User Manual for

EGO_VIII

(Release 2.0)

Markus Eichinger[†], Helmut Grubmüller[‡], and Helmut Heller

Leibniz-Rechenzentrum der Bayerischen Akademie der Wissenschaften
Barer Str. 21, D-80333 München, GERMANY

†: Buschingstr. 13, 81677 Munich, GERMANY

‡: Max-Planck-Institut für biophysikalische Chemie
Arbeitsgruppe für theoretische molekulare Biophysik
Am Faßberg 11, D-37077 Göttingen, GERMANY

1st printing, August 1995

Contents

1 Introduction **1**
1.1 What is EGO? . . . 1
1.2 What to Read . . . 2
1.3 Hardware & Software Requirements . . . 2
1.4 For Further Information . . . 2

2 Getting Started – Computing a Trajectory **5**
2.1 Installing EGO on Your Computer . . . 5
2.2 Preparing Simulation Data for EGO . . . 7
2.3 Creating lis-files for EGO . . . 8
2.4 Setting the Control Parameters . . . 9
2.5 Running EGO . . . 10
2.6 EGO Output . . . 12
2.7 Data Analysis . . . 13
2.8 Summary . . . 14

3 Starting and stopping EGO **15**

4 The Control File **17**
4.1 List of lis-files . . . 19
4.2 Requested Number of Nodes . . . 19
4.3 Frequency of Analysis Printout . . . 20
4.4 Frequency of Writing Restart-file . . . 21
4.5 Frequency of System Call . . . 21
4.6 Frequency of Energy Printout . . . 21
4.7 Number of Integration Steps . . . 22
4.8 Integration Time Step . . . 22
4.9 Order of Exclusion List . . . 22
4.10 Switch for Minimisation . . . 22
4.11 Switch for Equilibration . . . 23
4.12 Number of Distance Classes . . . 23
4.13 Used Type of Cluster Algorithm . . . 23
4.14 Number of Hierarchy Levels . . . 23
4.15 Number of Branches on Last Level . . . 23
4.16 Step for New Clustering . . . 24
4.17 Path for Output Data . . . 24
4.18 1-4 Electrostatic Damping . . . 24
4.19 Switch for Stochastic Boundary . . . 24

4.20 Switch for Harmonic Restraints . . . 24
4.21 Switch for SHAKE on Hydrogens . . . 25
4.22 Definition of SBOUND Region . . . 25
4.23 Maximum Friction Coefficient . . . 25
4.24 Switch for Using Flooding . . . 25
4.25 Switch for Using Translation and Rotation Correction . . . 26
4.26 Switch for Using Adaptive Flooding . . . 26
4.27 Initial Energy for Flooding in kT . . . 27
4.28 Final Energy for Flooding in kT . . . 27
4.29 Flooding energy increase in kT/ps . . . 27
4.30 Time constant for adaptive flooding in s . . . 27
4.31 Number of User Defined Integers and Doubles . . . 27
4.32 Switch for using immobilisation . . . 27
4.33 Pulling and stepping mode . . . 27
4.34 The Free Format Section . . . 28
4.34.1 Control of force output . . . 28
4.34.2 Control of minimization . . . 30
4.34.3 Control of constraints . . . 30
4.34.4 Calculation of the Hesse-Matrix . . . 31

5 Implementation **33**
5.1 Structure of EGO . . . 33
5.2 Structure of `xpl2lis` . . . 34

6 Methods **39**
6.1 Numerical Tasks in Molecular Dynamics Simulations . . . 39
6.1.1 Energy Function . . . 39
6.1.2 Integration Methods . . . 40
6.2 Methods to Increase Efficiency . . . 40
6.2.1 Distance Cut-Off Scheme . . . 41
6.2.2 Multiple Time Step Method . . . 41
6.2.3 Structure Adapted Multipole Method . . . 43
6.2.4 Fast Multiple Time Step Structure Adapted Multipole Method . . . 44
6.2.5 Conformational Flooding . . . 45
6.2.5.1 Theoretical Background . . . 46
6.2.5.2 Parameters in the control file relevant for 'flooding' . . . 49
6.2.5.3 Creating a 'flooding' matrix: `mkflood` . . . 52
6.2.5.4 A sample application . . . 53

A File Formats **59**
A.1 Input Files . . . 59
A.1.1 The units definition file `units.def` . . . 59
A.1.2 Brookhaven PDB Atom Coordinate Files . . . 61
A.1.3 X-PLOR Protein Structure Files (PSF) . . . 61
A.1.4 X-PLOR Parameter Files . . . 63
A.1.5 X-PLOR Topology Files . . . 64
A.2 Output Files . . . 66
A.2.1 EGO Trajectory Output Files (`.ego`) . . . 66

A.2.2 X-PLOR Trajectory Files (`.DCD, .crd`) 66
A.2.3 EGO Energy Summary Files (`.eny`) 67
A.2.4 Format of the 'flooding' matrix file `flooding.lis` 68

B Comparison of EGO and X-PLOR **71**
B.1 Important differences between EGO and X-PLOR 71
B.2 Common features of EGO and X-PLOR 71
B.3 Features Unique to EGO 72

C References **73**
C.1 Molecular Dynamics 73

Bibliography **73**

List of Figures

4.1 Definition of SBOUND potential 26

5.1 Program flow on the “master node” 35
5.2 Program flow on a “worker node” 36
5.3 Evaluation of data packets 37

6.1 Schematic plot of distance classes 42
6.2 Interaction scheme of SAMM 45
6.3 ‘Conformational Flooding’ 48
6.4 Harmonic effective Hamiltonian 49
6.5 Expected acceleration factor 50
6.6 Flooding example gramicidin 54

List of Tables

6.1 Computation scheme for the distance class algorithm 43

B.1 Important differences between EGO and X-PLOR 71

Chapter 1

Introduction

1.1 What is EGO?

EGO is a program to compute molecular dynamics trajectories. It is written in the programming language C [27]. EGO runs as a massively parallel program with high efficiency on a PowerXplorer (Parsytec) and CC (Parsytec) parallel computer using PARIX[33]. To be highly portable EGO uses PVM [16] and MPI [11] on workstation clusters, the Cray-T3D, Cray-T3E, the IBM-SP2 and other parallel computers. Additionaly EGO can be compiled to run sequential and, thus, may run on any UNIX workstation or Windows NT/95 system.

We have used EGO to compute trajectories for molecular systems containing more than $35,000$ atoms with extended atoms (i.e., non-polar hydrogens are handled implicitly in terms of special atom types) or with explicit hydrogen atoms. EGO uses a modified Verlet integration scheme (see Chapter 6). Input files to EGO consist of Brookhaven Protein Data Bank PDB files for the atomic coordinates, and X-PLOR-compatible PSF and parameter files for topology information and force constants.

Molecular dynamics simulations [29, 39, 31] have evolved in codes such as CHARMM [3, 4] and X-PLOR to model motions in small molecules, proteins, and nucleic acids in order to better understand molecular structure and function. Compared to EGO X-PLOR [6] is a more extensive package for macromolecule structure determination and refinement written by Professor Axel Brünger now at the Departments of Biophysics and Biochemistry at Yale University. X-PLOR has molecule structure manipulation capabilities and other useful features which complement the computing power of EGO for molecular dynamics (MD) work.

Because EGO and X-PLOR share common data file formats, it is important to understand what capabilities they share and how they differ. A more detailed comparison between them is contained in Appendix B.

In molecular dynamics calculations the equations of motion of atoms in molecules are solved by numerical integration. Electrostatic and van der Waals interactions represent non-bonded forces between atoms, and bonded interactions between (bonded) atoms are represented by stretching, torsion, and stearic hindrance potentials. The computational effort of the short range forces increases linear with the number N of atoms and is for sufficiently large systems ($N > 100$) negligible compared to the computational effort caused by Coulomb interaction, which increases with N^2. To reduce this huge computational effort we developed a method which combines a *Fast Multipole Method* (FMM) [18, 19, 28] and a *Multiple Timestep Method* [36, 43, 22] for rapid, yet sufficiently accurate evaluation of Coulomb interactions. The FMM is based on a multipole expansion of the Coulomb potential to a given order for a hierarchical subdivision of space. Rather than to use a cubic subdivision of space — as most implementations of the FMM do — we choose a *Structure Adapted* [32] decomposition method. This method takes

advantage of knowledge about structural and dynamical features of the biomolecules and helps to reduce the computational effort. The *Multiple Timestep Method* is based on the fact that the influence of far separated atoms varies slowly with time and, therefore, the contribution of these interactions can be evaluated less frequently. The combination of these algorithms, which is implemented in EGO we termed *FAst MUltiple timestep Structure Adapted Multipole Method* (FAMUSAMM) [9].

1.2 What to Read

This user manual starts with a semi-tutorial "Getting Started" (Chapter 2) followed by chapters which describe the organization and structure of the EGO program in more detail. Several appendices summarize other relevant information such as file formats, and the differences between EGO and X-PLOR program.

Read Chapter 2 for a quick introduction on how to create an EGO dataset and run a simple simulation. (You might have to use the structure editing features of the X-PLOR program to create the needed PSF topology description files for your PDB files and molecule structures).

Chapter 4 describes the dynamics control parameter settings in more detail.

Chapters 5 and 6 describe the implementation of the program and the modified Verlet integration method used by EGO, respectively.

1.3 Hardware & Software Requirements

EGO is able to run in a simple sequential version nearly on every platform which provides good C-Compilers. For Windows NT/95 systems we have successfully tested EGO using the GNU-C-Compiler.

As EGO uses PVM or MPI the program may run in the parallel form on any platform (with slight modificiations) if PVM or MPI runs on this system. We have successfully tested the parallel EGO version on workstation clusters (DEC, SUN, SGI, HP), a Cray-T3D, a Cray-T3E, an IBM-SP2 and a PowerXplorer and CC (Parsytec) parallel computer. For further information on PVM and the newest versions of PVM see http://www.epm.ornl.gov/pvm/. For further information on MPI see http://www.tc.cornell.edu/Edu/Talks/MPI/. To optimize performance of EGO on a PowerXplorer parallel computer, the PARIX-version of EGO should be used instead of the PVM-version. For further information on PARIX see http://www.parsytec.de/. The PARIX-version, PVM-version or MPI-version is selected during compilation via preprocessor directives (See Chapter 2.1) from the same source code.

1.4 For Further Information

Contact: Leibniz-Rechenzentrum
Barer Str. 21
D - 80333 Munich, (Germany)

or

Max-Planck-Institut für biophysikalische Chemie
Arbeitsgruppe für theoretische molekulare Biophysik
Am Faßberg 11
D-37077 Göttingen, (Germany)

- Helmut Grubmueller, hgrubmu@gwdg.de
- Helmut Heller, heller@lrz.de

Chapter 2

Getting Started – Computing a Trajectory

To compute a molecular dynamics trajectory with EGO, you need to:

1. If you will use only the sequential version of EGO you need just a C compiler. For the parallel version make sure that either PVM, MPI or PARIX is installed on the parallel computer you are going to use. For details on PVM, MPI or PARIX we refer to corresponding documentations. Here we give only a short description how the environment for EGO should look like.

2. install EGO on your computer. Compilation is done via a makefile written for GNU-make. So you need to install GNU-make. The executable must be called `gmake` and the location must be included to your `PATH` environment variable.

3. provide a molecular description of your system and dynamics parameter data sets:
   If you use X-PLOR the description consists of a PDB-file and a corresponding PSF-file. Dynamics parameters are given in parameter-files (e.g., `param19.pro`).

4. create lis-files from your data:
   EGO uses only these lis-files for molecular dynamics calculations. They are ASCII-files and contain all necessary data (molecule description, dynamics parameters, simulation parameter). Included with the EGO distribution is the utility-program `xpl2lis` which converts X-PLOR data to lis-data. A test data set of the protein BPTI is included in the distribution of the EGO program. We are going to use this data set as an example in our "Getting started" session.

5. adapt the simulation parameters in the ASCII-file `ctl.lis` to your needs (time step, output directory, etc.).

6. run it!

7. analyze data!

Command sequences in this manual are given in Unix notation. Users of MS-DOS or other systems should substitute equivalent commands where appropriate.

2.1 Installing EGO on Your Computer

Whenever any problems occur during installation or executing EGO, please have a look at the file `readme.txt`, which provides detailed and updated information to some known problems.

To install EGO do the following:

1. Unpack EGO:
   Create a subdirectory in your home directory $(HOME) (or elsewhere),
   `mkdir ego`
   `cd ego`
   Copy the latest EGO distribution to this directory and unpack[1] it:
   `gtar -xzvf ego_viii.taz`

2. Setup environment for EGO:
   A typical PVM installation for EGO will have the following files:
   `$(HOME)/pvm3`
   `$(HOME)/pvm3/bin/ALPHA, $(HOME)/pvm3/bin/SUN4, etc.`
   `$(HOME)/pvm3/include -> /usr/local/pvm3/include`
   `$(HOME)/pvm3/lib -> /usr/local/pvm3/lib`
   The shell script `$(HOME)/ego/ipvm`, which should be sourced by your .login-file (if you use csh or tcsh), sets the following environment variables:
   `PVM_ROOT = $(HOME)/pvm3` and `PVM_ARCH = ALPHA` or `SUN4` or `SGI`, etc.
   Your .login-file should also set `PVM_DPATH $(PVM_ROOT)/lib/pvmd`. The following directories should be added to the `PATH`-variable:
   `$(HOME)/pvm3/lib` and `$(HOME)/pvm3/bin/$(PVM_ARCH)`

   For PARIX you have to set the environment variable `PXSTACK` to 1500000, which you may include to your .login-file. All calls to PARIX (e.g., Compiler, program start) are done via the shell script `ppx`, which comes with PARIX. This shell script manages all other setup for PARIX automatically.

3. Compiling EGO:
   Compilation of EGO (sequential version) and all utility programs distributed with EGO is done by typing
   `aimk`

   The shell-script `aimk` sets the environment variable `PVM_ARCH` which determines the type of the computer you use (e.g. SUN4, ALPHA, SGI5, etc.) automatically and starts `gmake`[2] for compilation. If you want to compile for a different architecture (e.g., PARIX and not SUN4) you can specify this as a command line option:
   `aimk -sys PARIX`
   or
   `aimk -sys PVM`
   In addition you can pass all the standard make flags, e.g.:
   `aimk CC=gcc CFLAGS=-O2`
   to use gcc and optimization level 2.

   The makefile (`Makefile.aimk`) tries to do a decent job in assigning the correct compiler and flags, but sometimes you might have to help a bit. If you compile EGO on a hardware platform not yet supported, please let us know which compiler, flags, etc., you used and we will incorporate it into the makefile. The currently supported platforms are listed at the top of `Makefile.aimk` which looks like:

```
#######################################################################
```

[1] You need gnu-tar or gnu-unzip to do this
[2] We recommend to use GNU-make version 3.71 or later

```
# INFORMATION:
# set SYS from the command line like
# aimk [-sys <value>] [VAR=VALUE <goals>] [<goals>]
# to express what you want.
# Possible <value>s are:
# SEQ           sequential one node program
# SEQ_NANO      sequential one node program on the PARIX NanoKernal
# SEQ_SP2       sequential one node program for the SP2
# SEQ_DOS       sequential one node program for DOS/WIN95/WINNT (gcc)
# PVM           parallel PVM program on workstation cluster
# MPI           parallel MPI program on workstation cluster
# PARIX         parallel PARIX program on Parsytec system
# PPCPVM        parallel program under PVM on Parsytec system
...
```

The object files and the executable files of EGO and all auxilliary programs will be written to subdirectories named like the machine type on which the compiler was running (e.g., `$(HOME)/ego/SUN4/`, `$(HOME)/ego/ALPHA/` , etc.). The executables are also copied to `$(HOME)/pvm3/bin/<machine-type>`. There are two executable files for PVM named `ego` (executable for master-task) and `node` (exectubale for the slave tasks). There is only one executable `ego.px` for PARIXand for MPI. As the PARIX compiler is a cross compiler running for instance on a Sun workstation, `ego.px` is located in the correspodning directory. Also one executable, named `ego.<machine-type>`, is created for MPI. The filename of the sequential version of EGO is `ego_seq`. If you want to compile only EGO for a run on a PowerXplorer excluding the auxilliary programs you have to type
`aimk ego.px`

The utility programs which are distributed with EGO are not parallelized, but run sequentially in any standard UNIX environment. These utility programs usually are compiled through the makefile for the sequential EGO version (`aimk`). You don't need PVM for these utility programs, but the makefile copies the executables to the directory pointed to by `PVM_ROOT`. Thus you must have a directory `$(HOME)/pvm3/bin/<machine-type>`, even if you don't use PVM. If such a directory does not exist, it will be made for you.

If you call any of these utility programs without parameters, the program prints out detailed explanations. As the number of utilities is still growing, refer to the file `utilities.txt` for the actual list of available utility programs included to your current copy of EGO. Some important utility programs are listed below:

- `xpl2lis` converts X-PLOR-data to lis-data.
- `ego2crd` converts EGO-output files to crd/DCD-file format (trajectory file) which can be analyzed by other programs (X-PLOR, Quanta, etc.).
- `listest` prints out some important data on your simulation model (total mass, charge, bounding box, etc.).
- `mkmaxw` assigns a specific 'temperature' to your dataset.

2.2 Preparing Simulation Data for EGO

To describe a molecule or molecular system, you need to provide a description of the location of the atoms (the atom coordinates) and how they are connected together (the molecule topology).

Furthermore, you must also provide force constants for the interactions between atoms which are bonded together, and for interactions between atoms which are not bonded.

The former interactions are referred to as the *bonded* interactions, while the Coulomb and van der Waals interactions are referred to as the *non-bonded* interactions.

Van der Waals σ and ϵ parameters must be specified for interactions between all different types of 'like' atoms, and, from these, the parameters for interactions between 'different' atoms are derived from the 'like' atom parameters using the arithmetic average for sigma and the geometric mean for epsilon:

$$\sigma_{ij} = \frac{1}{2}(\sigma_{ii} + \sigma_{jj})$$

$$\epsilon_{ij} = \sqrt{\epsilon_{ii}\epsilon_{jj}}$$

.

X-PLOR Format Datasets

Included in the EGO distribution is the utility program `xpl2lis` which converts data of X-PLOR-compatible data files to lis-files used by EGO. X-PLOR data files include:

1. a .pdb (Brookhaven Protein Data Bank) file listing atom number, atom name, residue name, residue number, atom coordinates (x,y,z in units of angstroms), a friction coefficient and force constant,

2. a .psf (protein structure file) file listing each atom type, partial charge and mass, as well as a specification of which atom pairs are bonded, which sets of atoms form dihedral bond angles, exclusion lists, etc. A .psf file is commonly created from a starting .pdb file by using the structure editing functions of the X-PLOR program.

3. energy parameter file(s) which describe the bonded force parameters (such as certain hydrogen bond interactions, for example) as well as the van der Waals parameters for the different atom types. It is also possible for `xpl2lis` to use CHARMM parameter files.

The structure of .pdb, .psf and parameter files are further described in Appendix A of the EGO User Manual.

BPTI Test Dataset

A test data set for BPTI is included in the distribution of the EGO program. The data set is placed in the directory `$(HOME)/ego/testbpti/` and consists of the three X-PLOR-data files `pti.pdb, pti.psf, param19.pro`. We are going to use these data set in our "Getting started" session.

2.3 Creating lis-files for EGO

To create the lis-data files of the BPTI test dataset change to the directory containing the X-PLOR data

```
cd $(HOME)/ego/testbpti/
```

and start the conversion utility program `xpl2lis` by typing

```
$(HOME)/ego/ALPHA/xpl2lis ../utils/units.def pti.pdb pti.psf param19.pro
```

This example works if you have compiled EGO via `aimk` for a DEC-ALPHA computer.

During conversion some information on the newly created lis-files is given. Make shure that no warning errors occur during conversion and that xpl2lis does not stop with an fatal error. Each lis-file consists of a short header section which gives information on type and format of the included data. The file `units.def` located in the directory `$(HOME)/ego/utils` is a ASCII file and is needed for FAMUSAMM, which groups atoms into *structural units* in order to obtain rapidly converging multipole expansions. For every molecule structure (e.g., water, proteins, lipids, etc.) a corresponding subdevision into such structural units has to be given in the file `units.def`. In the `units.def` file of your EGO distribution such structural units are already defined for TIP3-water molecules, proteins and some lipids. If you intend to simulate molecules for which a proper subdevision into structural units is not already given, you first must edit the `units.def` file. A detailed description on this procedure is given in Sec. A.1.1.

2.4 Setting the Control Parameters

The lis-file `ctl.lis` is used to define all simulation parameters. All other lis-files are data-files of your simulation model. In Chapter 4 a detailed description of all control parameters in `ctl.lis` is given. For our first run we are only interested in a few settings:

1. the number of nodes to use,
2. the time step intervals for data output and restart saves, and
3. the directory path for output files.

In the following listing of `ctl.lis` two nodes are requested for calculation. Every 100 integration steps a full set of energies and coordinates is written to a newly created output file (ASCII). This is called an analysis step. The energies of all intermediate integration steps are added also to this file. The filename of an EGO-output file is built of a number with leading zeroes and the extension `.ego` (e.g. `00000451.ego`). In our example the simulation uses an integration time step of 1 fs and performs 5000 integration steps. This will lead to 50 EGO-output files (`00000000.ego, ..., 00000049.ego`). Every 10 analysis steps, that is in our case every 1000 integration steps, a restart file (`restart.lis`) is written. This restart file can be used to recover from an unexpected interruption of the calculation (see Section 2.6).

The output directory is specified as `../out/`. The trailing slash is *necessary* for EGO. If this directory does not exist, EGO will create it.

There are a lot of other Control Parameters which will be discussed in detail in Chapter 4.

Example: A Fraction of the Control File `ctl.lis`

```
...
2               Requested number of nodes.
568             Number of atoms.
100             Freq. of analysis-printout; next line is atom selection string:
A*
10              Frequency of writing restart-file every analysis-step!
0               Frequency of system call every analysis-step.
```

```
ego2crd.sh
-1              Frequency of energy-printout.
5000            Number of integration-steps.
1e-15           Integration time step in seconds.
5               Order of exclusion list.
FALSE           Switch for Minimisation.
...
```

2.5 Running EGO

Now we are ready to run EGO. If you are using the sequential version of EGO just type `ego_seq`.

If you use PVM you have to specify which and how many nodes should be used for EGO. This may be done by entering the PVM console by typing

```
pvm
```

Now you can add your nodes by typing

```
add node1 node2 node3
```

(Substitute your local machine names for node1, node2, etc.). Exiting the PVM console with

```
quit
```

will leave that configuration active for your calculation. If you add fewer nodes than you requested in your `ctl.lis` file, EGO spawns some "virtual nodes" on the existing nodes[3]. Always remember that one node is used as a *master node*. This node controls the *worker nodes*, writes analysis output, etc. So if you request N nodes for calculation in `ctl.lis` you should reserve $N + 1$ "physical nodes" of your parallel computer for EGO to get best performance. To start calculation type

```
ego
```

If you use PARIX you start the calculation by typing

```
ppx run -f0 4 4 /ego/PARIX/ego.px
```

This command starts EGO via the link 0 (`-f0`) on a Parsytec PowerXplorer with $4 * 4 = 16$ nodes which are connected via a 2-dimensional mesh having 4 nodes on each side.

If you use MPI calculation is started usually by a `mpirun` command. Depending on your MPI implementation you specify as argument either the program name or a configuration file which gives information on the hosts to use for your run, pathes and the program name. For further details look at the documentation of your MPI implementation.

The startup of EGO looks like:

```
EGO_VIII.2   A Molecular Dynamics Program
=========================================
C by M. Eichinger, H. Grubmueller and H. Heller, 1988-1995

Reading control file      : </home/mol/eichi/trans/data/pti/ctl.lis>

No of nodes used in calculation = 2 (out of 1 available nodes)
Found NO restart file     : </home/mol/eichi/trans/data/pti/restart.lis>
Lis-files are in directory: </home/mol/eichi/trans/data/pti/>
```

[3] This clearly will slow down calculation and should be used only for tests.

```
Writing data to directory : </home/mol/eichi/trans/data/pti/out/>
Current working directory : </home/mol/eichi/trans/data/pti/>

========== Start of control file ============================
2               Requested no of nodes.
568             Number of atoms.
...
========== End of control file ==============================

568 atoms will be written to output files.
Bounding-Box:  X: -6.9109    27.965   dx=  34.876
               Y:  10.077    34.601   dy=  24.524
               Z: -11.487    24.238   dz=  35.726
Level  0: Nr_of_clusters  105    Nr_of_Children  568   (Time:  0.1074)
Cl.-Statistic-file <cluster0.out> written: <RGyr>= 1.408 Sig = 0.236
Cl.-Statistic-file <cluster1.out> written: <RGyr>= 4.032 Sig = 0.4205
Number of atoms on Node   1:   300
Number of atoms on Node   2:   268
Reading bondfile ...
Reading exclusion-file ...
Reading exclusion-14-file ...
Reading angle-file ...
Reading dihedral-file ...
Reading improper-file ...
Reading shake-file ...
Found  114 hydrogen-atoms in shake-list.

Start now distributing data to nodes ...
Node number   1 is being loaded with  300 atom-coordinates now.
EXCLUSION list             :  1496
1-4 EXCLUSION list         :  1044
SHAKE list                 :  132
BOND list   (intern/extern):  226     24
ANGLE list  (intern/extern):  395     81
DIHEDRAL l. (intern/extern):  151     67
IMPROPER l. (intern/extern):  109     33
Branch for node  1 loaded.

...
Reading van der Waals parameters
Reading 1-4 van der Waals parameters
Reading bond potential parameters
Reading angle potential parameters
Reading dihedral potential parameters
Reading improper potential parameters

Switched over to calculation phase!
...
```

The last line signals that EGO's startup has finished. All necessary initialization of the master node and worker nodes has been done and EGO starts now to calculate the trajectory.

2.6 EGO Output

In an MD calculation, at *each* analysis step — as specified in the file `ctl.lis` — EGO displays the energy status on the terminal screen and writes the current coordinates to the trajectory file which are put to the output directory specified in the controls file. At *each* dynamics integration step, the energy status is written to the current EGO file.

Energy Status Display

The status display lists the energies of the system at the current integration step along with the temperature and the amount of wall clock time required per integration step. The following display shows a sample output from a BPTI run.

```
ELECTROSTATIC energy: -1414.745 kcal/mol.
VAN DER WAALS energy: -260.334 kcal/mol.
BONDED energy.......:  75.33292 kcal/mol.
ANGLE energy........:  94.56998 kcal/mol.
DIHEDRAL energy.....:  110.3339 kcal/mol.
IMPROPER energy.....:  10.42692 kcal/mol.
RESTRAINT energy....:  0 kcal/mol.
SBOUND energy.......:  0 kcal/mol.
HBOND energy........:  0 kcal/mol.
FLOODING energy.....:  0 kcal/mol.
KINETIC energy......:  518.2215 kcal/mol.
TOTAL energy........: -866.1942 kcal/mol.
TEMPERATURE.........:  329.27 K.
Integration step....:  0 (0 fs)
Time per step.......:  1.503 seconds.
```

Restart File

In the intervals specified in the control file, EGO writes a restart file called `restart.lis` to the directory containing the lis-files. A restart file contains all the information necessary to continue the calculation at the current integration step in the event of a subsequent interruption. At startup, EGO *automatically looks for a restart file* in the lis-file directory and continues the calculation from that point rather than starting over. If you have a restart file in your lis-file directory, but you don't want to use it, you have to rename the restart file, e.g., by typing
`mv restart.lis restart.sav`
or you specify in `ctl.lis` a restart-file that does not exist. See also Section 4.1.

Trajectory Files

At each analysis step, EGO creates a new trajectory file in the output directory. The trajectory files are named 'n.ego' starting from some number 'n' (eight digits, with leading zeroes) and increasing in sequence.

A trajectory file starts with two REMARK lines and the contents of the control file followed by the line '[BEGINCOORD]'. The next line contains four numbers (number of atoms,

integration step, step of analysis output, integration time step in fs) which are separated by white space. These data may be relevant for utility programs to analyze or convert the following atom positions (x, y, z coordinates in Å). Following the list of atom coordinates is an energy output for each integration step up to the next analysis step. Energy data is signaled by the line '[BEGINENERGY]' followed by a line with two numbers specifying the number of comment lines and the number of rows of energy output. The second comment line informs about the meaning of the different columns. Note that in EGO data in one row are separated by white space (spaces or tabs), that is, data columns are not given by a fixed column position!

Example Output Trajectory File Format

```
REMARK Output file of EGO_VIII on C-Version
REMARK C by M. Eichinger, H. Grubmueller and H. Heller, 1988-1995

... contents of control file ...

[BEGINCOORD]
568  552001  1  1.000000
9.930929 9.504575 -4.091603
8.971659 8.787418 -5.255728
...
8.105743 8.477091 -6.756792
9.998736 9.040694 -7.383424
[BEGINENERGY]
2  15
[Clock]=seconds, [Temperature]=K, [Energies]=kcal/mol
IntStep Clock Temperature Total Kinetic Electrostatic VDW Bonded Angle ...
 552001 0.3676 269.27 -1505.217 423.7917 -2009.109 -402.8278 109.4822 ...
...
```

2.7 Data Analysis

From the structure of the EGO files (see above), it should be fairly clear how to write simple programs which can extract trajectory or energy information from the EGO files and/or convert it into other formats. At present, the EGO distribution contains several programs for basic format conversion. These are C programs or simple Unix shell scripts.

The C program `ego2crd` is a utility program which converts EGO-output files into an X-PLOR/Quanta-compatible (FORTRAN UNFORMATTED) trajectory DCD or crd[4] file, and the trajectory energy information into an ASCII format '.eny' file. The source code of this utility is placed in `$(HOME)/ego/utils/ego2crd.c` and can easily be adapted to other output formats.

[4]The DCD file is an X-PLOR-compatible trajectory file which can be read by, e.g., Polygen Corp.'s Quanta molecular modeling software in combination with the PDB file information for the molecule (and a specification of the 'type' of molecule as protein, nucleic acid, etc.) for visualizing and rendering the molecule at different points in the trajectory, and for making trajectory animations, etc.

Other Utilities

As the number of utilities is still growing, please refer to the file `utilities.txt` for a full list of available programs. Some of them are described here:

`ego2pdb` is a utility to convert the atomic positions calculated by EGO during an MD-simulation to other file formats, e.g. to a PDB file or to coord.lis file. This utility is also able to read coordinates given in a free format. For further information type `ego2pdb` and see the given description. To provide an example, the comand line

```
ego2pdb pti.pdb 00000010.ego 00000010.pdb
```

will combine the coordinates from the given single EGO-output file `00000010.ego` with the original PDB file (used to create the EGO lis-files) to create corresponding new PDB files. These new PDB files are suitable for input to X-PLOR or Quanta, e.g., for further analysis or visualization.

`ego2crd` is a second important utility program which converts trajectories calculated by EGO to CHARMM compatible DCD-files or to X-PLOR compatible CRD-files. For usage information type `ego2crd`

2.8 Summary

This completes the tour through the basic steps of computing a trajectory using EGO. To recap, the basic steps are:

1. Create an EGO directory.
2. Create the EGO data lis-files from your input files.
3. Set appropriate control parameters for the simulation.
4. Run the simulation.
5. Analyze the trajectory output using utility programs or X-PLOR.

Chapter 3

Starting and stopping EGO

Usually, EGO is started simply by typing `ego` or `ego_seq`. In that case `ctl.lis` is the default for the control file name. But it is also possible to create control files with different names, e.g. `ctl_run1.lis` and start EGO by typing
`ego ctl_run1.lis`

If you provide a second filename, e.g., if you type
`ego ctl_run1.lis mailrestart.sh`
the second file can be a shell script which performs any task in that case EGO has stopped before the given number of simulation steps has been performed. That may be the case if you are on a computer with batch-job queing and your time limit has exceeded. In that case you can create a small shell-script which sends a mail to you in order to inform you and to setup the next batch job.

There are several ways to stop the execution of EGO before the given number of integration steps is performed. If you use 'Ctrl-C' or if you send a SIGTERM, SIGURG, SIGQUIT, SIGINT or a SIGHUP signal EGO catches that signal and stops as soon as the next integration step is finished. If you send SIGUSR1 then EGO stops as soon as the next restart file has been written.

Chapter 4

The Control File

The lis-file `ctl.lis` contains important simulation parameters, which influence your molecular dynamics calculation and, hence, we call it the control file. In this chapter all parameters which can be set in that control file will be described in detail. Since the computation of the energy function in EGO is based largely on that described in the X-PLOR software package, this chapter will contain frequent cross-references to the X-PLOR User's Manual Version 2.1 [6] for further clarifications.

The control file consists of three sections[1]. The **first section** is made up from a list of lis-files, which give the full description of the model system, e.g., the position of each atom (coord_300K.lis) and bond parameters (bondpara.lis). The **second section** specifies all necessary simulation parameters, e.g., the number of integration steps to be performed. This section has a fixed format and no line must be deleted or inserted. The **third section** is optional and follows after the last line containing 'DEBUG'. This section is in a free format.

The conversion program `xpl2lis` sets up a control file `ctl.lis` which may be regarded as a template file. In this template file most of the parameters given in the second section are set to reasonable values; no third section is included in that template file. Make a copy of the template file and change the parameters in order to meet your intended simulation.

Example control file listing

```
32              Number of files.
shake.lis
coord_300K.lis
1excl.lis
2excl.lis
3excl.lis
4excl.lis
14excl.lis
m1excl.lis
m2excl.lis
m3excl.lis
m4excl.lis
bondlist.lis
anglist.lis
dihelist.lis
```

[1] The usage of three sections does not have a very deep meaning, but is more or less a results of the program history.

```
imprlist.lis
vdw.lis
14vdw.lis
bondpara.lis
angpara.lis
dihepara.lis
imprpara.lis
hbtrlist.lis
hbacclis.lis
hbdonlis.lis
hbahmat.lis
hbbhmat.lis
hbhtype.lis
load.lis
typlist.lis
units.lis
flooding.lis
restart.lis
2               Requested number of nodes.
568             Number of atoms.
100             Freq. of analysis printout; next line is atom selection string:
A*
10              Frequency of writing restart file every analysis step!
0               Frequency of system call every analysis step.
ego2crd.sh
-1              Frequency of energy printout.
10000000        Number of integration steps.
1e-15           Integration time step in seconds.
5               Order of exclusion list.
FALSE           Switch for Minimisation.
1               Friction factor (1.0=>no friction, 0.0=>no motion).
0.08            Maximum position movement per integration step.
FALSE           Switch for Equilibration.
300             Target temperature in Kelvin.
1e-13           Coupling time constant in s.
8               Number of distance classes.
6.000000         0
9.500000         6
15.000000        4
24.000000        4
38.000000        4
60.000000        4
96.000000        4
0               Used type of cluster algorithm (0 is default).
-2              Number of hierarchy levels.
-17             Number of branches on last level.
500             Frequency of reclustering.
out/            Path for output data.
0.4             for special 1-4 electrostatic damping.
FALSE           Switch for stochastic boundary.
FALSE           Switch for harmonic restraints.
TRUE            Switch for SHAKE on hydrogens (Only bond length).
0.000000   0.000000             x- and x+ SBound-planes.
0.000000   0.000000             y- and y+ SBound-planes.
```

```
0.000000   0.000000             z- and z+ SBound-planes.
0.000000   0.000000   0.000000  Center of spheric SBound (x,y,z).
0.000000                        Radius of spheric SBound.
0.000000                        Curvature of SBOUND edges.
0.000000                        Additional stochastic boundary thickness.
0.000000                        Maximum friction coefficient.
FALSE           Switch for using flooding.
FALSE           Switch for using transl/rotation correction.
1               Flooding energy in kT.
300             Flooding temperature in Kelvin.
END
0               DEBUG: 1 == compare with exact forces
0               DEBUG: only should be used by a developer
!*** Here starts the third section, the free format section.
TOTAL A1-20
DIHE  A1-20
```

4.1 List of lis-files

The control file starts with a list of lis-files. EGO reads the specified lis-files given in the control file. The names of the lis-files are not important, but the order of the lis-files must not be changed. Thus you can have different versions of some lis-files and specify in the control file, which of them to use for the next calculation.

A typical lis-file name to change will be the coordinate lis-file (default name `coord.lis`). You can create coordinate files with different initial temperatures by using the utility program `mkmaxw`. Another important lis-file name you may specify is the name of the restart file (default `restart.lis`) to be written or read.

There is a special trick with the restart-file if you connect a '@' to the end of your restart-file name, e.g. 'restart.lis@'. In that case, EGO writes a restart-file named 'restart.lis@', but it will not use it the next time you start EGO. If you want to use that restart-file later you have to rename it to a file-name which does not end up with a '@'.

4.2 Requested Number of Nodes

This number specifies the number of requested "worker nodes" for your calculation. If EGO is not able to access that number of nodes, fewer nodes my be used by EGO. One node works as a master and does not directly participate in the molecular dynamics calculation. The "master node" handles initialization, data output, etc. Make sure that there is one additional node available for that task. One achieves best performance for n requested nodes, if there are $n+1$ "physical nodes". Minimum value for this variable is 2.

If, in case of PVM, EGO is not able to access the requested number of "physical nodes", some "virtual nodes" are created on a "physical node", which will clearly lead to a poor performance. However, this may be useful during program development, as small molecules can then be tested on a single workstation.

4.3 Frequency of Analysis Printout

As "Frequency of analysis printout" you specify after how many integration steps coordinates are written to an EGO-output file. The filename is built of a number with leading zeroes and the extension `.ego` (e.g. `00000451.ego`). An EGO-output file contains one set of coordinates and a number of energy printouts. The format of the EGO-output files is described in Appendix A.2 in detail. A typical value for this tag is 100.

It is possible to define a subset of atoms which are written to an EGO-output file by setting appropriate atom selection strings (ASS). This helps to reduce the size of EGO-output files. By default all atoms are selected (ASS='`A*`'). Atom selection strings are built from one or more atom selection units. The format of an atom selection unit is as follows:
ASS=$[+ \mid -]\{A \mid R\}\{number \mid number1 - number2 \mid string\}$ or
ASS=$[+ \mid -]\{DA \mid DR\}\{atomnumber\}\{<|>\}\{distance\}$

- `+` : include selected atoms/residues (default)
- `-` : exclude selected atoms/residues
- `A` : refer selection to atom numbers/names
- `R` : refer selection to residue numbers/names
- `DA`: refer selection by an atom-atom distance criterion
- `DR`: refer selection by an atom-residue distance criterion
- `number` : refer to a single atom or residue number
- `atomnumber` : refer to an atom number
- `number1-number2` : refer to an atom or residue range
- `string` : select atom/residue by string-match
- `distance` : distance in Å

Defined wildcards for string matching:

- '`*`' maches any string
- '`%`' maches a single character
- '`#`' maches any string of digits
- '`+`' maches a single digit

A '`/`' serves as the escape character, that is e.g. in the string-portion '`/*`' the character '`*`' is not interpreted as a wildcard. The atom selection units within an ASS are evaluated from left to right.

Examples of atom selection strings:

- ASS='`A*`' or '`+A*`': Selects all atoms.
- ASS='`A* -AH*`': Selects all atoms except hydrogen atoms (First, all atoms are selected. Second, all hydrogen atoms are deselected).

- ASS='`AC* AN* -RPRO`' : Selects all C and all N atoms except for those in prolines.
- ASS='`DA254<4.5`' : Selects all atoms which are closer than 4.5 Å to atom with number 254.
- ASS='`DR254<4.5`' : Selects all residues in which at least one atom is closer than 4.5 Å to the atom with number 254.

Note: The utility program `mkflood` uses the same atom selection notation.

4.4 Frequency of Writing Restart-file

As "Frequency of writing restart-file every analysis-step" you specify after how many analysis steps a restart file is written to the directory containing the other lis-files. The name of the restart file is given in the control file (default name `restart.lis`). A restart file contains all the information necessary to continue calculation at the current integration step in the event of a subsequent interruption. At startup, EGO automatically looks for the restart file specified in the control file and continues the calculation from that point rather than starting over. Typical values of writing restart files range between 10 to 1000. Note: At the end of a simulation run a restart file is created automatically!

4.5 Frequency of System Call

As "Frequency of System Call every analysis-step" you specifiy after how many completed EGO-output files a UNIX system-call[2] is invoked. This system call is used to start the shell script you specify in the subsequent line. Typical applications of such shell scripts my be the conversion, compression or moving of EGO-output files. For that reason some helpful variables are passed to the shell script. For more details look at the demo shell script `$(HOME)/ego/utils/ego2crd.sh`. If you enter 0 for this variable, no system call is made[3], otherwise typical values range between 10 to 500.

Calculation of EGO is halted until the shell script has finished. It is possible to send tasks you call in the shell script in to the background by putting an '&' at the end of your UNIX command, so EGO does not wait until the task has finished. If your task returns with an error (return-value not zero), EGO prints a warning, but continues with calculation!

4.6 Frequency of Energy Printout

As "Frequency of energy printout" you specify after how many steps energy information is printed to the display and to the EGO-output files. Due to the approximation algorithm used for long range forces the electrostatic and van der Waals energy is not calculated explicitly in every integration step. Such integration steps are flaged with a negative integration step number. A negative value of $-n$ for 'Step of energy printout' selects every n-th integration step to be print outed. A positive value of n selects only the next integration step after the n-th positve integration step. Zero leads to no energy printout at all.

[2] This is done by calling the C-function `system(char *string)`.
[3] The command line may be left blank, but must not be deleted!

4.7 Number of Integration Steps

This number specifies the number of integration-steps to be performed. After that many integration steps a restart file is written automatically.

4.8 Integration Time Step

This variable specifies the integration time step (in seconds) used by the verlet algorithm. Typical values range between 0.5 fs and 2 fs.

4.9 Order of Exclusion List

The exclusion list order takes a value out of $\pm 1, \pm 2, \pm 3, \pm 4, \pm 5$ which excludes certain non-bonded interactions between neighboring atoms. The meaning of this parameter is identical to the NBXMod variable in X-PLOR:

± 1 no nonbonded exclusions, that is, all nonbonded interactions are computed regardless of covalent bonds.

± 2 exclude nonbonded interactions between bonded atoms (1-2).

± 3 exclude 1-2 and 1-3 angle nonbonded interactions

± 4 exclude 1-2, 1-3, and 1-4 nonbonded interactions

± 5 same as ± 4 with 1-4 damping parameter for electrostatics and special 1-4 parameter for vdw interaction

The default exclusion list order is 5. A positive value causes explicit nonbonded exclusions to be taken account of, a negative value causes them to be discarded. (See also [6, pp. 57 & 97].)

4.10 Switch for Minimisation

If minimization is TRUE, the atom velocities are rescaled by the friction factor t at the end of each integration step and atoms are allowed to move no more than the maximum distance amount specified in "Maximum position movement per integration step" during one step (clipping). This clipping prevents local 'hot' spots from developing during minimization.

As a results of such a minimization procedure the simulation system constantly looses energy until the system will stay in a local structural minimum at temperature 0 K. In such a minimum the total force on each atom vanishes. Usually it is not possible and not necessary to find exactly the local minimum where all forces are zero. A measure of how far away a system is apart from a local minimum may be given by, e.g., the maximum force or the average force of all atoms in the simulation system. The smaller these values are, the closer the system is to the local minimum. During a minimization run these values are printed each analysis step to screen.

Usually, EGO performs the number of integration steps as given in 'Number of Integration Steps'. However, it is also possible to specify a stop criterion based on the maximum force or the average force acting on the atoms in the system. This is done with the keyword `MINIKRIT` in the free format section.

4.11 Switch for Equilibration

If TRUE, the atom velocities are rescaled during each integration step by the instantaneous temperature T of the system to the "target temperature" T_{target} with "coupling time constant" τ_T [6, p. 132]:

$$v_{new} = v_{old} \times \sqrt{1.0 + (T_{\text{target}}/T - 1.0) * (\Delta t/\tau_T)}$$

This procedure is described in more detail in [2]. There is no option for rescaling atom temperatures individually, or for rescaling or averaging temperature over time intervals other than the integration time step interval.

4.12 Number of Distance Classes

EGO uses a combination of a distance class algorithm and a "Fast Multipole Method" (FMM) to compute non bonded interactions between distant atoms. This algorithm is called FAMUSAMM and is described in more detail in Chapter 6.

There are 8 distance classes (class 0: 0–6.0 Å, class 1: 6.0–9.5 Å, etc.). The number of classes cannot be changed. The given values for the distance criterions (e.g., 6.0, 9.5, etc.) are default values chosen by the utility program `xpl2lis`. In general, if one choses larger values for these distance criterions the calculation of the electrostatic interactions is more accurate but also more compuational effort is needed. Usually, these values do not have to be changed and are a good tradeoff between accuracy and computational effort.

4.13 Used Type of Cluster Algorithm

As "Used type of cluster algorithm" you specify the type of cluster algorithm, which is used to build up a hierarchy of clusters. This is needed for FMM. Currently only a neural gas vector quantization algorithm is available. Set this variable to 0.

4.14 Number of Hierarchy Levels

As "Number of hierarchy levels" you specify the number of distance classes (hierarchy levels), which are used by FAMUSAMM for the given simulation data. The number of hierarchies grows with the number of atoms in your simulation system. A negative number indicates, that EGO should determine the number of hierarchy levels automatically at run time. The utility program `xpl2lis` choses a negative number of levels, thus, no manual changes are necessary.

4.15 Number of Branches on Last Level

As "Number of branches on last level" you specify the number of branches on the last level. A branch is a cluster of atom groups (units), which is structured into smaller sub-clusters on finer levels. The number of branches on the last level must be a multiple of the number of nodes. A negative number indicates, that EGO determines the number of branches on the last level automatically at run time. The utility program `xpl2lis` enters a negative number for the optimal number of branches, thus, no manual changes are necessary.

4.16 Step for New Clustering

As "Step for new clustering" you specify after how many integration steps new clustering of atom groups will be done. Such reclustering is necessary, because the size of clusters usually increases during a simulation as the atoms move around. Typical values range from 300 to 1000. Values greater then 500 should only be used if you simulate rigid proteins without water.

4.17 Path for Output Data

As "Path for output data" you specify the path for EGO-output files. The output path must be a subdirectory or a symbolic link of the directory containing the lis-files. If this directory does not exist, EGO creates it. Don't forget to put a trailing '/' at the end of your path. Default path name is 'out/'.

4.18 1-4 Electrostatic Damping

A scaling factor (e14) between 0 and 1 which smoothes the transition between excluded non bonded interactions and included interactions under the the exclusion list option ±5 [6, pp. 57, 97]. Default value is 0.4.

4.19 Switch for Stochastic Boundary

If TRUE, in accordance with the dissipation-fluctuation theorem, a random force is exerted on all atoms that are subjected to friction. Which atoms are subjected to friction depends on the harmonic restraints set in the `coord.lis` file (see below) or the setting of the SBOUND parameters.

If you use harmonic restraints only (all SBOUND parameters are set to zero), you can define the friction coefficient for every atom individually (see lis-file `coord.lis`). If the friction-coefficient of an atom is zero, no friction and no random force acts on that atom.

The utility program `xpl2lis` uses the nineth column (atom-property Q in X-PLOR) of a PDB file to define the friction coefficient. The EGO distribution includes the X-PLOR-script file `$(HOME)/ego/utils/boundary.inp`, which demonstrates how to set up friction coefficients for a selection of atoms.

If you use SBOUND, the friction coefficients in the `coord.lis` file are ignored and instead the stochastic forces act on atoms which are in the SBOUND region or closer then the distance value (in Å) specified in "Additional stochastic boundary thickness". The friction coefficient, given in ps^{-1}, increases linearly from 0 to the value specified in "Maximum friction coefficient". For details see Figure 4.1.

4.20 Switch for Harmonic Restraints

If TRUE, EGO uses harmonic forces to anchor atoms at the reference position defined in the coordinate file `coord.lis`. The strength of the harmonic potential can be set individually for each atom (see lis-file `coord.lis`).

The utility program `xpl2lis` uses the tenth column (atom-property B-factor in X-PLOR) of a PDB file to define the harmonic constant (in kcal/mol/$Å^2$. The EGO distribution includes the X-PLOR-script file `$(HOME)/ego/utils/boundary.inp`, which demonstrates how to set up harmonic restraints for a selection of atoms.

4.21 Switch for SHAKE on Hydrogens

If TRUE, EGO uses the SHAKE algorithm to constrain the bond length of any hydrogen atom to its partner atom. This allows the use of longer integration time steps, usually 1 fs to 2 fs. For details see [39, 25]. **Note**: Only the bond length, and not the angle, is constrained by SHAKE in EGO.

The utility program `xpl2lis` creates the file `shake.lis` which contains a list of all hydrogens and the respective heavy atoms to which they are bonded. EGO uses this list and shakes all hydrogens if this switch is set to TRUE. If, however, one does not like to shake all hydrogen atoms one can choose one of the following ways: The first way is to patch the `shake.lis` file, that is, one has to delete all hydrogen-heavy-atom list entries, to correct the total number of shaked atoms and to set the switch to TRUE in the control file. The other possibility is to use the keyword SHAKEOFF in the free format section (see Section 4.34.3).

4.22 Definition of SBOUND Region

SBOUND forces [5] restrain molecules to a given volume. The forces at the edge of the volume are designed to mimic the effects of solvent water. In EGO cubical and spherical boundaries can be defined (even simultaneously). The faces of a cube are defined by the positions of planes which are parallel to the coordinate system. Spheric SBOUNDs are defined by center and radius. A value of zero for a face position or for the radius indicates that no corresponding SBOUND will be used.

"Curvature of SBOUND edges" sets the radius in Å joining two or three SBOUND planes. If set to zero, SBOUND planes are joined rectangularly and you can select single planes, e.g., only infinite parallel XZ planes. If you have a positive radius, all planes will be active, even if they contain the origin. For details of SBOUND definition see also Figure 4.1.

Usually SBOUND is used together with stochastic boundary. Thus stochastic boundary should be set to TRUE and a "Maximum friction coefficient" greater then zero should be specified.

If any SBOUND is used, individual friction factors defined in `coord.lis` are disregarded. However, individual harmonic restraints can still be specified in addition to SBOUND if the switch for harmonic restraints is also set to TRUE.

EGO uses the SBOUND-potential

$$V(d) = Kd^2(d^2 - P) \tag{4.1}$$

with the constants $K = 0.2\,\mathrm{kcal/mol/Å^4}$ and $P = 2.25\,\mathrm{Å}$. These constants are defined in the constants file `ego.h`.

4.23 Maximum Friction Coefficient

As "Maximum friction coefficient" you specify for the SBOUND case the maximum friction coefficient in ps^{-1} for the the stochastic boundary.

4.24 Switch for Using Flooding

If TRUE, the flooding algorithm [21] is used to speed up transitions between conformational substates of proteins. You need to create a flooding file (default file name `flooding.lis`) with

A Shape of SBOUND potential.

B Friction coefficient for stochastic boundary increases linear and saturates at the position of the minimum.

1 Position of SBOUND is defined at the minimum.

2 Additional stochastic boundary thickness.

3 Maximum friction coefficient.

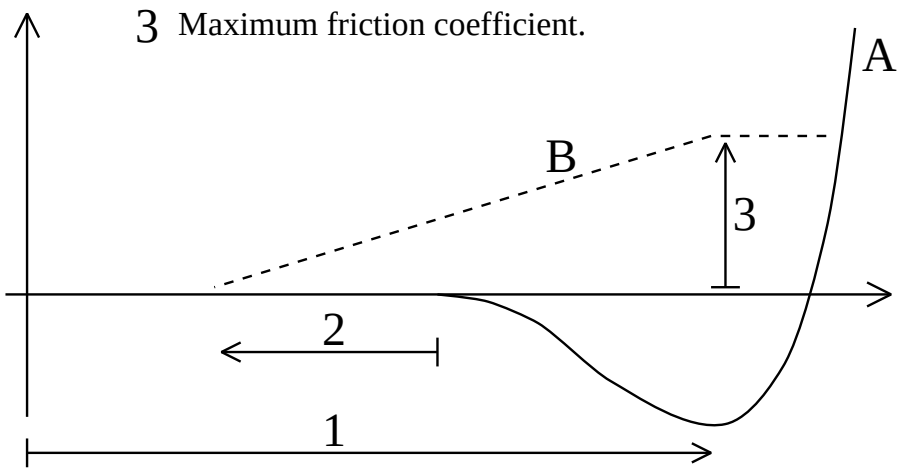

Figure 4.1: Definition of SBOUND potential and area of stochastic forces.

the utility program `mkflood`. The flooding file holds important information for the flooding process. For detailed information about the flooding algorithm see Chapter 6.2.5.

4.25 Switch for Using Translation and Rotation Correction

If TRUE, the translational and rotational degrees of freedom are eliminated by a method described in [7, 8]. This may be necessary for simulations of proteins in vacuum without any harmonic restraints or SBOUND forces. Due to the approximation algorithm used for long range forces, conservation of momentum is not fulfilled exactly. Via a dummy (=no flooding matrix) lis-file `flooding.lis` one can specify which atom coordinates will be used to inhibit translation or rotation. Such dummy flooding files, together with the full flooding files, are created by the utility program `mkflood`. For details see online help information given by `mkflood` as well as Chapter 6.2.5.

4.26 Switch for Using Adaptive Flooding

THIS OPTION IS NOT YET IMPLEMENTED.
PLEASE SET THIS SWITCH TO '`FALSE`'.

For details see Chapter 6.2.5.

4.27 Initial Energy for Flooding in kT

Initial flooding energy in units of thermal energy. **Target temperature in Kelvin** is used to translate that value into kcal/mol.

4.28 Final Energy for Flooding in kT

Maximum flooding energy used in units of thermal energy. "Target temperature in Kelvin" is used to translate that value into kcal/mol.

4.29 Flooding energy increase in kT/ps

The actual flooding energy used during the simulation is ("Initial energy for flooding in kT")+(elapsed time)∗ ("Flooding energy increase in kT/ps") as long as this expression is smaller than the "Final energy for flooding in kT". Otherwise, the flooding energy is set to "Final energy for flooding in kT". This allows (1) to have constant flooding energy (increase 0), (2) linear increase with time and (3) linear increase with time followed by a constant maximal value of the flooding energy.

4.30 Time constant for adaptive flooding in s

If "Switch for using adaptive flooding" is set `TRUE`, this time constant is used for averaging of the actual flooding potential in order to estimate the actual destabilization free energy ΔF. That average is compared with the target flooding energy, and E_{fl} is adjusted dynamically such as to minimize the difference of both.

4.31 Number of User Defined Integers and Doubles

A undocumented feature

4.32 Switch for using immobilisation

If "Switch for using immobilisation" is set to `TRUE`, the position of the center of mass of a set of atoms is constrained by employing a harmonic potential. The usage of this feature is usually necessary if one uses the pulling/stepping mechnism implemented in EGO. For more information see [23, 26]. The set of atoms from which the center of mass is calculated and which feel that harmonic potential can be defined by the usuall selection strings defined in Section 4.3. The integration time step at which this harmonic potential is switched on can be defined in 'Start integration step for flooding/immobilisation/pulling'. The harmonic constant is set in 'Harmonic constant for immobilisation'.

4.33 Pulling and stepping mode

This feature can be used to mimic force atomic microscopy experiments (pulling mode) as described in Ref. [23, 26]. In such computer experiments enzyme-ligand binding can be investigated by exerting a pulling force to the ligand while the center of mass of the enzyme is held fixed by a harmonic potential. If the pulling force is strong enough and points in the direction the enzyme-ligand complex will rupture.

There are two different modes, namely mode '1' (or '4') and mode '2' (or '5'), to manage pulling in EGO. If one enters '1' or '4' for the pulling mode a 'virtual' harmonic spring is used to excert a pulling force as described in [23, 26]. In that mode, one has to specify the atom to which one of the ends of the artifical spring is connected to. Note: Only **one** atom must be selected in the subsequent selection string! Furthermore, one has to specify the movement of the other end of the spring, i.e., one has to enter the speed (given in Å/ps) and the direction of movement into the following two lines. At least, one has to specify the spring constant, i.e., the spring constant acting parallel to the pulling direction and the spring constant perpendicular to the pulling direction. Usually the spring constant perpendicular to the pulling direction is set to zero.

The modes '1' or '4' differ only in the way how the pulling direction is defined. If one chooses mode '1' the pulling direction is given as a vector with carthesian coordinates. If one chooses mode '2' the pulling direction is defined by the interconection line of two atoms. That direction is determined from the positions of the two atoms at the start of EGO and will not change during a simulation even if the positions of the two atoms will change.
Example 1: (To atom 3 a 'spring' is attached pulling to the positive x direction)

```
1 Pulling mode (0=OFF,1=Pulling,2=Stepping).  Next line is atom sel.  string:
A3
10.0 Speed of pulling in A/ps.
1.000000 0.000000 0.000000 Pulling direction (dx,dy,dz).
20.00000 0.000000 Spring-constants (||,T) in kcal/(mol A2̂).
```

Example 2:(To atom 3 a 'spring' is attached pulling to the direction defined by the interconection line of atom 5 and 8.)

```
4 Pulling mode (0=OFF,1=Pulling,2=Stepping).  Next line is atom sel.  string:
A3
10.0 Speed of pulling in A/ps.
5 8 Pulling direction (dx,dy,dz).
20.00000 0.000000 Spring-constants (||,T) in kcal/(mol A2̂).
```

Corresponding to the two pulling modes there are also two 'stepping' modes (mode '2' or '5'). With 'stepping' one can move an atom or a group of atoms with a given velocity in a certain direction. Consequently, in that mode the selection string may define a set of atoms (e.g., a methyl-group). Any internal dynamics of such a selected group of atoms is frozen. The values set for the spring constant are meaningless in that modes.

4.34 The Free Format Section

The last section in the control file is optional and in a "free format" which works as follows: each line in the free format sections starts with a keyword written in capital letters followed by a listof space or tabulator separated parameters. The order of keywords in that section does not matter. There is also the possibility to use comments. Comment lines start with '!' or with '#'. Additionally the C comment style using '/*' and '*/' can be used.

4.34.1 Control of force output

The total force acting on an atom in an MD simulation is composed of several contributions, e.g., of forces arising from bonding terms, angle terms or from electrostatic interactions. In the

free format section of EGO one can specify that the sum of such a force contribution is written to file for later analysis. For example, if one likes to know the force arising from bonding contributions acting on a certain atom (or a set of atoms) one can specify the keyword `BOND` and a selection string:

```
BOND A5
```

Then, the sum of forces arising from bonding terms acting on atom 5 will be printed to the file `bond_force.out` located in the output directory. The file format is ASCII and easy to understand. The general format to control force output is

```
<FORCETYPE> <atom selection string>
```

Possible keywords for `<FORCETYPE>` are:

`BOND`	: Write bond forces to file bond_force.out
`ANGLE`	: Write angle forces to file angle_force.out
`DIHE`	: Write dihedral forces to file dihe_force.out
`IMPR`	: Write improper forces to file impr_force.out
`ELEC`	: Write electrostatic forces to file elec_force.out
`VDW`	: Write van der Waals forces to file vdw_force.out
`RESTR`	: Write restraint forces (arising from the SBOUND region) to file restr_force.out
`ELECVDW`	: Write the sum of electrostatic and van der Waals forces to file elecvdw_force.out
`INTERN`	: Write the sum of `BOND`, `ANGLE`, `DIHEDRAL` and `IMPROPER` to the file intern_force.out
`TOTAL`	: Write the total forces to the file total_force.out

For further information about the format of the atom selection string see Section 4.3.

In the free format section there is also the possibility to control the output of individual forces acting between atoms. For example, if one likes to know the bond force between atom 5 and 6 one has to specify

```
PICKBOND 5 6
```

Then, the force between the two atoms is printed to file `bond_pick.out` together with distance information. The file format is again ASCII. Note, that there will be only a force between atom 5 and 6 if there is a covalent bond specified in the structure file of your model system.

Other forces, which can be 'picked' like that, are the electrostatic and van der Waals forces, e.g., by specifying

```
PICKELEC 10 203
PICKVDW 10 203
```

However, due to the FAMUSAMM algorithm these forces are only explictily calculated if they are closer than about 9 Å. For more distant pairs of atoms these forces are approximated by a multipole scheme which calculates forces between groups of atoms rather than pairs of atoms. Thus, zero is printed to the output file if one selects two atoms more distant than about 9 Å.

If one likes to pick angle forces between a tripplet of atoms one, e.g., specifies,

```
PICKANGLE 4 6 8
```

Dihedral and improper forces can be picked by specifying

`PICKDIHE 4 6 8 9` or

`PICKIMPR 4 6 8 9`

There are two further keywords (`START`, `STRIDE` and `IDSTRING`) which are important for the control of force output. The keyword `START` specifies at which integration step EGO begins to write out the forces to file. The keyword `STRIDE` controls the frequency of writing out the forces. With the keyword `IDSTRING` one can set a string which modifies the name of the force output files. Example:

```
START 500
STRIDE 10
IDSTRING simulation1_
INTERN A5
ELEC A5
```

After integration step 500 and every 10th integration step the corresponding forces are written to file. The files will be named `simulation1_intern_force.out` and `simulation1_elec_force.out`.

Important note: Writing out forces to file considerably reduces the speed of your calculation and takes a lot of disk space if you select a large set of atoms.

4.34.2 Control of minimization

With the keyword `MINIKRIT` followed by two parameters one can set a stop criterion for a minimization run. The first parameter specifies the maximum force and the second parameter the average force acting on the atoms in the system. For example, if one has to find a minimum structure in which the maximum force is lower than 0.5 kcal/mol Å one specifies

```
MINIKRIT 0.5 -1.0
```

The second parameter -1.0 signals that the average force criterion is not used here. Correspondingly, if one specifies

```
MINIKRIT -1.0 0.5
```

the average force is used as the stop criterion. If both parametere are not equal -1.0 both criterions must be fulfilled.

4.34.3 Control of constraints

In the free format section there are three keywords (`SHAKEOFF`, `FORCEOFF` and `SPEEDOFF`) which control constraints.

The keyword `SHAKEOFF` is used to patch the list of hydrogen atoms with constraint bond length (SHAKE algorithm). Usually, if one sets TRUE in 'Switch for SHAKE on Hydrogens' all hydrogen atoms are shaked (see Section 4.21). If however, one wants to deselect a set of hydrogen atoms which should not be shaked, one can use the command `SHAKEOFF` in the free format section.

Example:

```
SHAKEOFF A* -RTIP3
```

The selection string 'A* -RTIP3' selects all atoms which do not belong to a TIP3 residue and, therefore, switches off the SHAKE algorithm for these atoms. Consequently, only TIP3 hydrogens are shaked (if, of course, TRUE is set in 'Switch for SHAKE on Hydrogens').

The keywords `FORCEOFF` and `SPEEDOFF` can be used to switch off the forces and the velocity of a selected set of atoms. Consequently, the selected atoms will not move while all other atoms will move. Usually both keywords are used together each selecting the same set of atoms. As an example application of these keywords one may think of a protein-water system. Let us suppose that a non-optimal setup of that protein-water system has 'produced' a lot of water molecules which have van der Waals overlaps with the protein. A minimization starting from such a structure will strongly affect the protein structure as the overlapping water molecules strongly repell the protein atoms. This can be avoided if one constraints the atom positions of the protein by selecting all protein atoms as given in the example below.

```
FORCEOFF A* -RTIP3
```

```
SPEEDOFF A* -RTIP3
```
Then only the water molecules can move and the protein structure will be unaffected.

4.34.4 Calculation of the Hesse-Matrix

There are two keywords (`HESSE` and `SELHESSE`) which control the calculation of the cartesian force constant Hesse-Matrix. Such a Hesse-Matrix can be used later to calculate a IR vibrational spectrum. The approximation of the Hesse-Matrix $H_{ij} = \partial^2 E/\partial q_i \partial q_j$ is done by finite differences of the first derivatives of the total energy E. This is either done via a one point approximation

$$\frac{\partial^2 E}{\partial q_i \partial q_j} \approx (\frac{\partial E(q_j + d)}{\partial q_i} - \frac{\partial E(q_j)}{\partial q_i})/d \tag{4.2}$$

or a two point approximation

$$\frac{\partial^2 E}{\partial q_i \partial q_j} \approx (\frac{\partial E(q_j + d)}{\partial q_i} - \frac{\partial E(q_j - d)}{\partial q_i})/2d \tag{4.3}$$

, whereas d denotes the step-length of each finite differene step in x, y and z direction.

With the keyword `HESSE` one can select between one of the two modes and set the step length given in Å. The keyword `SELHESSE` is manditatory and is used to specify a set of atoms for which the Hesse-Matrix should be calculated. Example:
```
HESSE 2 0.01
SELHESSE A*
```
In that example all atoms are selected for the calculation of the Hesse-Matrix. The calculation is done by the two point approximation and uses a step length of 0.01 Å. In the output-directory of EGO the file hesse.out is created with the corresponding Hesse-Matrix. The dimension of the elements in the Hesse-Matrix is N/m. In the util-directory there is the utiliy program `hess2hess` which is able to convert such a Hesse-Matrix to other formats and dimensions.

Note: Features which will affect the proper calculation of the Hesse-Matrix, like, e.g., Minimization, Equilibration or SHAKE are automatically switched off. Furthermore, the number of steps necessary to perform the calculation of the Hesse-Matrix is set automatically.

The Hesse-Matrix is stored in the file hesse.out. The header of that contains the atoms with their masses and positions. So, if you like to investigate isotipoc effects you just have to change the masses here in the file. After the keyword Matrix the hesse-matrix follows. Note: If you choose many atoms for a hesse-calulation the lines in that file get very long and some ASCII editors (like the good old vi will not work properly and will destroy the file).

It is also possible to restart a hesse calculation. Just restart again normally. EGO will look in the hesse.out file and will see how far it is. If it is not completed it will restart at the right point. EGO recognizes the status of the hesse-calcualtion from two numbers in the first line after the number of atoms. If it is not completed there will be for example 3 1 2 The first number specifies the number of atoms for which the hesse matrix should be calulated. The second number specifies the atom which is currently is moved and the third number specifies in which direction it is moved. Once it is completed only the number of atoms will be left.

```
Example for a completed hesse-matrix for H2O:
3
   1  OH2       15.99940      -0.60543    0.00000    0.77339
```

```
   2  H1         1.00800       0.00000    0.00000    1.54678
   3  H2         1.00800       0.00000    0.00000    0.00000
Matrix:
 620.76339    0.00555    0.12451   -302.92413    0.00365 -255.71554
-302.90517    0.00985  255.72955
   1.34366  -18.29916   -0.03854     -0.53269   -2.53697   -0.13334
-0.52906   -2.54235    0.14224
  -0.26595   -3.61815  867.78234   -346.08646   -0.31560 -440.00671
346.51745   -0.77901 -440.35201
-306.01033    0.02860 -349.09602    287.57882    0.00321  300.09174
15.91631   -0.01153   44.18106
   4.88112   25.75767   -1.80326      1.59560    3.98306    0.36573
-0.07680   -3.65888   -0.43499
-256.41766    0.18216 -426.99687    301.58252    0.77197  486.95572
-45.46118   -1.13272  -46.69626
-306.92754    0.00193  359.79018     15.84619    0.01383  -43.98345
288.33121   -0.01516 -300.36040
   4.50220   23.25857    1.67851     -0.02868   -2.56054    0.52004
1.64183    3.16829   -0.43899
 256.58000    1.09238 -426.33925     45.55342   -0.96294  -46.79383
-301.54461    0.63691  487.03730
```

Chapter 5

Implementation

All source file names which start with 'eg', like the file `egmaster.c`, are source files of the MD simulation program EGO. All source file names which start with 'wo', like the file `womain.c`, are source files of the utility program `xpl2lis`. An exception are the files `util_lib.c` and `timing.c` which are used by EGO, xpl2lis and other utility programs.

5.1 Structure of EGO

`egmach.c`:
EGO is designed to run on different parallel computers employing PVM, MPI or PARIX.This module contains our "Parallel Computer Interface" (PCI) of EGO employing one of above named parallel interface languages. This is done via a lot of preprocessor compiler directives and, therefore, this file is hard to read. All other functions in EGO call these PCI-functions defined in `egmach.c`.
`egmaster.c`:
This module contains the main loop over all integration steps on the master node. During every integration step the energy values sent from the worker nodes are collected, summed up and printed out. Periodically also atom coordinates are collected for saving in EGO-output files or restart files.
`egnode.c`:
This module contains initialization and the main loop over all integration steps performed on a worker node. After initialization multipole moments of any clusters on the own node are calculated. Together with atom positions these data are distributed to all other worker nodes via "data packets". During one integration step data packets from all other worker nodes have to be collected. The evaluation of such a data packet includes the summing up of short range forces like bonding and angle forces. The long range forces are calculated by FAMUSAMM. After the evaluation of each data packet the forces acting on all own atoms are known and a verlet integration step is performed. Energy and periodically the new coordinates are sent to the master node.
`egtree.c`:
This module contains all things concerning FAMUSAMM.
`egclust.c`:
This module manages the clustering of structural units to clusters and builds up the tree structure which is needed for the FMM.

5.2 Structure of `xpl2lis`

`womain.c`:
This module contains the main body of the utility program `xpl2lis`.

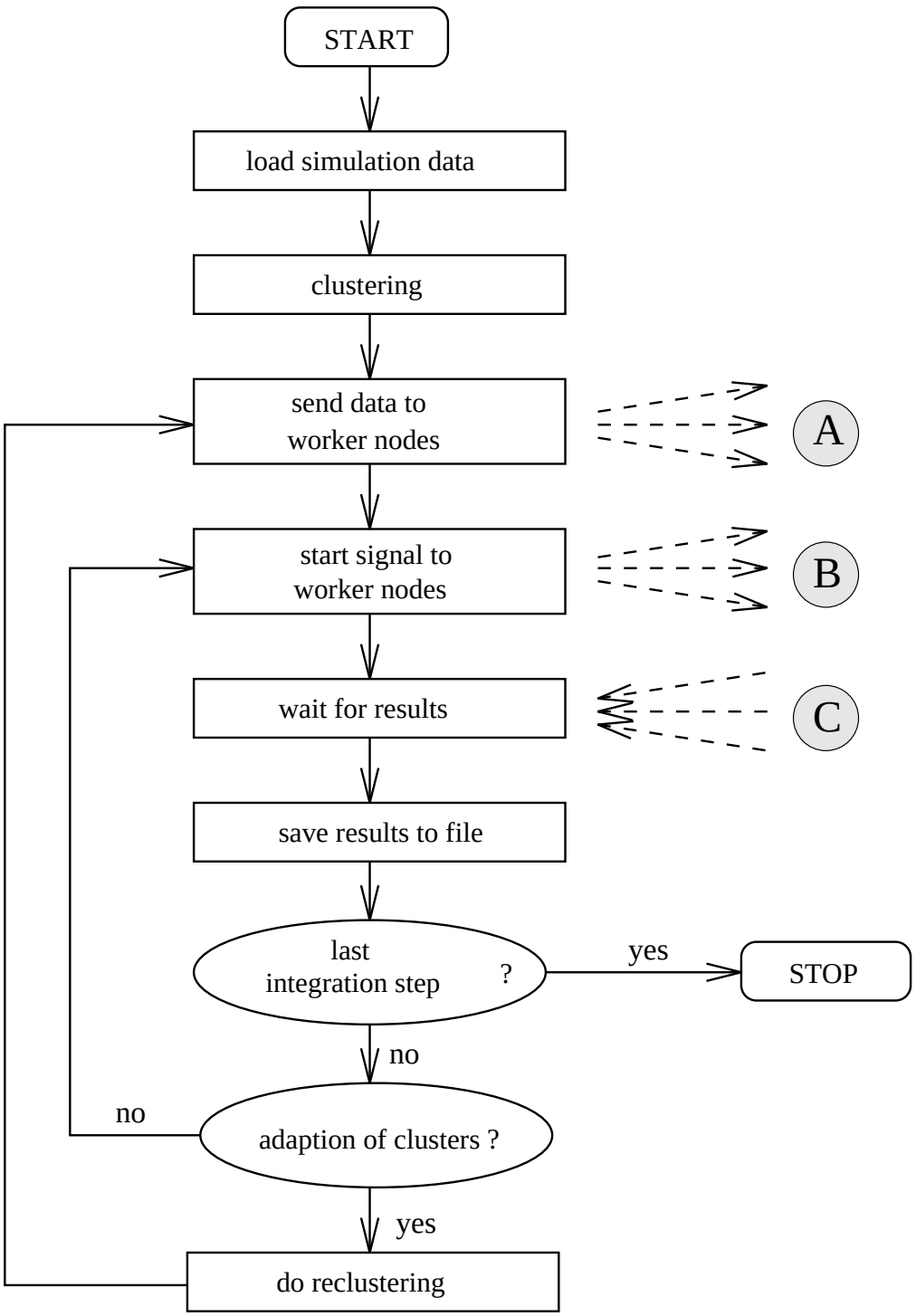

Figure 5.1: Program flow on the master node. The arrows marked with capital letters symbolize data exchange with worker nodes.

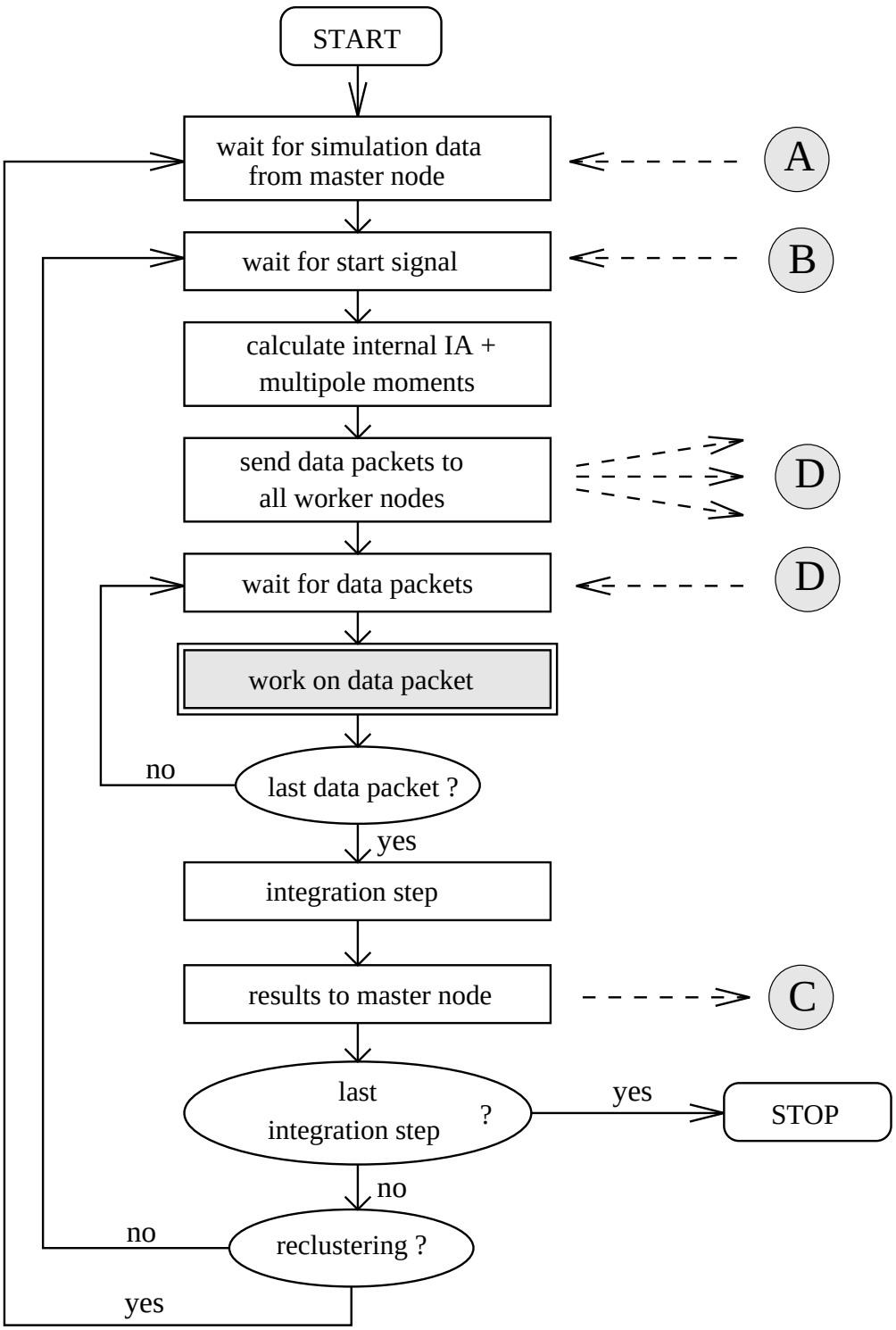

Figure 5.2: Program flow on a "worker node". During communication steps A, B and C data are exchanged with the master node; during communication step D data are exchanged with other worker nodes. The abbreviation "IA" stands for "InterAction".

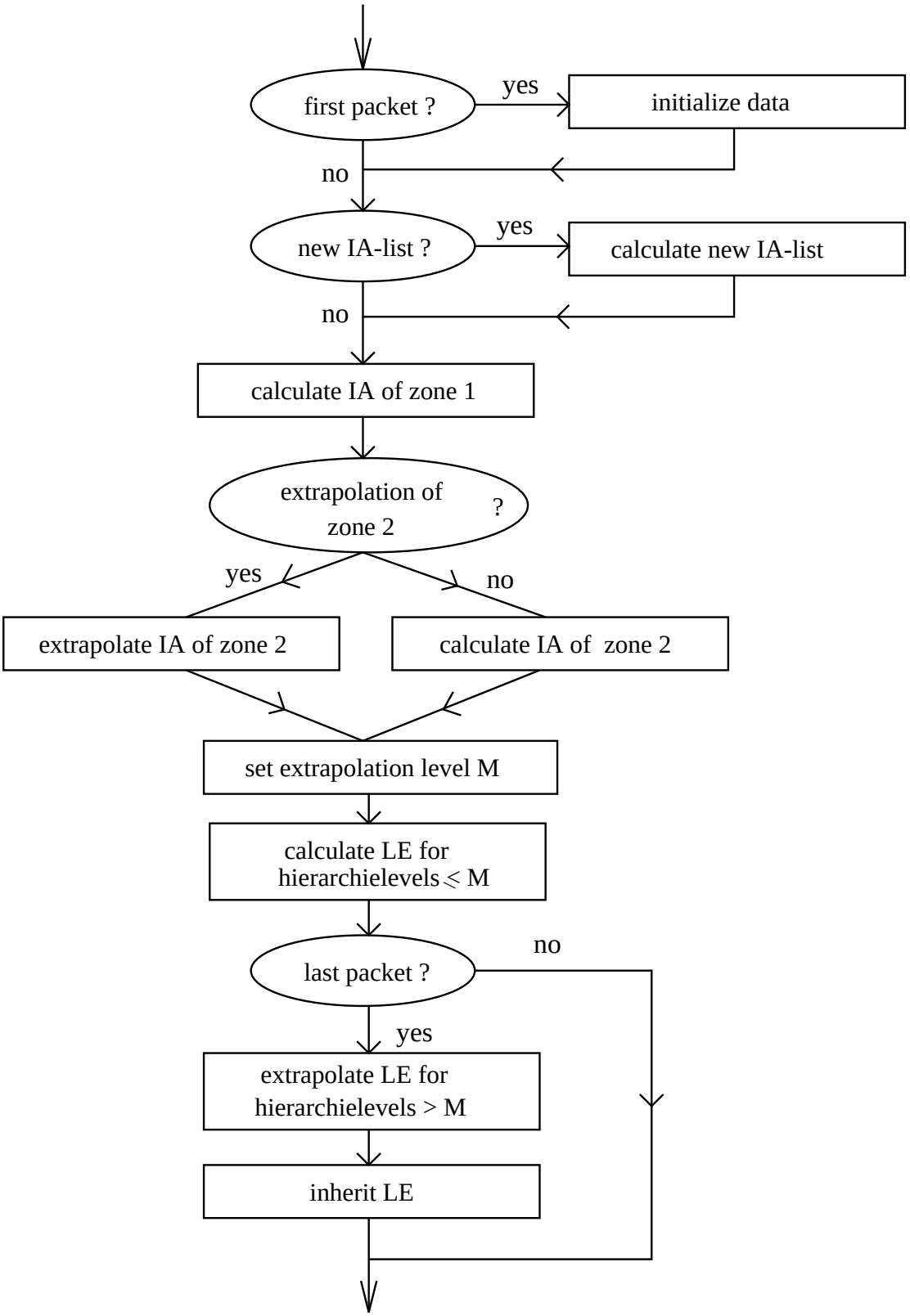

Figure 5.3: Work to be done on a received data packet. A data packet contains atom positions and mulipole expansions from other worker nodes. The abbreviation "LE" stands for "Local Expansion"

Chapter 6

Methods

EGO uses a modified Verlet method integration scheme using distance classes and a FMM to compute dynamics. This Chapter will discuss the energy function and integration method used by EGO.

6.1 Numerical Tasks in Molecular Dynamics Simulations

6.1.1 Energy Function

In this section we review the computational aspects of molecular dynamics simulations and discuss the relationship between the energy function used by EGO and that used by the programs CHARMM [3, 4] and X-PLOR [6], to which it is closely related.

Computer simulations of biological macromolecules are based on a classical mechanical model of biomolecules. For the nuclei of the N atoms of a molecule the Newtonian equations of motion ($i = 1, 2, \ldots N$) are assumed to hold

$$m_i \ddot{\vec{r}}_i \;=\; -\nabla_i E(\vec{r}_1, \vec{r}_2, \ldots, \vec{r}_N) \tag{6.1}$$

where $\vec{r}_i$ denotes the position of the i-th atom. Here we have used the notation $\nabla_i = \partial/\partial\vec{r}_i$. The function E

$$E \;=\; E_B \,+\, E_\theta \,+\, E_\phi \,+\, E_{El} \,+\, E_{vdW} \,+\, E_H \,+\, E_I \tag{6.2}$$

defines the total energy of the molecule. It is comprised of several contributions which correspond to the different types of forces acting in the molecule. The first contribution, E_B, describes the high frequency vibrations along covalent bonds, the second contribution, E_θ, the bending vibrations between two adjacent bonds and the third contribution, E_ϕ, the torsional motions around bonds. The fourth contribution, E_{El}, describes electrostatic interactions between partial atomic charges, the charges being centered at the positions of the atomic nuclei. The next term, E_{vdW}, accounts for the van der Waals-interactions between non-bonded atoms in the molecule, E_H stands for the energy of hydrogen bonds, and the last term, E_I, describes so-called improper motions of one atom relative to a plane described by three other atoms. Various research groups have developed functional representations and corresponding force constants which attempt to faithfully represent atomic interactions and dynamic properties of biomolecules [38, 3, 41, 42]. The program which we have developed is based on the energy representation of CHARMM [3, 4]. Actually, our program can read[1] a file of force parameters which has a format identical to that used by X-PLOR [6], a simulation program closely related to CHARMM. As a result, any adaptation of force constants suggested in the framework of CHARMM or X-PLOR can be readily transferred to our program.

[1] this is described in Section 2.3 of this manual.

6.1.2 Integration Methods

The integration method of the Newtonian equations of motion employed by our program is the Verlet algorithm [40]. This method determines the positions $\vec{r}_i(t+\Delta t)$, of atoms i at the instant $t+\Delta t$ according to the formula

$$\vec{r}_i(t+\Delta t) \;=\; 2\vec{r}_i(t) - \vec{r}_i(t-\Delta t) + \vec{F}_i(t)(\Delta t)^2/m_i \tag{6.3}$$

where $\vec{F}_i(t)$ stands for the sum of all forces acting on the i-th atom at time t, i.e.,

$$\vec{F}_i(t) = -\nabla_i E(\vec{r}_1(t), \vec{r}_2(t), ...\vec{r}_N(t)) \quad . \tag{6.4}$$

While integrating the Newtonian equations of motion computer time is spent mainly on evaluation of the two-particle interactions, i.e., of interactions originating from the Coulomb potential E_{El} and the van der Waals energy E_{vdW}. The programs CHARMM and X-PLOR avoid the prohibitive computational effort of an exact evaluation by allowing a cut-off for these interactions; this assumes that these interactions do not contribute much to the dynamics for pairs of atoms separated beyond a certain distance. (See below.)

We have not introduced such a criterion into our program. Rather than providing a cut-off option we introduced an option which makes it possible to evaluate the Coulomb interaction in a hierarchical way such that, according to a hierarchy of inter-particle distances, Coulomb forces are updated with different frequencies. Such an algorithm has been suggested in [43] and is described in [22, 20] and in Section 6.2.1 of this documentation. An alternative method for an efficient evaluation of Coulomb forces, the *Fast Multipole Algorithm*, has been developed by Greengard and Rokhlin [18, 19, 28] and is used in EGO simultanously for rapid evaluation of long range forces. The combination of both algorithms we termed FAMUSAMM.

Calculation of van der Waals and Coulomb forces is the most time consuming task in molecular dynamics calculations. The forces connected with the chemical bonds of biopolymers are determined much more rapidly during program execution. Because of the essentially linear arrangement of biopolymers the respective calculations can be readily ordered in a linear fashion and, therefore, a strategy for parallel computation of forces connected with chemical bonds is straightforward. Hence, we will not explain how these interactions are evaluated.

We would like to close this section with a brief description of the input-output requirements of our molecular dynamics program. As input the program needs a file of force parameters, a PDB file of atomic coordinates in protein data bank format, and a PSF protein structure file with definitions of bonds, dihedral and improper angles, etc. The file formats are identical to those of CHARMM and X-PLOR. As output the program delivers atomic coordinates in an internal format which may be converted on the host computer into any format for analysis of trajectory properties by CHARMM, X-PLOR or other programs.

6.2 Methods to Increase Efficiency

Computer simulations of a classical many-particle system are based on the solution of the Newtonian equations

$$m_k \frac{d^2}{dt^2}\vec{r}_k = \vec{F}_k(\vec{r}_1, \ldots, \vec{r}_{\mathcal{N}}) \quad , \tag{6.5}$$

where $\vec{r}_k$ denotes the position and m_k the mass of particle k, $k = 1, 2, \ldots, \mathcal{N}$, $\mathcal{N}$ being the number of atoms in the molecule. $\vec{F}_k(\vec{r}_1, \ldots, \vec{r}_{\mathcal{N}}) = -\vec{\nabla}_k E_{pot}(\vec{r}_1, \ldots, \vec{r}_{\mathcal{N}})$ represents the force acting on particle k.

A commonly used algorithm to integrate Equations (6.5) is the 'Verlet algorithm' [40]. According to this algorithm, the configuration $\vec{x}_{i+1} = (\vec{r}_1, \ldots, \vec{r}_{\mathcal{N}}) \in \boldsymbol{R}^{3\mathcal{N}}$ of a macromolecule (the index denotes the integration step) at instance $(i+1)\Delta t$ is

$$\vec{x}_{i+1} = 2\vec{x}_i - \vec{x}_{i-1} + (\Delta t)^2 \vec{f}(\vec{x}_i) \quad , \tag{6.6}$$

where $\vec{x}_i$ and $\vec{x}_{i-1}$ denote the configurations at time $i\Delta t$ and $(i-1)\Delta t$, respectively, and $\vec{f}(\vec{x}_i) = \left(\frac{\vec{F}_1}{m_1}, \ldots, \frac{\vec{F}_{\mathcal{N}}}{m_{\mathcal{N}}}\right)$ denotes the accelerations[2] acting at time $i\Delta t$. Δt is the integration step size. This algorithm is exact to second order in Δt.

$\vec{f}(\vec{x}_i)$ needs to be computed for every integration step. This vector, in general, will depend on the positions of all particles in the system. As a result, evaluation of $\vec{f}(\vec{x}_i)$ is the computationally most intensive part of a simulation. Furthermore, the computational effort increases with the square of the number of particles. It appears highly desirable to reduce the number of arithmetic operations in simulation calculations. Three approximation schemes which reduce the number of arithmetic operations will be described now.

6.2.1 Distance Cut-Off Scheme

The most commonly used approximation cuts-off all pair interactions beyond a certain maximum distance R_{cut}. The cut-off is achieved by multiplying the force by a 'cut-off'-function, which depends only on the distance and reduces the force to zero in a continous, differentiable, manner. A typical 'cut-off'-function assumes the constant value unity up to a distance R_{on}, then decreases to zero between R_{on} and R_{cut} and remains zero for all distances greater than R_{cut}. As a result, only interactions up to the distance R_{cut} have to be evaluated.

The appropriate choice of R_{cut} is a compromise between accuracy and computing time. On the one hand, the smaller R_{cut} is chosen, the faster the computation will be carried out. On the other hand, a large R_{cut} will reduce the error of the approximation.

The 'cut-off', although often employed, cannot be an appropriate approximation for molecular dynamics simulations since a neglect of the Coulomb interaction is believed to cause major rearrangements within the molecule. An improved approximation scheme involves ordering of the long-range interactions according to distance classes [43], as described below. A similar algorithm had been suggested in [36] for short-range forces in Lenard-Jones liquids.

6.2.2 Multiple Time Step Method

A 'distance class' is defined as follows: Let $\{R_0 = 0, R_1, \ldots, R_n\}$ be a set of radii with $R_j < R_{j+1}$ for all $j = 0, 1, \ldots, n-1$. Then the set of particles k with positions $\vec{r}_k$ satisfying the condition $R_j \leq |\vec{r}_i - \vec{r}_k| < R_{j+1}$ is called the *distance class j with respect to particle i* . The class scheme is illustrated in Figure 6.1.

The basic idea of the distance class algorithm relies on the observation that the change in time of pair interactions acting between two particles in general is smaller the further they are separated. Thus, it is possible to approximate the time development of pair interactions by a 'step function', i.e. the interactions are assumed to remain constant during a number of integration steps. For closely spaced particles the 'step function' extends only over a single instance in time, for particles separated further the 'step function' extends over several instances in time, the number of steps it spans increasing with the increasing separation between the particles. The 'step function' extending over j steps implies that the force has to be computed

[2] We will refer to this quantity as a force, even though it is strictly an acceleration.

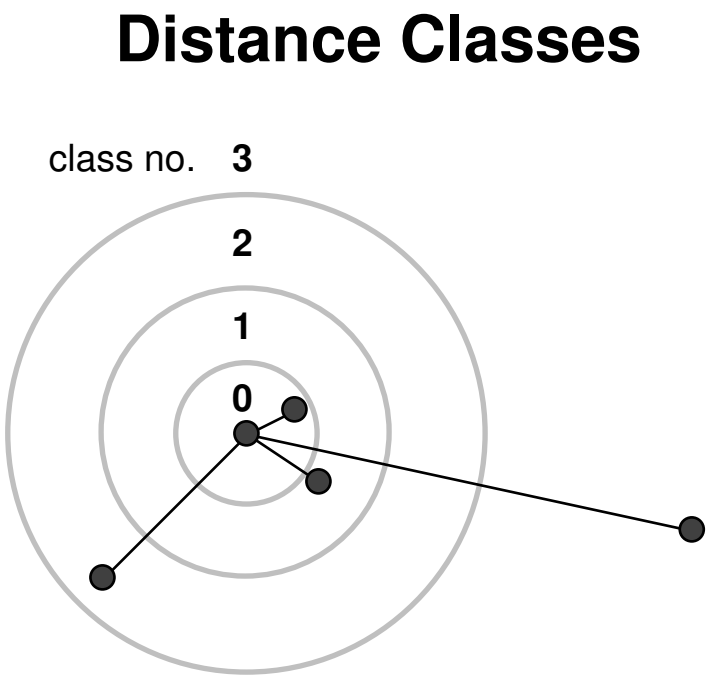

Figure 6.1: Scheme of the distance classes for a particle located at the center. The distance classes are numbered $0, 1, 2, 3$. One other particle and its distance to the center particle is shown for each distance class.

only once during j integration steps. For any given particle one can order all other atoms according to the distance classes introduced above. The interactions with atoms in the inner distance classes have to be computed more often than those of the outer classes. Assuming that the forces originating from the particles within distance class j are evaluated once every 2^j integration steps, an obvious choice of a distance class algorithm is of the form

$$\vec{x}_{n_j i+k+1} := 2\vec{x}_{n_j i+k} - \vec{x}_{n_j i+k-1} + (\Delta t)^2 \sum_{j=0}^{n} \vec{f}^{(j)}_{n_j i+2^j \lfloor k/2^j \rfloor} \quad , \tag{6.7}$$

where k, denoting the integration step, runs from 0 to $n_j - 1$. The index i counts groups of integration steps, each group consisting of $n_j = 2^n$ single integration steps.

One such group will be referred to as a 'macro-integration step'. The number of distance classes is $n+1$. The force $\vec{f}^{(0)}$ is caused by the particles of the innermost (0-th) distance class, $\vec{f}^{(1)}$ by the particles of the class next to the innermost and so on. Force $\vec{f}^{(n)}$ is caused by the particles of the outermost (n-th) distance class. The brackets ($\lfloor \ldots \rfloor$) denote the floor function, i.e., the positive integer less than or equal to the argument.

The corresponding computation scheme is illustrated in Table 6.1, where the forces used for the actual integration step are listed for the different distance classes. The numbers represent the integration step, during which the corresponding force has been computed. The boxed numbers represent forces that need to be evaluated, while the non-boxed forces need not be computed, since they can be copied from previous integration steps.

As explained above, the forces of the outer distance classes, according to the distance class algorithm, are not updated within *every* integration step. Instead, the most recently evaluated forces are used as an extrapolation for the actual forces. As a result of this approximation

integration step	distance class				
	0	**1**	**2**	**3**	...
0	$\boxed{0}$	$\boxed{0}$	$\boxed{0}$	$\boxed{0}$	...
1	$\boxed{1}$	0	0	0	...
2	$\boxed{2}$	$\boxed{2}$	0	0	...
3	$\boxed{3}$	2	0	0	...
4	$\boxed{4}$	$\boxed{4}$	$\boxed{4}$	0	...
5	$\boxed{5}$	4	4	0	...
6	$\boxed{6}$	$\boxed{6}$	4	0	...
7	$\boxed{7}$	6	4	0	...
8	$\boxed{8}$	$\boxed{8}$	$\boxed{8}$	$\boxed{8}$	...
9	$\boxed{9}$	8	8	8	...
10	$\boxed{10}$	$\boxed{10}$	8	8	...
$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\ddots$

Table 6.1: Computation scheme for the distance class algorithm. Shown are the forces originating from the different distance classes required during several integration steps. Boxed forces need to be computed; the others can be copied from previous integration steps.

the total energy of the system will not be constant. Other effects on MD simulation of such extrapolation schemes are discussed in [20].

In EGO two recently evaluated forces are used for extrapolation, employing the DC-1d method. As in [20] shown, MD simulations are least affected by the extrapolation method DC-1d. Let $\vec{f}^{(j)}_{n_j i+k}$ be the force on an atom originating from distance class j for an integration step $n_j i + k$ between the two recently evaluated forces $\vec{f}^{(j)}_{n_j i}$ and $\vec{f}^{(j)}_{n_j(i-1)}$.

The DC-1d method is defined through

$$\vec{f}^{(j)}_{n_j i+k} = a_k^{(j)} \vec{f}^{(j)}_{n_j i} + b_k^{(j)} \vec{f}^{(j)}_{n_j(i-1)} \tag{6.8}$$

and the coefficients through

$$a_k^{(j)} = \frac{3n_j^2 - 2n_j + 1}{n_j(n_j+1)} - 3k\frac{n_j - 1}{n_j(n_j+1)} \qquad \text{and} \qquad b_k^{(j)} = 1 - a_k^{(j)} \quad . \tag{6.9}$$

6.2.3 Structure Adapted Multipole Method

The *Structure Adapted Multipole Method* (SAMM) [32] takes advantage of knowledge about structural and dynamical features of biomolecules and helps to reduce the number of orders of multipole expansions for an approximate representation of the Coulomb potential. Large biomolecules are decomposed into small *structural units*. The electrostatic properties of these units can be approximated by their monopole or dipole moment. In our approach the force field generated by a group of atoms is approximated by an electrostatic monopole if the net charge does not equal zero (charged group of atoms), otherwise the force field is approximated by the dipole moment (dipolar group of atoms).

As the first non vanishing multipole moment does not depend on the selected reference point, one can minimize the error used by this approximation by selecting an *optimal* reference

point. In [32] for a charged group of atoms the optimal reference point

$$r_c = \frac{\sum_i q_i \vec{r}_i}{\sum_i q_i} \tag{6.10}$$

was derived, where $\sum_i$ denotes the sum over all atoms which belong to the specified atom group. As a result of this optimization, the dipole moment vanishes, and we effectively are accurate up to the dipole moment. For dipolar units the optimal reference point

$$\vec{r}_d = \frac{1}{3p^2} \overline{\overline{Q}}^0 \vec{p} - \frac{1}{12p^4} (\vec{p} \overline{\overline{Q}}^0 \vec{p}) \vec{p} \tag{6.11}$$

was derived, where $\vec{p}$ denotes the dipole moment $\sum_i q_i \vec{r}_i$. The components of the quadrupole tensor $\overline{\overline{Q}}^0$ are given by

$$Q^0_{\alpha\beta} = \sum_i q_i (3 r_{i\alpha} r_{i\beta} - \vec{r}_i^{\,2} \delta_{\alpha\beta}) \quad . \tag{6.12}$$

For a hierarchical treatment of electrostatic interaction these small structural units are grouped into larger aggregates — we call such aggregates *clusters* — at different levels of resolution. Usually EGO groups six structural units into a cluster, four clusters into a supercluster, and so on. In order to minimize the error of the electrostatic representation by the first non vanishing multipole moment, one has to maximize the compactness of the clusters. We want all clusters to have approximately the same size on every hierarchy level. This is a non trivial problem. In EGO a neural network algorithm for vector quantization [30] is used to solve this problem.

The FMM [19, 28] uses a local Taylor expansion — often called 'local expansion' — of the electrostatic potential for every level of resolution to sum up forces generated by sufficiently far separated structural units or aggregates. In our implementation forces arising from atoms which are closer then about 10 Å, called the innermost interaction zone (IIZ), are calculated exactly. In Figure 6.2 the interaction scheme of the Structure Adapted Multipole Method is illustrated. It shows how atoms, structural units and clusters contribute to the force acting on a selected atom.

6.2.4 Fast Multiple Time Step Structure Adapted Multipole Method

A closer look at the algorithm reveals that the computational effort for the evaluation of local expansions is of the same magnitude as for the forces from the IIZ. It is possible to use the Multiple Time Step Method described in Section 6.2.2 to avoid unnecessarily frequent computations of the local expansions and of forces from the IIZ. In our approach we divided the IIZ in two distance classes. The forces between all atoms which are closer then 5 Å are calculated exactly in every integration step. The sum of forces from atoms of the second distance class (5 - 10 Å) however is calculated exactly only every other integration step. The values of two previously calculated integration steps are used to extrapolate the forces for the steps between two exact integration steps. As a result of this the computational effort for the IIZ is reduced by a factor of almost two.

As explained above, the influence of atoms separated by more then 10 Å is represented through the local expansions. Analogous to the time development of forces the time development of the coefficients of local expansions varies slowly and can be extrapolated for a certain number of integration steps without risking too large errors. Thus the time consuming calculation of local expansions is avoided periodically.

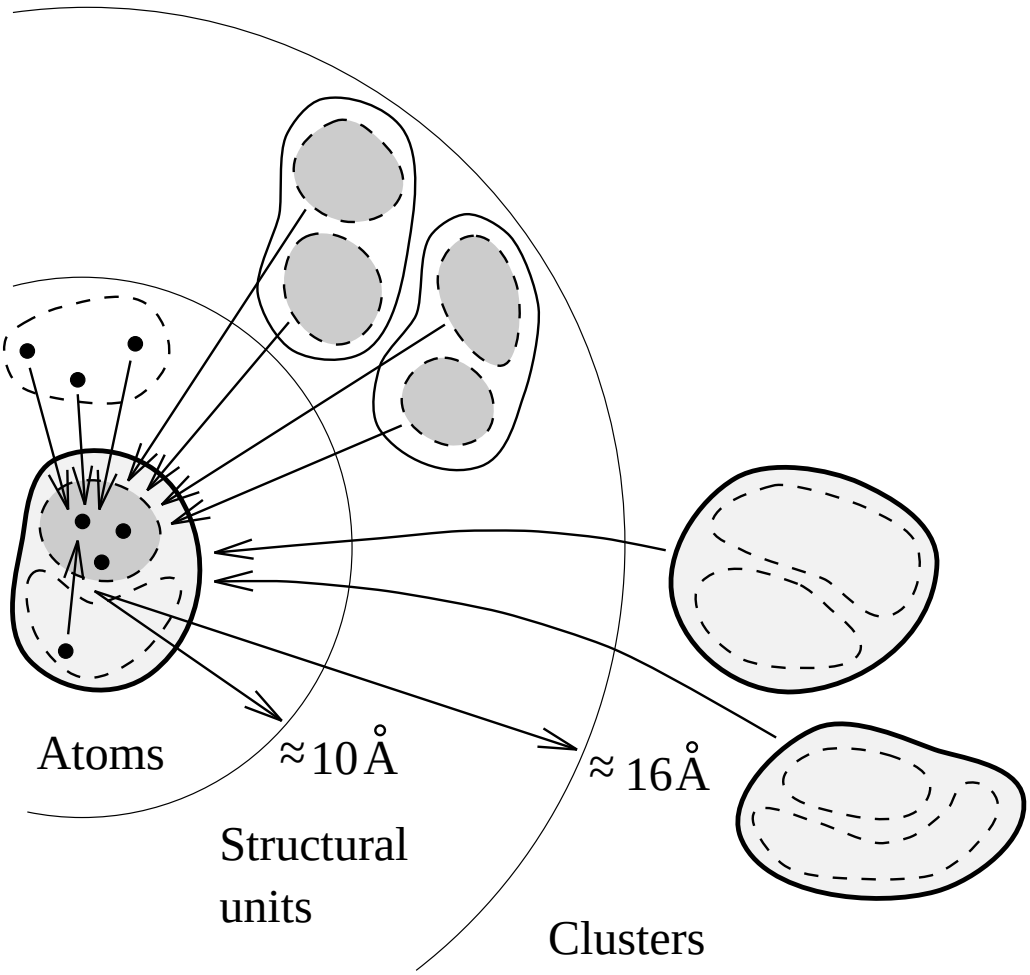

Figure 6.2: Interaction scheme of the "Structure Adapted Multipole Method" (SAMM). For clearity only two structural units per cluster are shown. The arrows starting from clusters and structural units indicate contributions to the local expansions of the cluster and structural unit which contains the selected atom. Interactions of atoms closer than about 10Å are calculated exact.

The combination of the Multiple Time Step Method and the Structure Adapted Fast Multipole Method is called *FAst MUltiple time step Structure Adapted Multipole Method* [9] (FAMUSAMM). The computational effort of this method scales with $O(N)$.

6.2.5 Conformational Flooding

'Conformational Flooding' is a novel method to study and to predict conformational motions in macromolecular systems (especially in proteins) on a microsecond time scale. Such motions typically occur as conformational (structural) transitions between distinct conformational substates [12, 1]. Such motions may be localized, as for example ring flips, or collective in nature and quite complex, like T-R-transitions, or the gating of ion channels. With few exceptions, conformational motions are slow on the MD accessible time scale with mean transition times ranging from nanoseconds to hours. For a review on conformational transitions, see, e.g., Ref. [13]. For theoretical studies, see Refs. [10, 17, 35, 14, 24].

'Conformational Flooding' aims at these rare events, which, at present, cannot be predicted

with traditional molecular dynamics (MD) simulations. Given an initial conformation of the system, the method identifies one or more product states, which may be separated from the initial state by free energy barriers that are large on the scale of thermal energy. It also provides approximate reaction paths, which can be used to determine barrier heights or reaction rates with the usual techniques like umbrella sampling [37]. The method employs an artificial potential that destabilizes the initial conformation and, thereby, lowers free energy barriers of structural transitions. As a result, transitions are accelerated by several orders of magnitude and thus may be observed in MD-simulations.

Conformational flooding has a variety of applications in several fields, e.g., as a tool for protein structure determination or conformational search, to check the stability of protein models, to predict functional motions, or to improve estimates of thermodynamic quantities such as free energies and entropies for proteins, polymers or glasses.

A typical 'flooding-simulation' involves several steps.

(1) Prepare the system and carry out conventional MD-simulations.

(2) After spending many CPU-hours you realize that the conformational motion of interest does not occur within available simulation time; you decide to use 'conformational flooding'.

(3) Decide which atoms the destabilizing forces shall be acted upon (e.g., C-alpha atoms).

(4) Use an equilibrated trajectory of an MD-run (as long as possible, e.g., several 100 ps) to generate an approximate description of the initial conformational substate by 'mkflood'. This description is referred to as a 'flooding matrix'.

(5) Estimate an appropriate flooding strength.

(6) Switch on the destabilizing flooding potential (derived from the flooding matrix) and run a 'flooding' simulation.

(7) Observe the value of the flooding potential as it evolves during the flooding simulation. A sudden jump to small values indicates the conformational transition you look for.

(8) If no conformational transition is observed within available computer time, the flooding strength was probably chosen too small. Go to (5), maybe to (3).

(9) Analyze the observed transition, look for further transitions starting with the new structure, or write a paper.

These steps are explained in detail below.

6.2.5.1 Theoretical Background

This section reviews the relevant statistical mechanics underlying 'conformational flooding'. It is certainly not necessary to understand everything in detail here, however, knowledge on how and why things work helps to obtain better results.

We will treat conformational transitions in a general framework, but focus on *collective* conformational transitions. Mainly due to entropic barriers, these are generally slow in terms of transition rates and occur on time scales above nanoseconds. However, an actual event of barrier crossing may be as fast as a few picoseconds. For a study of these motions one has (a) to search for distinct low-energy conformations, (b) to find reaction paths connecting the

conformations in configuration space, and (c) to estimate transition rates or mean transition times. 'Conformational flooding' provides a means to carry out tasks (a) and (b). Estimates can be derived for task (c), but here established techniques are more accurate and, therefore, are recommended.

The potential energy landscape of proteins is quite complex, and so is the free energy landscape. To reduce that complexity, we develop an effective, coarse-grained description with an adjustable level of coarse graining. We will proceed in three steps. First, we will introduce the notion of 'conformation space' as a subspace in configuration space. On this subspace we will consider a free energy landscape. Second, we will motivate a proper choice of linear collective coordinates. Third, these 'conformational coordinates' will serve to construct a substate model in terms of a harmonic free energy.

From a conventional MD-simulation one obtains an ensemble of S structures ('snapshots') $\mathbf{x}_i$, $i = 1 \ldots S$, where $\mathbf{x}_i$ denotes the $3N$ dimensional configuration vector consisting of the N atomic positions of the N atoms within the system selected to be subjected to the flooding forces. This ensemble is used to estimate the configuration space density $\rho^{\mathbf{x}}(\mathbf{x})$, which characterizes the initial (known) conformational substate.

We can obtain a coarse-grained description of $\rho^{\mathbf{x}}$ (i.e., of the initial substate) by considering a number m ($1 \leq m \leq 3N$) of 'conformational' (typically collective) degrees of freedom $(c_1, \ldots, c_m)$, which are assumed to be involved in the conformational motion. Here, m enters as an adjustable parameter, as does the selection of the N atoms to be affected during the flooding simulation.

The m conformational coordinates are extracted from an unperturbed MD simulation by means of a principal component analysis [15]. Below, the averages $\langle \ldots \rangle$ denote ensemble averages over the unperturbed MD trajectory.

From the covariance matrix $\mathbf{C} := \langle (\mathbf{x} - \langle \mathbf{x} \rangle)(\mathbf{x} - \langle \mathbf{x} \rangle)^T \rangle$ we derive a symmetric, positive definite $3N \times 3N$-matrix $\mathbf{A}^{-1} := \mathbf{C}$, which serves to approximate the configuration space density $\rho^{\mathbf{x}}$ in terms of a multivariate Gaussian distribution,

$$\rho^{\mathbf{x}}(\mathbf{x}) \approx Z^{-1} \exp[-\frac{1}{2}(\mathbf{x} - \langle \mathbf{x} \rangle)^T \mathbf{A} (\mathbf{x} - \langle \mathbf{x} \rangle)] \,, \tag{6.13}$$

where Z is an appropriate normalization.

Care has to be taken that rigid body motions (rotations and translations) are prohibited during that process, since these would 'smear out' the configuration space density.

Diagonalizing $\mathbf{A} = \mathbf{Q}^T \mathbf{\Lambda} \mathbf{Q}$ with orthonormal $\mathbf{Q} \in \mathcal{R}^{3N \times 3N}$ and diagonal $\mathbf{\Lambda} = (\delta_{ij} \lambda_i)_{i,j=1,\ldots,3N}$ yields collective coordinates $\mathbf{q} = \mathbf{Q}(\mathbf{x} - \langle \mathbf{x} \rangle)$, which serve to simplify Eq. (6.13):

$$\rho^{\mathbf{x}}(\mathbf{q}) \approx Z^{-1} \exp[-\frac{1}{2} \mathbf{q}^T \mathbf{\Lambda} \mathbf{q}] \,. \tag{6.14}$$

For our coarse-grained description we select the m collective coordinates $\mathbf{c} = (q_1, \ldots, q_m)^T$ with smallest eigenvalues λ_i. That number m of conformational degrees of freedom $\mathbf{c}_i$ which are explicitly considered determines the level of coarse-graining.

On the subspace of the c_i we define a 'conformation space density' $\rho^{\mathbf{c}}(\mathbf{c})$ as the projected configuration space density

$$\rho^{\mathbf{c}}(\mathbf{c}) = \int d^{3N} x' \, \rho^{\mathbf{x}}(\mathbf{x}') \delta[\mathbf{c} - \mathbf{f}(\mathbf{x}')] \tag{6.15}$$

$$\approx Z^{-1} \exp[-\frac{1}{2} \mathbf{c}^T \mathbf{\Lambda}_c \mathbf{c}] \,, \tag{6.16}$$

from which we derive a free energy landscape $F(\mathbf{c})$,

$$F(\mathbf{c}) = -k_B T \ln \rho^{\mathbf{c}}(\mathbf{c}) \tag{6.17}$$

$$\approx \frac{1}{2} k_B T \mathbf{c}^T \mathbf{\Lambda}_c \mathbf{c}\,, \tag{6.18}$$

where k_B is the Boltzmann constant and T the temperature. The latter result is a harmonic approximation, which is used as a model for the initial substate. Note that this harmonic approximation of the *free* energy landscape differs from the harmonic approximation of the *potential* energy, which is employed in normal mode analysis. The main difference is that the former includes entropic contributions, whereas the latter does not.

The coarse-grained substate model in terms of $F(\mathbf{c})$ serves to design a 'flooding'-potential $V_{\mathrm{fl}}(\mathbf{c})$, which is to be included into the force field during MD-simulations, and which is supposed to accelerate conformational transitions. In agreement with our assumption that the conformational coordinates c_i describe conformational transitions sufficiently accurate, we define the flooding potential as a function of only these m degrees of freedom. Note that during a flooding simulation *all* (not only the m c_i) degrees of freedom are considered, so no real elimination of degrees of freedom takes place here.

Qualitatively, we modify the free energy landscape F as indicated in Fig. 6.3. In the figure, the bold line represents $F(\mathbf{c})$ in the vicinity of a substate (well) as a function of one particular c_i. Also shown is a free energy barrier separating the initial substate from other substates (which are not shown). The purpose of the flooding potential V_{fl} is to rise the free energy within the substate (thin line) as to destabilize that initial substate and to drive the system into another substate. As is also indicated in the figure, we require V_{fl} to be short-ranged, so that the barrier is unaffected. With that assumption, the free energy barrier height is reduced by a destabilization free energy ΔF indicated in the figure and defined below, and one expects a corresponding acceleration $\exp(\Delta F/k_b T)$ of conformational transitions. That process is termed 'conformational flooding'.

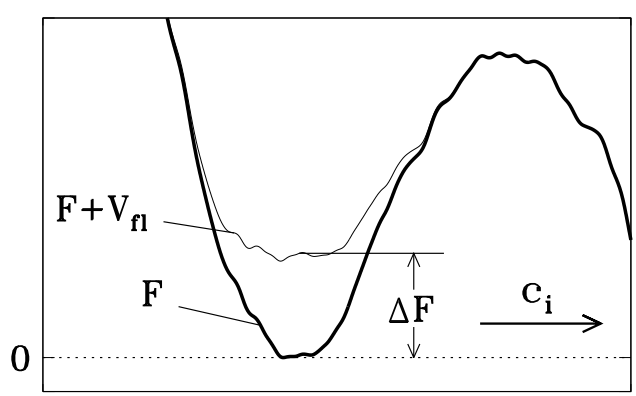

Figure 6.3: 'Conformational Flooding' lowers free energy barriers of conformational transitions and thus accelerates the transitions. The figure shows a cut through the free energy landscape $F(c_i)$ (bold line) along a particular conformational coordinate c_i in the vicinity of a conformational substate (well). To the right, a free energy barrier separates the substate from another one (not shown). Inclusion of the artificial flooding potential V_{fl} into the Hamiltonian of the system reduces the barrier height by an amount ΔF (thin line).

To ensure that $V_{\rm fl}$ 'fits' into the initial substate (cf. Fig. 6.4) we chose a (multivariate) Gaussian,

$$V_{\rm fl}(\mathbf{c}) := E_{\rm fl} \exp[-\frac{1}{2}\mathbf{c}^T \mathbf{\Lambda}_{\rm c}\mathbf{c}/\gamma^2] \tag{6.19}$$

(not to be mixed up with the density model $\rho^{\rm c}(\mathrm{c})$, which only accidentally happens to have the same functional form), where $E_{\rm fl}$ is the strength of the flooding potential, and $\gamma = \sqrt{E_{\rm fl}/k_B T}$ determines the overall extension of the flooding potential. With that choice, $V_{\rm fl}$ reduces the depth of the energy well uniformly without extending much into the high energy regions of conformation space, where barriers are to be expected.

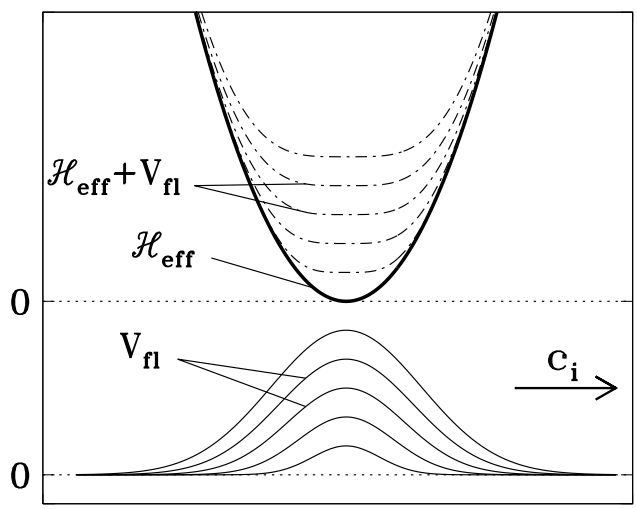

Figure 6.4: Harmonic effective Hamiltonian $\mathcal{H}^k_{\rm eff}$ (bold line) and Gaussian-shaped flooding potential $V_{\rm fl}$ for various flooding strengths $E_{\rm fl}$ (thin, solid lines) as a function of one conformational coordinate c_i. Adding $V_{\rm fl}$ to the effective Hamiltonian of the system decreases the depth of the substate well (dashed-dotted lines) uniformly.

We would like to stress that our flooding potential will not push the system towards any *preselected* destination in configuration space; hence, no bias is included as to which product state the system will move. Rather, the method is likely to follow transition paths of low free energy and thus should identify those neighboring conformational substates, to which also the unperturbed system ($V_{\rm fl} = 0$) would move at much slower time scales.

As a rule of thumb, Fig 6.5 provides an upper limit for the expected acceleration factor $\tilde{\alpha}$ for various flooding strengths *per degree of freedom*, $\epsilon_{\rm fl} = E_{\rm fl}/m$. Also given is the corresponding destabilization free energy $f = \Delta F/m$ per degree of freedom.

For a more detailed description, for estimates of the acceleration factor, and for two sample applications, see [21].

6.2.5.2 Parameters in the control file relevant for 'flooding'

The following parameters in the control file `ctl.lis` are used for conformational flooding:

```
Switch for using flooding
```

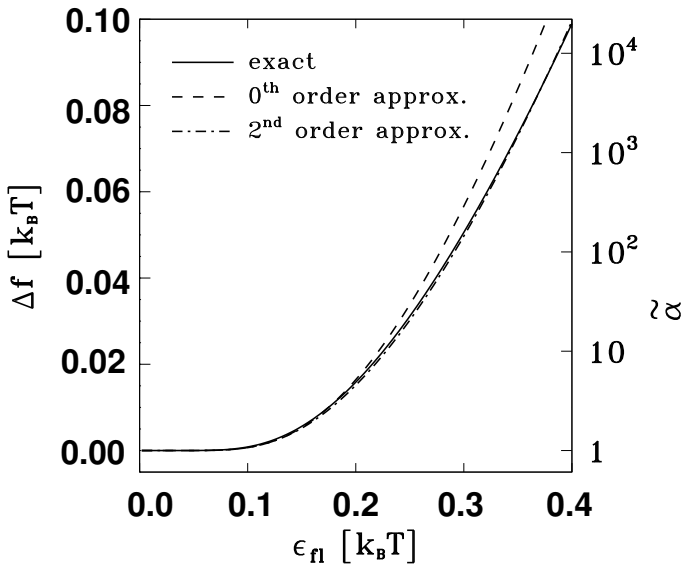

Figure 6.5: Expected acceleration factor $\tilde{\alpha}$ due to flooding strength $\epsilon_{\mathrm{fl}} = E_{\mathrm{fl}}/m$ *per degree of freedom.*

Set this switch to `TRUE` in order to include a flooding potential into the force field. The flooding matrix `flooding.lis`, which can be generated using `mkflood`, is required. Translation/rotation correction should always be switched on. Appropriate flooding energy strengths (see below) must be given. Set this switch to `FALSE` to perform a conventional MD simulation. In this case all parameters given below are ignored, except for the `Switch for using transl/rotation correction`

`Switch for using transl/rotation correction`

If set to `TRUE`, this switch eliminates rigid body movements like translations and rotations of the selected atoms during a simulation. This feature is required for performing flooding simulations. Since the atom selection is given in the file `flooding.lis`, that file is always required if rigid body movements are to be eliminated. Rigid-body elimination is also necessary for the unperturbed MD-run required to generate a flooding matrix [step (4) above]. Don't be confused by the fact that here, of course, you will not have a flooding-matrix at hand. Instead, a dummy-file `flooding.lis` can be created using `mkflood`, in which only the atom selection is contained (see the sample application given below).

`Switch for using adaptive flooding`

THIS OPTION IS NOT YET IMPLEMENTED.
PLEASE SET THIS SWITCH TO '`FALSE`'.

Adaptive flooding is an issue where indeed some knowledge on the theory section is beneficial. If you have not yet read that section, we recommend setting this switch to `FALSE`, in which case the `Time constant for adaptive flooding` is ignored. We emphasize, however, that

there are cases in which conformational flooding will work much better when using adaptive flooding.

When performing a flooding simulation, two quantities can be used to characterize and to adjust the flooding strength: (a) the flooding energy E_{fl}, which determines the 'height' of the flooding potential V_{fl} and (b) the average value (a running average with an appropriate time constant) of the *actual* value of the flooding potential V_{fl}, which is a function of configuration and, therefore, of time. It can be shown that this average is a good estimate for the flooding destabilization free energy ΔF, which in turn determines the expected acceleration factor $\alpha \approx \exp(\Delta F/k_b T)$ via the Boltzmann factor. In contrast, the relation between the flooding energy E_{fl} and the expected acceleration is not quite clear and depends sensitively on the quality of the Gaussian configuration space density approximation.

Thus setting a definite value for the flooding energy E_{fl} (switch: `FALSE`) makes it somewhat difficult to estimate its effect and typically entails a couple of trial-and-errors.

On the other hand, dynamically adjusting the flooding energy E_{fl} such that a definite value for the destabilization free energy ΔF (and, therefore, for the expected acceleration) is obtained (switch: `TRUE`) makes life much easier, but requires more experience in the interpretation of the development of the flooding energy in time.

`Initial energy for flooding in kT`

Initial flooding energy in units of thermal energy. `Target temperature in Kelvin` is used to translate that value into kcal/mol. Typical values are zero or small values, if increasing flooding strength is required, or a value equal to `Final energy for flooding in kT`, if constant strength is required. Depending of the `Switch for using adaptive flooding`, this energy is interpreted in two different ways: If the switch is set to `FALSE`, the energy denotes the flooding energy E_{fl}; if set to `TRUE`, a target value for the *actual* averaged value (running average with time constant `Time constant for adaptive flooding in s`) of the flooding potential V_{fl} is given here.

`Final energy for flooding in kT`

Maximum flooding energy used in units of thermal energy. `Target temperature in Kelvin` is used to translate that value into kcal/mol. `Flooding energy increase in kT/ps` is used to determine how fast this maximum value is reached. Depending on the `Switch for using adaptive flooding`, this energy is interpreted in two different ways: If set to `FALSE`, the energy denotes the flooding energy E_{fl}; if set to `TRUE`, a target value for the *actual* averaged value (running average with time constant `Time constant for adaptive flooding in s`) of the flooding potential V_{fl} is given here.

`Flooding energy increase in kT/ps`

The actual flooding energy used during the simulation is (`Initial energy for flooding in kT`)+(elapsed time)∗(`Flooding energy increase in kT/ps`) as long as this expression is smaller than the `Final energy for flooding in kT`. Otherwise, the flooding energy is set to `Final energy for flooding in kT`. This allows (1) to have constant flooding energy (increase 0), (2) linear increase with time and (3) linear increase with time followed by a constant

maximal value of the flooding energy.

`Time constant for adaptive flooding in s`

If `Switch for using adaptive flooding` is set `TRUE`, this time constant is used for averaging of the actual flooding potential in order to estimate the actual destabilization free energy ΔF. That average is compared with the target flooding energy, and E_{fl} is adjusted dynamically such as to minimize the difference of both.

6.2.5.3 Creating a 'flooding' matrix: `mkflood`

The program `mkflood` carries out the principal component analysis on an ensemble of structures (ego-files) and creates the appropriate flooding matrix `flooding.lis` required to carry out a flooding simulation. Typically, the quality of the flooding matrix will be the better, the larger the used structure ensemble is, i.e., the longer the used MD trajectory is.

To further enhance the accuracy of the flooding matrix, `mkflood` allows to extrapolate the parameters of the flooding matrix to infinite times. It does so by considering subsets i of the structure ensemble of increasing size ΔT_i and by performing a principal component analysis on each of these subsets. The function $\lambda(\Delta T) = a - b/\Delta T$ is then fitted to each of the appropriate parameters (the eigenvalues) of the matrices, and an estimate $\lambda(\Delta T \to \infty) = a$ is obtained. The sub-ensembles used are chosen to increase in size logarithmically, starting with a given minimal size, and up to the total ensemble available.

In 'dummy-mode', `mkflood` allows to generate a file `flooding.lis`, containing solely a list of selected atoms. This feature is useful to carry out a conventional MD-simulation with rotation/translation correction, as, e.g., required to generate an ensemble for the initial conformational substate.

The format of the program call is

```
mkflood <pdb-file> <psf-file> <start-num> <end-num> <min. #files>
        <#steps> <nr_of_dofs> ['<Sel.-string>']
```

with the following arguments:

`<pdb-file>`:	Coordinate file used to determine the numbers of the selected atoms.
`<psf-file>`:	Structure file used to determine the numbers of the selected atoms.
`<start-num>`:	Number of the first ego-file of the structure ensemble to be used for the principal component analysis.
`<end-num>`:	Number of the last ego-file used
`<min. #files>`:	minimal ensemble size used for the extrapolation to infinite times. The first flooding matrix will be computed from the ego-files `<start-num>...<start-num>+<min. #files>-1`. **Attention:** In order to avoid singularities (which result in an error message), take care that the minimal ensemble size is larger than the dimension of the configuration space considered, i.e., larger than three times the number of selected atoms.

<#steps>: Number of flooding matrices to be computed. Each of the matrices is computed from a different number of ego-files, these numbers are distributed logarithmically in the range <min. #files>...<end-num>-<start-num>. Use <#steps>=0 in order to avoid any extrapolation. In this case only one flooding matrix is computed from the complete ensemble given, and the value of <min. #files> is ignored.

<nr_of_dofs>: The number m of degrees of freedom used as 'flooding degrees of freedom' c_i. That number must be smaller than or equal to $3N-6$, where N is the number of selected atoms.

<Sel.-string>: Expression to select those atoms from the system, which (a) are used for the translation/rotation correction, (b) which are used for the principal component analysis to create the flooding matrix, and (c) onto which the flooding potential acts upon. A typical selection would include only one representative of groups of atoms, whose motion is strongly correlated (to save computer time), e.g., the alpha carbon of each group, and exclude those atoms of the system, which definitely do not actively take part in the conformational motion to be studied (e.g., solvent atoms).

The general syntax of the selection string is

[+|-]{A|R}{<number|num1-num2|string},

where

+ : includes the selected atoms/residues (default),
- : excludes the selected atoms/residues,
A : refers selection to atom numbers/names,
R : refers selection to residue numbers/names,
number : selects a single atom/residue,
num1-num2 : selects an atom/residue range, and
string : selects an atom/residue by string-match. Allowed wild-cards are * (string), % (single character), # (string of digits), and + (single digit), similar to the XPLOR selection scheme.

Example: The selection ' AC* AN* -RPRO' selects all carbon (C) atoms and all nitrogen (N) atoms except for those in prolines.

To create a dummy flooding-matrix-file to be used for translation/rotation correction, use:

```
mkflood <pdb-file> <psf-file> 0 0 0 0 0 ['<Selection-string>']
```

6.2.5.4 A sample application

Assume we want to predict the gating reaction of gramicidin A, a small ion channel. The initial structure entered into the flooding simulation described below is depicted in Fig. 6.6 (upper two panels); the predicted final configuration is depicted in Fig. 6.6 (lower two panels). Water molecules are drawn in blue. Note the red (dark) ring, which has moved outside the channel to enable the passage of ions through the channel. This gating has been observed by patch clamp measurements.

For the flooding simulation we proceed as follows:

Steps 1 and 2:

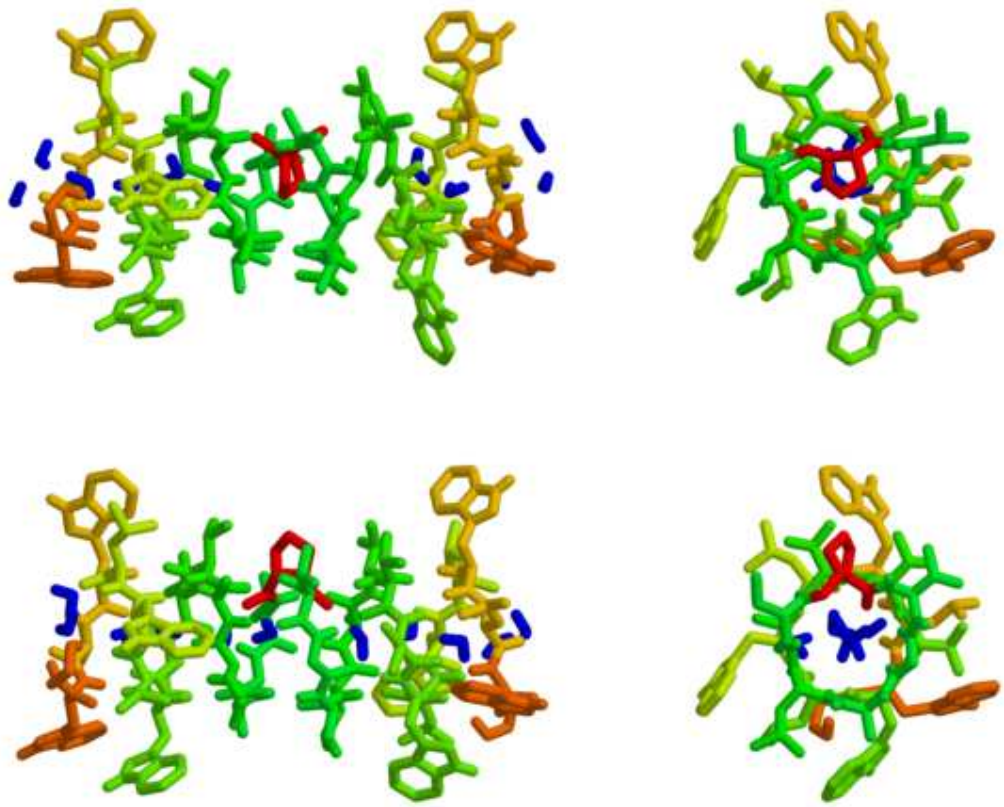

Figure 6.6: Initial structure (upper panels) and final structure (lower panels) of the gating reaction of gramicidin A. The initial structure has been modeled on the basis of NMR data; the final structure has been suggested by 'conformational flooding'.

We assume an equilibrated structure (`gram.pdb` and `gram.psf`) is available, which does not show considerable conformational motions within nanoseconds. In particular, no gating transition is observed on that time scale. The system contains a total of 353 atoms. The necessary `*.lis`-files are available or have been created with `xpl2lis` and, eventually, with `mkmaxw`.

Steps 3 and 4:

We perform an unperturbed MD-simulation of 300 picoseconds length to create an ensemble of structures, from which the flooding potential can be derived. For that purpose, coordinate files (ego-files) are written every 100 steps. A slight coupling to a heat bath (room temperature, 300 K) is used, and, **very importantly**, translation/rotation correction is switched on. To tell `EGO` which atoms to consider for the translation/rotation correction, we have to create a dummy flooding file `flooding.lis` first:

```
mkflood gram.pdb gram.PSF 0 0 0 0 0 'RDIOX ACA -R3 -R5 -R8 -R9 -R12'
```

Here, the selection string selects all atoms of the dioxilane ring (which has residue name `DIOX` in the pdb-file) as well as all alpha carbons of the backbone with several exceptions given explicitly to save computer time. This choice is motivated by the expectation that mainly the dioxilane ring and the protein backbone are actively involved in the gating conformational

transition. This is not to say that all other atoms will not be allowed to move; it is just that the force driving the destabilization of the initial configuration will not act upon the unselected atoms.

The dummy flooding file can now be used by `EGO` to create the 300 picosecond ensemble. The following control file `ctl.lis` is used:

```
33              Number of files.
shake.lis
coord.lis
...
forceout.lis
2               Requested number of nodes.
353             Number of atoms.
100             Freq. of analysis printout; next line is ...
A*
10              Frequency of writing restart file every analysis step!
0               Frequency of system call every analysis step.
ego2crd.sh
-5              Frequency of energy printout.
300000          Number of integration steps.
1e-15           Integration time step in s.
5               Order of exclusion list.
FALSE           Switch for Minimisation.
1               Friction factor (1.0=>no friction, 0.0=>no motion).
0.08            Maximum position movement per integration step in A.
TRUE            Switch for Equilibration.
300             Target temperature in Kelvin.
1e-12           Coupling time constant in s.
8               Number of distance classes.
...
0               Used type of cluster algorithm (0 is default).
-2              Number of hierarchy levels.
-18             Number of branches on last level.
500             Frequency of reclustering.
out/            Path for output data.
0.4             Scaling factor for special 1-4 electrostatic damping.
TRUE            Switch for stochastic boundary.
TRUE            Switch for harmonic restraints.
TRUE            Switch for SHAKE on hydrogens (Only bond length).
...
FALSE           Switch for using flooding.
TRUE            Switch for using transl/rotation correction.
FALSE           Switch for using adaptive flooding.
0               Initial energy for flooding in kT.
0               Final energy for flooding in kT.
0               Flooding energy increase in kT/ps.
1e-12           Time constant for adaptive flooding in s.
```

```
0 0             Number of user defined integers and doubles.
END
0               DEBUG: 1 == compare with exact forces
0               DEBUG: only should be used by a developer.
```

From the ensemble of 3000 structures (ego-files) we chose all except for the first 500 (which might be perturbed due to a non-relaxated initial structure) to create the flooding matrix with `mkflood`. We use extrapolation of the eigenvalues with a minimal number of 1000 structures and a total of 10 steps for the extrapolation. $m = 20$ degrees of freedom shall be 'flooded':

```
mkflood gram.pdb gram.PSF 500 3000 1000 10 20 'RDIOX
        ACA -R3 -R5 -R8 -R9 -R12'
```

Note that we used a selection string identical to the one used to create the dummy flooding file for the translation/rotation correction. This should always be done.

Steps 5 through 8:

After some trial-and-error and through observing the time development of the flooding potential after flooding has been switched on (as described below) we find that a flooding strength of $E_{\mathrm{fl}} = 50\,k_BT$ results in an average value of the flooding potential of about $10\,k_BT$. We observe the typical behaviour that the system 'feels' the flooding potential and within few picoseconds relaxes towards a slightly different structure, but still stays within the initial substate. This relaxation induces a drop of the initially large (about $25\,k_BT$) flooding energy down to the stable value of roughly $10\,k_BT$.

Taking this value as an estimate for the destabilization free energy ΔF, we expect an acceleration factor of $e^{10} \approx 20\,000$. This seems to be a proper value, since the patch clamp experiments tell us that the gating proceeds on a time scale of several microseconds.

[Instead of this trial-and-error we could alternatively have used adaptive flooding (set switch to `true`) and directly enter the desired destabilization free energy of $10\,k_BT$. We will, however, not consider this further here.]

We thus chose to use a constant flooding energy of $E_{\mathrm{fl}} = 50\,k_BT$ and carry out the flooding simulation. The following control file `ctl.lis` is used:

```
33              Number of files.
shake.lis
coord.lis
...
forceout.lis
2               Requested number of nodes.
353             Number of atoms.
1000            Freq. of analysis printout; next line is ...
A*
10              Frequency of writing restart file every analysis step!
0               Frequency of system call every analysis step.
ego2crd.sh
```

```
-50             Frequency of energy printout.
1000000         Number of integration steps.
1e-15           Integration time step in s.
5               Order of exclusion list.
FALSE           Switch for Minimisation.
1               Friction factor (1.0=>no friction, 0.0=>no motion).
0.08            Maximum position movement per integration step in A.
TRUE            Switch for Equilibration.
300             Target temperature in Kelvin.
1e-12           Coupling time constant in s.
8               Number of distance classes.
...
0               Used type of cluster algorithm (0 is default).
-2              Number of hierarchy levels.
-18             Number of branches on last level.
500             Frequency of reclustering.
out/            Path for output data.
0.4             Scaling factor for special 1-4 electrostatic damping.
TRUE            Switch for stochastic boundary.
TRUE            Switch for harmonic restraints.
TRUE            Switch for SHAKE on hydrogens (Only bond length).
...
TRUE            Switch for using flooding.
TRUE            Switch for using transl/rotation correction.
FALSE           Switch for using adaptive flooding.
50              Initial energy for flooding in kT.
50              Final energy for flooding in kT.
0               Flooding energy increase in kT/ps.
1e-12           Time constant for adaptive flooding in s.
0 0             Number of user defined integers and doubles.
END
0               DEBUG: 1 == compare with exact forces
0               DEBUG: only should be used by a developer.
```

After about 100 picoseconds (the actually observed transition time is a stochastic quantity and, therefore, may vary considerably; it is distributed very much like decay times of individual radioactive atoms) we observe a sudden jump of the flooding energy to very small values, indicating the conformational transition we wanted to study. Indeed, inspecting the structure after this energy jump reveals the ring flip shown in Fig. 6.6.

The ego output files written during the drop of the flooding energy provide an approximate reaction path of the conformational transition. It can be used to calculate a free energy profile (e.g., with umbrella sampling techniques), which in turn provides a better estimate for the transition rate.

If the value for the flooding potential were chosen too small, e.g., $30\,k_BT$, no conformational transition would have been observed.

A very large value for the flooding potential can cause artificial deformations of the structure, which can be detected either by visual inspection or through a significant increase of the

short-range energy contributions like 'bond', 'vdw', 'angle', or 'dihedral'.

In rare cases it may happen that 'conformational flooding' drives the system into a 'dead end', i.e., a path starting with low free energies, which does not traverse an energy barrier, but rather ends at a barrier. In such a case the flooding energy will decrease significantly, but it will not drop down to very small values as in the case of a conformational transition. In most cases re-running the simulation with slightly different parameters (e.g., flooding strength) will help here. Alternatively, one can try to reduce the number of selected atoms in such a way as to focus more and more at those atoms involved in the transition. Note also that frequent 'dead ends' can indicate that the value for the flooding strength has been chosen too large, and the system does not have enough time to 'search' for a low energy pathway.

Always keep in mind:

The conformational motions you would like to see within a flooding simulations may be too slow as to be observed. Although we did not yet find any limit for the acceleration factor, it seems unlikely that transitions on the time scale of seconds can be accelerated by as much as nine orders of magnitude.

Finally:

Should you succeed in predicting a conformational transition using conformational flooding we would appreciate receiving a note from you.

Appendix A

File Formats

A basic description of the input and output data files and their use is given in Section 2.2. In the following section we give a more detailed description of these files and their format. With the exception of the X-PLOR trajectory output files (DCD or crd), all of these data files are in ASCII format.

A.1 Input Files

The input files of EGO for a molecular dynamics simulation consist of atom coordinates, connectivity, energy parameter and a definition file for the so-called *structural units*. The atom *coordinate* and residue information is taken from Brookhaven Protein Data Bank (PDB) format files. The atom type, mass, partial charge, and connectivity information (bonds, angles, etc.) is taken from X-PLOR PSF files. The energy parameters defining force constants and equilibrium coordinates are taken from X-PLOR parameter (paramxxx.xxx) files. Furthermore, the structure adapted multipole method on which FAMUSAMM is based and which is responsible for the approximate but fast calculation of Coulomb forces, needs the definition of so-called *structural units*. These structural units are defined in the input file `units.def`.

A.1.1 The units definition file `units.def`

The file `units.def` located in the directory `$(HOME)/ego/utils` is a ASCII file and is needed for FAMUSAMM, which groups atoms into *structural units* in order to obtain rapidly converging multipole expansions. For every molecule structure (e.g., water, proteins, lipids, etc.) a corresponding subdevision into such structural units has to be defined in the file `units.def`. The utility program `xpl2lis` creates with the help of the `units.def` file the corresponding lis-file `units.lis`. In the `units.lis` file every structural unit is defined by a list of atom numbers which belong to that unit. Note that in EGO atom numbers start with '0', X-PLOR, PDB, PSF atom numbers start with '1'. Usually between three to seven heavy atoms belong to a structural unit. Units which contain no charged atoms are called *neutral units*. Units which contain charged atoms, but have no net charge are called *polar units*. All other units exhibit a net charge and are called *charged units*.

Currently structural units are defined for simple proteins, TIP3-water, POPC- and POPE lipids, retinal of the protein Bacteriorhodopsin, Gramicydin and Biotin. Here we give some rules you must follow if you want to define structural units for molecules not yet defined in the your `units.def` file:

- All atoms belonging to one unit *must* be conected via covalent bonds.
- The structural units should be compact in size.

- The best choice for the number of atoms in a structural unit is about four heavy atoms.
- The net charge of charged units should be bigger then 0.1 e.

The file format of the units definition file `units.def` is as follows:

- Tabulators and newlines work as a white space.
- Comments are introduced by an '!'. All characters up to the end of the line are skipped.
- For every residue name (e.g., TIP3, ALA, VAL, etc.) the subdevision into structural units is controlled by a hierarchy of three '{' levels. Note, that due to the PDB format a residue name has at maximum four characters.
- The first '{' level includes alternative versions of a unit definition. Different definitions are required, as different atom names may apear for one residue name. This happens usually for residues which term amino acids in a protein: An amino acid at the O-terminus of the protein contains two atoms with atom name 'OT1' and 'OT2'. The same amino acid contains only one atom named 'O' if it occurs in the chain.
- The next '{' level includes the structural units of one alternative.
- The third '{' level includes all heavy atoms, which make up a structural unit. Note, that only heavy atoms must be indicated, as hydrogen atoms automatically are included to the structural unit to which the heavy partner atom belongs. Thus, the same `units.def` file can be used for all-atom models and compound-atom models.
- The atom names within the third '{' level must be separated by ','.

Below, a section of the `units.def` file is listed:

```
TIP3 {  !*** The TIP3 water model consits of one O atom and two H atoms.
        !*** only one alternative.
        {       !*** only one structural unit.
                {OH2}
        }
}

VAL {   !*** two alternatives for Valin.
        !*** 1. alternative: VAL in the chain.
        {      !*** two structural units.
               {N, CA, C, O} {CB, CG1, CG2}
        }
        !*** 2. alternative: VAL at the O-terminus.
        {       !*** two structural units.
                {N, CA, C, OT1, OT2} {CB, CG1, CG2}
        }
}
```

A.1.2 Brookhaven PDB Atom Coordinate Files

The Brookhaven PDB file format can contain various information to describe molecular systems, but EGO uses only ATOM records to identify atom coordinates and stochastic boundary parameters (when used on a per atom basis, see Section 4.19) and Section 4.20. For a given atom, the atom type, mass and partial charge information is provided in the corresponding PSF file generated by X-PLOR from the PDB file and topology files.

ATOM records appear in the PDB file as follows (as an example we show parts of the file `pti.pdb`):

```
REMARK FILENAME="pti.pdb"
REMARK BPTI COORDINATES TAKEN FROM CRISTALLOGRAPHIC DATA W/O WATERS
REMARK HYDROGEN POSITIONS GENERATED USING HBUILD (2 ITERATIONS)
REMARK RMS FLUCTUATIONS (T. ICHEYE) FOR HEAVY PROTEIN ATOMS INCLUDED
REMARK DATE:24-Apr-89  02:35:12       created by user: heller
ATOM      1  HT1 ARG     1      27.077  26.629  -3.076  1.00  0.00      MAIN
ATOM      2  HT2 ARG     1      27.163  28.116  -2.262  1.00  0.00      MAIN
ATOM      3  N   ARG     1      26.522  27.417  -2.687  1.00  0.57      MAIN
...
ATOM    567  OT1 ALA    58      26.421  31.411  -8.486  1.00  0.74      MAIN
ATOM    568  OT2 ALA    58      25.216  33.274  -8.900  1.00  0.78      MAIN
END
```

The REMARK lines are comments which are ignored by EGO. The ATOM record consists of atom number, atom name, residue name, residue number, the x, y, z coordinates of the atom in Å, followed by friction parameter (9th column) and harmonic force (10th column) constant. The friction parameter is only used in case that the stochastic boundary is enabled in the control file `ctl.lis`, and atoms with non-zero harmonic force constants are bound to their reference position, taken from `coord.lis`, with the harmonic force constant. Units are generally given in terms of Å for distance, and kcal/mol for energy, therefore, a harmonic force constant, k, is given in units of $kcal/mol/\text{Å}^2$. The friction constant is given in ps^{-}1. Included in the EGO distribution is the X-PLOR-script `$(HOME)/ego/utils/boundary.inp`, which demonstrates how to set up harmonic restraints and friction factors for a selection of atoms.

A.1.3 X-PLOR Protein Structure Files (PSF)

Protein Structure Files (PSF) are used by EGO as a summary of the atom type, mass, partial charge and connectivity of the molecular system. PSF files are generated from the original PDB file in combination with X-PLOR topology file using X-PLOR.

The topology data files used by X-PLOR specify the atom parameters and connectivity for all amino acids and nucleotides. X-PLOR extracts all the information necessary (along with patches and modifications from the default configuration) for a given molecule in the PSF file in the form of:

1. a list of atoms with the atom types (CH3E, CH1E, O, N, ...), partial charges and masses,
2. a list of atom number *pairs* representing bonds,
3. a list of atom number *triples* representing the angles between pairs of bonds,

4. a list of atom number quadruples representing *dihedral* angles,
5. atom number quadruples representing *improper* angles[3, 4],
6. atom numbers defining hydrogen bond donors and acceptors,
7. explicit nonbonded interaction exclusions (See also Section 4.9).

An example PSF file follows (`pti.psf`):

```
PSF
       4 !NTITLE
 REMARKS FILENAME="pti.psf"
 REMARKS BPTI COORDINATES TAKEN FROM CRISTALLOGRAPHIC DATA W/O WATERS
 REMARKS HYDROGEN POSITIONS GENERATED USING HBUILD (2 ITERATIONS)
 REMARKS RMS FLUCTUATIONS (T. ICHEYE) FOR HEAVY PROTEIN ATOMS INCLUDED
 REMARKS DATE:24-Apr-89  02:34:58       created by user: heller

     568 !NATOM
       1 MAIN 1    ARG  HT1  HC     0.260000       1.00800           0
       2 MAIN 1    ARG  HT2  HC     0.260000       1.00800           0
       3 MAIN 1    ARG  N    NH3    0.000000E+00   14.0067           0
       4 MAIN 1    ARG  HT3  HC     0.260000       1.00800           0
...

     582 !NBOND: bonds
       3       5       5      18      18      19       5       6
       6       7       7       8       8       9       9      10
...

     834 !NTHETA: angles
       3       5      18       3       5       6       5      18      19
      18       5       6       5       6       7       6       7       8
...

     351 !NPHI: dihedrals
       3       5       6       7       5       6       7       8
       6       7       8       9       7       8       9      11
...

     259 !NIMPHI: impropers
       5       3      18       6       9       8      11      10
      11      12      15       9      22      20      25      23
...

     114 !NDON: donors
       9      10      12      13      12      14      15      16
      15      17       3       1       3       2       3       4
...

      79 !NACC: acceptors
      19      18      26      25      32      31      33      31
      35      34      47      46      54      53      63      62
```

```
...

      24 !NNB
      45      44      43      97      96      95     210     209
     208     224     223     222     236     235     234     328
...

     222       0 !NGRP
       0       0       0       5       0       0       7       0       0
...
```

A.1.4 X-PLOR Parameter Files

The parameter files contain the parameters which correspond to the various bonds and angles listed in the PSF file above. The files contain parameters which are appropriate for different types of molecules and molecular environments (proteins, solvent, etc.). The units for the angle energy constants is $kcal/mol/rad^2$. An example parameter file follows below (`param19.pro` for proteins):

```
remark - parameter file PARAM19 -
remark PEPTIDE GEOMETRY FROM RAMACHANDRAN ET AL BBA 359:298 (1974)
remark TORSIONS FROM HAGLER ET AL JACS 98:4600 (1976)
remark JORGENSEN NONBOND PARAMETERS JACS 103:3976-3985 WITH 1-4 RC=1.80/0.1

set echo=false end
!! - PEPTIDE GEOMETRY TO GIVE RAMACHANDRAN ET AL BBA 359:298 (1974)
!! - PEPTIDE TORSIONS FROM HAGLER ET AL JACS 98:4600 (1976)
!! - NONBONDED TERMS JORGENSEN JACS 103:3976 W/ RC1-4 = 1.80 EC1-4 = 0.1
!!  The default h-bond exponents are now 6-repul 4-attr
!! ++++++++ ATOMTYPE OS (IN METHYL ESTER) ADDED FOR CHARMM COURSE /LN ++++
!! SOLVENT PARAMETERS: SUPPORTING ST2 AND MODIFIED TIP3P MODEL
!! Switched from Slater-Kirkwood to simple mixing rules - AB
!! Hbond parameters based on comparisons of dimer results with
!!   ab initio calculations. - WER  12/19/84
!! Grouping of atom types for VDW parameters - BRB 1/3/85


bond C    C      450.0  1.38!  B. R. GELIN THESIS AMIDE AND DIPEPTIDES
bond C    CH1E   405.0  1.52!  EXCEPT WHERE NOTED.  CH1E,CH2E,CH3E, AND CT
bond C    CH2E   405.0  1.52!  ALL TREATED THE SAME. UREY BRADLEY TERMS ADDED
...

angle C    C    C       70.0 106.5!  FROM B. R. GELIN THESIS WITH HARMONIC
angle C    C    CH2E    65.0 126.5!  PART OF F TERMS INCORPORATED. ATOMS
angle C    C    CH3E    65.0 126.5!  WITH EXTENDED H COMPENSATED FOR LACK
...

dihe CH1E C    N    CH1E 10.0      2     180.0! PRO ISOM. BARRIER 20 KCAL/MOL.
dihe CH2E C    N    CH1E 10.0       2     180.0
dihe CR1E C    C    CR1E 5.0       2     180.0! => TRP OOP. VIB 170CM 1
...
```

```
impr C    C    CR1E CH2E  90.0    0   0.0!GIVE 220 CM 1 METHYL OOP FOR TOLUENE.
impr C    CR1E C    CH2E  90.0    0   0.0!USED HERE FOR TRP CG OUT OF PLANE
impr C    CR1E CR1E CH2E  90.0    0   0.0!               PHE, AND TYR CG OOP
...

{* nonbonding parameter section *}
{* ========================== *}
!! for use with:
!! NBXMOD=5  ATOM CDIEL SHIFT vswitch
!!    CUTNB=8.0  CTOFNB=7.5  CTONNB=6.5  EPS=1.0  E14FAC=0.4  WMIN=1.5
!!

 !                  eps     sigma       eps(1:4) sigma(1:4)
 !                  (kcal/mol) (A)
 !                  ---------------------------------------
 NONBonded  H       0.0498   1.4254      0.0498   1.4254
 NONBonded  HA      0.0450   2.6157      0.0450   2.6157 !- charged group.
 NONBonded  HC      0.0498   1.0691      0.0498   1.0691 !   Reduced vdw radius
 !
 NONBonded  C       0.1200   3.7418      0.1000   3.3854 ! carbonyl carbon
 NONBonded  CH1E    0.0486   4.2140      0.1000   3.3854 ! \
 NONBonded  CH2E    0.1142   3.9823      0.1000   3.3854 !  extended carbons
...

set echo=true end
```

A.1.5 X-PLOR Topology Files

An example X-PLOR topology file for protein residues shown below illustrates how the topology information is defined there:

```
REMARKS TOPH19.PRO ( protein topology )
REMARKS Charges and atom order modified for neutral GROUPs.
REMARKS Histidine charges set to Del Bene and Cohen sto-3g calculations.
REMARKS Amide charges set to match the experimental dipole moment.
REMARKS Default for HIStidines is the doubly protonated state

set echo=false end
!! for use with PARAM19 parameters ( no special hydrogen bonding potential )
!! donor and acceptor terms just for analysis

AUTOGENERATE  ANGLES=TRUE  END
{*==========================*}

{* protein default masses *}
MASS   H      1.00800! hydrogen which can h-bond to neutral atom
MASS   HC     1.00800!    ="=     ="=     ="=    to charged atom
MASS   C     12.01100! carbonyl carbon
MASS   CH1E  13.01900! extended atom carbon with one hydrogen
```

```
...
MASS   S     32.06000! sulphur
MASS   SH1E  33.06800! extended atom sulfur with one hydrogen

!some empirical rules for the following topologies:
! 1. angles are taken between all permutations of atoms bonded to
!    a particular atom. Exception: 2 angles linking the THR double ring
! 2. each bond with non-terminal atoms creates one dihedral. Exception:
!    ring bonds in aromatic side chains (but not PRO).
! 3. each planar atom vertex creates one improper-planar term
!    execption: ARG head groups.
! 4. each 1-extended-H carbon atom creates one improper-tetrahedral term
!    (for chirality)
! 5. Each bond in an aromatic ring creates one improper-torsion term
!    (exception: PRO)
! 6. LYS head groups and methyl head groups create one dihedral for
!    each hydrogen
! 7. all 1:2 and 1:3 nonbonded interactions are assumed to be excluded
! 8. All 1:4,1:5,... non-bonded interactions in aromatic rings
!    (or double rings) are explicitly excluded
! ----------------------------------------------------------------------

RESIdue ALA
 GROUp
  ATOM N    TYPE=NH1   CHARge=-0.35   END
  ATOM H    TYPE=H     CHARge= 0.25   END
  ATOM CA   TYPE=CH1E  CHARge= 0.10   END
 GROUp
  ATOM CB   TYPE=CH3E  CHARge= 0.00   END
 GROUp
  ATOM C    TYPE=C     CHARge= 0.55   END
  ATOM O    TYPE=O     CHARge=-0.55   END
 BOND N    CA
 BOND CA   C
 BOND C    O
 BOND N    H
 BOND CA   CB
 IMPRoper  CA     N    C  CB  !tetrahedral CA
 DONOr H    N
 ACCEptor O C
 IC   N    C    *CA  CB     0.0000    0.00  120.00    0.00   0.0000
END {ALA}

!-------------------------------------------------------------------
RESIdue ARG
 GROUp
...
!-------------------------------------------------------------------
set echo=true end
```

A.2 Output Files

Output from EGO consists of ASCII EGO files, which can be converted into X-PLOR '.DCD' (FORTRAN UNFORMATTED) trajectory files (binary) and EGO '.eny' energy summary files (see Section 2.7).

A.2.1 EGO Trajectory Output Files (`.ego`)

The EGO output file format is self-explanatory. Energies are given in units of kcal/mol, temperatures in Kelvin, and coordinates in Å. The energy is given as a total, and also in its components. An EGO file starts with two REMARK lines and the contents of the control file followed by the line '[BEGINCOORD]'. The next line contains the number of atoms, the integration step, the step of analysis output and the integration time step in fs), which are separated by white spaces. These data may be relevant for utility programs to analyze or convert the following atomic coordinates (x, y, z coordinates in Å). Following the list of atomic coordinates is an energy output for each integration step up to the next analysis step. Energy data is signaled by the line '[BEGINENERGY]' followed by a line with two numbers specifying the number of comment lines and the number of rows of one energy output statement. The second comment line informs about the meaning of the data in columns. Note that in EGO data in columns are separated by white space, that is, data rows are not given by any fixed row position ! An example listing (taken from Section 2.6) is shown below:

```
REMARK Outputfile of EGO_VIII on C-Version
REMARK C by M. Eichinger, H. Grubmueller and H. Heller, 1988-1995

... contents of control file ...

[BEGINCOORD]
568  552001  1  1.000000
9.930929 9.504575 -4.091603
8.971659 8.787418 -5.255728
...
8.105743 8.477091 -6.756792
9.998736 9.040694 -7.383424
[BEGINENERGY]
2  15
[Clock]=seconds, [Temperature]=K, [Energies]=kcal/mol
IntStep Clock Temperature Total Kinetic Electrostatic VDW Bonded Angle ...
 552001 0.3676 269.27 -1505.217 423.7917 -2009.109 -402.8278 109.4822 ...
...
```

A.2.2 X-PLOR Trajectory Files (`.DCD, .crd`)

By default, X-PLOR generates binary (FORTRAN UNFORMATTED) trajectory files. X-PLOR can also convert these files into a more easily portable ASCII format, but both files contain the same information.

Other programs (including Quanta, a molecular visualization and modelling package), can read these X-PLOR binary trajectory files, and, therefore, as described in Section 2.7, the

utility program `ego2crd` is provided to convert EGO format trajectory output files into the '.DCD' format.

The DCD format is structured as follows (FORTRAN UNFORMATTED, with Fortran data type descriptions):

```
HDR     NSET    ISTRT   NSAVC   5-ZEROS NATOM-NFREAT    DELTA   9-ZEROS

'CORD'  #files  step 1  step    zeroes  (zero)          timestep  (zeroes)
                        interval

C*4     INT     INT     INT     5INT    INT             DOUBLE  9INT

==========================================================================
NTITLE          TITLE

INT (=2)        C*MAXTITL
                (=32)

==========================================================================
NATOM
#atoms
INT
==========================================================================
X(I), I=1,NATOM         (DOUBLE)
Y(I), I=1,NATOM
Z(I), I=1,NATOM
==========================================================================
```

A.2.3 EGO Energy Summary Files (`.eny`)

The '.eny' energy summary files created by the utility program `ego2crd` list the energy breakdown at each integration step in the usual units. The file starts with three numbers giving the number of coment lines, the number of energy columns and the number of energy lines. Energy columns are separated by white space. For energy analysis use only the positive integration step numbers.

```
2 15  1293
[Clock]=seconds, [Temperature]=K, [Energies]=kcal/mol
IntStep Clock Temperature Total Kinetic Electrostatic VDW Bonded Angle ...
 0 4.341 329.27 -866.1942 518.2215 -1414.745 -260.334 75.33292 94.56998 ...
 1 6.11 299.24 -900.6289 470.962 -1413.93 -262.3529 83.48541 98.46636 ...
 2 4.822 286.46 -900.3844 450.8502 -1413.766 -266.1185 94.69375 107.5519 ...
...
```

A.2.4 Format of the 'flooding' matrix file `flooding.lis`

The file `flooding.lis` is used by `EGO` for performing rotation/translation correction and computation of the flooding forces. It contains:

- the number of flooding matrices to be used (currently `mkflood` supports only one matrix),
- the total number N_{total} of atoms within the system,
- the number N of selected atoms,
- a list of N_{total} selection flags (0:not selected, 1:selected) denoting which atom has been selected,
- the $3N$ averaged coordinates $\angle\mathbf{x}\rangle$ of the selected atoms,
- the $3N \times 3N$ elements of the (symmetric) flooding matrix $\mathbf{A}$, which determines the flooding potential,
- and (optionally) a list of the eigenvalues (eventually extrapolated) of the covariance matrix $\mathbf{C}$ (only for the human user).

The format of `flooding.lis` is:

```
[HEADER]
<nr_of_flooding_matrices>
<total number of atoms>
<number of selected atoms>
[USINGARRAY]
<selection flag of atom 1>
<selection flag of atom 2>
  ...
<selection flag of atom Ntotal>
[CENTER_OF_MASS]
<x-coordinate of 1st selected atom>
<y-coordinate of 1st selected atom>
<z-coordinate of 1st selected atom>
<x-coordinate of 2nd selected atom>
<y-coordinate of 2nd selected atom>
<z-coordinate of 2nd selected atom>
...
<x-coordinate of Nth selected atom>
<y-coordinate of Nth selected atom>
<z-coordinate of Nth selected atom>
[FLOODING_MATRIX]
<x1x1>
<x1y1>
<x1z1>
<x1x2>
<x1y2>
```

```
<x1z2>
...
<x1xN>
<x1yN>
<x1zN>
<y1x1>
<y1y1>
<y1z1>
...
<y1xN>
<y1yN>
<y1zN>
...
...
...
<zNxN>
<zNyN>
<zNzN>
[COVARIANCE EIGENVALUES]
<LAMBDA 1>  (smallest eigenvalue)
<LAMBDA 2>
...
<LAMBDA 3N> (largest eigenvalue)
<EOF>
```

Appendix B

Comparison of EGO and X-PLOR

The energy function implemented in EGO and the one implemented in the widely used molecular dynamics program X-PLOR [6] are identical, and EGO uses X-PLOR parameter files.

B.1 Important differences between EGO and X-PLOR

EGO does not share the same command language with X-PLOR, but rather is controlled by the setting of parameters in a control file for the molecule to be simulated. This control file contains all of the information necessary to compute molecular dynamics trajectories. X-PLOR is useful for the interactive analysis of the MD results.

X-PLOR is a large (60,000 lines) FORTRAN program which runs in interactive and batch modes. X-PLOR has molecular structure manipulation, crystallographic and NMR NOE refinement features which EGO does not have.

EGO	X-PLOR
control file	command language
only molecular dynamics simulation	molecular dynamics simulation and analysis, crystallographic refinement, NOE restraints
SHAKE for H-atoms	SHAKE for bonds and angles between arbitrary atom types available
	free energy calculations
	periodic boundaries

Table B.1: Important differences between EGO and X-PLOR

B.2 Common features of EGO and X-PLOR

At the heart of every molecular dynamics program lies the energy function (see also Section 6.1) which describes the potential energy surface which itself determines the movement of the atoms. The energy functions in EGO and X-PLOR are identical sums of inter-atom electrostatic, van der Waals, and harmonic bond energies. Therefore the same sets of simulation parameters can be used. The most important *options* for molecular dynamics simulations from X-PLOR are also included in EGO: the exclusion mode for non-bonded interactions, and the attenuation factor for special 1–4 interactions.

Both programs allow Newtonian dynamics as well as Langevin dynamics. For Langevin

dyanamics, however, EGO allows only selected atoms to be exposed to random forces[1], (e.g., to form a stochastic boundary). Harmonic constraints can be used to bind atoms to certain positions in a boundary region.

To manipulate the temperature of the system, it is possible to rescale the velocities in different ways.

EGO complements the many features of X-PLOR as a powerful computational engine for trajectory calculations. Both programs share common file formats including PDB atom coordinate files and PSF topology files[2]. EGO can also read restart files from X-PLOR, and X-PLOR (as well as QUANTA[3] [34]) can read the trajectory files from EGO.

The following features are common to both programs:

- Same Energy function.
- Same Parameter sets can be used.
- Initial atom coordinates taken from Brookhaven PDB files.
- Molecule topology taken from protein structure file (.psf).
- X-PLOR reads EGO trajectory files.
- Exclusion modes (NBXMod: $-5, \ldots, +5$).
- Attenuation factor for special 1–4 interactions.
- Newtonian dynamics and Langevin dynamics.
- Stochastic boundary capability.
- Velocity rescaling to manipulate temperature.
- Harmonic coordinate restraints.
- Definition of cubical and spherical SBOUND region.

B.3 Features Unique to EGO

Besides implementing the most important features of X-PLOR, EGO also offers few features which are not available in X-PLOR. The most important one is that EGO runs concurrently on many parallel computers.

To efficiently compute long-range interactions EGO uses a distance class algorithm (see Section 6.2.1) and a Structure Adapted Multipole Method (see Section 6.2.3). This algorithm is superior to the simple cut-off scheme used by X-PLOR and describes long-range interactions more accurately.

Features unique to EGO include:

- Runs as a parallel program on many parallel computers (PVM, MPI, PARIX).
- Uses a distance-class method and a FMM to compute long-range, nonbonded interactions.
- The Flooding algorithm is used to speed up transitions between conformational substates in a protein.

[1] This is no limitation, as *all* atoms may be selected.
[2] Generated by X-PLOR.
[3] QUANTA is a visualization program for molecular modelling copyrighted by Polygen Corp.

Appendix C

References

C.1 Molecular Dynamics

J. A. McCammon and S. C. Harvey, *Dynamics of proteins and nucleic acids*, Cambridge University Press, Cambridge, 1987.

C. L. Brooks, Martin Karplus, and B. M. Pettitt, *Proteins*, Wiley & Sons, Inc., New York, 1988.

Bibliography

[1] A. Ansari, J. Berendzen, D. Braunstein, B. R. Cowen, H. Frauenfelder, M. K. Hong, I. E. T. Iben, J. B. Johnson, P. Ormos, T. B. Sauke, R. Scholl, A. Schulte, P. J. Steinbach, J. Vittitow, and R. D. Young: "Rebinding and relaxation in the myoglobin pocket", *Biophys. Chem.*, **26**, 337–355 (1987).

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

[3] B. R. Brooks, R. E. Bruccoleri, B. D. Olafson, D. J. States, S. Swaminathan, and M. Karplus: "CHARMM: A Program for Macromolecular Energy, Minimization, and Dynamics Calculations", *J. Comp. Chem.*, **4**(2), 187–217 (1983).

[4] B. R. Brooks and M. Hodoščeк: "Parallelization of CHARMM for MIMD Machines", *Chemical Design Automation News (CDA News)*, **7**(12), 16–22 (1992).

[5] C. L. Brooks III and M. Karplus: "Deformable stochastic boundaries in molecular dynamics", *J. Chem. Phys.*, **79**, 6312–6325 (1983).

[6] A. T. Brünger: "X-PLOR, Version 2.1", The Howard Hughes Medical Institute and Department of Molecular Biophysics and Biochemistry, Yale University, 260 Whitney Avenue, P.O. Box 6666, New Haven, CT 06511, 1990.

[7] C. Eckart: "Some Studies Concerning Rotating Axes and Polyatomic Molecules", *Phys. Rev.*, **47**, 552–558 (1935).

[8] N. Ehrenhofer: "Untersuchung der Konformationsdynamik eines vereinfachten Proteinmodells", Master's thesis, Ludwig-Maximilians-Universität, München, Germany, 1994.

[9] M. Eichinger: "Paralleler schneller Multipolalgorithmus mit Mehrschrittverfahren für Molekulardynamiksimulationen", Master's thesis, Ludwig-Maximilians-Universität München, 1995.

[10] R. Elber and M. Karplus: "Multiple Conformational States of Proteins: A Molecular Dynamics Analysis of Myoglobin", *Science*, **235**, 318–321 (1987).

[11] M. P. I. Forum: "MPI: A Message-Passing Interface Standard", University of Tennesse, Knoxville, Tennessee 37831, May 1994.

[12] H. Frauenfelder, G. A. Petsko, and D. Tsernoglou: "Temperature-dependent X-ray diffraction as a probe of protein structural dynamics", *Nature (London)*, **280**, 558–563 (1979).

[13] H. Frauenfelder, S. G. Sligar, and P. G. Wolynes: "The Energy Landscape and Motions of Proteins", *Science*, **254**, 1598–1603 (1991).

[14] A. E. García: "Large-Amplitude Nonlinear Motions in Proteins", *Phys. Rev. Lett.*, **68**, 2696–2699 (1992).

[15] C. W. Gardiner: "Handbook of Stochastic Methods". Springer-Verlag, Berlin, Heidelberg, New York, 1985.

[16] A. Geist, A. Beguelin, J. Dongarra, W. Jiang, R. Manchek, and V. Sunderam: "PVM 3 user's guide and reference manual", Oak Ridge National Laboratory, Tennessee 37831, May 1994.

[17] N. Gō and T. Noguti: "Structural Basis of Hierarchical Multiple Substates of a Protein", *Chem. Scr.*, **29A**, 151–164 (1989).

[18] L. Greengard and V. Rokhlin: "A Fast Algorithm for Particle Simulations", *J. Comp. Phys.*, **73**, 325–348 (1987).

[19] L. Greengard and V. Rokhlin: "On the Evaluation of Electrostatic Interactions in Molecular Modeling", *Chem. Scr.*, **29A**, 139–144 (1989).

[20] H. Grubmüller: "Molekulardynamik von Proteinen auf langen Zeitskalen", PhD thesis, Technische Universität München, Germany, Jan. 1994.

[21] H. Grubmüller: "Predicting slow structural transitions in macromolecular systems: conformational Flooding", *Phys. Rev. E*, **52**, 2893 (1995).

[22] H. Grubmüller, H. Heller, A. Windemuth, and K. Schulten: "Generalized Verlet Algorithm for Efficient Molecular Dynamics Simulations with Long-Range Interactions", *Mol. Sim.*, **6**, 121–142 (1991).

[23] H. Grubmüller, B. Heymann, and P. Tavan: "Ligand Binding: Molecular Mechanics Calculation of the Streptavidin-Biotin Rupture Force", *Science*, **271**(5251), 997–999 (1996).

[24] H. Grubmüller and P. Tavan: "Molecular Dynamics of Conformational Substates for a Simplified Protein Model", *J. Chem. Phys.*, **101**, 5047–5057 (1994).

[25] H. Heller: "Simulation einer Lipidmembran auf einem Parallelrechner", Doctoral Dissertation, Technical University of Munich, Germany, December 1993.

[26] B. Heymann: "Beschreibung der Streptavidin-Biotin-Bindung mit Hilfe von Molekulardynamiksimulationen", Master's thesis, Ludwig-Maximilians-Universität München, 1996.

[27] B. W. Kernighan and D. M. Ritchie: "The C programming language". Prentice Hall Software Series, London, 1988.

[28] J. F. Leathrum and J. A. Board: "The Parallel Fast Multipole Algorithm in Three Dimensions", Technical report, Dept. of Electrical Engineering, Duke University, Durham, 1992.

[29] M. Levitt and S. Lifson: "Refinement of Protein Conformation using a Macromolecular Energy Minimization Procedure", *J. Mol. Biol.*, **46**, 269–279 (1969).

[30] T. M. Martinetz and K. J. Schulten: "A 'Neural Gas' Network Learns Topologies.", in O. Simula, editor, "Proceedings of the Int. Conf. on Artificial Neural Networks, ICANN-91, Espoo, Finland, 24–28 June 1991", pages 397–402, Amsterdam, 1991. Elsevier Science Publishers.

[31] J. A. McCammon, B. R. Gelin, and M. Karplus: "Dynamics of Folded Proteins", *Nature (London)*, **267**, 585–590 (1977).

[32] C. Niedermeier and P. Tavan: "A Structure Adapted Multipole Method for Electrostatic Interactions in Protein Dynamics", *J. Chem. Phys.*, **101**, 734–748 (1994).

[33] PARSYTEC Computer GmbH, Germany: "PARIX Version 1.3-PPC reference manual", December 1994.

[34] Polygen Corporation, 200 Fifth Av., Waltham, MA 02254, U.S.A.: "Quanta", 1988.

[35] E. Shakhnovich, G. Farztdinov, A. M. Gutin, and M. Karplus: "Protein Folding Bottlenecks: A Lattice Monte Carlo Simulation", *Phys. Rev. Lett.*, **67**, 1665–1668 (1991).

[36] W. B. Streett, D. J. Tildesley, and G. Saville: "Multiple time step methods in molecular dynamics", *Mol. Phys.*, **35**, 639–648 (1978).

[37] J. P. Valleau and G. M. Torrie, in B. J. Berne, editor, "Statistical Mechanics Part A: Equilibrium Techniques", page 137. Plenum Press, New York, 1977.

[38] W. F. van Gunsteren and H. J. C. Berendsen: "GROMOS Manual", BIOMOS b.v., Biomolecular Software, Groningen, The Netherlands.

[39] W. F. van Gunsteren and H. J. C. Berendsen: "Algorithms for macromolecular dynamics and constraint dynamics", *Mol. Phys.*, **34**(5), 1311–1327 (1977).

[40] L. Verlet: "Computer "Experiments" on Classical Fluids. I. Thermodynamical Properties of Lennard-Jones Molecules", *Phys. Rev.*, **159**(1), 98–103 (1967).

[41] A. Wallquist and G. Karlström: "A New Non-Empirical Force Field for Computer Simulations", *Chem. Scr.*, **29A** (1989).

[42] S. J. Weiner, P. A. Kollman, D. A. Case, U. C. Singh, C. Ghio, G. Alagona, J. S. Profeta, and P. Weiner: "A New Force Field for Molecular Mechanical Simulation of Nucleic Acids and Proteins", *J. Am. Chem. Soc.*, **106**, 765–784 (1984).

[43] A. Windemuth: "Dynamiksimulation von Makromolekülen", Diploma Thesis, Technical University of Munich, Physics Department, T 30, James-Franck-Street, D-8046 Garching / Munich, August 1988.