lab2 lab3 lab4 lab5 lab6 lab7 lab8 ...

...

end

DESCRIPTION

In more modern biomolecular force fields all possible torsion angles are included in the interaction potential (see **AUTO_DIHEDRAL** of the environment **&SOLUTE**). **dihed** includes only selected proper torsions in the potential as it was required by earlier force fields. Each proper torsion is defined by a quadruplet of atom labels (see synopsis). Labels starting with a - or a + refer to atoms belonging to the preceding and following residue in the solute sequence.

EXAMPLES

dihed

-c n ca cb n ca cb cg1 n ca c +n

end

WARNINGS

The keyword **bonds** must appear before **dihed**. If **AUTO_DIHEDRAL** of the environment **&SOLUTE** is selected, the keyword **dihed** has no effect.

imphd

NAME

imphd – Define improper torsions list for the residue

SYNOPSIS

imphd

...

lab1 lab2 lab3 lab4 lab5 lab6 lab7 lab8 ...

...

end

DESCRIPTION

The keyword includes only selected improper torsions. Following **imphd** a list of improper torsions ended by the keyword **end** must be provided. Each improper torsion is defined by a quadruplet of atom labels (see synopsis). Labels starting with a - or a + refer to atoms belonging to the preceding and following residue in the solute sequence.

EXAMPLES

imphd

-c ca n hn ca +n c o

end

WARNINGS

The keyword **bonds** must appear before **imphd**.

backbone

NAME

backbone – Define the backbone atoms for the residue

SYNOPSIS

backbone *lab1 lab2 lab3 ...*

DESCRIPTION

With **backbone** a list of atom labels (*lab1*, *lab2*, *lab3*) is provided which belong to the biomolecule backbone. The corresponding atoms are uniquely identified. The backbone atoms are only used by ORAC in the calculation of run time properties. The command can be repeated as many times as necessary.

WARNINGS

The keyword **bonds** must appear before **backbone**.

termatom

NAME

termatom – Define a pair of atoms which are covalently bound to other residues

SYNOPSIS

termatom *lab1 lab2*

DESCRIPTION

termatom is used to define two atoms, whose labels are *lab1* and *lab2*, which are connecting the residue to the rest of the biopolymer. If the residue has only one connecting atom or has none, one of the labels or both must be replaced by a *****.

EXAMPLES 1

Connecting atoms for an amino acid:

termatom n c

EXAMPLES 2

Connecting atoms for a residue not covalently connected with the others residues of any solute sequence: **termatom** * *

WARNINGS

This keywords must be always present in any **RESIDUE** environment.

acc

NAME

acc – List the hydrogen bond acceptor atoms. Experimental - **Unsupported**

SYNOPSIS

acc *lab1 lab2*

DESCRIPTION

The labels *lab1*, *lab2* are string character indicating the atom types (see command **atom**). If only one label is specified, *label1* refers to the hydrogen bond acceptor. If also *label2* is specified, the latter is the acceptor and *label1* while refers to the conjugate acceptor bonded atom (e.g. N and H (acceptor) in the C-O bond)

don

NAME

don – List the hydrogen bond donor atoms. Experimental - **Unsupported**

SYNOPSIS

don *lab1 lab2*

DESCRIPTION

The labels *lab1*, *lab2* are string character indicating the atom types (see command **atom**). If only one label is specified, *label1* refers to the hydrogen bond donor. If also *label2* is specified, the latter is the donor and *label1* while refers to the conjugate acceptor bonded atom (e.g. C (acceptor) and O (donor) in the C-O bond)

Chapter 9

Compiling and Running ORAC

9.1 Compiling the Program

9.1.1 Serial version

ORAC has been written *mostly* in FORTRAN 77. The present release 5.0 includes some fortran90 code and can no longer be compiled with the g77 compiler. However, ORAC 5.0 can be compiled with **gfortran**, the Gnu Fortran compiler for GCC, the Gnu Compiler Collection. ORAC 5.0 is currently supported only for Linux operating systems. The source code is distributed along with a Makefile which has been tested on several Linux platforms. You must have the Gnu version of **make** to make the executable. As a **configure** file is not provided in this release, for other UNIX platforms the Makefile may need some hacking. The ORAC distribution file is a tar archive containing the ORAC source code and a few examples which illustrate most of the important features of the program.

The untarring of the distribution file using the command `tar -xvf orac5.0.tar.gz` will create a directory with the following sub-directories:

```
./orac5.0
./orac5.0/src
./orac5.0/doc
./orac5.0/RELEASE5.0/tests
./orac5.0/RELEASE5.0/lib
./orac5.0/RELEASE5.0/pdb
```

The directory `./orac5.0/src` contains the source code. Read the copyright agreement **COPYRIGHT** before modifying or distributing the code.

The directory `./orac5.0/doc` contains this manual in pdf and html format.

The directory `./orac5.0/RELEASE5.0/tests` contains input/output examples

The directory `./orac5.0/RELEASE5.0/lib` contains the parameters and topology files according to the the AMBER force field[2] (version AMBER03).

The directory `./orac5.0/RELEASE5.0/pdb` contains The Protein Data Bank format coordinate files for running the input examples in `./orac/examples`

In order to see the list of the available compilation targets do

```
% cd $HOME/orac5.0/src
% make show
```

make with no arguments will show the main targets with a short help. To compile ORAC just do

```
% cd $HOME/orac5.0/src
% make default
```

In this case the fortran compiler is by default **gfortran**. The current release of the Makefile supports also the Intel® fortran compiler. To compile ORAC with the Intel fortran compiler do

```
% cd $HOME/orac5.0/src
% make Intel
```

9.1.2 Parallel version

The parallel version of ORAC has been written using the message passing library interface in its Open MPI version which has full MPI-2 standard conformances. ORAC *must* be compiled with MPI extension for running replica exchange simulation (see Chapter 4.4). In order to do this, you have to have the Open MPI package installed in your multiprocessor computer or in your computer cluster. To compile the parallel version of ORAC, starting from the directory where you have untarred the distribution just do

```
% cd orac5.0/src
% make PARALLEL
```

The default underlying fortran compiler is that implied in your local **mpif90** wrapper. In order to know which compiler **mpif90** is actually using just do

```
% mpif90 -compile-info
```

To compile the parallel version of the executable using the Intel® fortran compiler, starting from the directory where you have untarred the distribution do

```
% cd orac5.0/src
% make Intel_PARALLEL
```

When launched in parallel, ORAC creates in the directory from which it was launched, *nprocs* **PARXXXX** new directories where the main input file is copied and all output of the replicas are written. The only two files that *need* to be in the directory from which ORAC is launched are the main input and the **REM.set** file (only if the a REM simulation is started from scratch and the scaling factors of the replicas are assigned manually and not automatically (see **SETUP(&REM)**).

9.2 How to set dimensions in ORAC : The config.h file

Being written mostly in fortran77 language, the ORAC program does not dynamically allocate the required memory. Memory allocation is done statically and dimensions throughout the code are given in a single file named **config.h**. To adapt the size of the program to other problems the **config.h** file need to be changed and the program recompiled. In the current distribution an ancillary **awk** script that builds the **config.h** file has been provided. This script is called **configure** and can be found in the **tests** directory. **configure** parses a general input file for ORAC and produces, to the standard output, the corresponding **config.h** file.

A certain number of ORAC routines contains **INCLUDE** statements. The corresponding include files, which may contain **PARAMETER**, **COMMON** and general dimension statements (**REAL**, **INTEGER** etc.), have by convention a **.h** suffix and are generated by the standard preprocessor (**/lib/cpp**) from **.inc** files and the **config.h** file. The **.inc** files are templates of include files. where constants are initialized to character symbols (some are listed below). When making the executable, these character symbols are replaced by the standard preprocessor with their numeric values assigned in the **config.h** file.

The meaning of most of the character symbols contained in the **config.h** is explained in the file itself. Here, it is worth mentioning a few:

- **PRESSURE**

The statement **#define PRESSURE** is found in the distribution **config.h** file. It implies that the single time step non-bonded force routines will be generated including the pressure computation section. Since force routines not including the pressure calculation are faster of about 10-20 %, it might be useful in simulation at constant volume to replace the statement with:

`#undef PRESSURE`

With the current version of ORAC , after this change all the `*.CPP.f` files must be removed by hand and the program recompiled.

- `_SIT_SOLU_`

This is the maximum number of atoms in the system (it includes the solvent and solute atoms. The highly misleading name is due to historical reasons.

- `_TYP_SOLU_`

This is the maximum number of possible *different* units type as coded in the topology database.

- `_NRES_`

This is the maximum number of possible units in the solute (i.e. the number of entries in the JOIN structured command).

- `_TGROUP_`

This is the maximum number of groups in the system.

- `_LMAX_ _MMAX_ _NMAX_`

These parameters controls the dimension of the sine/cosine work in the standard Ewald Method. In the `config.h` provided in the ORAC distribution archive, written for SPME simulations, these parameters are all defined to be 1

- `_NAT_WW_ _NAT_WP_ _NAT_PP_`

These parameters control the neighbor list dimensions. E.g. the three neighbor lists for the solvent (since a maximum of three shell for r-RESPA are allowed) are integer arrays of dimensions `_NAT_WW_×_MOL_SOLV_`.

- `_FFT1_ _FFT2_ _FFT2_ MORD`

These parameters control the dimensions of the Q charge array and of the M polynomials for PME computation (see section 4.1)

The total size of the code depends on the number of particles in the system and on the kind of calculation to be carried out. To give an idea, an 8000 atoms system, running with PME, linked cell and computing e.g. the VACF, requires about 25 Mb of memory. The equilibration of the solvated reaction center (33000 atoms) requires around 85 Mb.

Index

- alpha*-carbon, 94
- `acc`, 133
- acceptance ratio, 69, 70, 97
- `ADD_BEND`, 69
- `ADD_BOND`, 69
- `ADD_STR_BENDS`, 80
- `ADD_STR_BONDS`, 79
- `ADD_STR_TORS`, 81
- `ADD_TORS`, 69
- adding a bending, 80
- adding a harmonic distance constraint, 79
- adding an harmonic torsion, 81
- `ADD_TPG SOLUTE`, 73
- `ADD_UNITS`, 121
- `ADJUST_BONDS`, 82
- alkanes, 14
- AMBER force field, 30
- `&ANALYSIS`, 60
- `ANDERSEN`, 110
- Andersen, H.C., 110
- `angular_cutoff`, 92
- angular cutoff, 92
- animation
 - using ORAC generated file, 64
- animation from xyz file, 90
- `ANNEALING`, 110
- `UPDATE`, 60
- `ASCII`, 61
- asymmetric unit, 120
- `atom_record`, 62
- atomic charges
 - subtracting excess charge, 118
- atomic scaling, 21, 113
 - Liouvillean split for, 24
- `atoms`, 130
- `AUTO_DIHEDRAL`, 82
- `average`, 91
- `averaged`, 93
- B-spline interpolation, 33
- backbone, 94
 - writing the coordinates of [] atoms, 63
- `backbone`, 133
- barostat, 18
- bending
 - printing out, 75
- `BENDING`, 82
- bending potential, 29, 100
- `BENDINGS`, 124
- Bennett acceptance ratio, 45, 55
- Berendsen H., 22
- Berne B. J., 3
- `BOND`, 125
- bonded potential, 28
 - subdivision of, 28
- `bonds`, 131
- `BPTI`, 26
- B-spline
 - of the direct lattice potential, 84
- canonical transformations, 7
- center of mass, 106
- `CG`, 112
- `CHANGE_CELL`, 105
- Ciccotti G., 22
- `cofm`, 115
- compiling ORAC , 135
- `compute accessibility`, 95
- `compute contac_solute`, 95
- `compute neighbors`, 95
- `compute volume`, 95
- `config.h`, 136
- `config.h` file, 136
- configure file, 136
- conjugate gradient, 112, 115
- constant pressure
 - scaling, 113
- constant pressure simulation, 17
- constant temperature simulation, 17, 115
- `CONSTRAINT`, 83
- constraints, 82, 83
 - printing out, 75
 - with r-RESPA, 13
- `CONTROL`, 100
- coordinates
 - of the solvent, 121, 123
- `COORDINATES`, 117, 121
- Crooks theorem, 51
- `CRYSTAL`, 105
- crystal structure, 106
- crystal symmetry, 119
- crystal to orthogonal matrix, 107

- crystallographic parameters, 105
- cutoff, 3, 32, 35, 84
 - for hydrogen bonds, 92
 - in the reciprocal lattice, 38
 - reciprocal lattice, 84
- CUTOFF, 83
- cutoff, 91, 95
- DEBUG, 100
- decaalanine, 5
- defaults, 115
- DEF_FRAGMENT, 90
- DEF_SOLUTE, 118
- density of states, 30
- dielectric constant, 3, 38, 93
- diffusion, 94
- diffusion coefficient, 90
- dihed, 132
- dihedral angle, 30
- dihedral angle in torsions, 128
- dimensions
 - changing the, in ORAC , 136
- dipole, 90
- direct lattice potential, 32
- direct potential
 - subdivision of, 35
- dirty, 67
- discrete time propagator, 9, 11, 36
- DIST_FRAGMENT, 90
- dist_max, 112
- dived_step, 95
- don, 134
- driven thermal changes, 64, 103
- driving external potential, 79
- DUMP, 62
- dumping the restart file, 64
- DYNAMIC, 62
- dynamical matrix, 111
- eigenvectors, 111, 112
- electrostatic correction, 84
- electrostatic corrections, 36
- electrostatic potential, 32
 - subdivision of, 35
- energy equipartition, 115
- energy_then_die, 67
- enhanced sampling, 4
- equations of motion, 7
 - for Parrinello-Rahman-Nosé Hamiltonian, 18
- equilibration, 104
- ERF_CORR, 83
- ERFC_SPLINE, 84
- error function, 32, 84
- EWALD, 84
- Ewald method, 4, 32
 - electrostatic corrections, 36
 - in multiple time scales integrators, 66
 - intramolecular correction, 83
 - intramolecular self term, 32
 - self energy, 37
 - setting work array dimensions, 137
 - smooth particle mesh, 32
- excess charge, 118
- extended Lagrangian, 17
- fluctuation theorem, 51
- force breakup, 12
- force field, 28, 124
 - input parameters from ASCII file, 76
 - input parameters from binary file , 76
- force field printout, 62
- FORCE_FIELD, 91
- fractional translations, 119
- fragment
 - writing coordinates of, 63
- free energy, 4
- FREQUENCIES, 111
- fudge factor, 86
- GENERATE, 122
- glycine, 78
- GOFR, 91
- group scaling, 20, 21, 25, 113
 - Liouvillean split for, 24
- GROUP_CUTOFF, 85
- Hamilton
 - equations, 7, 9
- harmonic constraints, 79, 80
- harmonic frequencies, 111
- HBONDS, 92
- heavy_atoms, 95
- Hermitian operator, 10
- histogram , 92
- history file, 62, 65, 90
 - auxiliary file, 62
- H-MASS, 85
- hydrogen bond, 92, 133
 - acceptor and donor, 134
- imphd, 132
- improper torsion, 30, 75, 101, 132
 - definition of, in the parameter file, 128
- &INOUT, 61
- INSERT, 122
- inst_xrms, 94
- &INTEGRATOR, 66
- integrator
 - reversible, 10
 - symplectic, 6, 9
- interchange matrix, 119
- ISEED, 110

- ISOSTRESS, 111
- isothermal-isobaric ensemble, 16
- I-TORSION, 85
- jacobian, 8
- Jarzynski identity, 53
- JOIN, 74
- JORGENSEN, 86
- KEEP_BONDS, 86
- k-ewald, 66
- leap frog algorithm, 10
- Legendre transformation, 19
- Lennard-Jones, 13, 126
 - cutoff, 35
 - parameters, 126
- linked cell, 87, 89
- LINKED_CELL, 87
- Liouville
 - formalism, 9
- Liouvillean, 3, 11, 36
 - split of Parrinello-Rahman-Nosé, 23
- liquid water, 38
- LJ-FUDGE, 86
- Lucy's functions, 49
- lysozyme, 74
- Markovian process, 51
- Martyna G, 3
- mass
 - of the Nosé thermostat, 115
 - specifying the type atomic, 126
- maximum likelihood, 55
- MAXRUN, 101
- MDSIM, 113
- mean square displacement, 94
- memory demand in ORAC, 136
- Message Passing library interface, 136
- &META, 69
- metadynamics, 5, 47, 69
 - Gaussian and Lucy's function, 49
 - multiple walkers, 5
 - well-tempered metadynamics, 50
- MINIMIZE, 112
- mixing rules, 125
- molecular scaling, 25, 113
 - Liouvillean split for, 24
- MPI, 136
- MTS_RESPA, 66
- multiple Bennett acceptance ratio, 45
- multiple restarts in parallel simulation, 64
- multiple time steps, 3, 27
 - for Parrinello-Rahman-Nosé Hamiltonian, 18
- neighbor list, 87, 89
 - for hydrogen bonds, 92
 - setting work arrays dimensions, 137
- non bonded potential
 - subdivision of, 34
- non-bonded potential, 28
- Nosé thermostat, 18, 115
- no_step, 112
- NPT ensemble, 17, 115
- NVT ensemble, 17, 26, 64, 115
- occupy, 62
- omit_angles, 131
- Open MPI, 136
- pair correlation function, 91
- parallel version
 - compiling, 136
 - REM algorithm, 99
 - steered molecular dynamics simulations, 80
- &PARAMETERS, 73
- Parrinello-Rahman-Nosé Extended Lagrangian, 17
- PARXXXX directories, 99, 136
- PDB, 106, 107
 - generating a file, 61
 - writing the `□` file to disk, 63
- PLOT, 63
- PLOT_FRAGMENT, 90
- PMF, 51
- position Verlet, 10
- potential
 - bending, 29
 - bonded, 28
 - non-bonded, 28
 - of mean force, 51
 - stretching, 28
 - subdivision of, 10
- &POTENTIAL, 79
- potential of mean force
 - determination of via the Crooks theorem, 54
- potential subdivision, 28, 36
 - for the AMBER force field, 30, 36
- PRESSURE
 - parameter in the `config.h` file, 136
- PRINT, 101
 - replica exchange method, 97
- print
 - for harmonic calculations, 112
- print, 92
- PRINT_DIPOLE, 93
- print_histo, 92
- printing the force field parameters, 62, 100
- printing topology information, 100
- PRINT_TOPOLOGY, 75
- propagator, 9
 - discrete time, 9
 - stepwise, 9

- proper torsion, 29, 75, 101
 - definition of, in the parameter file, 127
 - frequency range, 30
- &PROPERTIES, 90
- PROPERTY, 102
- protein
 - printing out the sequence, 75
 - giving the input sequence in ORAC , 74
- p_test, 67
- QQ-FUDGE, 87
- r-RESPA, 13
 - energy conservation, 14
 - for NPT ensemble, 23
 - input examples, 67
 - performances, 14
 - use in ORAC, 66
 - with Parrinello-Rahman-Nosé Hamiltonian, 18
- radial_cutoff, 92
- radial distribution function, 90
- RATE, 70
- RATTLE, 13
- reaction coordinate, 5, 51
- reaction field, 3
- READ, 70
- READ_CO, 107
- reading the restart file, 64, 100
- READ_PDB, 106
- READ_PRM_ASCII, 76
- READ_SOLVENT, 123
- READ_TPG_ASCII, 77
- READ_TPGPRM, 76
- reciprocal lattice, 35
- reciprocal lattice potential, 32
- REDEFINE, 123
- reference system, 10, 35
- REJECT, 102
- &REM, 97
- REMSSTEP, 97
- replica exchange method, 38, 97, 98
 - Hamiltonian REM, 42
 - local scaling and global scaling, 99
 - temperature REM, 40
- REPLICATE, 106
- REPL_RESIDUE, 77
- RESET_CM, 106
- RESIDUE, 129
- residue, 77, 129
 - definition of, in the tpg file, 129
 - sequence, 74
- residue, 92
- residue sequence, 100
- RESTART, 64, 100
- restart file, 64, 100, 105
 - parallel simulation, 64
- restricted canonical transformation, 7
- reversible integrator, 10
- rigid, 130
- root mean square displacement, 92, 93, 109, 118
- &RUN, 100
- Ryckaert J.-P., 22
- SAVE, 71
- saving coordinates to disk, 62
- SCALE, 113
- SCALING, 113
- SCALE_CHARGES, 118
- scaling
 - equivalence of atomic and group, 21
- scaling method
 - for constant pressure simulation, 113
- SD, 112
- SEGMENT, 98
- SELECT_DIHEDRAL, 88
- SETUP(&REM)
 - replica exchange method, 98
- &SETUP, 105
- SHAKE, 3, 13, 83
- &SIMULATION, 110
- simulation box, 105
- smooth particle mesh Ewald (see also SPME), 32
- solute
 - defining a fragment of, 90
 - input examples, 107
 - input topology from ASCII file, 77
 - input topology from binary file, 76
 - inserting in solvent, 122
 - pair correlation function, 91
 - setting up the unit cell, 117
 - topology, 105
 - total charge, 118
- SOLUTE, 107
- &SOLUTE, 117
- solute
 - thermostatting solute atoms, 115
- solvent
 - thermostatting solvent atoms, 115
- solute tempering, 98
- SOLVENT, 107
- solvent
 - generating the coordinates, 117, 122
 - input examples, 107
 - reading the coordinates of, 121, 123
 - setting up the unit cell, 122
- &SOLVENT, 121
- space group, 106, 117, 119
- SPACE_GROUP, 119
- spherical cutoff, 35
- SPME, 4, 31, 32, 84
 - accuracy, 34

- B-spline interpolation, 33
- in multiple time scales integrators, 66
- memory demand, 34
- performances, 34
- setting work arrays dimensions., 137
- START, 60
- STOP, 60
- steepest descent, 112
- STEER, 102
- steered molecular dynamics, 5, 51
 - adding a time dependent bending, 80
 - adding a time dependent stretching, 79
 - adding a time dependent torsion, 81
 - along a curvilinear coordinate, 88
 - printing out the work, 64
 - restart, 103
 - thermal changes, 64
- STEER_PATH, 88
- step, 66
- s_test, 67
- STRESS, 114
- stretching, 125
 - printing out, 75
- STRETCHING
 - for the solute, 88
- stretching potential, 28, 100
- structure factor, 33, 90, 91, 94
- structured commands
 - definition of, 58
- STRUCTURES, 93
- VORONOI, 95
- symplectic
 - building integrators, 9
 - condition, 10
 - condition for canonical transformations, 8
 - integrators, 6
 - notation of the equations of motion, 7
- TEMPERATURE, 114
- temperature scaling
 - with Nosé thermostats, 115
- TEMPERED, 71
- TEMPLATE, 108
- termatom, 133
- test-times, 67
- thermal changes, 103
- thermal work, 103
- thermalization, 104
- THERMOS, 115
- thermostat, 18, 115
 - Andersen, 110
- TIME, 104
- time dependent bending, 80
- time dependent stretching, 79
- time dependent torsion, 81
- TIME_CORRELATIONS, 94
- TIMESTEP, 68
- topology, 77, 124, 128
 - adding extra topology, 73
 - from ASCII file, 77
 - from binary file, 76
 - printing, 75
- torsion
 - definition of dihedral angle, 128
 - improper, 132
 - TORSION IMPROPER, 128
 - printing out, 75
 - proper, 127
- torsional potential, 14, 29, 100
- TORSION IMPROPER, 127
- TORSION PROPER, 127
- total, 92
- TRAJECTORY, 65
- trajectory file, 60, 62, 90
 - auxiliary file, 62
- Trotter formula, 9
- Tuckerman M., 3
- unit cell, 117, 122
 - replicating along selected directions, 106, 117, 122
- unitary transformation, 10
- UPDATE, 89
- use_neighbor, 91
- use_neighbors
 - for hydrogen bonds, 92
- vacf, 95
- valine, 129
- velocity
 - rescaling, 113
- velocity autocorrelation function, 90, 94
- velocity Verlet, 10
- Verlet
 - neighbor list, 89
- VERLET_LIST, 89
- very_cold_start, 67
- virtual variables, 19
- Volume calculation, 95
- Voronoi, 26
- Voronoi Polihedra, 95
- Wang-Landau algorithm, 47
- water, 26
 - properties of, 38
- work
 - in a SMD simulation, 64
- write, 62
- WRITE_GRADIENT, 112
- WRITE_GYR, 96
- WRITE_PRESSURE, 116

WRITE_TPGPRM_BIN, 78

WTEMPERED, 71

Xmol animation, 90

X_RMS, 118

xyz format, 64

Bibliography

- [1] S. J. Wiener, P. A. Kollmann, D. T. Nguyen, and D. A. Case. *J. Comput. Chem.*, 7:230, 1986.
- [2] W. D. Cornell, P. Cieplak, C. I. Bavy, I. R. Gould, K. M. Merz Jr., D. M. Ferguson, D. C. Spellmeyer, T. Fox, J. W. Caldwell, and P. Kollmann. *J. Am. Chem. Soc.*, 117:5179, 1995.
- [3] B. R. Brooks, R. E. Bruccoleri, B. D. Olafson, D.J. States, S. Swaminathan, and M. Karplus. *J. Comput. Chem.*, 4:187, 1983.
- [4] W.F. van Gunsteren and H. J. C. Berendsen. *Groningen Molecular Simulation (GROMOS) Library Manual*. Biomos, Groningen, 1987.
- [5] A. D. MacKerrel, J. Wirkeiwicz-Kuczera, and M. Karplus. *J. Am. Chem. Soc.*, 117:11946, 1995.
- [6] J. J. Pavelites, P. A. Gao, and A. D. MacKerrel. *Biophysical J.*, 18:221, 1997.
- [7] A. D. MacKerell Jr., D. Bashford, M. Bellott, R. L. Dunbrack, J. D. Evanseck, M. J. Field, S. Fischer, J. Gao, H. Guo, S. Ha, D. Joseph-McCarthy, L. Kuchnir, K. Kuczera, F. T. K. Lau, C. Mattos, S. Michnick T., Ngo, D. T. Nguyen, B. Prodhom, W. E. Reiher III, B. Roux, M. Schlenkrich, J. C. Smith, R. Stote, J. Straub, M. Watanabe, J. Wiorkiewicz-Kuczera, D. Yin, and M. Karplus. *J. Phys. Chem. B*, 102:3586, 1998.
- [8] J. P. Ryckaert, G. Ciccotti, and H. J. C Berendsen. *J. Comput. Phys.*, 23:327, 1977.
- [9] G. Ciccotti and J. P. Ryckaert. *Comp. Phys. Report*, 4:345, 1986.
- [10] M. P. Allen and D. J. Tildesley. *Computer Simulation of Liquids*. Oxford University Press, Walton Street, Oxford OX2 6DP, 1989.
- [11] P. Procacci, T. Darden, and M. Marchi. *J. Phys. Chem*, 100:10464, 1996.
- [12] W. B. Street, D.J. Tildesley, and G. Saville. *Mol. Phys.*, 35:639, 1978.
- [13] O. Teleman and B. Joensonn. *J. Comput. Chem.*, 7:58, 1986.
- [14] M. E. Tuckerman, G. J. Martyna, and B. J. Berne. *J. Chem. Phys.*, 94:6811, 1991.
- [15] M. E. Tuckerman and B. J. Berne. *J. Chem. Phys.*, 95:8362, 1991.
- [16] M. E. Tuckerman, B. J. Berne, and A. Rossi. *J. Chem. Phys.*, 94:1465, 1990.
- [17] H. Grubmuller, H. Heller, A. Winemuth, and K. Schulten. *Mol. Simul.*, 6:121, 1991.
- [18] M. E. Tuckerman, B.J. Berne, and G.J. Martyna. *J. Chem. Phys.*, 97:1990, 1992.
- [19] M. E. Tuckerman, B. J. Berne, and G. J. Martyna. *J. Chem. Phys.*, 99:2278, 1993.
- [20] D. D. Humphreys, R. A. Friesner, and B. J. Berne. *J. Phys. Chem.*, 98:6885, 1994.
- [21] P. Procacci and B. J. Berne. *J. Chem. Phys.*, 101:2421, 1994.
- [22] P. Procacci and M. Marchi. *J. Chem. Phys.*, 104:3003, 1996.

- [23] G. J. Martyna, M. E. Tuckerman, D. J. Tobias, and M. L. Klein. *Mol. Phys.*, 87:1117, 1996.
- [24] P. Procacci and B. J. Berne. *Mol. Phys.*, 83:255, 1994.
- [25] M. Marchi and P. Procacci. *J. Chem. Phys.*, 109:5194, 1998.
- [26] M. Saito. *J. Chem. Phys.*, 101:4055, 1994.
- [27] H. Lee, T. A. Darden, and L. G. Pedersen. *J. Chem. Phys.*, 102:3830, 1995.
- [28] J. A. Barker and R. O. Watts. *Mol. Phys.*, 26:789, 1973.
- [29] J. A. Barker. The problem of long-range forces in the computer simulation of condensed matter. volume 9, page 45. NRCC Workshop Proceedings, 1980.
- [30] P. Ewald. *Ann. Phys.*, 64:253, 1921.
- [31] S.W. deLeeuw, J. W. Perram, and E. R. Smith. *Proc. R. Soc. London A*, 373:27, 1980.
- [32] T. Darden, D. York, and L. Pedersen. *J. Chem. Phys.*, 98:10089, 1993.
- [33] U. Essmann, L. Perera, M. L. Berkowitz, T. Darden, H. Lee, and L. G. Pedersen. *J. Chem. Phys.*, 101:8577, 1995.
- [34] R. W. Hockney. *Computer Simulation Using Particles*. McGraw-Hill, New York, 1989.
- [35] H.G. Petersen, D. Soelvanson, and J. W. Perram. *J. Chem. Phys.*, 101:8870, 1994.
- [36] L. Greengard and V. Rokhlin. *J. Comput. Phys.*, 73:325, 1987.
- [37] J. Shimada, H. Kaneko, and T. Takada. *J. Comput. Chem.*, 15:28, 1994.
- [38] R. Zhou and B. J. Berne. *J. Chem. Phys.*, 103:9444, 1996.
- [39] Y. Duan and P. A. Kollman. Pathways to a protein folding intermediate observed in a 1-microsecond simulation in aqueous solution. *Science*, 282:740–744, 1998.
- [40] R. H. Swendsen and J. S. Wang. *Phys. Rev. Lett.*, 57:2607, 1986.
- [41] C. G. Geyer. in *Computing Science and Statistics, Proceedings of the 23rd Symposium on the Interface*, edited by E. M. Keramidis, page 156, 1991.
- [42] E. Marinari and G. Parisi. *Europhys. Lett.*, 19:451, 1992.
- [43] K. Hukushima and K. Nemoto. *J. Phys. Soc. Jpn.*, 65:1604, 1996.
- [44] G. M. Torrie and J. P. Valleau. *Chem. Phys. Lett.*, 28:578–581, 1974.
- [45] A. Laio and M. Parrinello. Escaping free-energy minima. *Proc. Natl. Acad. Sci. USA*, 99:12562–12566, 2002.
- [46] S. Marsili, A. Barducci, R. Chelli, P. Procacci, and V. Schettino. *J. Phys. Chem. B*, 110:14011–14014, 2006.
- [47] F. Wang and D. P. Landau. *Phys. Rev. Lett.*, 86:2050–2053, 2001.
- [48] J. Henin and C. Chipot. *J. Chem. Phys.*, 121:2904–2914, 2004.
- [49] A. Laio, A. Rodriguez-Forte, F. L. Gervasio, M. Ceccarelli, and M. Parrinello. Assessing the accuracy of metadynamics. *J. Phys. Chem. B*, 109:6714–6721, 2005.
- [50] C. Jarzynski. Nonequilibrium equality for free energy differences. *Phys. Rev. Lett.*, 78:2690–2693, 1997.
- [51] G. E. Crooks. *J. Stat. Phys.*, 90:1481–1487, 1998.

- [52] G. Hummer and A. Szabo. *Proc. Natl. Acad. Sci. USA*, 98:3658–3661, 2001.
- [53] M. R. Shirts, E. Bair, G. Hooker, and V. S. Pande. *Phys. Rev. Lett.*, 91:140601, 2003.
- [54] D. Chandler. *Introduction to Modern Statistical Mechanics*. Oxford University Press, 1987.
- [55] J. M. Sanz-Serna. *Acta Numerica*, 1:243, 1992.
- [56] S. K. Grey, D. W. Noid, and B. G. Sumpter. *J. Chem. Phys.*, 101:4062, 1994.
- [57] J. J. Biesiadecki and R. D. Skeel. *J. Comp. Physics.*, 109:318, 1993.
- [58] P. J. Channel and C. Scovel. *Nonlinearity*, 3:231, 1990.
- [59] H. Goldstein. *Classical Mechanics*. Addison-Wesley, Reading MA, 1980.
- [60] V. I. Arnold. *Mathematical Methods of Classical Mechanics*. Springer-Verlach, Berlin, 1989.
- [61] H. F. Trotter. *Proc. Am. Math Soc.*, 10:545, 1959.
- [62] H. de Raedt and B. De Raedt. *Phys. Rev. A*, 28:3575, 1983.
- [63] H. Yoshida. *Phys. Letters A*, 150:262, 1990.
- [64] S. J. Toxvaerd. *J. Chem. Phys.*, 87:6140, 1987.
- [65] H.C Andersen. *J. Comput. Phys.*, 52:24, 1983.
- [66] M. E. Tuckerman and M. Parrinello. *J. Chem. Phys.*, 101:1302, 1994.
- [67] S. Nose and M. L. Klein. *Mol. Phys.*, 50:1055, 1983.
- [68] G. Herzberg. *Spectra of Diatomic Molecules*. Van Nostrand, New York, 1950.
- [69] M. Watanabe and M. Karplus. *J. Phys. Chem.*, 99:5680, 1995.
- [70] J. K. Kjems and G. Dolling. *Phys. Rev. B*, 11:16397, 1975.
- [71] F. D. Medina and W. B. Daniels. *J. Chem. Phys.*, 64:150, 1976.
- [72] G. Cardini and V. Schettino. *Chem. Phys.*, 146:147, 1990.
- [73] D. Frenkel and B. Smit. *Understanding Molecular Simulations*. Academic Press, San Diego, 1996.
- [74] D. C. Rapaport. *The Art of Molecular Dynamics Simulation*. Cambridge University Press, Cambridge (UK), 1995.
- [75] S. Nosé. In M. Meyer and V. Pontikis, editors, *Computer Simulation in Materials Science*, page 21. Kluwer Academic Publishers, 1991.
- [76] S. Nosé. *Prog. Theor. Phys. Supp.*, 103:1, 1991.
- [77] M. Ferrario. In M.P.Allen and D.J.Tildesley, editors, *Computer Simulation in Chemical Physics*, page 153. Kluwer Academic Publishers, 1993.
- [78] G. J. Martyna, D. J. Tobias, and M. L. Klein. *J. Chem. Phys.*, 101:4177, 1994.
- [79] H. C. Andersen. *J. Chem. Phys.*, 72:2384, 1980.
- [80] M. Parrinello and A. Rahman. *Phys. Rev. Letters*, 45:1196, 1980.
- [81] S. Nose. *Mol. Phys.*, 52:255, 1984.
- [82] M. Ferrario and J.-P. Ryckaert. *Mol. Phys.*, 78:7368, 1985.
- [83] M. E. Tuckerman, C. J. Mundy, and M. L. Klein. *Phys. Rev. Letters*, 78:2042, 1997.

- [84] S. Melchionna, G. Ciccotti, and B. L. Holian. *Mol. Phys.*, 78:533, 1993.
- [85] H.J.C.Berendsen. Lectures notes unpublished; reported by G.Ciccotti and J.P.Ryckaert, *Comp. Physics Report* 4 (1986) 345, 1986.
- [86] E. Paci and M. Marchi. *J. Phys. Chem.*, 104:3003, 1996.
- [87] S. Toxvaerd. *Phys. Rev. B.*, 47:343, 1993.
- [88] J.-P. Hansen. Molecular-dynamics simulation of coulomb systems in two and three dimensions. In *Molecular Dynamics Simulation of Statistical-Mechanics Systems*, Proceedings of the International School of Physics "Enrico Fermi". North Holland Physics, 1986.
- [89] H.G. Petersen. *J. Chem. Phys.*, 103:3668, 1995.
- [90] P. Procacci, T. Darden, E. Paci, and M. Marchi. *J. Comp. Chemistry*, 18:1848, 1997.
- [91] S. J. Stuart, R. Zhou, and B. J. Berne. *J. Chem. Phys.*, 105:1426, 1996.
- [92] P. Procacci, M. Marchi, and G. J. Martyna. *J. Chem. Phys.*, 108:8799, 1998.
- [93] A. Rahman and F. H. Stillinger. *J. Chem. Phys.*, 55:3336, 1971.
- [94] P. Liu, B. Kim., R. A. Friesner, and B. J. Berne. *Proc. Acad. Sci.*, 102:13749–13754, 2005.
- [95] M. R. Shirts and J. D. Chodera. *J. Chem. Phys.*, 129:124105, 2008.
- [96] M. Mezei. *J. Comput. Phys.*, 68:237, 1987.
- [97] G. H. Paine and H. A. Scheraga. *Biopolymers*, 24:1391, 1985.
- [98] T. Huber, A. E. Torda, and W. F. van Gunsteren. *J. Comput.-Aided Mol. Des.*, 8:695, 1994.
- [99] S. Marsili, A. Barducci, R. Chelli, P. Procacci, and V. Schettino. *J. Phys. Chem. B*, 110:14011, 2006.
- [100] M. Watanabe and W. P. Reinhardt. *Phys. Rev. Lett*, 65:3301, 1990.
- [101] N. G. Van Kampen. *Stochastic Processes in Physics and Chemistry*. North-Holland, 1992.
- [102] M. Iannuzzi, A. Laio, and M. Parrinello. *Phys. Rev. Lett.*, 90:238302, 2003.
- [103] V. Babin, C. Roland, T. A. Darden, and C. Sagui. *J. Chem. Phys.*, 125:204909, 2006.
- [104] L. B. Lucy. *Astronom. J.*, 82:1013, 1977.
- [105] Wm. G. Hoover. and C. G. Hoover. *Phys. Rev. E*, 73:016702, 2006.
- [106] D. J. Earl and M. W. Deem. *J. Phys. Chem. B*, 109:6701, 2005.
- [107] C. Zhou and R. N. Bhatt. *Phys. Rev. E*, 72:0205701(R), 2005.
- [108] R. E. Belardinelli and V. D. Pereira. *Phys. Rev. E*, 75:046701, 2007.
- [109] A. Barducci, G. Bussi, and M. Parrinello. *Phys. Rev. Lett.*, 100:020603, 2008.
- [110] P. Raiteri, F. L. Gervasio, C. Micheletti, and M. Parrinello. *J. Phys. Chem. B*, 110:3533, 2006.
- [111] B. Isralewitz, M. Gao, and K. Schulten. *Curr. Op. Struct. Biol.*, 11:224–230, 2001.
- [112] D. J. Evans and D. J. Searls. *Phys. Rev. E*, 50:1645–1648, 1994.
- [113] M. Sprik and G. Ciccotti. *J. Chem. Phys.*, 109:7737–7744, 1998.
- [114] S. Park and K. Schulten. *J. Chem. Phys.*, 120:5946–5961, 2004.
- [115] R. W. Zwanzig. *J. Chem. Phys.*, 22:1420, 1954.

- [116] R. H. Wood and W. C. F. Muehlbauer. *J. Phys. Chem.*, 95:6670–6675, 1991.
- [117] C. H. Bennett. *J. Comp. Phys.*, 22:245, 1976.
- [118] A.M. Ferrenberg and R. H. Swendsen. *Phys. Rev. Lett.*, 63:1195, 1989.
- [119] D. D. Minh and A. B. Adib. *Phys. Rev. Lett.*, 100:180602, 2008.
- [120] R. Chelli and P. Procacci. *Phys. Chem. Chem. Phys.*, 11:1152–1158, 2009.
- [121] C. Brot B. Quentrec. *J. Comp. Phys*, 13:430, 1975.
- [122] M. P. Allen and D. J. Tildesley. *Computer Simulation of Liquids*. Oxford University Press, Walton Street, Oxford OX2 6DP, 1989.