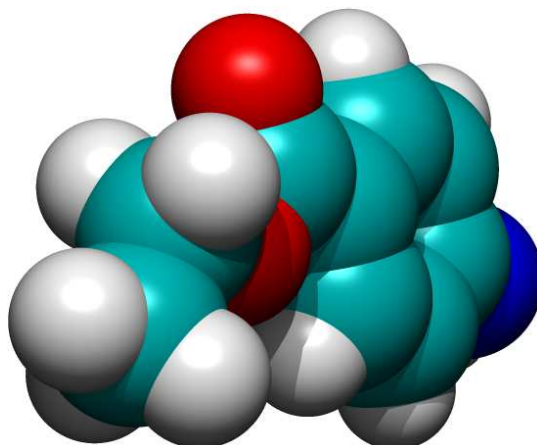


# MACSIMUS manual

benzocaine (ethylaminobenzoate)

parameter\_set = charmm21

```
      HA
      |
HA-CT-HA
      |
HA-CT-HA
      |
    OSn.2
      |
Cp.7=On.5
      |
C6R--C6R-HA
      |   |
HA-C6R   C6R-HA
      |   |
HA-C6R--C6R-NPn.5^-Hp.25
                  |
                  Hp.25
```



## Most often used links:

[2.2](#) Blend synopsis and options

[9.2](#) Cook synopsis and options

[9.2.5](#) Cook input data

MACromolecule SIMulation Software

© Jiří Kolafa 1993–2010

MACSIMUS may be distributed under the terms of the  
GNU General Public Licence

## Credits:

- **ray**: Mark VandeWettering “reasonably intelligent raytracer”
- CHARMM force field (files: charmm\*.par, charmm\*/\*.rsd)
- GROMOS force field (files: gromos\*.par, gromos\*/\*.rsd)
- DOS mouse library by unknown author (mouselib.[ch])
- amoeba implementation by Z. Wagner
- moil support by J. Schofield
- several bug fixes by T. Trnka
- gtk-based GUI by K. Matas

# Contents

<b>I</b>	<b>Program ‘blend’ version 2.1i</b>	<b>14</b>
<b>1</b>	<b>Introduction</b>	<b>16</b>
1.1	Force fields . . . . .	16
1.2	‘blend’ overview . . . . .	16
1.3	Versions . . . . .	17
<b>2</b>	<b>Running blend</b>	<b>18</b>
2.1	Environment . . . . .	18
2.2	Synopsis . . . . .	19
2.2.1	Global options . . . . .	19
2.2.2	<u>par-options</u> and parameter files . . . . .	20
2.2.3	<u>mol-options</u> and molecular files . . . . .	22
2.2.4	<u>Extra-options</u> . . . . .	30
2.3	File extensions . . . . .	36
2.4	Run-time control . . . . .	39
2.4.1	<i>get data</i> format for input . . . . .	39
2.4.2	Scrolling . . . . .	40
2.4.3	Error handling . . . . .	41
2.4.4	Interrupts . . . . .	41
2.5	Showing molecules graphically . . . . .	42
2.5.1	X11 and Turbo C Graphics . . . . .	42
2.5.2	Playback output . . . . .	46
2.6	Energy minimization . . . . .	46
2.7	Missing coordinates . . . . .	47
<b>3</b>	<b>Force field and the parameter file</b>	<b>48</b>
3.1	Structure of the parameter file . . . . .	48

3.2	Force field generation options . . . . .	49
3.3	Table of atoms . . . . .	51
3.4	Non-bonded forces . . . . .	52
3.4.1	Selection of site-site and Coulomb energy terms . . . . .	52
3.4.2	Combining rules for the Lennard-Jones parameters . . . . .	53
3.4.3	Table of site-site parameters . . . . .	54
3.4.4	Non-bonded fixes . . . . .	55
3.4.5	Table of polar atom parameters . . . . .	55
3.4.6	Table “shellrep” of repulsive counterparts . . . . .	55
3.4.7	Table of axially polar bonds . . . . .	56
3.4.8	Table of 1–3 axially polar groups . . . . .	56
3.4.9	Table defining water models . . . . .	56
3.4.10	Table defining the protein backbone types . . . . .	59
3.5	Non-bonded potential cutoff . . . . .	59
3.6	Bond potential . . . . .	60
3.7	Bond angle potential . . . . .	60
3.8	Torsions . . . . .	61
3.8.1	Torsion angle . . . . .	61
3.8.2	Torsion potential . . . . .	62
3.8.3	Cis and trans-dihedrals . . . . .	63
3.8.4	Implementation of the torsion potential . . . . .	63
3.8.5	Chirality . . . . .	63
3.8.6	Dihedrals in aromatic rings . . . . .	64
3.8.7	Tables of dihedrals and impropers . . . . .	65
3.8.8	Atom matching rules for finding the energy terms . . . . .	66
<b>4</b>	<b>Description of molecules</b>	<b>67</b>
4.1	Molecular file (mol-file) format . . . . .	67
4.2	Chemical file format . . . . .	68
<b>5</b>	<b>Output format (ble-file)</b>	<b>71</b>
5.1	Global parameters . . . . .	71
5.2	Site types . . . . .	72
5.3	Non-bonded fixes . . . . .	73
5.4	Header of molecule . . . . .	73

5.5	One species (molecule) data . . . . .	74
5.6	Table of sites . . . . .	75
5.7	Tables of bonds and bond angles . . . . .	75
5.8	Tables of dihedrals, impropers and aromatics . . . . .	76
5.9	Table of axial polarizability tensors . . . . .	77
5.10	Table of dependants . . . . .	78
5.10.1	Lone (out-of-plane) dependants . . . . .	79
<b>6</b>	<b>Examples</b>	<b>81</b>
6.1	Example 1: Protein in water . . . . .	81
6.2	Example 2: Cluster Na <sub>4</sub> Cl <sub>4</sub> . . . . .	83
<b>7</b>	<b>Problems</b>	<b>84</b>
7.1	Bugs and caveats . . . . .	84
7.2	Trouble shooting . . . . .	84
7.3	Frequently asked questions . . . . .	87
7.3.1	Free molecules . . . . .	87
7.3.2	Prevent molecules from evaporating . . . . .	87
7.3.3	One or more molecules? . . . . .	87
<b>II</b>	<b>Program ‘cook’ version 2.6</b>	<b>88</b>
<b>8</b>	<b>Overview</b>	<b>90</b>
8.1	Features of <b>cook</b> . . . . .	90
8.2	History . . . . .	91
8.3	Compile-time versions of <b>cook</b> . . . . .	92
8.4	Caveats and plans . . . . .	93
8.5	Disclaimer . . . . .	94
<b>9</b>	<b>Running cook</b>	<b>95</b>
9.1	Where is . . . . .	95
9.2	Synopsis . . . . .	95
9.2.1	File parameters . . . . .	95
9.2.2	Options . . . . .	96
9.2.3	File extensions . . . . .	103
9.2.4	Program flow . . . . .	107

9.2.5	Input data . . . . .	108
9.2.6	Interactive and batch control . . . . .	132
9.2.7	Interrupt . . . . .	132
<b>10</b>	<b>Parallelization</b>	<b>133</b>
10.1	Compiling . . . . .	133
10.2	Running . . . . .	133
10.3	Linked-cell list and Ewald parallelized . . . . .	133
10.4	Ewald $k$ -space and $r$ -space running in parallel . . . . .	134
10.5	Pair sums for a single big molecule parallelized . . . . .	134
<b>11</b>	<b>How to set parameters</b>	<b>135</b>
11.1	Accuracy . . . . .	135
11.1.1	Errors of constraints . . . . .	136
11.1.2	Energy conservation . . . . .	137
11.1.3	Self-consistent field accuracy . . . . .	137
11.2	How to set Ewald parameters $\alpha$ and $\kappa$ . . . . .	138
11.2.1	Simple way . . . . .	138
11.2.2	More accurate way . . . . .	138
11.2.3	Most accurate way . . . . .	138
11.3	How to set accuracy for constraint dynamics . . . . .	140
11.3.1	The SHAKE algorithm with Verlet integration . . . . .	140
11.3.2	Constraint forces by Lagrangian multipliers . . . . .	141
11.3.3	Correcting constraints . . . . .	142
11.3.4	Dependants . . . . .	143
11.4	Site-site potential cutoff . . . . .	144
11.5	The timestep . . . . .	144
11.6	Functions for $r$ -space Ewald sums . . . . .	145
<b>12</b>	<b>NVT and NPT ensembles</b>	<b>147</b>
12.1	Constant temperature simulations . . . . .	147
12.1.1	The friction thermostat . . . . .	147
12.1.2	Decoupled translational and intramolecular thermostats . . . . .	149
12.1.3	The Nosé–Hoover canonical ensemble . . . . .	149
12.1.4	Maxwell-Boltzmann thermostat . . . . .	150
12.1.5	Langevin thermostat . . . . .	151

12.1.6 Which thermostat . . . . .	151
12.2 Constant pressure simulations . . . . .	151
12.2.1 Friction (Berendsen) barostats . . . . .	152
12.2.2 MTK thermostat and barostat . . . . .	152
<b>13 Initial configuration . . . . .</b>	<b>155</b>
13.1 Random-shooting algorithm . . . . .	155
13.2 Crystal initial configuration . . . . .	156
13.3 Immersing a large solute into solvent . . . . .	156
<b>14 Measurements . . . . .</b>	<b>158</b>
14.1 Units of measurements . . . . .	158
14.2 Convergence profile . . . . .	158
14.3 Analysis of statistical errors . . . . .	159
14.4 Kinetic quantities from equilibrium molecular dynamics . . . . .	160
14.4.1 Diffusivity . . . . .	160
14.4.2 Conductivity . . . . .	162
14.4.3 Viscosity . . . . .	163
14.5 Kinetic quantities from the Einstein relations . . . . .	164
14.5.1 Requirements . . . . .	164
14.5.2 Usage . . . . .	164
14.5.3 Results . . . . .	165
14.5.4 Analysis of results . . . . .	165
14.6 Structure factor . . . . .	167
14.6.1 Structure factor for pure simple fluids . . . . .	167
14.6.2 Structure factor for mixtures . . . . .	168
14.7 Radial distribution functions . . . . .	169
14.8 Cluster (oligomer) analysis and bond kinetics . . . . .	171
14.8.1 Cluster overview . . . . .	171
14.8.2 Compilation and synopsis . . . . .	171
14.8.3 Input data . . . . .	171
14.8.4 Results . . . . .	173
14.8.5 Bugs and caveats . . . . .	174
14.8.6 Bond kinetics . . . . .	174
14.9 Dihedral angle distribution . . . . .	175

<b>15 Special versions</b>	<b>176</b>
15.1 Fixing positions of selected atoms . . . . .	176
15.2 Notes on water models . . . . .	177
15.3 Cut off electrostatic forces (CUTELST) . . . . .	178
15.4 Gravity simulation (STARS) . . . . .	179
15.5 Polarizable dipoles . . . . .	179
15.5.1 Polarizability models . . . . .	180
15.5.2 Integration methods . . . . .	181
15.6 Pressure tensor . . . . .	182
15.6.1 Pressure tensor for Drude oscillators . . . . .	184
15.7 Slab geometry and surface tension . . . . .	185
15.7.1 Surface tension via virial pressure . . . . .	186
15.7.2 Surface tension via virtual volume change . . . . .	187
15.7.3 Surface tension via virtual area change . . . . .	187
15.7.4 Surface tension via pressure tensor . . . . .	188
15.7.5 Cutoff corrections in the slab geometry . . . . .	188
15.7.6 Interfacial Gibbs energy by the cleaving method . . . . .	189
15.8 Simulations at walls . . . . .	190
15.8.1 Atom-surface force field . . . . .	191
15.8.2 Atom-metal force field . . . . .	191
15.8.3 Using WALL and GOLD versions . . . . .	192
15.8.4 Wall versions and integrals of motion . . . . .	193
15.8.5 Initial configuration . . . . .	193
15.8.6 Input data for the WALL versions . . . . .	194
15.8.7 Wall visualization and more . . . . .	194
15.9 Anchor sites and axes and measure forces . . . . .	195
15.10 Analyze pair energies . . . . .	197
15.11 Viscosity by shear stress . . . . .	198
15.12 Widom and scaled insertion particle method . . . . .	200
15.13 Metals . . . . .	203
<b>16 File formats</b>	<b>204</b>
16.1 Configuration . . . . .	204
16.2 Convergence profile . . . . .	205
16.3 Playback file . . . . .	206

<b>17 Examples</b>	<b>207</b>
17.1 Example 1: Protein in water . . . . .	207
17.2 Example 2: Melting point of a model of NaCl . . . . .	209
17.2.1 Force field and molecules . . . . .	209
17.2.2 Crystal Na <sub>4</sub> Cl <sub>4</sub> . . . . .	210
17.2.3 Preparation of data for the simulation . . . . .	210
17.2.4 Crystal Na <sub>108</sub> Cl <sub>108</sub> . . . . .	210
17.2.5 Equilibrium simulation of the crystal . . . . .	211
17.2.6 Melt . . . . .	212
17.2.7 Melt–crystal equilibrium . . . . .	212
<b>18 Source C files and making cook</b>	<b>214</b>
18.1 General software in directory <code>gen</code> . . . . .	214
18.2 Simulation software in directory <code>sim</code> . . . . .	215
18.3 Particular <code>cook</code> source code . . . . .	216
18.4 Making <code>cook</code> . . . . .	216
18.5 Library interface . . . . .	217
<b>III Utilities</b>	<b>220</b>
<b>19 General utilities</b>	<b>222</b>
19.1 <code>makemake</code> : Makefile and project interface . . . . .	222
19.2 <code>plot</code> : Plot a graph (with formulas and mouse-rescaling) . . . . .	222
19.3 <code>tabproc</code> : Command-prompt spreadsheet . . . . .	228
19.4 <code>mergetab</code> : Merge columns of data . . . . .	229
19.5 <code>tab</code> : Column table of consecutive numbers . . . . .	229
19.6 <code>ev</code> : Calculator . . . . .	230
19.7 <code>endian</code> : Change endian (order of bytes) in binary files . . . . .	230
19.8 <code>start</code> : Start application according to file extension . . . . .	230
19.9 <code>sortcite</code> : Sort LaTeX citations . . . . .	231
<b>20 Program ‘pdb’ version 1.4a</b>	<b>232</b>
20.1 Running . . . . .	232
20.1.1 Environment . . . . .	232
20.1.2 Synopsis . . . . .	232



20.1.3	Files	233
20.1.4	Options	234
20.2	Residues	237
20.2.1	Format of residues	237
20.2.2	Termini	238
20.2.3	List of residues	239
20.3	Bugs and caveats	241
<b>21</b>	<b>Data analysis</b>	<b>242</b>
21.1	showcp: Show and analyze convergence profiles	242
21.2	cp2cp: Manipulate convergence profile files	245
21.3	rdfg: Analyze and show radial distribution functions	246
21.4	smoothg: Smooth histogram-based RDF	246
21.5	staprt: Print a sta-file	247
21.6	sfourier: Structure factor from RDF	247
21.7	coordn: Coordination number	248
21.8	hbonds: H-bonds for liquid water	248
21.9	cppak: Loss (de)compression of convergence profile files	248
21.10	autocorr: Statistical analysis using autocorrelation function	249
21.11	spectrum: Spectrum (Fourier transform)	251
<b>22</b>	<b>Working with playback files</b>	<b>252</b>
22.1	plbinfo: Get information on plb-files	252
22.2	plbcheck: Some checks on binary playback files.	252
22.3	plbconv: Converts old and new plb formats	252
22.4	plb2plb: Extract selected sites	253
22.5	plb2asc: Conversion of plb-files to ASCII	253
22.6	asc2plb: plb-files from ASCII	253
22.7	frame: Extract one frame from a plb-file	253
22.8	cutplb: Edit plb-files	253
22.9	plbcut: extracts parts of a playback file	254
22.10	plbbox: Change box size of a plb-file	254
22.11	densprof: Selected density profile angular correlations.	254
22.12	plb2cryst: Sort sites to files according to crystal-like structure.	255
22.13	plb2nbr: Sort sites to files according to the number of neighbors.	255

22.14	<code>plbmerge</code> : Merge several plb files . . . . .	255
22.15	<code>atomdist</code> : Atom-atom distances . . . . .	255
22.16	<code>smoothpl</code> : Smooth the playback file . . . . .	255
22.17	<code>plbmsd</code> : Mean square displacement of atoms . . . . .	256
22.18	<code>density</code> : Calculate local density . . . . .	257
22.19	<code>mergeplb</code> : Merge several plb files into one . . . . .	257
22.20	<code>filtplb</code> : Convert a plb-file for a subset of atoms . . . . .	257
22.21	<code>plbpak</code> : Loss (de)compression of playback files . . . . .	257
22.22	<code>plb2diff</code> : Diffusion, conductivity, and viscosity calculations . . . . .	258
22.23	<code>shownear</code> : re-color atoms according to their distance . . . . .	258
22.24	<code>tomoil</code> : Conversion to MOIL . . . . .	259
<b>23</b>	<b>Molecule visualization</b>	<b>260</b>
23.1	<code>molcfg</code> : Create configuration mol- and gol-files . . . . .	260
23.2	<code>show V 1.0c</code> : Viewing playback (trajectory) files . . . . .	261
23.3	<code>ray</code> : The raytracer . . . . .	270
23.4	<code>ppm2ps</code> : PPM, PBM, PGM to PostScript conversion . . . . .	273
23.5	<code>ppminfo</code> : Get information on ppm,pbm,pgm-files . . . . .	275
23.6	<code>showppm</code> : View a (sequence of) ppm-file(s) . . . . .	275
23.7	<code>stereo</code> : Stereogram . . . . .	275
<b>24</b>	<b>Miscellaneous utilities</b>	<b>277</b>
24.1	<code>pdb2pdb</code> : Rearrange pdb-files . . . . .	277
24.2	<code>ramachan</code> : Ramachandran plot from blend and playback files . . . . .	277
24.3	<code>makepept</code> : Makes a peptide in che-format . . . . .	278
24.4	<code>blefilt</code> : Blend-file filter. . . . .	279
24.5	<code>bonds</code> : Make (show-able) mol-file from coordinates . . . . .	279
24.6	<code>cutprt</code> : Shorten a prt-file . . . . .	280
24.7	<code>lattice</code> : Make a cubic lattice . . . . .	280
24.8	<code>showpro</code> : Show sorted pro-files . . . . .	281
24.9	<code>view3db</code> : View 3db files or convert to 3dt . . . . .	281
<b>IV</b>	<b>Appendixes</b>	<b>282</b>
<b>25</b>	<b>Ewald summation</b>	<b>283</b>

<b>26 MD of Polarizable Force Fields</b>	<b>285</b>
26.1 Notation . . . . .	285
26.2 Polarizability . . . . .	285
26.3 Pair operators of electrostatic interaction . . . . .	286
26.4 Electrostatic energies . . . . .	286
26.5 Electrostatic field . . . . .	287
26.6 Total energy . . . . .	287
26.7 Forces . . . . .	288
26.7.1 Gradient of the repulsive antipolarization . . . . .	288
26.7.2 Gradient of the polarizability tensor . . . . .	288
26.7.3 Fluctuating charge . . . . .	289
26.7.4 Implementation . . . . .	290
<b>27 Time-reversible predictors for Verlet+SHAKE with a velocity-dependent rhs</b>	<b>291</b>
27.1 The task . . . . .	291
27.2 MACSIMUS solution . . . . .	291
27.3 Algorithm . . . . .	293
<b>28 Always Stable Predictor-Corrector (ASPC) instant</b>	<b>294</b>
28.1 Task . . . . .	294
28.2 ASPC . . . . .	294
<b>29 Specific heat <math>C_V</math> in the molecular dynamics microcanonical ensemble</b>	<b>296</b>
<b>30 Dielectric constant in SI</b>	<b>298</b>
30.1 Dielectric constant from external field . . . . .	298
30.2 Fluctuation formulas . . . . .	299
30.2.1 Notes . . . . .	301
<b>31 Fourier transform</b>	<b>302</b>
31.1 Basic formulas . . . . .	302
31.2 Implementation . . . . .	302
31.3 Structure factor . . . . .	303
31.3.1 Mixtures . . . . .	303
<b>References</b>	<b>304</b>

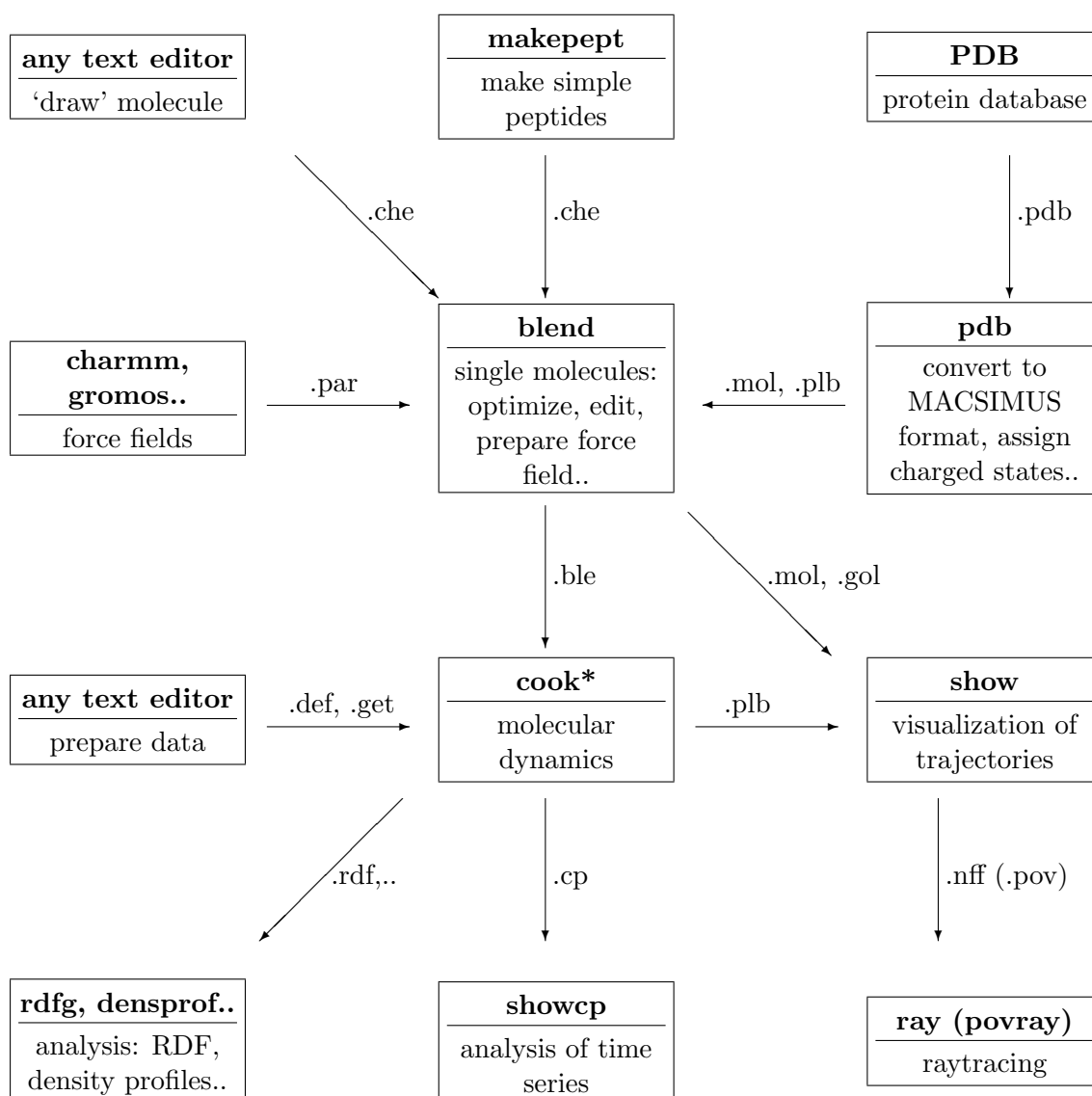


# Foreword

MACSIMUS is essentially a “one man software”. Its development started in 1991 in the John Perram’s group in Odense, Denmark, under project PROSIS (PROtein SIMulation Software). The first sufficiently general version is from 1993. In 1999, PROSIS was renamed to MACSIMUS (MACromolecule SIMulation Software), the name invented by John Perram. The latest development, however, has declined from macromolecular simulations and focused rather on fluids, ionic systems, polarizability, surface and two-phase phenomena, etc.

MACSIMUS makes use of the traditional Unix (now Linux) front-end of command prompt; graphics is based on hot keys (with simple online help available). There are no mouse-driven menus and cards you know from “standard” software.

Basic components of MACSIMUS with the most important file types follow:



# Part I

Program ‘blend’ version 2.1i

Odense, October 1993 (V0.663), November 1995 (V1.5), August 1996 (V1.6g)  
Guelph April 1996 (V1.6d)  
Evanston December 1995 (V1.6c), July 1996 (V1.6f)  
Prague 1996 (V1.7)  
1999 PROSIS renamed to MACSIMUS  
More updates: 2001–

Program **blend** assigns a force field to (a mixture of) molecules and prepares thus data for molecular dynamics simulation programs of the **cook** family.

# Chapter 1

## Introduction

### 1.1 Force fields

Force fields describing complex organic molecule are composed of many elementary energy terms, typically including Lennard-Jones and Coulomb interactions for non-bonded forces, and bond, bond angle, dihedral and improper torsion potentials for intramolecular bonded forces.

The parameters of elementary energy terms for different combinations of atom types are listed in tables. Yet these tables do not define uniquely the force field because often different terms can be used for the same effect. This problem is often solved (e.g., in CHARMM) by defining these terms separately for each *residue* (molecular building block). The problem of assigning a force field then reduces to simple looking in tables.

Our task, however, is to assign a force field to a molecule given by a chemical formula without additional information with the exception of partial charges attached to atoms which cannot be easily derived from the chemical formula only and must be defined for each residue. All other energy terms are generated automatically. This requires a comprehensive analysis of the molecular structure. Since the rules of assigning a force field to the molecular structure differ from one package to another, and even from one version of the same package to another, and are often on purpose obscured by the authors to prevent any other use of the force fields than buying their expensive and inefficient programs, program **blend** has become much more complicated than was originally intended.

### 1.2 ‘blend’ overview

Program **blend** assigns a force field to (a mixture of) molecules and prepares thus data for molecular dynamics simulation program **cook** (better said, many versions of this).

The force field consists of Lennard-Jones parameters, force constants and values of bond lengths, bond angles, torsions (dihedral potentials) and improper torsions. **blend** analyses the structure of a molecule and finds the required energy terms by looking in tables. Partial charges are usually not part of the force field and are given separately.



The molecules are described internally in a table-based format (originally produced by graphical Molecule Editor **MEDIT**). They can be input by a user in a text format close to the chemical structure formulas or via external sources (PDB interfaced by utility **pdb**).

Using the constructed force field, program **blend** can find the (local) energy minimum of the molecule (from given configuration or from random input) and also fill missing atom coordinates (typically hydrogens).

Several molecules can be blend-ed together and the prepared mixture can be simulated by the MD program **cook**. In other words, **blend** creates a file that contains a full description of the force field of given molecules and this file is used by **cook**.

X-Windows and PC (compiled by Borland/Turbo C) versions can show molecules graphically and allow configuration editing.

Although program **blend** was developed originally with the CHARMM force field in mind, an attempt was made to be more general if possible.

During years, many features have been added to **blend**: support of polarizability, molecular editing features, essential dynamics, normal mode vibrations, inertia matrices and radii of gyration, internal coordinates, etc.

## 1.3 Versions

See `macsimus/readme.txt` and `macsimus/blend/metamake` for compilation instructions.

Major versions of ‘blend’ are:

**POLAR** This version supports scalar and tensor polarizabilities of atoms. With the Busing (exp-6) potential, it supports the ‘shell-core’ model. The polarizability may also be saturated (see Sect. 26). Use `#define POLAR` for compiling

**TINY** (Obsolete) The TINY version omits some less used functions and is suitable, e.g., for DOS. Use `#define TINY` for compiling

**site-site potential** The used site-site potential can be selected in file `macsimus/blend/metamake`: Lennard-Jones, exp-6, WCALJ, Busing etc.

**X11** Version with X11 graphics

**DOS** (Obsolete) Version for DOS with Turbo graphics

# Chapter 2

## Running blend

### 2.1 Environment

Environment variable `BLENDPATH` can be set to point to the path that contains the parameter files (with extensions `.bin` and `.par`). If it is empty, the parameter files are looked for in the working directory.

Example for linux/unix (csh, tcsh):

```
setenv BLENDPATH /home/jiri/blend/data
```

Example for linux/unix (sh, bash):

```
export BLENDPATH=/home/jiri/blend/data
```

Example for DOS:

```
set BLENDPATH=D:\JK\BLEND\DATA
```

(Obsolete) To enable graphics under Turbo C/DOS, the graphics driver must be in the working directory or pointed to by an environment variable `BGI`, example:

```
set BGI=C:\TC20
```

Tested with VGA only (driver `egavga.bgi`).

To remap mouse buttons, use the environment variable called `MOUSEMAP`. Example (DOS):

```
set MOUSEMAP=132
```

will swap the middle and right mouse buttons. This is recommended for Microsoft Mouse with only two buttons: the right button will move the molecule/atoms and the less important function, rescaling and z-rotating, will not be available by mouse.

WARNING: On some UNIX systems there is a system command of the same name, `blend`. Make sure that your path contains the correct order of subdirectories.

## 2.2 Synopsis

```
blend \  
[-o file | -s] \  
[-lnumber] [-vnumber] [-znumber] [-gnumber] \  
[ par-options ] [ parset.par | parset.bin ] \  
[ charges.pch ] [ reactions.rea ] \  
[ mol-options ] { species | species.mol | species.che } \  
[ mol-options { species | species.mol | species.che } ] \  
...
```

The options are parsed from left and thus, e.g., **-s** must precede any parameter that generates output. Option **X** is switched on (i.e., set to value 1) by one of **-X** **-X+** **-X1** and switched off (cleared to value 0) by **-X-** or **-X0**. The options of **blend** are case sensitive (most other MACSIMUS programs use case-insensitive options). Numerical options may be given in decimal, octal (number begins with 0) or hexadecimal (number begins with 0x).

If **blend** is run without any parameter, a brief description of options is printed.

### 2.2.1 Global options

These options apply to the whole run and should be placed before any molecule is processed.

- gnumber** Graphical show of minimization (by number steps). Valid for X-Windows or DOS Turbo C versions. The default is **-g3** for DOS but **-g0** (no graphics) for UNIX (X-Windows)
- lnumber** Size of the line buffer for reading files (default=256). You may find this option useful if you have a complicated formula in the chemical format.
- o file** This option sets the batch mode. Output from **blend** is written to file; if file does not have extension, **.ble** is appended. No keyboard input is requested; in the case of error, the calculations exit with a nonzero return code. The space between **-o** and the file name is optional. Cannot be combined with option **-s**.
- snumber** Enable scrolling (screen buffer = number kB). If no number is given, the default is 31 kB (this is also maximum for DOS). Cannot be combined with option **-o**. **blend** must be compiled with option **-DSCR** to enable this feature.
- v** Sum of (note that the option may be octal or hexadecimal, e.g., **-v11** = **-v013** = **-v0xb**)
  - v0** Neither force field information nor the table of sites are printed. This is the default for the interactive mode (no output file specified by **-o** option).
  - v1** List sites and their parameters. Tables of bonds, angles, dihedrals and impropers are not printed.

- v2 Print tables of bonds, angles, dihedrals, impropers etc.
- v3 The same as -v1 + -v2: Print the full force field information (as required by `cook*`). This is the default for the batch mode (`file.ble` is created).
- v4 Verbose (info on topological analysis, selection of terms, etc.).
- v8 List the used pair site-site parameter terms.
- v16 POLAR only: print convergence rates.
- v256 List all pair site-site parameter terms (even for atoms not present in the molecules), may be very lengthy.
- v4 -J1e-11 -v0 To be checked again: Special for functions using Jacobi diagonalization: shows the diagonalization progress and turns off verbose mode otherwise. See -J, -N, -E for details.
- znumber (0) Seed for random number generator. If -z is not specified, the seed is derived from time so that it is different for each run.
- digit (6) (Underscore) Output precision of force constants. Most of the force field terms (like force constants K) are printed with digit decimal digits, some of them (to have comparable relative precision) with precision digit-1 or digit+1. Must be  $0 < \text{digit} < 9$ .  
In addition, negative values of digit force fine adjustment of rounding of partial charges when a mol-file is (re)created so that their sum for a molecule is an exact integer. Useful for charges pasted to a mol-file from a quantum-mechanical calculation.

## 2.2.2 par-options and parameter files

Normally, the parameter set to be used is defined in the mol-file (keyword `parameter_set`). If not defined, it can be specified as a parameter (but note that all molecules blended together must use the same parameter set!). The options that control the force field assignment logic have their defaults defined in the parameter file and may be changed using a command line parameter. See Sect. 20.1.4, for more information. A list of par-options follows:

- parset.par Name of the parameter file in text format.
- parset.bin Name of the parameter file in binary format. The default is `charmm21.bin`.  
The binary file is created from the text file if option -b is given. It is faster to use binary files for parameters (well, in 21st century the difference is negligible, but on 386 it used to make a difference). WARNING: binary files of full and TINY versions are not compatible.
- anumber For number  $\geq 0$ : scale all angle force constants number% times. For number  $< 0$ : set all angle force constants to  $-\text{number}$  kcal/mol. For number = 0: constrain angles (inefficient minimization algorithm).  
See also -b. Note that the changed -a0 value is exported to a ble-file. -a0 does not work accurately enough in some cases like planar molecules (e.g., TIP4P water); a large value may help, but the algorithm becomes inefficient.

-b

-b1 Make binary image parset.bin from parset.par. The name of the source parameter file parset.par must be specified after option -b. **blend** version number is written to the header of parset.bin and two major digits (e.g., 1.2) are tested whenever **blend** reads the binary file.

-bnumber For number > 1: scale all bond force constants number times. For number < 0: set all bond force constants to  $-\text{number}$  kcal/mol/Å<sup>2</sup>. For number=0: constrain bonds (inefficient minimization algorithm).  
Warning: -b1 is reserved for making binary files (see above).

-fnumber Fabricate (make up) bond or angle terms that are missing in the parameter file.

-f0 Missing bond or angle terms issue a warning and no term is added.

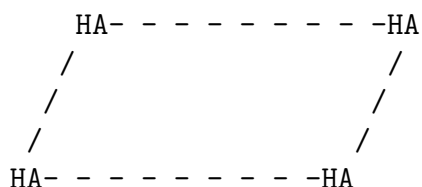
-f1 A missing bond parameter is added with length 1.5 Å and force constant  $K = 50$  kcal/mol.

-f2 A missing angle parameter is added with equilibrium angle 120° and force constant  $K = 20$  kcal/mol.

-f3 Both -f1 and -f2.

-inumber (default=5) Calculate the maximum angle potential energy (over all angle potential terms in a molecule) and print a warning if this energy exceeds number kcal/mol. Such a warning for “normal” molecules means that the configuration is wrong, e.g., methane is in a wrong “pyramidal” conformation:

CT



The default should detect such cases. Use -i0 if it is OK and the warnings bother you.

-xnumber Lennard-Jones scaling; the default is -x1000100 (all scaling factors are unities).

-xEEE (three digits) All Lennard-Jones energy terms ( $\epsilon = -E_{\min}$ ) are multiplied by factor  $EEE/100$ .

-xRRRREEEE (more than 3 digits) In addition, all Lennard-Jones radii ( $\sigma$  or  $R_{vdW}$ ) are multiplied by factor  $RRRR/1000$ .

-x-DDD Energy terms are scaled  $x = \exp(DDD/1000)$  times and radii  $x^{-1/3}$  times. This means that density is scaled  $x$ -times with the structure unchanged.

- @number POLAR only: Output multiplication factor in % for the value of **shell** (number of shell electrons for auxiliary charges for MD with polarizabilities). Only exported to MD because **blend** uses exact point dipoles. Default=100%.
- ^number
- ~number POLAR only: polarizability multiplication factor in %. Default=100%.
- [number POLAR only: saturation energy multiplication factor in %. Default=100%. Rarely used.
- [number Active with **-n-1** only: reorder sites in molecules of number sites in increasing site mass. E.g., **-[3 -n-1** will reorder TIP3P water to HHO.

### 2.2.3 mol-options and molecular files

These options can be changed before each molecule is processed. Once an option is set or cleared, its new value remains active until it is cleared or set again (i.e., it is not re-set to the default).

species.mol Molecule file (*mol-file*) generated by **blend**, **pdb** (and originally by the molecule editor **MEDIT**) is read and processed.

species.che File in the chemical format (*che-file*) is read and processed. The corresponding file species.mol is created or rewritten (in this case the old species.mol is backed-up as species.mol (species.mox for DOS). See Sect. 4.2.

species If no extension is specified, species.mol is tried first. If it exists, it is read and processed, otherwise species.che is processed instead and the corresponding species.mol is created. Thus, if you call **blend** for the second time, species.mol and not species.che is read.

-cnumber This option affects how chirality information on CH1E and other chiral atoms is treated. See Sect. 3.8.5.

- c0 Chirality info is taken from the .mol file or determined by using z-coordinates in the .che file (default). If a 3D configuration is available and the calculated chirality differs, a warning is printed.
- c1 Calculate unknown (i.e., not given in the .mol file) chiralities from the configuration (.plb, .che, etc.) and write them to the .mol file. The old species.mol is backed-up as species.mol (species.mox for DOS).
- c2 Calculate all chiralities from the configuration (.plb, .che, etc.) and write them to the .mol file. The old species.mol is backed-up as species.mol (species.mox for DOS). Warning is printed if the calculated chirality does not match the former chirality from the .mol file.

-dnumber Automatic bond generation. For positive number, a bond is added if the atom-atom distance is less than number% of the bond equilibrium distance. For negative number, all bonds are erased first. Useful, e.g., if some bonds are missing in the PDB file but the coordinates are good, or if ionic bonds are created (cation-anion strong force) and these should be turned into explicit bonds. Bond info must be present in the par-file. Useful range of number is between 110% and 120%; values larger than 130% may lead to extra bonds when atoms are accidentally close together. Values less than 100% issue a warning. Note: bonds are not created if there is a record in the PAR file for the bond, but the force constant is zero. This is the case of fake “bonds” as H-H in water to define a rigid molecule.

-e [no 2D input] Selects editing (adding/removing atoms/bonds). Cannot be combined with a 2D input (see below). If file `species.edt` exists then the edit information is read from this file, otherwise from standard input.

Note that any editing is performed BEFORE the molecule is minimized and shown. The recommended style of interactive work is to run two instances of **blend**, one for showing atom ID's, and the second for editing. After editing, kill the first **blend**.

Available commands are

? Print brief help

aa id type charge [id1 [id2 [...]]] Add Atom and connect it to atoms id1, id2, ... It will be marked as ‘missing’ and its position will be calculated later (cf. option -k). Note: thus, after short minimizing, you should start minimizing again with **keep=0** or re-run **blend**.

af id type charge [id1 [dist]]

af id type charge = X Y Z Add Free atom (not connected). It will be placed dist from atom id1 on the opposite side than id1's neighbors (or randomly if id1 does not have neighbors) and will not be marked as ‘missing’. The second form adds the atom with given absolute coordinates X,Y,Z. Blend should be called with -y0 or -y2

aw [id1 [dist [#]]]

aw = X Y Z Add Water molecule (TIP3P model). Its oxygen will be placed dist from atom id1 on the opposite side than id1's neighbors (or randomly if id1 does not have neighbors) and will not be marked as ‘missing’. Both water hydrogens will be marked as ‘missing’. The atom ID's are W#-O, W#-H1, and W#-H2; if # is missing, it is advanced by 1 from previous **aw** command (or is 1 for the 1st **aw** command). The default dist is 2. See the note for **aa** statement. The second form places the water oxygen to X,Y,Z specified. Note: Equivalent to the following sequence of commands:

```
af W#-O OW -.834 ID1 DIST
aa W#-H1 HT .417 W#-O
aa W#-H2 HT .417 W#-O
```

ra id Remove Atom.

**rm** id Remove whole Molecule. id is the ID of any atom in the molecule.  
**ab** id1 id2 Add Bond.  
**rb** id1 id2 Remove Bond.  
**i** id newid Change ID (rename atom)  
**q** id charge Change charge.  
**t** id type Change type.  
**p** id Print info on atom(s) id.  
**pf** id type charge [grid [shell [NforPLB]]]  
**pw** [grid [shell [NforPLB]]] Probe using Free atom / Probe using Water.  
 Given probes (atom for **pf**, TIP3P water for **aw**) are placed to a grid around the molecule and the energy is calculated; in the case of water, the water molecule is first rotated to give the minimum energy. The minimum energies are then printed and optionally the playback files containing the molecule and NforPLB minimum energy probes are generated. This may be optimized by a special patch: compile-time option causing omitting all atoms kept on place from energy calculations (`#ifdef OMITKEPT` in `blendmin.c`)  
**react** rea-file where rea-file is a reaction-file (extension `.rea`, see Sect. 2.3).  
 Thus, 'reaction' can be performed during editing. Example:

```

af prot HP 1 16-OAC .8
react p.rea
.

```

**ff** parset Sets the force field (parameter set), also `parameter_set=parset`. If set, the editing commands are performed only if the current force field matches the `ff` command. Turned off by empty parset (i.e., editing is on). This allows to have different editing statements for different force fields in one `.edt` file. Example:

```

ff charmm22
! this applies for charmm22 only
t CYS13aSG S
ff charmm21
! this applies for charmm21 only
t CYS13aSG ST
ff
! this applies for any force field
q CYS13aSG -0.8

```

! Any comment.

.

**end** End of editing (not needed for input from species.edt)

Blank lines are simply ignored. Wildcards `?` (matches exactly one character) and `*` (only at the end of id, matches any number of characters) can be used in id's. Example: if your mol-file has been created for the che-file (in some previous



`blend` run), it has atom ID's of the form 12-CH2E. Specify 12-\* if you know the number but not the atom type.

`-enumber` [2D input] Scaling for 2D input given in % of the default values. If number is positive, the plane configuration is “waved” to prevent numerical problems with pure 2D starting configuration. This can be switched off by using negative number. Active only if 2D input has been specified (option `-r2` or `-r12`, or `.che` file as the only source of information: in this case option `-e` does not mean editing!).

`-hnumber` Constraint bond angles containing light atoms (typically hydrogens). This information is written to a ble-file applies for a subsequent `cook*` simulation, not the optimization performed by `blend`!

$\text{abs}(\text{number})+0.5$  is the mass limit (in atomic units) to treat bond angles constrained. Thus, `-h-` (default) sets the limit to 0.5 (all bond angles with hydrogens can bend), `-h` or `-h1` sets the limit to 1.5 (bond angles with hydrogens are fixed but all other atoms not). This applies only to output passed to `cook` where the bonds corresponding to constrained angles are marked by `K=0.0` in the table of bonds.

If the argument of `-h` is negative, all angles containing light atoms are constrained, irrespective of overdetermination and singularities.

Normally the argument of `-h` is positive. Then the program removes the overdetermined bond in groups like `-CH3` and constraints only one bond of two possible in a chain like `-C-NH-C-` and does not constraint the `H-N-H` angle in `-NH2`. No general algorithm detecting overdetermination or singularities is used but the used algorithm constraints correctly hydrogens in proteins and TIP3P water.

Using `-h` improves generally energy and constraint conservation with a marginal impact on the generated trajectory. Higher values of `-hnumber` are not recommended.

`-jnumber` This option has two purposes:

1. Allows adding special bonds (harmonic springs between atoms). Useful e.g. for keeping certain distances (nearly) constant (if the force constant is high). These bonds are used while energy minimization and, if option `-o` is used simultaneously, exported to `cook`. These ‘bonds’ are specified either in file `species.jet`, or, if this file does not exist, may be given from keyboard. The format of data is similar to the format of the table of bonds, See Sect. 3.6, only atom IDs replace atom types. Example:

```
!REMARK : Distances to keep tail from folding up:
! id      id      K length
ASN3CA  TYR159CA  1    61
LEU22CA TYR159CA  1    52
ALA26CA TYR159CA  1    84
end
```

where K is in [kcal/mol] and length is in [Å]. Single statement **end** or **.** marks the end of data (not needed if data are read from file).

HINT: if you want to use these constraints only within **blend** and not in **cook**, re-run **blend** with these options: **-m- -w- -o file**, but without **-j**.

2. Dihedrals to constrain can be also given in this file. Example:

```
! ID ID ID ID ANGLE/degree(0,180)
5-CT 6-CT 7-CT 8-CT 120
```

BUGs (in V2.1k):  $\cos(\text{angle})$  is used internally, 0 and 180 are not allowed and the sign of ANGLE is not distinguished.

**-knumber** This options allows to keep some atoms in fixed positions during minimization.

**-k0** All atoms (incl. those marked by \* in the 1st column in species.mol) are free and can move during minimization.

**-k1** The \*marked atoms are kept fixed; however, atoms with missing coordinates are always free to allow energy minimization.

**-k2** Atoms with missing coordinates are free, known ones are kept fixed (marking is irrelevant).

**-k3** If there is any missing atom, **-k2** applies, otherwise **-k1**. This is the default.

**-k4** Read keep info from file species.keep. Should be combined with **-k1** to be active (**keep=1**), i.e., use **-k5** to read the keep info and minimize using it.

**-k8** Read mark info from file species.mark.

**-k-1 -k-2 -k-3** As above with the exception that single atoms are always free. Particularly, option **-k-3** is intended to initial minimizing a protein with charges compensated by counterions: protein skeleton is kept, unknown hydrogens and counterions are free (note that hydrogens are filled by **blend** while counterions have been exported from **pdb** and are inaccurate and may clash with hydrogens).

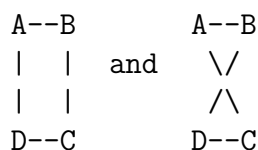
**-mnumber** Energy minimization (optimization). If number is positive, then number steps of the conjugate gradient method is performed. If number is negative, number/4 steepest descent steps are performed and then number steps of the conjugate gradient method. The default is **-m-100**.

**-nnumber** Number of molecules. Not used by **blend**, only passed via a ble-file to program **cook**.

**-n-number** If the argument is negative, **blend** tries to split a 'molecule' into submolecules (clusters). The format of output ble-file is changed and understood by **cook** as a configuration. PDB-file with crystal water or insulin consisting of two identical chains are typical examples.

BUG: the algorithm for determining whether two clusters are identical is simplified. Clusters having the same numbers and types of atoms and bonds on

each atom are considered identical. Thus, clusters written with different order of atoms are not recognized as identical. On the other hand, the following two clusters will be incorrectly considered as identical:



WARNING: If the mol-file has been generated from a che-file, it may happen that numbers in atom ID's do not match the real numbering because sites might have been renumbered to have consecutive numbering of clusters.

TECHNICAL NOTE: since **blend** does not know the final number of species, it first writes `nspec=<number> !?` to the ble-file. After all molecules are processed, and if `-o` has been specified, it re-reads the ble-file and writes the correct final number of species in the form `nspec=<number> !fixed`. This does not work if `-o` has not been specified (i.e., you use option `-v0` and either `-s` with `$w` command or redirection of standard output).

- pnumber Write configuration in the playback-compatible format. Optional number > 0 means that number% of the van der Waals radii of atoms will be used as the radii of atom spheres (default is 70%). Two files, species.plb and species.go1, will be created.
- p-number As above with reversed endian of species.plb. To be used for transferring binary data between computer architectures using the same format of floating point numbers (e.g., IEEE) but different order of bytes (endianness, sex). E.g., x68 (IBM-compatibles) and Dec Alpha are little endians while Sun, SGI are big endians. If this does not help, ASCII data should be transfered (utility plb2asc).
- qnumber Scaling of charges in %. The default is 100%. Note that scaling is done on input, i.e., the scaled charges are written to output. At the same time, warning on fractional charges is suppressed. Hint: use `-q100` to suppress the warning and keep charges unchanged
- rnumber Read configuration option:
  - r0** Initial configuration will be read from available configuration (tried in the order of types species.plb, .pla, .3db, .3dt), if not available then derived from 2D data in species.che (or deprecated species.2db, species.2dt), and if this does not help, then filled with random numbers. This is the default.
  - r1** Initial configuration is random. No action is taken to prevent atoms overlap. Note that certain force fields (Busing) may fail for atoms very close together.
  - r2** Initial configuration will be obtained from 2D screen coordinates. If the molecule is given by species.mol file, then species.2db (binary) file will be read, if the molecule is given by species.che, then the 2D coordinates are derived from it. See also option `-e`.

- r12 If the molecule is given by species.mol file, then species.2dt (text) file containing screen coordinates will be used to determine the initial 3D configuration. See also option -e. (Obsolete)
- r3 Initial configuration will be read from species.3db (binary) file. Note that format .3db is deprecated since V2.1a and replaced by .plb.
- r13 Initial configuration will be read from species.3dt (text) file.
- r4 Initial configuration will be read from the playback file species.plb. Special sub-options can be specified:
  - r4:frame Selected frame is read from the playback file. frame = 1 is the 1st frame, frame = -1 denotes the last frame, frame = -2 the second last, etc.
  - r4:from:to:by[:file] For special functions (see options -A -E -G -R) as well as the PDB output (options -w10, -w20, -w30), the range of frames analyzed is specified. The default is [-r4:1:-1:1]. Parameter file must be given with extension (usually .plb); if omitted, species.plb is the default.

NOTE: for PDB output, see also `pdb -pFROM:T0:BY`.
- r-3 Initial configuration will be read from deprecated species.3db file with reversed endian.
- r-4 Initial configuration will be read from the playback species.plb file with reversed endian. Use for transferring data between machines with the same format of floating point numbers (now usually IEEE) but different order of bytes (endian, sex).
- r5 Alpha helix, old algorithm
- r6 (Added in version 1.6d, changed in 1.7k; see also -k7 and -k5.) The initial configuration is alpha-helix. The molecule must be a protein (or peptoid, i.e., N-substituted glycine; the N-backbone may need finer minimization). The backbone is recognized by atom types (see Sect. 3.4.10). The alpha-helix configuration is generated and the backbone atoms are marked as to be kept during minimization (cf. option -k2). The side-chains are very approximate and thus the first step should be energy minimization which should not include the backbone: the keep status is set to -k1 (see option -k. As the second step, the backbone atoms should be included in the minimization (use `keep=0` or re-run `blend`). Example:
 

```
blend -r6 -g -t10 polyval.che # creates alpha-helix
                                # + optimizes side-chains
blend -o xxx -g polyval      # optimizes whole config.
```
- r7:phi:psi:omega (Added in version 2.0k.) The initial configuration is described by dihedral angles phi (-C-N-Ca-C-), psi (-N-Ca-C-N-), and omega (-Ca-C-N-Ca-) (in degrees). If no phi, psi, omega are given, the default is phi=psi=-51, omega=180, which is approximate alpha-helix (to be optimized). If more molecules are processed, the default is the values of previous molecules.

- tnumber The range [C1,C2] for the cutoff switch function will be  $C1=\text{number}-1$  and  $C2=\text{number}+1$ . The default is no cutoff. See Sect. 3.5.
- t-number The range [C1,C2] for the cutoff switch function will be  $C1=\text{int}[\text{number}/100]/10$  and  $C2=C1+(\text{number} \bmod 100)/10$ . E.g., -t-12345 gives  $C1=12.3$ ,  $C2=12.3+4.5$ . Normally the -tnumber form is preferred.
- unumber All non-bonded pair energies (i.e., site-site [typically Lennard-Jones] and Coulomb ones) are recorded and first number highest terms are printed. Thus, possible atom overlaps are easily visible. If number is negative then first -number terms with lowest energy are printed.
- w Write configuration in some special formats:
  - w Write configuration to species.3db. If this file already exists, the old one is backed up as species.3db (species.3dx for DOS). This was the default prior V2.1a; the 3db format is now deprecated—plb is recommended instead.
  - w- w0 Do not write configuration; does not apply to the plb format (use -p0 to suppress writing species.plb). This is the default.
  - w-1 As option -w with reversed endian.
  - wdigit If digit > 1 then write configuration to ascii species.3dt. digit is the number of decimal digits printed. If this file already exists, the old one is backed up as species.3dt (species.3dx for DOS).
  - w10[:Hstyle] Creates PDB file species.pdb using backbone analysis. To be used for che-file based peptides created, e.g., by makepept. BUGS: This is a simple code that may fail in complex cases. Does not recognize PRO as the 1st residue. Uses non-standard numbering of H and perhaps also other atoms. None or very limited support for residues like HEM, etc. Knows only some termini. Some problems can be solved by hand-editing the resulting PDB file.
    - w10:0 Hydrogen on beta C is ' HCB1'
    - w10:1 Hydrogen on beta C is ' HB1 ' (C as Carbon omitted). Hydrogen on the backbone N is always NH1 (N is not omitted).
    - w10:2 Hydrogen on beta C is '1HCB '
    - w10:3 Hydrogen on beta C is '1HB '
    - w10:4 Hydrogens are omitted (default)
 Can be used with option -r4 to create a series of PDB files.
  - w20 Re-creates PDB file species.pdb using id names in mol-file, to be used for files created by the pdb converter. BUGS: 1 backbone only; changes order of atoms (in this case, try utility pdb2pdb, see Sect. 24.1.) Can be used with option -r4 to create a series of PDB files.
  - w30 Selects -w10 or -w20 automatically
  - w40 Write file species.atm with a header containing the number of atoms, a blank line, and then 1 line/atom in 4 columns At x[A] y[A] z[A].

**-w80** Write a `cfg`-file in the `cook` format (V2.7a and newer. Cf. `plb2cfg`).

**-w160** Write a `cfg`-file in the `cook` format (older than V2.7): deprecated.

The **-w** options may be combined (**-w125** = `.3dt` in 5 digits + `.atm` + `.cfg`); however, `.3db` and `.3dt` cannot be combined (only one is written).

**-y`number`** Center molecule or create box. Sum of flags:

- 1 Center molecule before minimizing; box size (if contained in the `plb`-file) is not changed. Recommended to show a molecule.
- 2 Center molecule after minimizing and set box size (written to the `.plb` file) to zero (free boundary conditions). Note that `cook` with `init<0` reads only the contents from such a `plb`-file (the box size is derived from `cook` initialization).
- 4 Make all coordinates positive, incl. Lennard-Jones sigma. This means that min and max in all coordinates are determined and the molecule is moved so that all coordinates are positive.
- 8 Box molecule after minimizing, incl. Lennard-Jones sigma. Maximum of all coordinates is written as box size to the `plb`-file. Recommended with **-y4**. Note that `cook` with `init<0` reads such a `plb`-file including the box size.
- 16 Modifies **-y4** and **-y8**: do not add Lennard-Jones sigma to coordinates (may lead to overlaps in periodic b.c.). (To modify the box size, use `plbbox`).

The default, suitable for making a `blend`-file, is **-y3**. To work with a configuration in periodic b.c., use **-y13**

## 2.2.4 Extra-options

**-A** Generate `species.ang` of independent internal coordinates. File `species.plb` (and `species.mol`) will be read (see option **-r4**) and the following files will be generated:

`species.ang` protocol about angle analysis

`species.1`, `species.2` ... angle files

Option **-A** implies **-w0**, **-p0** (no write), **-m0** (no minimization), and `all_dihedrals=0` (one dihedral per bond).

The angle analyzing algorithm takes into account only the following structures:

**3- and 4- bonded atoms** 1 dependent angle is removed, 2 angles and 1 improper are left

**6-cycles** of 6 dihedrals and 6 angles, 6 are removed

**5-cycles** of 5 dihedrals and 5 angles, 6 are removed

**merged rings** 1 additional improper removed

NOT SOLVED other cycles than 5- and 6- (like if there are CYS-CYS bridges);  
complex merged rings (like in the heme)

Actually, the topological analysis to get only independent angles is not easy. The formula  $2*ns-5$  works for trees only (no cycles). WARNING (?): tested with CHARMM21 only.

-Cnumber (-1) Reference atom for dipole and quadrupole calculations, and for the second virial coefficient (see -V; in this case, both molecules must have the same atom as the reference). Note that the dipole value depends on the reference point for ions only, and similarly the quadrupole value depends on the reference point for dipolar molecules.

number>=0 Atom number.

-1 The reference point is the centroid of atomic charges (default).

-2 The reference point is the center of mass.

-3 The reference point is the geometric center.

-D File species.msd of mass-weighted MSD (mean square displacement) is calculated. Implies -w0, -p0 (no write), -m0 (no minimization).

-E Essential dynamics. The essential dynamics (ED) algorithm analyzes the trajectory (in playback file, see option -r4) and by diagonalizing the covariance matrix calculates the “essential motions”. Implies -w0, -p0 (no write), -m0 (no minimization).

-EA use all atoms in ED

-EH use all heavy atoms (not hydrogens)

-EC use only C-alpha (cheapest but usually sufficient)

-EB use all atoms in the backbone (omits all sidechains)

It is assumed that atom IDs have been derived from a PDB file.

### Algorithm:

1. reads frames and matches frames with the 1st frame which means that r is rotated and displaced so that  $\sum(r_i - r_{0i})^2$  is min (simple MC minimization)
2. calculates the covariance matrix  $rmCov(r_i, r_j)$  (all coordinates separate, i.e., rank=3\*#\_of\_atoms)
3. calculates its eigenvalues

## More about ED:

- Essential dynamics files **ess####.plb** use the average frame as the center
- This average frame is also written as **essr0.plb** (if **-F#** is odd, this is the same as the central frame)
- The amplitude of essential motion in **ess####.plb** is the same as to reproduce the stdev (mean square amplitude), i.e., cos-amplitude is  $\sqrt{2} \times \text{stdev}$  and linear  $\diagup \diagdown \diagup \diagdown$  amplitude is  $\sqrt{3} \times \text{stdev}$  (if the essential motions were harmonic or  $\diagup \diagdown \diagup \diagdown$ , respectively, the **ess####.plb** would show the correct motion). This holds for **-M1** (the default), **-M#** just multiplies the motion by **#**
- File **ess.cp** is written containing development of coordinates in the basis of eigenvectors. There are **-P#** items (columns) in **ess.cp**. Use **showcp -a** or **showcp -p** etc. to analyze **ess.cp**, ignore bad headers of columns 0 (Etot) and 1 (T)
- the matched trajectory (with rotations and translations removed) is written to **essmatch.plb** if option **-P**. **blendess.c** must be compiled with **#define WRITEMATCH**.

Note: try **blend/esstest.c**

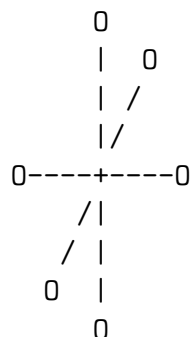
**-Fnumber** For essential dynamics and normal mode analysis. Number of frames in the output playback files used to visualize the motion. Positive number denotes harmonic motion, negative denotes linear motion between min and max. Cheap: use **-F2**, better: **-F7** or so. See also **-M** and **-P**. Of course, **-F-2=-F2**, **-F-3=-F3**.

**-G** Calculate eigenvalues of the inertia matrix. Three files are produced:

**inertia.cp** Convergence profile of 7 items:  $I_{xx}$ ,  $I_{yy}$ ,  $I_{zz}$ ,  $R_x$ ,  $R_y$ ,  $R_z$ ,  $R_{\text{gyr}}$ .

**inertia.mol** Mol-file file for **show** to visualize **inertia.plb**. 'inertia' is viewed as a 'molecule' of 6 atoms

**inertia.plb** Playback file for 'inertia'. The following Cartesian cross or 'molecule':



where the masses of 'atoms' O are 1/2 and the lengths of axes (O-center) are  $R_x$ ,  $R_y$ ,  $R_z$ , has the same tensor (matrix) of inertia as the original molecule



BUG: the sign of both vectors in each O-O ‘bond’ is undefined. This does not cause any problems while showing, but might cause problems in further calculations

**Theory:** The inertia matrix  $M$  is calculated first

$$M = \sum_i m_i \Delta \vec{r}_i \Delta \vec{r}_i$$

where

$$\Delta \vec{r}_i = \vec{r}_i - \vec{r}_{\text{center-of-mass}}$$

The diagonalized inertia tensor is

$$I = U M U^{-1} = \begin{pmatrix} E_1 & 0 & 0 \\ 0 & E_2 & 0 \\ 0 & 0 & E_3 \end{pmatrix} = \begin{pmatrix} E_2 + E_3 & 0 & 0 \\ 0 & E_3 + E_1 & 0 \\ 0 & 0 & E_1 + E_2 \end{pmatrix}$$

In the same spirit of “radius of gyration”, the following “partial radii of gyration”  $R_a$  are defined:

$$R_a = \sqrt{E_i / m_{\text{tot}}},$$

where  $m_{\text{tot}}$  is the total mass of the molecule. Finally,

$$R_{\text{gyr}} = \sqrt{R_x^2 + R_y^2 + R_z^2}$$

- Inumber POLAR ONLY: maximum number of self-field iterations before an error is printed. The default is 200. With virial calculation (see -V), divergence is treated as infinity potential. Since the internal accuracy is 1e-8, this means that the convergence rate must be better than  $1e-8^1/200 = 0.91$ ; increase -I if you have such an ill-defined model. On the other hand, you may decrease -I if you are sure the convergence is fast enough.
- Jnumber Essential dynamics and normal mode analysis: accuracy for the diagonalization (Jacobi method)  
Essential dynamics:  $\text{sqrt}(\text{number}) \cdot 0.2$  is used as the error threshold for configuration matching when the configurations are translated and rotated to match best each other (before the covariance matrix is calculated)
- Mnumber For essential dynamics and normal mode analysis. Amplitude of the visualized motions. The units are Å for normal mode analysis, and standard deviation of the essential motion for the essential dynamics. The default is 1 (Å or standard deviation).
- M-number For normal mode analysis, the same as  $\text{number} \cdot \text{ns}^{1/2}$ . -M-0.2 gives a reasonably good amplitude irrespective of molecule size.
- Nnumber Calculate the normal mode frequencies (vibrational spectrum). If number is given, it denotes dr for numerical calculation of second derivatives; default **dr=1e-5** Å. File species.nmf is created. See also -M, -P, and -F

**Theory:** At energy minimum

$$U = U_{\min} + \frac{1}{2} \frac{\partial^2 U}{d\vec{r}_i d\vec{r}_j} (\vec{r}_i - \vec{r}_{i,\min})(\vec{r}_j - \vec{r}_{j,\min})$$

where the derivative is calculated by the simplest 2nd order formula with step  $\Delta r$ .

By diagonalizing matrix

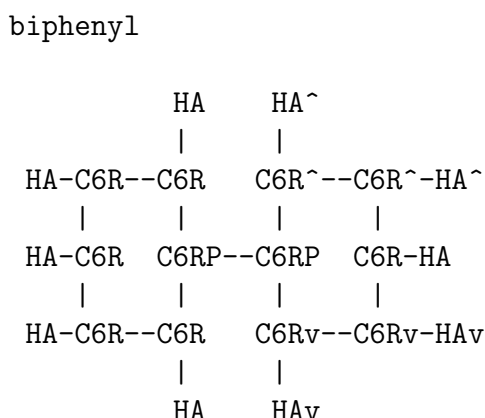
$$U'_{ij} = \frac{1}{2} \sqrt{\frac{1}{m_i m_j}} \frac{\partial^2 U}{d\vec{r}_i d\vec{r}_j}$$

one gets the squared circular frequencies  $(2\pi\nu)^2$  (the eigenvalues) at the diagonal. The input configuration should be well minimized: use `-mBIG_NUMBER` or `cg=BIG_NUMBER` and wait until the minimization stops; to be on a very safe side, repeat once more with `-m-100` or `sd=100`. The output file `species.nmf` contains frequencies  $\nu$  in THz and wave numbers in  $\text{cm}^{-1}$ , ordered. If the eigenvalue is negative (saddle point—not enough minimized or numerically imprecise), then  $\nu$  printed is negative, too.

**Accuracy:** If the step  $\Delta r \approx (\text{machine precision})^{1/3} \approx 10^{-5}$ , then  $U'$  is known with precision of 10 digits. Six (for a general molecule) eigenvalues corresponding to rotations and translations should be zeros  $\Rightarrow$  they will be calculated with relative (to other eigenvalues) precision of about  $1\text{e-}10$ . Since the frequencies are square roots of eigenvalues, they will be given with relative precision of  $1\text{e-}5$ . Consequently, frequencies slower than  $1\text{e-}5$  of the fastest ones, i.e., slower than about 1GHz, cannot be calculated.

**Diagonalization method:** threshold Jacobi method

**Example (especially didactic):** Create file `biphenyl.che`:



then do the following:

```
blend -p -N -M.4 -P99 -F7 -m-9999 biphenyl
(biphenyl.nmf and nm0000.plb, ... created)
show biphenyl nm0006
rotate the molecule (left mouse button)
type hot keys: '$' 'i' 'w' ... Esc Esc $
```

BUG: cannot be used for models with zero masses of some particles (as TIP4P water)

**-Pnumber** For essential dynamics and normal mode analysis. Number of playback files generated. number of the most significant eigenvalues is considered (number least significant if number is negative). Files are named **ess####.plb** or **nm####.plb**, where **####** is 0000, 0001, ... Use also 'blend -p' to get the gol-file, then **show** to view the motion. The default is **-P0** (no playback files).

**-Rnumber** Core size for the second virial coefficient (see **-V**). The center of this spherical core is given by **-C**.

**-Tnumber** Temperature (in K) for the second virial coefficient (see **-V**).

**-Vnumber** Number of random samples for the second virial coefficient. Negative value means that twice as many samples are used and every second is generated with molecule 1 reflected. Use this if both molecules are dipolar.

Caveat: both molecules are treated as rigid (after separate optimizations).

To calculate the second virial coefficient, use 1, 2, or 3 mol-files as blend arguments:

- 1 The second virial coefficient of pure compound is calculated.
- 2 The cross second virial coefficient (for a mixture) is calculated.
- 3 For development purposes: the third argument must correspond to a mol (or che) file with both molecules merged (in the same order of sites).

An example of the 2nd virial calculation at 373 K is:

```
blend -C1 -V-1000000 -T373 spce
```

The algorithm is based on the formula:

$$B_2 = \frac{2\pi}{3}R^3 - 2\pi \int_0^1 /R \langle f(1/x, Omega) \rangle_{\Omega} 1/x^4 dx$$

where  $f(r, \Omega) = \exp(-u(r, \Omega)/kT) - 1$ ,  $\Omega$ , stands for all angles, and  $R$  is the core size (option **-R**; the potential can be assumed infinity inside the core).

First, the configuration is split into molecules. Both are minimized separately (in two steps, with the other molecule shifted far away).

Then, the above integral is calculated by the naive Monte Carlo (random shooting) If the **-V** option has a negative value, every second configuration is created instead by inverting ( $r \rightarrow -r$ ) the first molecule. This is necessary for

both molecules dipolar because the integrand is then limiting to a constant as  $r \rightarrow \infty$  (otherwise it would be infinity and the integral would be unsuitable for the MC method).

Works with polarizability, too; note that a diverging self-field is treated as infinity potential. (See -I for the number of iterations.)

## 2.3 File extensions

File names of all molecule-related files are derived from species or species.mol or species.che specified at command line. A list of extensions follows. ‘(i)’ means that the file is can be read by blend, ‘(o)’ that it is created or rewritten.

- .2db (i) OBSOLETE! 2D screen configuration (binary). Produced by MEDIT (?).
- .2dt (i) 2D screen configuration (2 column x-y ASCII file). Produced by MEDIT (?).
- .3db (io) 3D configuration (binary file). It consists of `number_of_atoms` vectors, each consisting of three single precision floating-point numbers containing x-y-z.
- .3dt (i) 3D configuration in ASCII (3 column x-y-z).
- .3db (o) Backup of .3db (UNIX, OS/2 etc.).
- .3dt (o) Backup of .3dt (UNIX, OS/2 etc.).
- .3dx (o) Backup of .3db or .3dt (DOS only).
- .che (i) Chemical format. See Sect. 4.2.
- .edt (i) Edit molecule file. See option -e.
- .dep Dependant info file, with lines of the form `dependants : base sites`. See Sect. 5.10.
- .keep Info on atoms kept (frozen) while minimizing. Can be written by hot key `@k`, read by `@K` or option -k4. The file format is simple:  

```
! comment
2 ! atom number
3 ! atom number
```
- .mark Info on marked atoms. Can be written by hot key `@m`, read by `@M` or option -k8. The file format is the same as for .keep
- .pch (i) Partial charge replacement file, must be in the current directory or in the directory given by BLENDPATH. If present in the input line, all patterns for groups of atom types replace the respective patterns found in the molecule, overriding any charge specified in the .che or .mol files. Mol-file will be rewritten with the new charges. Example of the pch-file:

```

! >C=O
C=0.55      O=-0.55

! -NH2
NT=-0.3     H=0.15     H=0.15

```

The first atomtype=charge item on line is the ‘central’ atom, all other atoms are connected by bonds. Ordering of atoms but the first one defining the center is irrelevant. The charge assignment algorithm takes sites in a molecule one by one and for each tries to match all possible groups of this site and its nearest neighbors to patterns in the .pch file. If there are several matches possible, the first one applies; thus, the following trick is possible:

```

CT=-0.4 HA=0.1 HA=0.1 HA=0.1 HA=0.1
CT=-0.3 HA=0.1 HA=0.1 HA=0.1

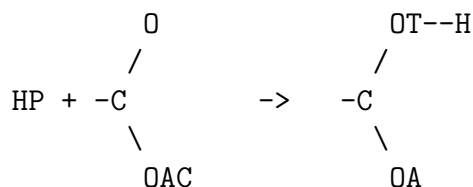
```

Using this, the first line will be used for (full-atom) methane and the second one for ethane.

If an atom is a member of several groups, its charges is calculated as a sum over all group contributions and info is printed.

**.pro** (o) Results of probing. See 2.2.3, option **-e**, commands **pa** and **pw**. See also utility **showpro** (24.8).

**.rea** (i) Chemical reaction file. Performs a special form of ‘chemical reaction’ on groups defined by reaction patterns. Can be given also by command ‘react’ during editing. Example: to perform the following reaction



the rea-file should contain a line like this:

```
HP + O <1.2 *C OAC += H-.6 OT+.35 C+.05 OA+.2 > -100
```

**<** the reaction takes place if the distance of HP and O is less than the distance given by the ‘<’ command (here 1.2 Å, default = previous line, if nothing given = 1 Å)

**\*** \*C means that C is the central atom of the group (if no \* is given, 1st atom after ‘+’ is central)

**+=** += means that partial charges (numbers after atom types) will be added to already present partial charges

**=** = instead of += means assignment of partial charges

> energy of the reaction follows (negative=exothermic). This energy is written to the mol-file as the 'zero\_energy' field.

\ EXTRA lines: if the charge redistribution extends over one group, 'continuation' lines may be added. They start with '\', e.g.

```
\C5RE + N5R H C5R += C5RE+.05 N5R+.1 H+0.05 C5R+0.05
```

Here, '+' does not denote a reaction but an atom that has been marked in previous reaction step (=line without '\' + any number of lines with '\'); it must be bonded to the next atom (here C5RE–N5R) which itself must not be marked. Marking of other atoms in the group is irrelevant. Note that any such marking is canceled by a new reaction (=line without '\'). The above example changes charges in the group:

```
H--N5R---C5R
      /
     C5RE
```

where C5RE has been affected by a reaction but N5R not

.geo (i) If this file is present, **blend** prints selected distances and angles on hotkey 'E' and after minimizing is finished. species.geo is just file of lines with sites of bonds or angles

```
! comment
1 2 # bond 1-2
1 2 3 # angle 1-2-3
```

The comments are printed, too.

BUGS: very ugly implementation, should unify with 'clicking', should add dihedrals...

.gol (o) File containing a 2-line header followed by colors and radii of atoms. Used by **show**. Originally the goal file **goal.fil** for **PlayBack**.

.jet (i) File determining additional constraints. See Sect. 2.2.3, option -j.

.mol (io) Molecule description file. See Sect. 4.1.

.mox (o) [DOS]

.mol (o) [UNIX etc.] Backup of .mol. (**blend** in several cases rewrites the .mol file; in these cases, the old version is backed-up.

.plb (io) 3D configuration (binary file). It has 2 float numbers as a header, otherwise it is identical to the .3db file. The first float of the header contains the number of sites, the second should contain 0e0 (in other cases, it contains box size). For **show** and Niels' program **PlayBack**.

.sym (i) File defining some symmetry conditions to be kept while minimizing. Very primitive. Turned off if -N specified. Example:

```

z 0 2
z 3-4
z 5
.

```

z 0 2 means that sites 0 and 2 are kept symmetric with respect to  $z=0$  plane. z 5 means that site is kept in  $z=0$  plane. z 3-4 means that sites 3 and 4 are kept in  $z=0$  plane. . stops Instead of x,y,z one may use X,Y,Z or 0,1,2.

Other extensions (not related to individual species):

```

.par (i) Parameter file in text format
.bin (io) Its binary image.
.ble (o) Default extension for the output file.

```

## 2.4 Run-time control

If no option `-o` is given, program `blend` runs in the *interactive mode*. The user can interactively control the course of minimization and respond to error messages. The output is written to `stdout`. If option `-s` is also given, the user can watch several last screens of output. (Do not confuse with key and mouse control in the graphical window!)

If option `-o` is given, only few messages are printed to `stderr`. The only user interaction is typing `ctrl-C` (see Sect. 2.4.4).

### 2.4.1 *get data* format for input

The *get data* format resembles the `GET DATA` statement in PL/1 or `NAMelist` in FORTRAN. It is used as user-friendly input format in several places.

The input data consist of assignments of the form variable=numeric expression.

A set of such assignments is ended by a semicolon. The numeric expression may contain numbers, variables, operators `+` `-` `*` `/` `^` `(` `)` `#` `%` and elementary functions, where symbol `^` denotes exponentiation and `#` the result of previous assignment. Operators `+=` `-=` `*=` `/=` `^=` `%=` are also allowed if `#` is not used.

If used interactively, some additional commands are available so that you can use the *get data* module as a calculator.

Example:

```

ran=1/3    ! this is comment
by *= 2;   ! by is doubled

```

List of commands:

? Get help

?? List all variables.

expression Evaluate expression and store it into #

variable=numeric expression Evaluates expression and assigns the result to variable

?variable=numeric expression Evaluates expression, assigns the result to variable, and prints the result

?= Toggle auto echo mode (NOTE: this is useful in case of errors during input, as wrong identifier or syntax)

?format Set output format (C-style), e.g. ?%.15g is g-format with 15 digits

! Comment to the end of line

; End of data

\$command To enter scrolling (see below).

## 2.4.2 Scrolling

Scrolling is available if the program has been compiled with option `-DSCR` (i.e., `#define SCR`) and run with option `-s`.

Scrolling is called from the get data unit by \$ and a command. Then, prompt \$ appears; it is not repeated in next scroll commands. Scrolling is also available if an error occurs. Standard screen of 25 rows and 80 columns is the default screen.

A list of scroll commands follows:

`$?` `$h` Get help

`$u` Scroll by one screen up

`$number` Number lines down (up if negative)

`$-` One line up

`$t` `$g` Go to the top of stored screens

`$d` `$SPACE` Scroll by one screen down

`$b` `$G` Go to the bottom of stored screens

`$wfile` Write whole buffer to file file

`$ifile` Include file as if typed in

`$/string` Find string

`$/` Repeat last find



`$Lnumber` Set number of lines on screen

`$Cnumber` Set number of columns on screen. Lines longer than this limit are truncated; if you see garbage on the screen, try to re-set the number of columns. WARNING: Some screens have that bad habit of eating ends of lines longer than number of characters/line which does not fit the scroll's assumptions. If this is the case, set C to a large number.

`$!number` Set clrscr mode (try if the screen clears badly or slowly)

`$Q` Emergency quit (=exit(1))

`$.` `$q` Return control to the get data unit or error handling

`$exit` Kill program (nothing saved!)

### 2.4.3 Error handling

Error messages (Warnings, Errors and Disasters) are self-explanatory and contain also the name of the source file in C and the line number. They are printed both to **stderr** and the file defined by option `-o` (or **stdout** if not specified); the latter contains more information needed for analyzing the source of the error.

When an Error occurs in the batch mode (option `-o file`), the program exits with a nonzero return code. When a Warning occurs, message is printed and execution continues.

When either an Error or Warning occurs in the interactive mode, the user is prompted for an appropriate action (by typing a letter followed by `RETURN`).

(a)bort Abort (with a core dump).

(e)xit Exit without a core dump (see above for the return code).

(c)ontinue (Warnings only) If you wish, it is safe to continue.

(i)gnore Ignore the Error and continue. No further check is made whether it is possible to continue so that this selection is **strongly discouraged**.

(s)croll Use scroll (see above). Active only if option `-s` has been specified.

### 2.4.4 Interrupts

You can interrupt energy minimization by pressing `ctrl-C`; if the course of minimization is shown graphically, then `ESC` is preferred (in the X11 environment, `ESC` must be typed in the graphical window). `ctrl-C` pressed in the graphics window kills the program; on X11, `ctrl-C` pressed in the text window works as `ESC` in graphics.

In the interactive mode (no option `-o`) you will get back to get data.

In the batch mode (option `-o file`) just the current minimization is interrupted; type `ctrl-C` twice to break the program.

Under UNIX, the interrupt is caught immediately and you only have to wait until one minimization step is finished. Under DOS, you must wait until the first I/O; for some extenders and Windows no `ctrl-C` is accepted at all.

## 2.5 Showing molecules graphically

`blend` can generate output for program `show` (both for X11 and DOS). (Originally X11 program `PlayBack` written by Niels Jacobsen was used). Program `tomoil` can be used to translate the `blend`-format to format understood by `moil` (by C.Simmerling, University of Chicago). In the X11 and DOS/Turbo C versions there is a simple on-line graphics available.

### 2.5.1 X11 and Turbo C Graphics

Graphics is available if `blend` has been compiled by Turbo C with option `-DDOS` (i.e., `#define DOS`), or for an X-Windows system with option `-DX11`.

If `blend` runs with option `-gnumber`, the course of minimization is shown each number-th step. Pressing `ESC` interrupts the minimization and you can watch the molecule, rotate it, etc. From the interactive input for minimization, graphics is called by entering `gr=1`; (bonds only, fast), `gr=2`; (balls-and-sticks, slower), `gr=3`; (shaded balls, slow), or `gr=4`; (stereo bonds – blue-red glasses needed). On-line help for hot keys (rotating and moving the molecule, rescaling, etc.) is obtained e.g. by pressing ?.

#### Mouse:

`drag left button` Rotate molecule (or marked atoms in the ‘move’ mode) around x and y axes

`drag middle button` Move molecule (or marked atoms in the ‘move’ mode) in x and y directions

`drag right button horizontally` Rotate molecule (or marked atoms in the ‘move’ mode) around z axis

`drag right button vertically` Resize (rescale)

`click left button` Print info on the selected atom and mark the atom (e.g., for moving). Second click on any atom gives atom-atom distance, third click gives the angle. Does not work in the stereo (anaglyph) mode (`gr=4`).

`click middle button` Flood-mark, i.e. marks all atoms connected by bonds until any other marking. See also hot key `[M]`. To mark a group, left-click first the nodal atom(s) (which connects the group to the rest of the molecule), then middle-click any atom in the group.

click right button Unmarks the atom.

### Hot keys:

**cursor arrows** Move the molecule in x and y directions; if the ‘move’ function is active (see yellow info bar in the bottom of the screen), the marked atoms are moved.

**Home** **End** Move the molecule in z direction, i.e., move the eye for central projection

**x** **X** **y** **Y** **z** **Z** Rotate molecule around given axis. Lowercase rotates in positive direction (anticlockwise), uppercase in negative direction.

**w** Write (remember) molecule position (in fact, erases a list of translations and rotations)

**W** Undoes all translations and rotations (but those made with parts of a molecule)  
**W** and **w** are intended for editing a configuration which can be pasted back to cook\*. Example:

```
blend -o SYS A B C                # prepare SYS.ble
cook SYS                          # simulate+write SYS.plb
molcfg A B C SYS                  # generates SYS.mol
blend -o SYS -n-1 -g -y0 -r4:-1 SYS # move molecules, then type W .
```

NOTES: If original molecules A B C consist of non-connected parts, file **SYS.def** (variable **N[]**) will have to be changed. Option **-y0** to blend is essential. Option **-n-1** is required in the periodic b.c. These are not treated properly in blend so that **-m0** or **-mSMALL** is needed. Then, **-tSMALL** is recommended. BUG: inefficient implementation, memory consuming (problems with DOS version)

**+** **-** Resize (rescale) the screen. Do not confuse with **Home** **End**!

**\*** Double step size for the above functions

**/** Halve step size for the above functions

**r** **R** Decrease/increase radii of atoms (draw styles 2 and 3 only)

**1** Show bonds only (fastest)

**2** Stick-and-ball model

**3** Stick-and-ball model, spheres are shaded (slowest)

**4** Anaglyph, bonds shown in blue and red for the right and left eye, respectively. To view in 3D, blue and red glasses are needed

**g** **G** Toggle among the above four graphics modes

- i Label atoms by atom identifiers. For proteins taken from PDB database, this is something like ALA12CA
- t Label atoms by atom types
- n Label atoms by atom numbers
- q Label atoms by (partial) atomic charges
- l Toggle labeling style
- f Find atom. One of the above labeling styles must be set. For atom identifiers and types, wildcards ? and \* are accepted, however, \* works only as the last character. For charges, you must enter two numbers, charge value and inaccuracy. Thus, e.g., 0.3 0.05 will find all charges in the range [0.25,0.35].
- F Cancel finding, i.e., label ALL atoms again
- d POLAR only: Show induced dipoles in eA
- D POLAR only: Show induced dipoles in Debye
- p POLAR only: Show polarizabilities in Å<sup>3</sup>
- s POLAR only: Show saturations (in the saturated polarizability model only)
- SPACE Cancel any labeling and redraw
- m Toggles the ‘move’ function. Atoms to be moved must have been marked: click the atom centers by mouse or use the ‘find atom’ function (hotkey f), alternatively use hot keys M and u. The selected atoms are moved either by cursors and hot keys x X y Y z Z or by the mouse. Rotations twist around the least clicked atom (=in green) as the center. Type m again to leave the moving mode; hot keys like . ESC , cancel moving. BUG: there is no ‘undo’
- L Mark all labeled atoms (see hot key f)
- M Flood mark. Marks all bonded atoms connected by bonds until an already marked atom is found. Groups can be readily marked. Equivalent to middle button click.
- u Unmark all marked atoms
- I Invert marking (swap marked ↔ not marked)
- k Show keep status. Atoms kept while minimizing (cf. option -k, variable ‘keep’, and \* in the 1st column of the mol-file or in front of atom id in the che-file) are marked.

- K** Inverse to hot key k: Will keep the marked atoms while minimizing. In addition, ‘get data’ is forced after **ESC**.
- A** Average charges of marked atoms and unmark. Useful, e.g., for editing quantum-mechanical charges: mark a group of equivalent atoms (e.g., all three hydrogens in a -CH<sub>3</sub> group), then press **A**.
- b** **B** Recolor bonds, cycles through 9 preferably light colors: yellow (default), white, gray, dark gray, blue, green, cyan, red, magenta
- c** Recolor carbons, cycles through 3 colors: cyan (default), gray, dark gray. BUG: dump screen functions fail for dark gray.
- #** **=** Toggle none/ 1 Å / 0.1 Å grid (graphic modes 2 and 3 only).
- ,** Cycles through different styles of showing atom centers (graph modes 2,3 only) or turns centers off.
- e** Erase/draw-over toggle. Useful while minimizing to see conformational changes. The default is writing over with gradual shading off old positions, first value is ‘animation’ (old molecule is erased, new drawn), second value is writing over (without shading, fastest option).
- P** PPM (raw Portable PixMap, P6) dump of screen to file **blend000.ppm**; further dump is **blend001.ppm**, etc. Colors are inverted for draw styles (graphic modes 1 and 4, with the exception that the background (black on screen) is always white. Bug: dark gray for carbons (see hotkey c) does not work well.
- C** PostScript dump of screen to file **blend000.ps**, see above for more info.
- 0** PostScript black-and-white dump of screen to file **blend000.ps**. Colors are translated using 0.3R+0.59G+0.11B. See above for more info. NOTE: for fancy pictures, use **show** and **ray**.
- E** Calculate Energy, dipole moment, quadrupole moment, and geometry (according to the **.geo** file). The quadrupole moment tensor  $Q$  is defined by

$$Q = \frac{1}{2} \sum_i q_i (3 \vec{r}_i \vec{r}_i - r^2 I)$$

where  $I$  is the unit tensor. Tensor  $Q$  is then diagonalized.

NOTE: This form is common in chemistry. In physics, it is used without factor 1/2.

NOTE: the diagonalized tensor is not ordered and the molecule is not rotated into the principal axes. To get also the non-diagonalized tensor, use option **-v** (verbose).

- ESC** Escape showing: if option **-s** has been specified, you may enter data and continue interactively.

- , Minimize again (conjugate gradients); if `cg=0`, then `cg=1` is used. The same as ESC and ; Enter
- ; As above with preceding steepest descent minimizing (with `sd=cg`, or `sd=1` if `cg=0`)
- ' Toggle center points (when atoms are shown). (The center points help clicking.)
- ctrl-C Kill blend (nothing saved!)
- h ? Display brief help for hot keys

DOS: The graphics driver must be in the working directory. Tested with VGA only (driver `egavga.bgi`).

## 2.5.2 Playback output

Output in the playback-compatible format is generated if option `-pnumber` is specified. Optional number means that number% of the van der Waals radii of atoms will be used as sphere radii (default is 70%). File names for output are derived from molecule names: the `goal.fil` file (ASCII) takes extension `.gol`, the binary playback file (with 1 configuration only) takes `.plb`. The files are written to the working directory.

Playback files with long trajectories are generated by `cook` and can be also read in by `blend`.

## 2.6 Energy minimization

Energy minimization (optimization) module implements two common methods: the greedy (steepest descent) gradient method and the conjugate gradient method. The steepest descent method simply moves by certain step in the direction of forces; the step size increases in the case of success and decreases on failure. The conjugate gradient method is basically the Polak-Ribiere version [12]; the algorithm to find a minimum along a line was optimized so that in most cases only two evaluations of forces are required by one conjugate gradient step.

The steepest descent method is usually less efficient than the conjugate gradient method. It is recommended for initial high energy configurations with atom overlaps. Once energy drops, the conjugate gradients should be used.

The force field used by this minimization is the full force field including bond vibrations which will be replaced by bond-length constraints in simulations.

The energy minimization is called by `-m` option. If the molecular configuration is to be generated from random input, `-r1` is used. There is no guarantee that the resulting configuration is close to global energy minimum, a locked high-energy configuration may arise for complicated molecules. It is best seen using graphics, and it is discernible also from energy terms printed to output.

In the interactive mode, the user is after one minimization cycle prompted for entering new data. (See Sect. 2.4.1, for the data format.) On-line help is obtained by entering

```
help=1;
```

While minimizing, a protocol is printed. You can interrupt minimization by pressing `ctrl-C`. See Sect. 2.4.4.

If the process of minimization is shown graphically, press `ESC` rather than `ctrl-C` to stop minimization. Now you can watch the molecule. A second `ESC` then moves control to input data as above.

## 2.7 Missing coordinates

The initial configuration can have missing atom coordinates. These appear as blank lines or as values bigger than 9e5 in the `.3dt` file, or as values bigger than 9e5 in the `.3db` file. Program `blend` calculates approximately the missing coordinates by using known positions of bonded atoms; if it has not got enough information, a random position is generated. The configuration should be energetically minimized, preferably with the atoms with known coordinates kept in place (see option `-k`). Especially fitted to fill missing hydrogens.

# Chapter 3

## Force field and the parameter file

Program **blend** first reads in the parameter file (or its binary image); while reading the file, various checks (e.g., on duplicated entries) are made.

Then, **blend** analyses the structure of the molecule and finds both bonded and non-bonded energy terms. It looks in the tables to assign the potential parameters. The rules that apply as well as the format of the parameter file are described in this section.

### 3.1 Structure of the parameter file

The parameter file is recognized by extension **.par**. Its format is close to the format provided by CHARMM. Its general structure is:

```
<options>
<table "atoms">
<table of site-site terms>
<table "bonds">
<table "angles">
<table "dihedrals">
<table "impropers">
<table "nbfixes">    [obsolete name: <table "NBFIX">]
<table "polaratoms"> [POLAR only]
<table "polarbonds"> [POLAR only]
<table "waters">
<table "backbone">
```

The exclamation mark **!** denotes a comment.

Each table begins with a header (single keyword on a line) and ends by a blank line; hence, blank lines are not allowed within tables (any lines without a header after a blank line are omitted). Detailed explanation is given below.



## 3.2 Force field generation options

Example of table <options> is:

```
all_dihedrals=0  ! one dihedral per central bond
ar_dih_limit=1   ! inclusion limit for dihedrals in aromatic rings
all_impropers=0  ! one improper per central atom
column_X=0       ! column X disabled (all atoms match = all X=1)
sqrt_rule=0      ! no sqrt combining rule for LJ energies
LJsigma=0        ! vdW radii used
factor14=0.5     ! multiplicative factor for 1-4 interactions
distance14=4     ! 1-4 interactions are really 1-4
ar_14_limit=0    ! 1-4 also for aromatic rings (use 3 for GROMOS)
polar=0          ! polarizability not included
;               ! end of data set
```

The assignments may be in any order and must end by ; (see Sect. 2.4.1).

Description of options:

**all\_dihedrals** `all_dihedrals=1` All dihedrals are generated for each central bond. Thus, ethane has 9 dihedrals. Used in newer versions of CHARMM. (This is the default if `all_dihedrals` is missing).

`all_dihedrals=0` Only one dihedral is generated for each central bond. Thus, ethane has one dihedral. Required by older versions of CHARMM. The rules which dihedral from generally several possibilities to select are mainly aesthetic. It is preferred that the four atoms of the dihedral be part of a longer chain.

`all_dihedrals=-1` No dihedrals nor aromatics are generated.

`all_dihedrals=2,3` As `all_dihedrals=0,1` and dihedrals with multiple periodicities are shifted (i.e. absolute term `K[0]` set) to give `Emin=0`

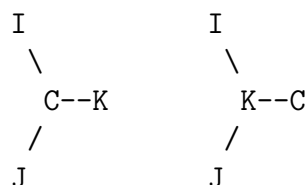
**ar\_dih\_limit** Maximum aromaticity of dihedrals. The default is `ar_dih_limit=-1` which turns off special treatment of dihedrals in aromatic rings. See Sect. 3.8.6, for detailed explanation.

**all\_impropers** `all_impropers=-1` No impropers are generated.

`all_impropers&1 = 1` All three impropers are generated for each central atom.

`all_impropers&1 = 0` Only one improper is generated for each central atom (default). The rules which improper from three (or six if `all_impropers&2`) possibilities to select are mainly aesthetic—symmetry and bond length criteria are applied. See the comments in the `blendgen.c` for details.

`all_impropers&2 = 2` Swapping of central and ending atom is allowed when generating the impropers. Thus, one line `C I J K` is enough in the table of impropers for both these cases:



This is used for CHARMM21.

**all\_impropers&2 = 0** Swapping of central and ending atom is not allowed. Thus, the second improper in the above figure will not match **C I J K**: a separate term for **K I J C** (or **K J C I** or **K C I J**) is required.

**column\_X** Enable/disable column X in the atom table in parset.par.

**column\_X=0** Any atom can match wildcard atom X in the tables of dihedrals and impropers (default).

**column\_X=1** Only the atoms marked by 1 in column X in the table of atoms can match X. Rarely used option.

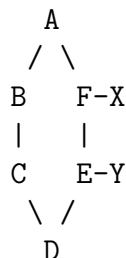
**factor14** 1-4 interactions are interactions for atoms connected by three bonds (1-2-3-4). All Coulomb interactions for 1-4 atoms are multiplied by **factor14**. Lennard-Jones interaction for which no explicit 1-4 terms or **nbfixes** are available are also multiplied by **factor14**. The default is **factor14=0.5**.

**distance14** Normal value for this option is 4. If, however, **distance14=3** or **distance14=5** then everything written in this manual about 1-4 interaction (incl. **factor14**) applies for 1-3 or 1-5 interactions, respectively.

**ar\_14\_limit** Controls 1-4 interactions in aromatic rings (see Sect. 3.8.6). Implemented with **distance14=4** only.

**ar\_14\_limit=4** The 1-4 interactions of atoms that are all in an aromatic ring are excluded (their nonbonded interactions are zero).

**ar\_14\_limit=3** As above and also atoms attached to the aromatic ring do not interact with the aromatic atoms (X does not interact with B but X and Y do interact).



**ar\_14\_limit=2** As above and also atoms attached to the aromatic ring do not interact themselves (X and Y do not interact). This is used for GROMOS96.

**ar\_14\_limit=0** No special treatment of aromatic atoms. A and D, X and B, and X and Y in the above molecule interact by 1-4 interactions. This is the default.

**LJsigma** If **LJsigma=1** then the **RvdW** column in the Lennard-Jones table is interpreted as Lennard-Jones sigma and the **sigvdW** column in **NBfix** table also as Lennard-Jones sigma. Affects reading the **.par** file only.

**polar** **polar=1** denotes a polarizable force field; **POLAR** version of **blend** must be used. **polar=0** is the nonpolarizable default; do not confuse with CHARMM old ‘polar’ force field, it is NOT polarizable!

**polar=2** no longer active (see table “shellrep” as a more flexible replacement)

**polar=4** (in addition to 1, i.e., **polar=5**) (1+4) requests ADIM (induced dipole–induced dipole terms within a molecule)

### 3.3 Table of atoms

The first table in the parameter file lists all atom types. Example:

```
atoms
!i name  X A    mass  Z  col description
  1 H     0 0    1.00800 1   H  Hydrogen bonding hydrogen (neutral group)
  2 HC    0 0    1.00800 1   H  Hydrogen bonding hydrogen (charged group)
...
 10 CT    1 0   12.01100 6   C  Aliphatic carbon (tetrahedral)
 11 CH1E  1 0   13.01900 7   C  Extended atom carbon with one hydrogen
...
```

Description of columns follows:

**i** Number of atom, extended atom or interaction site. The atom numbers must be positive. They need not be ordered but must not duplicate.

**name** Symbolic name of atom, extended (united) atom or interaction site. It must be composed of max. 4 UPPERCASE letters and digits (1st char must be letter). (Maybe the length limit is 6, I am not sure now :-())

**X** 1 if the atom will match wildcard **X** in the tables of dihedral and improper torsions, otherwise 0 (as for hydrogens). Active only if **column\_X=1**.

**A** 1 if the atom is used in aromatic rings.

**mass** Atomic mass in g/mol.

**col** (version 1.7 or higher) This is color type. Valid values are:

```
C    Carbon (cyan or gray)
H    Hydrogen (white or light gray)
O    Oxygen (red)
N    Nitrogen (blue)
```

- S** Sulphur (yellow)
- M** Metal (magenta)
- X** Halogen and other gases (green)
- Z** Atomic number. Used for determining the atomic charge centroid serving as the reference point in dipole and quadrupole calculations (cf. hot key **E**).
- description** Any text.

## 3.4 Non-bonded forces

Non-bonded forces consist of the site-site and Coulomb forces. Typical site-site forces are the Lennard-Jones forces, **blend** however supports any forces that are written in a separate module. (MACSIMUS/sim/MODULE/sitesite[ch]).

Parameters for the site-site potentials are calculated from the atom data in the parameter file using *combining rules*. The values of partial charges needed to calculate the Coulomb forces are not (and cannot be easily) given in the parameter file and must be specified with the molecule.

The following text assumes the Lennard-Jones potential as an example.

### 3.4.1 Selection of site-site and Coulomb energy terms

The following rules, ordered from high to low priority, apply for selection of the site-site (e.g., Lennard-Jones) interactions between two atoms:

1. All non-bonded forces between directly bonded atoms (1-2) and between second neighbors (1-3) are zero.
2. If the pair is separated by three bonds (1-4 interaction), energy terms are looked for in the following order:
  - (a) 1-4 value from table **nbfixes**
  - (b) General value from table **nbfixes** with energy multiplied by **factor14**
  - (c) Atom values are taken from 1-4 columns of table **Lennard-Jones** and combined according to combining rules. If no 1-4 value is available in table **Lennard-Jones**, it is derived from the general value by multiplying the energy **E<sub>min</sub>** by **factor14**.
3. Other interactions than 1-2, 1-3, 1-4 (both intra- and intermolecular): If the pair of atoms is listed in table **nbfixes**, these values for the Lennard-Jones parameters are used, otherwise table **Lennard-Jones** and the combining rules are applied.

The following rules apply for calculating the Coulomb interactions between two atoms:

1. All Coulomb forces between directly bonded atoms (1-2) and between second neighbors (1-3) are zero.
2. Coulomb forces between atoms separated by three bonds (1-4 interaction) are multiplied by `factor14`.
3. In all other cases (intra- and intermolecular), normal Coulomb forces are used.

### 3.4.2 Combining rules for the Lennard-Jones parameters

The combining rules depend on the site-site module used and you should read the comments of the respective module.

For the most usual Lennard-Jones module (`MACSIMUS/sim/lj/sitesite[ch]`), they are controlled by variable `sqrt_rule`:

1. `sqrt_rule & 1`: 0=polarizability rule for Emin (see below), 1=geometric mean for Emin (TIPS style)
2. `sqrt_rule & 2`: 0=additive diameters (see below), 2=geometric mean for diameters (TIPS style)

The Lorenz-Berthelot combining rule (`sqrt_rule=1`) and geometric mean square rule (`sqrt_rule=3`) are typical for modern force fields. Older versions of CHARMM used the ‘polarizability rule’ (`sqrt_rule=0`) based on the Kirkwood-Slater formula. The Lennard-Jones interaction of two atoms  $i$  and  $j$  is then given by:

1. If the values of both polarizabilities `alpha` in table `Lennard-Jones` are nonzero and neither `sqrt_rule` nor option `-l` is specified:

$$u(r_{ij}) = \frac{A_{ij}}{r_{ij}^{12}} - \frac{B_{ij}}{r_{ij}^6}$$

where the combining rules for the coefficients are

$$A_{ij} = \frac{B_{ij}}{2}(R_{vdW,i} + R_{vdW,j})^6$$

$$B_{ij} = \frac{\alpha_i \alpha_j}{\gamma_i + \gamma_j}$$

$$\gamma_i = \frac{-\alpha_i^2}{256 E_{min,i} R_{vdW,i}^6}$$

2. If any of polarizabilities `alpha` is zero then `sqrt_rule=1` applies, i.e., geometric mean of energy and arithmetic mean of diameters (Lorentz-Berthelot rule).

$$B_{ij} = (-2E_{min,i})^{1/2}(-2E_{min,j})^{1/2}(R_{vdW,i} + R_{vdW,j})^6$$

and  $A_{ij}$  is as above.

NOTE: The original article [9] contains the effective number of outer shell electrons  $N$  instead of  $E_{min}$  and a different formula. The conversion (see also program `alphatoe.c`) is:

$$E_{min} = -\frac{K}{4}\alpha^{3/2}N^{1/2}(2R_{vdW})^{-6}$$

where

$$K = \frac{3}{2}(4\pi\epsilon_0 m_e)^{-1/2}e\hbar = 362.3376 \text{ kcal mol}^{-1}\text{\AA}^{3/2}$$

### 3.4.3 Table of site-site parameters

**Lennard-Jones** is the second table in the CHARMM and GROMOS parameter files. For other site-site interactions, **Lennard-Jones** is replaced by the corresponding name, e.g., **Busing**, **Busing-12** – see file `metamake` in the `blend` directory. `blend` checks the table name not to use a wrong force field. Example:

Lennard-Jones				
!	alpha	Emin	Rmin	These columns used for
!name	[A^3]	(kcal/mol)	(A)	1-4 interactions
H	0.044	-0.04983	0.800	
HC	0.044	-0.04983	0.800	
HA	0.100	-0.00447	1.468	
CT	0.980	-0.09027	1.800	
...				

Description of columns follows:

**name** Symbolic name of atom, extended (united) atom or interaction site.

**alpha** Polarizability  $\alpha$  in  $\text{\AA}^3$  to be used by some combining rules

**Emin** Minimum energy  $E_{min}$  of the Lennard-Jones (LJ) atom-atom potential in kcal/mol

**RvdW** Van der Waals radius  $R_{vdW}$  (half of the atom-atom distance of the LJ potential minimum), in  $\text{\AA}$ . Note that if `LJsigma` is set then this column contains the Lennard-Jones sigma instead!

**1-4 interactions** If another three columns are given, they contain **alpha Emin Rmin** for 1-4 interactions

For other site-site interactions, the **Emin** and **RvdW** columns may have different meaning and more columns may be added; all columns but **atom** are repeated for 1-4 interactions.

The **alpha** column is used for combining rules and does not mean the polarizability in the POLAR version. Often it is irrelevant.

### 3.4.4 Non-bonded fixes

Site-site forces between certain types of atoms cannot be accurately calculated using the combining rules and must be listed separately in table **nbfixes** (old name **NBFIX**). If the 1–4 columns are missing, **factor14** multiple of the normal value is taken for the 1–4 interactions. If **sigvdW=0** for the normal interaction (not 1–4), the interaction is calculated from site-site (e.g, Lennard-Jones) tables using the combining rule; thus, only 1–4 terms can be given separately in the **nbfixes** table while other terms are regular. Example (from **charmm22.par**):

```
nbfixes
! OT   OT      -0.152073  3.5365 not needed
HT   HT      -0.04598   0.4490
HT   OT      -0.08363   1.9927
```

where **sigvdW** is twice the van der Waals radius. Note that if **LJsigma** is set then this column contains the Lennard-Jones sigma instead!

### 3.4.5 Table of polar atom parameters

This applies for the POLAR version only.

```
polaratoms
!atom  alpha  shell  Esat      arep
!      A^3    e      kcal/mol  1/e
CL      2    -1000   0          0.22439024
I       4.5  -1000   0          0.37037037
```

**atom** Atom name

**alpha** Dipolar polarizability  $\alpha$  in Å<sup>3</sup>

**shell** Number of outer shell electrons in *e* (normally negative because the electron shell is negative). This value is copied to the output atoms table and used by the polar version of cook as the charge of auxiliary site mimicking polarizability by a dipole.

**Esat** Saturation energy of the induced dipole. **Esat=0** turns off the saturation (the same as infinity saturation energy)

**arep** Parameter for the shell-core model (repulsive antipolarization), in  $e^{-1}$ . Also denoted  $\kappa$ .

### 3.4.6 Table “shellrep” of repulsive counterparts

This applies for the POLAR&1 version only. The “repulsive antipolarization” or “shell-core” term, equation (26.2), is not symmetric and consists of a polarizable ion

(usually anion),  $i$ , and its “repulsive” counterpart (usually cation with none or small polarizability),  $j$ . MACSIMUS uses this term if:

1. the atoms are more than `distance14` bond apart (normally `distance14=4` and 1–4 interactions cannot have the shell-core term),
2. the anion  $i$  has nonzero parameter `arep` alias  $\kappa_i$ ,
3. the cation  $j$  is listed in table “shellrep” (one atom type per line).

Example:

```
shellrep
!atom
NA
LI
```

(In principle also anions (with  $\kappa > 0$ ) may appear in this table. In this case there may be two shell-core terms per a pair of atoms. This is not recommended.)

### 3.4.7 Table of axially polar bonds

This table describes the axial polarizability tensor of a pair of atoms connected by a bond. The tensor is located at the first atom of the pair. `alphazz` is the polarizability in the direction of the axis, `alpha = alphaxx = alphayy`.

```
polarbonds
! atom --> atom  alpha  alphazz Esat
!              A^3    A^3    kcal/mol
H              CL    1      2      300
```

### 3.4.8 Table of 1–3 axially polar groups

The same as `polarbonds` but the second atom is a second connected neighbor (1–3). The middle atom does not enter the formulas for induced dipoles and is used only in topological analysis to assign the force field terms.

```
polarangles
! atoms      alpha  alphazz Esat
S  C  N      1.778  5.670    0
N  C  S      0.867  0.801    0
```

### 3.4.9 Table defining water models

Table `waters` defines water model(s). The reason is that water molecule is recognized from molecule topology and appropriate information is exported so that `cook` is able to use an optimized code for water which speeds up the calculations.



```
waters
! must be in standard order recognized by cook
! name NS atom charge atom charge ...
TIP3P 3   HT .417      HT .417   OT -.834
```

`name` is the model name, `NS` number of sites. Atom types and partial charges follow.

More about water compatibility:

1. Command `aw` adds water sites in order H H O which is ‘standard’ and recognized by `cook` to be rigid TIP3P water model (it only is recognized if `blend -o SYSNAME` is called with `-n-1` option!)
2. To treat old files with water added by `aw` with order of sites O H H, option `-[3` should be used (in addition to `-n-1`). (Precisely, `-n-1 -[#` causes #-atom clusters to be reordered so that H go first)
3. Then, in the result of `blend -n-1`, water line ‘species xxx.1’ should be edited to ‘species TIP3P’ unless the appropriate table `water` is present.

Supported water models are collected in `water.par`

The molecules are (in standard versions) rigid; thus, option `-h` should be given to `blend` to prepare the force field ble-file, e.g.: `blend -o spc -h spc.che`

**SPC** Simple Point Charge by Berendsen et al., used by GROMOS. The che-file `spc.che` for `blend` is:

```
SPC water model
parameter_set=gromos

HW.41      HW.41
 \        /
  OWn.82
```

**TIP3P** Three-site model by Jorgensen. File `tip3p.che` is:

```
TIP3p water model
parameter_set=water
!parameter_set=charmm21 is with LJ on H

HT.417 HT.417
 \        /
  OWn.834
```

**TIP3P/CHARMM** The CHARMM version contains additional H-H and O-H Lennard-Jones terms. `cook` must be run with option `-x` to include these terms (see below).

TIP3p water model

```
HT.417 HT.417
 \    /
  OWn.834
```

WARNING: in CHARMM21, the TIP3p water oxygen is called OW, in CHARMM19 and CHARMM22 it is OT.

TIP4P Four site model by Jorgensen. File `tip4p.che` is:

```
TIP4P water model
parameter_set = water
```

```
      H4p.52000
     /
O4-M4n1.04
 \
      H4p.52000
```

ST2 Classical but now obsolete model by Stillinger et al. File `st2.che` is:

```
ST2 water
parameter_set=water

! use blend with -r option until energy=0

      OH2--LPn.2357
     / | \
H.2357 | LPn.2357
      H.2357
```

F3C Model by Levitt et al., to be used with vibrating bonds and angles (`cook -u999`). Therefore there is no rigid version.

```
F3C water model by Levitt etal
parameter_set=water
      OFn.82
     /    \
HF.41  HF.41
```

**blend** caveat: crystal water in PDB files is not (now) recognized. If **blend** has been called with option `-n-1`, it is possible to edit the corresponding **species** command in the **ble-file**.

**cook** notes: virial pressure for ST2 is not correct – pressure by virtual volume change should be used. **ERFCPLUS** (see `ewald.c`) must be at least 1.8 for ST2 !!!

In versions of `cook` newer than Sep 96 there are special optimized functions available for water-water interactions for standard rigid models. If option `-x` is given to `cook`, using of these functions is turned off and general slower functions are used (ST2 is not available with `-x`). The CHARMM version of TIP3P and any modifications with flexible angles and/or bonds are available with `-x` only.

### 3.4.10 Table defining the protein backbone types

This table contains three lines with atom types used respectively for N, C<sub>α</sub>, and carbonyl C. Example:

```
backbone
! types of atoms for the backbone
! peptide N types:
NH1 NH2 NH3 N
! Calpha types:
CT1 CT2 CP1
! carbonyl (>C=O) C types:
C CC CD
```

This information is used for:

- Writing a PDB file ‘from scratch’ (option `-w10`), e.g., when the source of molecule was a che-file.
- Creating  $\alpha$  helix conformation (option `-r6`).

## 3.5 Non-bonded potential cutoff

If option `-tnumber` is specified, the non-bonded interactions within `blend` are calculated with cutoff. This speeds up the calculations.

The cutoff is smooth between site-site distances  $C_1$  and  $C_2$  given by option `-t` (see Sect. 2.2.3). The switch function which multiplies the potential  $u(r)$  is:

$$s(r) = \begin{cases} 1, & \text{for } r < C_1 \\ A[(r^2 - C)^2 - B]^2, & \text{for } C_1 < r < C_2 \\ 0 & \text{for } r > C_2 \end{cases}$$

where  $C = (C_1^2 + C_2^2)/2$ ,  $B = (C_2^2 - C_1^2)/2$  and constants  $A$ ,  $D$  are given by the requirements that  $s(C_1) = 1$ ,  $s(C_2) = 0$ . Then,  $s'(C_1) = s'(C_2) = s''(C_1) = s''(C_2) = 0$  and the cut-off potential and the corresponding force are sufficiently smooth.

Note that this cutoff is used only within `blend` to increase performance for large molecules and is not exported to `cook`, which uses a different cutoff algorithm.

Recommended values:

- `-t0` (Default) No cutoff, recommended for molecules smaller than 500–1000 sites.

- t11** Minimum reasonable value for larger molecules. Typical cutoff is then 3 LJ sigma, minimum (for larger atoms as triply bonded carbon) is 2.5 LJ sigma.
- t6** This is sufficient for filling missing hydrogens (in this case, all other atoms are kept fixed and the most important forces are bond angles).

## 3.6 Bond potential

The bond potential is:

$$U(r) = K(r - r_0)^2$$

where  $r$  is the bond length,  $r_0$  the equilibrium bond length, and  $K$  the force constant.

This potential is used for minimization in this program. In simulations by **cook**, bond-length constraints may be used instead.

The format of the table of bonds is given by the example:

```
bonds
!atom atom K length
C      C      450  1.38
C      CH1E,CH2E,CH3E 405  1.52
```

Description of columns follows:

**atom** Atom type. Comma separated lists of atom types are allowed. Both atoms may be lists, however, repeating terms are not allowed (e.g., **C1,C2 C1,C2 100 1.5** is incorrect because it expands into **C1 C2 100 1.5** and **C2 C1 100 1.5** which are equivalent.)

**K** Force constant  $K$  in  $\text{kcal mol}^{-1}\text{\AA}^{-2}$ .

**length** Equilibrium bond length  $r_0$  in  $\text{\AA}$ .

An error condition is raised if **blend** cannot find a bond in the table.

## 3.7 Bond angle potential

The bond angle potential is

$$U(\theta) = K(\theta - \theta_0)^2 + K_{\text{Urey-Bradley}}(|r_1 - r_3| - s)^2$$

The algorithm to calculate the bond angle  $\theta$  uses the scalar product of both bonds to calculate  $\cos \theta$ . There is a substantial (mechanical) singularity for  $\theta$  close to 0 or  $\pi$ ; for the equilibrium bond angle  $\theta_0 = 0$  or  $\theta_0 = \pi$  it becomes a numerical singularity only and is fixed. In the Urey-Bradley term,  $s$  is the 1-3 equilibrium constant

The table of bond angles in the parameter file looks like:

angles						
!atom	atom	atom	K	angle	[Kub	s]
C	C	C	70	106.5		
C	C	CH2E,CH3E	65	126.5		
C	C	CR1E	70	122.5		
S	C	N	40	180	100	2.879

Description of columns follows:

**atom** Symbolic atom name.

**K** Force constant  $K$  in kcal/mol.

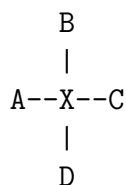
**angle** Equilibrium angle  $\theta_0$  in degrees.

**Kub** Force constant  $K$  in kcal/mol  $\text{\AA}^2$ . If not present, Kub=0.

**angle** Equilibrium 1-3 distance in  $\text{\AA}$ .

An error condition is raised if **blend** cannot find an angle in the table.

For planar groups (like Fe in the heme):



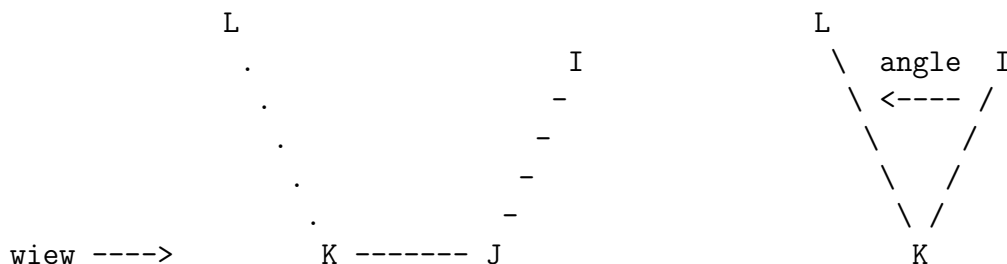
where bond angles are  $90^\circ$ , it is necessary to suppress generating angle terms for angles A-X-C and B-X-D. This is not a problem if (approximate) coordinates are known: an angle larger than  $150^\circ$  is removed from the list of angle terms and a message is printed. If the coordinates are not known, a warning message is printed; it may help to run **blend** again.

## 3.8 Torsions

Torsions are force field terms defined by four atoms. Different types of torsions (dihedrals, impropers, and also *aromatics*) differ by bonding of the atoms and by energy terms used but not by the definition of the torsion angle.

### 3.8.1 Torsion angle

The *torsion angle* of four atoms I J K L (written in this order but not necessarily connected by bonds in this order) is the oriented angle of plane IJK from plane JKL. The orientation is defined as follows: let us watch the four atoms from the KJ direction so that atom K is closer to the eye than atom J. Then the angle is the oriented angle of line segment LK from IK:



If the four atoms are connected by bonds as in I-J-K-L, the torsion angle is called *dihedral (torsion) angle* or *proper torsion angle*; if atom I is bonded to J, K, and L, the torsion angle is called *improper (torsion) angle*. Atom I is thus the central atom.

### 3.8.2 Torsion potential

The *torsion potential* for four atoms I J K L with the torsion angle  $\phi$ , equilibrium torsion angle  $\phi_0$ , force constant  $K$  and *periodicity*  $n$  is:

$$U(\phi) = \begin{cases} K(\phi - \phi_0)^2 & \text{for } n = 0 \\ K[1 + \cos(n\phi - \phi_0)] & \text{for } n > 0 \end{cases}$$

The torsion potential with  $n > 0$  is sometimes called *dihedral (torsion) potential*. It is used to describe hindered rotation as e.g. of the methyl groups in ethane.

EXCEPTION:  $n = 5$  denotes a special cis- and trans-dihedrals (see Sect. 3.8.3).

CAVEATS: Only  $\phi_0 = 0^\circ$  and  $180^\circ$  occurs. The case  $\phi_0 = 180^\circ$  is replaced by the following formula

$$U(\phi) = |-K| + (-K) \cos(n\phi)$$

which is equivalent as soon as  $K > 0$ . If, however,  $K < 0$ , which is the case of some CHARMM22 terms, this formula for  $U(\phi)$  gives shifted values of energy.

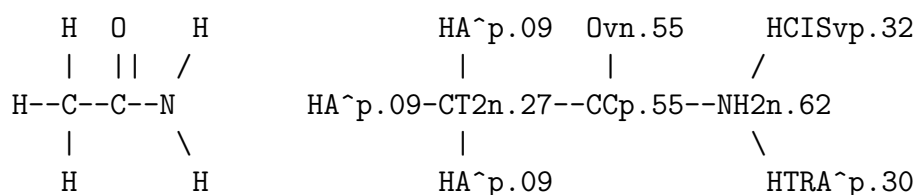
The torsion potential with  $n = 0$  is sometimes called *improper (torsion) potential*. It is used for three purposes:

1. With non-zero improper torsion angle it is used to define chirality of sp<sup>3</sup> chiral atoms in the extended atom representation (e.g., CH1E in aminoacids).
2. With zero improper torsion angle it is used to keep the sp<sup>2</sup> hybridized bonds planar (e.g., carbon in ketones).
3. In some force fields the improper torsion with zero dihedral angle is used misleadingly to keep 5- and 6- aromatic rings planar. The angle is in fact the dihedral angle of four atoms around the ring; for instance in benzene there are six such *aromatic dihedrals*. The problem is that the energy terms occur in the table of impropers and special hooks must be used to find them. See Sect. 3.8.6.

Note that the output from **blend** uses slightly different formulas for the torsion potentials (see Sect. 5.8).

### 3.8.3 Cis and trans-dihedrals

Sometimes it is necessary to guarantee the cis and trans conformation around a bond. For instance, CHARMM22 distinguishes both amide hydrogens in amides:



This is a problem of the initial configuration only—filling unknown hydrogens (heavy atoms are known). To do it correctly, both hydrogen types are distinguished and special cis dihedral terms are added:

$$U_{cis}(\phi) = K \cos^2 \phi \quad \text{for } K \cos \phi < 0$$

$$U_{cis}(\phi) = 0K \quad \text{for } K \cos \phi > 0$$

$K < 0$  (or  $K > 0$  and  $\phi_0 = 180$ ) denotes the trans dihedral.

The cis-dihedral energy term is 0 if the atoms are in the correct cis position and large if cis is encountered.

These dihedrals are not exported (in a ble-file) to `cook` because they are not needed once the bond is in the correct conformation.

### 3.8.4 Implementation of the torsion potential

There are two different algorithms to implement the torsion potential. For  $\phi_0 = 0$  or  $\phi_0 = \pi$  the potential is even in  $\phi$  and it is sufficient to calculate  $\cos \phi$  (which requires evaluation of certain scalar product); both  $(\phi - \phi_0)^2$  and  $\cos(n\phi - \phi_0)$  (the former is implemented for the most usual case of  $\phi_0 = 0$  only) is then expressed via  $\cos \phi$ .

The improper torsion used for keeping chirality has  $n = 0$  and arbitrary  $\phi_0$  and in this case the sign of  $\phi$  is important; another (and more complicated) algorithm based on  $\sin \phi$  has to be used in this case.

### 3.8.5 Chirality

The chirality of an extended atom I (usually of type CH1E) with three bonds to atoms J, K, L (listed in this order) is defined as follows: if the triple of vectors (IJ, IK, IL) has the same arrangement as the standard right-hand orthogonal coordinate system x-y-z, the chirality is +1. In other words, let us watch the molecule from that side of plane JKL so that we see J-K-L arranged anti-clock-wise (in the positive direction). If then atom I lies below plane JKL, the chirality is +1.

The sign of the equilibrium improper torsion angle of four atoms I J K L is the same as chirality of atom I (see Sect. 3.8.1).

The chirality of a tetrahedral atom I (usually of type CT) with four bonds to atoms J, K, L, M (listed with this order in column `bound_atoms`) is defined in the same way. It is naturally assumed that I and M are not separated by plane JKL. Not implemented yet.

Normally, `blend` finds the requested chiralities in `.mol` file or calculates them from z-coordinates in `.che` file. If option `-c` is given, `blend` calculates chiralities from the 3D configuration. File `species.mol` is then rewritten (the old one is named `species.mox`).

Chirality calculations and checking are performed in two stages: first, 3D configurations around all three-bonded atoms are considered; second, tables of impropers are searched to find whether the atom is really chiral. It may happen in the first stage that an atom is considered chiral just because the planar configuration was distorted and this is fixed in the second stage.

Since the chirality depends on atom numbering, it is not suitable to human hackers. Thus, we define *alphabetical chirality* as the chirality with respect to alphabetical order of the three neighbors of the chiral atom, and *mass chirality* as the chirality with respect to ordering by atomic masses. All three chiralities are printed to output. The alphabetical chirality of chiral alpha CH1E atom in the main chain of naturally occurring aminoacids (but glycine) is +1 if the atoms are named as in CHARMM and united atom representation is adopted for aliphatic hydrogens. Our simple algorithm does not solve cases when some of the three neighbors of the chiral atom are the same (the atom may still be chiral); in these cases, ? is printed.

### 3.8.6 Dihedrals in aromatic rings

This section applies only to force fields using impropers (=terms listed in the table of impropers) also for aromatic dihedrals. Normally `ar_dih_limit=-1` and then all energy terms listed in table `dihedrals` apply to atom chains I-J-K-L (including case `n=0` for aromatic dihedrals) while all energy terms listed in table `impropers` are for atoms J,K,L bonded to I.

Let us define *aromaticity* of the dihedral angle I-J-K-L as the number of atoms that lie in the same aromatic ring, provided that bond J-K lies in the same aromatic ring too. For instance, benzene (full-atom model) has 6 dihedrals with aromaticity=4, 12 dihedrals with aromaticity=3, and 6 dihedrals with aromaticity=2.

Specifying `ar_dih_limit>0` has two effects:

1. Aromatic dihedrals are looked for in the table of impropers.
2. Variable `ar_dih_limit` controls the way in which dihedrals are assigned to bonds that are parts of aromatic rings. `ar_dih_limit` is the maximum aromaticity allowed for dihedrals to be included. `ar_dih_limit` must be in the range 1–4. The default is 1 (no dihedrals with central bond in aromatic ring) because normally the aromatic dihedrals should be enough to keep aromatic rings planar. `ar_dih_limit=4` means that aromaticity is irrelevant when counting dihedrals—all are included. `ar_dih_limit=3` means that if all atoms I, J, K, L lie in the same aromatic ring, they are not included in dihedrals (they have already been



included in aromatic dihedrals). If, however, at least one of atoms I and L points out of the aromatic ring, the dihedral is included.

A 5-ring is called aromatic if it contains at least 3 aromatic atoms. A 6-ring is called aromatic if it contains at least 4 aromatic atoms. This cumbersome definition is necessary in older versions of CHARMM because non-aromatic (general) atom types are used sometimes in aromatic rings. In new versions of CHARMM, all atoms in aromatic rings are of aromatic type so that a simpler criterion could be used.

### 3.8.7 Tables of dihedrals and impropers

Example (from charmm.par):

```
dihedrals
!atom atom  atom  atom    K    n angle
CH1E  C      N      CH1E   10    2  180
CH2E  C      N      CH1E   10    2  180

... lines deleted ...

X      S      S      X      4      2    0

impropers
! Improper torsions. See dihedrals above.
!atom atom  atom  atom    K    n angle
C      CR1E  C      CH2E   90    0    0
C      CR1E  CR1E  CH2E   90    0    0

... lines deleted ...

NR      X      X      CT      25    0    0

cisdihedrals
0      CC      NH2  HCIS      1000      5  0.00  !  cis
0      CC      NH2  HTRA      1000      5 180.00 !  trans
```

Description of columns follows:

**atom** Symbolic atom name

**K** Force constant  $K$  in kcal/mol.

**n** Periodicity number. Note that  $n = 5$  is exceptional—see above.

**angle** Equilibrium angle  $\phi_0$  in degrees.

### 3.8.8 Atom matching rules for finding the energy terms

If `column_X=1` has been specified, wildcard atom `X` in the tables of dihedrals and impropers matches any atom but hydrogen (to be more precise, it matches all atoms with 1 in column `X` in the atom table); if `column_X=0` has been specified, `X` can match any atom.

If there are several matches possible, the match with less wildcards is selected; if this selection is not unique, an error message is printed.

If the dihedral or improper torsion is not found in the table, it is assumed that it is zero; if `-v` has been specified, a message is printed.

Example: In the improper torsion table [11], there are the following two entries:

!atom	atom	atom	atom	K	n	angle
C	OC	OC	CH2E	100.0	0	0.0
C	X	X	CH2E	60.0	0	0.0

Then, parameters for `C-OC-OC-CH2E` will be taken from the first line, while the parameters for `C-C-OC-CH2E` from the second line.

If there are several periodicities in the table of dihedrals, they are summed; note that the entries may have different numbers of wildcards. It is not allowed to combine the improper potential ( $n = 0$ ) with the dihedral potential ( $n > 0$ ).

Example:

!atom	atom	atom	atom	K	n	angle
CUA1	CUA1	CUA3	CUA3	0.43	1	0.0
X	CUA1	CUA3	X	0.76	2	180.0

In this case, both terms are summed for `CUA1-CUA1-CUA3-CUA3`.

# Chapter 4

## Description of molecules

The structure of molecules is described by listing all atoms and bonds. In addition, information to determine conformation of chiral atoms and partial charges are given.

**blend** supports two formats: The molecular format is intended for interaction with other programs. The chemical format allows simple molecules to be easily build up using a text editor only.

### 4.1 Molecular file (mol-file) format

This file with extension `.mol` describes the topology of a molecule and contains also partial charges. Program **blend** checks consistency of bonds, removes possible multiple bonds, and calculates the number of clusters. Thus, you can treat e.g. fullerene with an atom inside the cage or a water dimer as one “molecule” consisting of more clusters.

```
alanine - aliphatic hydrogens in united atom representation
```

```
! total charge = 0.00
```

```
parameter_set = charmm21
```

```
zero_energy = -1
```

```
number_of_atoms = 9
```

```
atoms
```

```
! i a-id  a-type  charge  chir  nbonds  bound_atoms
  0 H1     HC     0.3500   0     1     2
  1 H2     HC     0.3500   0     1     2
  2 N      NT    -0.3000   0     4     1 3 0 4
  3 H3     HC     0.3500   0     1     2
  4 CA     CH1E   0.2500  -1     3     5 2 6
  5 CB     CH3E   0.0000   0     1     4
  6 C      C      0.1400   0     3     7 4 8
  7 O1     OC    -0.5700   0     1     6
  8 O2     OC    -0.5700   0     1     6
```

The first non-comment (i.e., not beginning with `!`) line is the header.

Optional `parameter_set` defines the name of the parameter file without extension. Extension `.bin` is added, and if the binary file does not exist, extension `.par` is tried. The default is `charmm21`.

Optional `zero_energy` is the energy (in kcal/mol), or energy offset, which is to be added to the intramolecular energy. It is typically a result of ‘chemical reactions’ – see group replacement as defined in `.rea`. Missing `zero_energy` means zero.

`number_of_atoms` refers to the number of atoms in the following table.

`atoms` introduces the table of atoms. The table columns are:

`i` Order number.

`a-id` Names that the user can give to the atoms. They have no special meaning for program `blend` but that they can label atoms in a picture of molecule.

`a-type` Atom (extended-atom, interaction site) types.

`charge` (Partial) charge of atom in e.

`chir` Atom chirality. See Sect. 3.8.5.

`nbonds` Number of chemical bonds.

`bound_atoms` Indices of atoms bound by a chemical bond.

If there is an asterisk `*` in the 1st column of the `atoms` table, the corresponding atom is *marked* and can be kept fixed during minimization. See option `-k` for details. Example:

!	i	a-id	a-type	charge	chir	nbonds	bound_atoms
*	4	CA	CH1E	0.2500	-1	3	5 2 6
*	5	CB	CH3E	0.0000	0	1	4
*	6	C	C	0.1400	0	3	7 4 8

## 4.2 Chemical file format

The chemical file format is again best seen from an example. The extension of the file must be `.che`.

```

dipeptide
parameter_set=charmm21
  HC^.35
    |
  HCv.35-NTn.3:nter-HCv.35
    |
  CH1Ev.25---CH3E                                     ! ALA
    |
  C.55-0vn.55      H.3
    |              |
  NPn.35-----+Hv.25
    |              |
    |              N5Rn.4==C5RE.3
    |              /      |
  CH1Ev.1-CH2E-C5R.1      |      ! HIS
    |              \      |
  C.14:cter          C5RE.1==N5Rn.4
 /      \
OC^n.57  OCvn.57

```

The first two lines have the same meaning as for the mol-file. They must not be separated by any blank line.

The symbolic atom names must be in uppercase and may contain digits. Numbers mean partial charges in electrons, negative charge is marked by **n** or **m** (because - denotes a bond), positive optionally (to distinguish the charge from a digit as a part of identifier) by **p** (e.g., **MNAp1**). The circumflex  $\wedge$  means that the atom goes 1 Å up the plane, **v** that it goes down; you can use several **v** or  $\wedge$ . The field after a colon means optional identifier (column **a-id** in the mol-file); if not present, the identifier is derived from atom number and type (e.g., **O-HC**).

The bonds cannot bend, they may, however, cross; character **+** should be used for the crossing (**x** is no longer allowed since V 2.1h: use **+** for crossing of bonds of any direction). Additional bonds can be added by statement **#connect**. Example:

```

...
---S.2:cys
#connect 33-S 44-S 125-S cys

```

There may be any number of **#connect** statements, but the total number of id's must be even – the id's are interpreted in pairs. No wildcards are allowed.

Unless changed by option **-r**, the initial configuration is derived from the layout of atoms on the paper and the z-coordinates given by  $\wedge$  and **v** used; this is not a good initial configuration, it is, however, sufficient to determine chirality for each chiral atom. (There are two chiral **CH1E**'s in the above example; try to check that the absolute chiralities of alpha-carbons in the above dipeptide are the same as in alanine given is above — see Sect. 4.1). Energy minimization must follow. Normally one line separation is 1 Å and the character pitch is 0.4 Å. This can be rescaled by option **-enumber** (in %). The z-coordinates are not rescaled.

Statement `#include file` can be used in che-files: `file` should contain (a part of) molecule without header, just as included physically to the place of `#include` statement. The BUG: `#include` statements cannot be nested.

An asterisk `*` in front of atom type is exported to a mol-file and means that the atom will be kept in place while minimization. Examples:

```
blendbus -g5 -r2 -e200 AlCl3SCN.che
parameter_set=busing
*ALp3
```

```
CLn1:Cl1          CLn1:Cl2
      CLn1:Cl3
```

```
LIp1
```

```
#include "SCN.inc"
```

To create files in the chemical format, it is recommended to use a text editor that supports rectangular blocks. As complicated molecules as adamantane  $C_{10}H_{16}$  or fullerene  $C_{60}$  with an atom in the cavity have been easily prepared using e.g. QEDIT.

If the molecule is given by this chemical format, the corresponding species.mol file is generated (the old one is backed up as species.mox).

# Chapter 5

## Output format (ble-file)

The output file contains full description of the mixture of molecules **blend**-ed together. It serves as input for MD programs **cook** and **cooks**. In addition, it contains a lot of comments (marked by !) for human hackers. Its general structure is:

```
<global data>
<table on site-types>
<table 'nbfixes' of non-bonded fixes>

<header of molecule 1>
<data of molecule 1>
<table 'sites' of molecule 1>
<table 'bonds' of molecule 1>
<table 'angles' of molecule 1>
<table 'dihedrals' of molecule 1>
<table 'impropers' of molecule 1>
<table 'aromatics' of molecule 1>

<header of molecule 2>
<data of molecule 2>
etc.
```

The user is encouraged to edit this file, like adding partial charges, deleting unnecessary potential terms, etc.

### 5.1 Global parameters

Global data are in *get data* format (see Sect. [2.4.1](#)). Example:

```
nspec=2  nsites=8  nmbfixes=3  factor14=0.500000 ;
```

where

**nspec** Total number of molecules blended together. (The name ‘nspec’ originates from ‘Number of (molecular) SPECies’. Though here ‘number of molecules’ just means ‘number of species’, in molecular dynamics there are many *molecules* of the same *species*.)

**nsites** Total number of types of interaction sites contained in all molecules.

**nmbfixes** Number of non-bonded fixes.

**nparms** Number of additional parameters (over two basic denoting atom size and energy) for the site-site interactions. For Lennard-Jones **nparms=0**.

**factor14 distance14 polar sqrt\_rule** See Sect. 3.2.

**polar** Polarizability version, see also file `blend/metamake`. Flags can be summed in the polarizable versions.

**polar=0** Standard non-polarizable version

**polar&1** Basic dipolar polarizability support (incl. axial and saturated)

**polar&2** : No longer used, functionality can be mimicked by table **shellrep**

**polar&4** : Also intramolecular 1-2 and 1-3 induced dipole–induced dipole terms are included. Note that then the molecular polarizability is not a sum of atomic polarizabilities!

## 5.2 Site types

This table lists all site types in all molecules processed. It’s name depends on the site-site module linked; for proteins, **Lennard-Jones** is most usual. The column headers are similar as in the parameter file. The number of columns depends on **nparms**. Example for a typical protein force field is (**nparms=0**, **polar=0**):

Lennard-Jones

!i	atom	mass	alpha	Emin	RvdW	alpha[1-4]	Emin[1-4]	RvdW[1-4]
2	HC	1.0080	0.044	-0.0498	0.6000	0.0440	-0.02490	0.6000
4	HT	1.0080	0.044	-0.0498	0.9200	0.0440	-0.02490	0.9200
11	CH1E	13.0190	1.350	-0.0486	2.3650	1.3500	-0.10000	1.8100
13	CH3E	15.0350	2.170	-0.1811	2.1650	2.1700	-0.10000	1.7600
14	C	12.0110	1.650	-0.1410	1.8700	1.6500	-0.07050	1.8700
36	NT	14.0067	1.100	-0.0900	1.8300	1.1000	-0.10000	1.6300
43	OC	15.9994	2.140	-0.1591	1.5600	2.1400	-0.07955	1.5600
46	OW	15.9994	0.840	-0.1521	1.7680	0.8400	-0.07605	1.7680

Site-site force fields with **nparms>0** have the columns **atom mass alpha Emin RvdW parm...parm alpha[1-4] Emin[1-4] RvdW[1-4] parm[1-4]...parm[14]**. For PO-LAR force fields (**polar>0**), four columns are appended:



**alphap01** Polarizability in Å<sup>3</sup> to be used by polar version of **cook**. Do not confuse with **alpha** which serves for combining rule only.

**shell** Auxiliary charge ('number of outer shell electrons') for replacing a point dipole by a finite-size dipole (in **cook**). See variable **shell** in the parameter file and option -@).

**Esat** Saturation energy, in kcal/mol. For the model of 'saturated polarizability' only (see Sect. 26).

**arep** Parameter for the shell-core model (repulsive antipolarization), in  $e^{-1}$ . Also denoted  $\kappa$ .

## 5.3 Non-bonded fixes

This table has exactly the same format as table **nbfixes** in the parameter file. As it contains only non-bonded fixes applying to atoms in the blended molecules, it is usually shorter or even empty. See Sect. 3.4.4.

## 5.4 Header of molecule

The header consists of keyword **species** followed by a name that is derived from the name of the molecular file (**species.mol** or **species.che**). At the same time, a summary of energy terms and a minimization protocol is printed as comments. Example:

```
!!!!!!!!!!!!!!
species valine
!!!!!!!!!!!!!!
! valine

! mass=117.1485 g/mol
! potential energy summary (in kcal/mol)
!
! Lennard-Jones:   -0.085      bonds: 9197.366   dihedrals:      2.818
!      Coulomb:  -33.921      angles: 198.001   impropers:     21.795
! sum nonbonded:  -34.006  sum bonded: 9419.979   aromatics:      0.000
! total U: 9385.973
! 27 steps of steepest descent:
!   5 :  U=9095.6037      h=5.89786e-06

... lines deleted ...

!   95 :  U=-100.85253      h=0.000915236
!  100 :  U=-100.86482      h=0.000864632
!           U0=-100.865
! potential energy summary (in kcal/mol)
```

```

!                                     dihedrals:      1.245
! Lennard-Jones:      4.013      bonds:      0.358      impropers:      0.152
!      Coulomb: -113.163      angles:      6.530      aromatics:      0.000
! sum nonbonded: -109.150 sum bonded:      8.285      total U: -100.865

```

Here, U is the potential energy in kcal/mol and h the step size used in minimization.

## 5.5 One species (molecule) data

The molecular data are in *get data* format (see Sect. 2.4.1). Example:

```

i=0  N=5
ns=11  nc=10  nangles=15  ndihedrals=14  nimpropers=3  naromatics=0
ndependants=0;

! 0 dihedrals, 0 impropers, and 0 aromatics zero or not found
! 25 pairs excluded (1-2 and 1-3)  14 interactions 1-4  0 non-bond fixes
! total charge = 0.00

```

Where

i Species order number

N Number of molecules of the species. The value of option **-n** is thus passed to MD programs; it has no meaning to **blend** because **blend** is able to process only one molecule of each species.

**config config=1** denotes that table **config** will be present containing a configuration of all species. This is generated for **blend -n-1**, i.e., when one input molecule is split into clusters (molecules) and these are treated separately by **cook**. The configuration initializer of **cook** then reads this table and ignores the coordinates of table **sites**.

ns Number of interaction sites (atoms) in the molecule

nc Number of constraints. Unless **-h** has been specified, this is the number of bonds.

nangles Number of bond angles (those that are not constrained, see option **-h**).

ndihedrals Number of dihedral torsions; note that terms with different positive periodicities for the same dihedral angle are merged.

nimpropers Number of improper torsions (does not include aromatics)

naromatics Number of ‘aromatic dihedrals’ used to keep aromatic rings planar

ndependants Number of ‘dependants’. See Sect. 5.10.

naxials POLAR only: number of atom pairs defining the axial polarizability tensor

Details on some tables follow

## 5.6 Table of sites

The table of sites consists of keyword **sites** followed by **ns** lines. An example follows:

```
sites
!i atom    charge      x        y        z    #  excluded *1-4  chir:nam
  0 HC      0.3500    -0.5158    2.4968    0.7781  0
  1 HC      0.3500    -1.7505    2.0381   -0.2218  1  0
  2 NT     -0.3000    -0.7538    2.3600   -0.2184  2  0 1
  3 HC      0.3500    -0.5890    3.1709   -0.8336  3  0 1 2
  4 CH3E    0.0000     2.1067    2.2152   -0.1489  1  *2
  5 CH1E    0.2500    -0.1139    1.1323   -0.6603  5  0 1 2 3 4 +++
  6 CH1E    0.0000     1.2201    0.9911     0.0936  6  *0 *1 2 *3 4 5 +??
  7 CH3E    0.0000     1.9662   -0.2889   -0.2953  4  *2 4 5 6
  8 C       0.1400    -1.1452    0.1554   -0.1754  8  *0 *1 2 *3 *4 5 6 *7
  9 OC     -0.5700    -2.1028    0.6459     0.4269  4  *2 5 *6 8
 10 OC     -0.5700    -1.0187   -1.0326   -0.4039  5  *2 5 *6 8 9
```

The columns are:

**i** Order number. It is used in other tables to refer to the site.

**atom** Atom type

**charge** Partial charge in electrons

**x y z** Coordinates of the site

**#** Number of exceptional site-site terms

**excluded \*1-4** List of exceptional site-site terms. Those marked by **\*** are 1-4 interactions, those without **\*** are exclusions (1-2 and 1-3 pairs that do not interact by bonded forces). Only sites with indexes lower than **i** are listed.

**chir:nam** This field appears for chiral atoms only. The first character is the chirality with respect to atom numbering (= the chirality that appears in **.mol** file), the second character is the alphabetical chirality, the third character is the mass chirality. This field is not used by MD programs. See Sect. 3.8.5, for details.

## 5.7 Tables of bonds and bond angles

Tables of bonds and angles have a similar format. Shortened examples follow:

```
bonds
!i atom    i atom  K[kcal/mol/A^2]  r[A]    calc.    Upot
 10 OC      8 C      455.00    1.2300    1.2187    0.058
   9 OC      8 C      455.00    1.2300    1.2349    0.011
```

... lines deleted ...

#### angles

!i atom	i atom	i atom	K[kcal/mol]	angle[deg]	calc.	Upot	[Kub s]
5 CH1E	8 C	10 OC	85.00	118.5000	120.15	0.071	
9 OC	8 C	10 OC	85.00	123.0000	123.72	0.013	

... lines deleted ...

i atom Refers to the notation of the table of sites

K The force constant. See Sect. 3.

r Equilibrium bond length

angle Equilibrium bond angle

calc. Calculated values of bond length or angle

Upot The energy of bond or angle potential in kcal/mol

If the angle potential contains the Urey-Bradley term, the force constant Kub and equilibrium length s appear as extra two columns in table **angles**.

## 5.8 Tables of dihedrals, impropers and aromatics

Tables of dihedrals, impropers and aromatics have the same format. Shortened examples follow (note that the table of aromatics is empty in this example):

#### dihedrals

!i atom	i atom	i atom	i atom	nX	n	K K[0]	calc.angle	Upot	angle K[1]
10 OC	8 C	5 CH1E	2 NT	2	3	-0.100	171.74	0.191	
10 OC	8 C	5 CH1E	6 CH1E	2	3	-0.100	54.88	0.196	

... lines deleted ...

#### impropers

!i atom	i atom	i atom	i atom	nX	n	K K[0]	calc.angle	Upot	angle K[1]
8 C	9 OC	10 OC	5 CH1E	0	0	100.000	0.72	0.016	-0.0000
6 CH1E	7 CH3E	4 CH3E	5 CH1E	2	0	55.000	-32.57	0.122	-35.2644

... lines deleted ...

#### aromatics

!i atom	i atom	i atom	i atom	nX	n	K K[0]	calc.angle	Upot	angle K[1]
---------	--------	--------	--------	----	---	--------	------------	------	------------

Where:

**i atom** Refers to the notation of the table of sites

**nX** Number of matches of wildcard atoms **X** while looking for this torsion in the tables of dihedrals or impropers. For negative **n** (see below), the maximum number over all periodicities.

**n** The periodicity number. For nonzero **n** the potential form differs from that in the parameter file (see Sect. 3.8.2).

**n=0**

$$U(\phi) = K(\phi - \phi_0)^2$$

**n>0**

$$U(\phi) = |K| + K \cos(n\phi)$$

**n<0** Several dihedrals with different periodicities have been summed and the potential is

$$U(\phi) = \sum_{i=0}^{|n|} K_i \cos^i \phi$$

**K|K[0]** If  $n \geq 0$  then the force constant  $K$  in kcal/mol, if  $n < 0$  then the value of  $K_0$ .

**calc.angle** The calculated torsion angle

**Upot** The potential energy in kcal/mol

**angle|K[1]** **n=0** Equilibrium improper torsion angle

**n>0** The field is empty

**n<0** The values of  $K_1, K_2, \dots K_{|n|}$

## 5.9 Table of axial polarizability tensors

Applies to POLAR version only.

**axials**

!i atom-->i atom	arep	alpha	alphazz	Esat
3 S 4 C	0.26998	1.7781	5.6704	0.000
5 N 4 C	0.22019	0.8672	0.8005	0.000

**alphazz** is polarizability in the direction of the pair of atoms, **alpha** in both perpendicular directions. For other parameters see Sect. 5.2.

## 5.10 Table of dependants

A dependant is a massless site which is a linear combination of two, three or four other sites: a typical example is site M in the TIP4P water model. **blend** automatically generates dependants in the following circumstances (D=massless dependant, X=any atom which may be bonded to more atoms):

```
linear patterns:  X-D-X   X-X-D

planar patterns:  X-D-X   X-X-X   (as e.g. TIP4P water)
                  |       |
                  X       D

3D patterns:      X-D-X   X-X-X
                  / \     / \
                  X   X   D   X
```

**blend** itself does not use dependant for anything, these are only exported in the ble-file to be used by **cook**.

Caveats:

1. It is necessary to mark angles not containing the massless site as constrainable by option **-h** or **-h-1**.
2. Badly determined multiple dependants not checked. (?? - probably they are in newer versions)
3. For **blend**, it is generally necessary to add an improper or other terms to guarantee planarity/linearity

Field **err=** in the table of dependants (in ble-file) contains the deviation from linearity/planarity and should be small ( $< 1e-12$  for free simple molecules as e.g. TIP4P water,  $< 1e-2$  if the group is a part of more complex molecule). For 3D patterns, **err** should be always very small ( $< 1e-12$ ).

NOTES/BUGS:

dependants do not work well in certain ‘nonstandard’ cases like **Q-S-Q---C---Q-N-Q**, where Q is supposed to be a dependant

Any site dependent on another dependant is now removed (see **#define REMOVED** in **blenddep.c**)

In the case above, bonds S-C and C-N are missing and should be added by hand from a ble-file generated with the masses of dependants nonzero (may be small; it is necessary to use parallelly this ‘mass’ version of the force field also if **nmf** is calculated)

dependants

```
!i atom # i atom weight...
 2 05   4 0 H5 0.2041   4 E5 0.2959   3 E5 0.2959   1 H5 0.2041   err=5e-13
```

This table means that atom O5 (here of TIP5P water) is a linear combination of two H5 sites and two E5 sites, with weights given (their sum is 1).

**blend** uses a MC algorithm to determine the weights, the error is printed as **err=**.

It is also possible to specify dependants in a special file of extension **.dep**. This cannot be combined with automatic selection of massless dependent sites! Example of species.dep:

```
! this is comment
7 8 9 : 11 12 13 14 ! sites 7,8,9 depend on 11,12,13,14 (using site numbers)
H1 : CA1 CA2 CA3 ! site H1 depends on CA1,CA2,CA3 (using atom ID)
```

Another example – rigid molecule of benzene. The che-file **benzene.che** is

```
benzene
HA.1:b1    HA.1    HA.1:b2
|          |          |
C6Rn.1--C6Rn.1--C6Rn.1
|          |          |
C6Rn.1--C6Rn.1--C6Rn.1
|          |          |
HA.1    HA.1:b3    HA.1
```

and using file **benzene.dep**

```
* : b1 b2 b3
```

all sites but b1,b2,b3 are dependent on b1,b2,b3. **cook** programs then use constrained dynamics of triangle b1,b2,b3 and all forces acting on other atoms are propagated to b1,b2,b3. See also utility **makedep.c**.

### 5.10.1 Lone (out-of-plane) dependants

Cook now supports out-of-plane (“lone”, from lone electron pair) dependants, based on 3 parents. See Sect. 11.3.4 The extended format of table “dependants” is (not generated by **blend** automatically):

```
dependants
M 3 M6NE 3 2 O6NE 0.60071493 1 H6NE 0.19964254 5 H6NE 0.19964254 e
L 0 L6NE 3 2 O6NE 1.86320976 1 H6NE -.43160488 5 H6NE -.43160488 -1.73602206
0.86801103 0.86801103 0.00000000 0.63064693 -0.63064693 -0.73718021 0.00000000
0.63988055 0.63988055 0.00000000 0.46490043 -0.63988055 e
L 4 L6NE 3 2 O6NE 1.86320976 1 H6NE -.43160488 5 H6NE -.43160488 -1.73602206
0.86801103 0.86801103 0.00000000 0.63064693 -0.63064693 0.73718021 0.00000000
-0.63988055 -0.63988055 0.00000000 -0.46490043 0.63988055 e
```

The “M” lines are the old (“Middle” or linear) dependants

The “L” lines (“Lone”) have another 13 numbers appended after the list of three

parents and weights (See Sect. [11.3.4](#), for the notation):

$x_1, x_2, x_3$

$y_1, y_2, y_3$

$w_z$

$t_{x,1}, t_{x,2}, t_{x,3}$

$t_{y,1}, t_{y,2}, t_{y,3}$

Then, letter 'e' should follow to mark the end of the line.



# Chapter 6

## Examples

The source files of these (and other) examples are in directory `examples/`.

### 6.1 Example 1: Protein in water

We want to simulate protein like crambin in water. The structure and 3D coordinates of crambin are taken from the PDB database (file `macsimus/examples/crambin.pdb`). By running the PDB-converter to the charmm21 force field,

```
pdb -fcharmm21 crambin
```

we get the following two files:

`crambin.plb` 3D coordinates of atoms; coordinates of hydrogens are missing.

`crambin.mol` Molecular file (see Sect. 4.1) in `charmm21`. Chiralities of extended tetrahedral atoms are missing.

In addition, we must have a file describing water copied to our working directory: `blend/data/charmm21/hoh.che` (or `hoh.mol` which can be obtained by copying `blend/data/charmm21/hoh.rsd`). It will simplify later work if this file is renamed to `TIP3P.che`.

```
cp ~/macsimus/blend/data/charmm21/hoh.che TIP3P.che
```

We have to run `blend` twice. In the first step (optional `-g` selects graphics)

```
blend -c -t6 -g crambin
```

the following calculations are performed:

1. Files `crambin.mol` and `crambin.plb` are read.

2. Missing chiralities are calculated (this step is empty for full-atom force fields like charmm22). Note that (unknown) coordinates of hydrogens are not required to do this unless for tetrahedral NP groups, where a warning is issued; if ignored, the system chooses the conformation at random which is fully OK here because both H of NH<sub>2</sub> are equivalent.
3. Coordinates of missing hydrogens are filled by (not very good) values.
4. Energy is minimized by 25 steps of steepest descent and 100 steps of conjugate gradient method. This is the default (see option `-m`) which is fully sufficient for obtaining good hydrogen positions. During minimization, positions of atoms with known coordinates are kept fixed; this is the default action when an unknown atom is found (see option `-k` for details). Short value of cutoff `-t6`, i.e., C1=5, C2=7, is sufficient for filling missing hydrogens. Detailed behavior depends on the version (UNIX/X11/DOS) and whether graphics is selected (option `-g`)
5. If graphics is on (`-g`), stop showing by pressing . (period) in the graphics window.
6. Files `crambin.mol` and `crambin.plb` are rewritten by new ones with chiralities and hydrogen positions; the old files are backed-up.

In the second step,

```
blend -o crambinw -m30 -n crambin -h -n999 TIP3P
```

the following calculations are performed:

1. Files `crambin.mol` (with correct chiralities) and `crambin.plb` (with hydrogens) are read. Since all atom coordinates are now defined in `crambin.plb`, all atom coordinates are subject of minimization. 30 steps of the conjugate gradient method are performed, which is sufficient to fix inaccurate bond lengths and angles from the PDB file.
2. Unless `TIP3P.mol` and `TIP3P.plb` are already present, `TIP3P.che` is read and the water molecule is energetically minimized (by max 30 steps of the conjugate gradient method). Files `TIP3P.mol` and `TIP3P.plb` are created.
3. File `crambinw.ble` containing a description of the force field for both `crambin` and water is created. It can be used by program `cook`. Options `-n` (= `-n1`) and `-n999` define the numbers of molecules of `crambin` and water, respectively (these may be omitted and defined later in MD by `cook`). Option `-h` specifies that the TIP3P water molecule will have the H–O–H angle constrained, that is, it will be treated as hard body.

NOTE: this example is continued by MD simulation in the manual of `cook`, see Sect. 17.1.

## 6.2 Example 2: Cluster Na4Cl4

Let us study local energy minima of a cluster of 4 cations Na<sup>+</sup> and 4 anions Cl<sup>-</sup>. Note that **blend** can treat this cluster as one molecule. File **Na4Cl4.che** contains the following definition:

```
Na4Cl4
parameter_set=charmm21
MNAPp1 XCLMn1
      XCLM^n1 MNAP^p1
XCLMn1 MNAPp1
      MNAP^p1 XCLM^n1
```



First try:

```
blend -g Na4Cl4
```

This statement reads **Na4Cl4.che**, derives 3D configuration from it, and minimizes it energetically. You should get a cube.

Then, try:

```
blend -r -g Na4Cl4
```

Now, the starting configuration is random and you will get different local energy minima, like a strip 2x4, a ring (octagon), etc. Try to repeat the same calculation several times. If not converged enough, continue by pressing , (comma), end by .. Use mouse for rotating the configuration, right button or   for rescaling.

NOTE: The CHARMM-atoms MNA and XCL are neutral and it is not accurate to use them with charges. Therefore TIPS-based MNAP and XCLM were added.

# Chapter 7

## Problems

### 7.1 Bugs and caveats

1. Special hydrogen bonding potentials (as in old versions of CHARMM) are not supported.
2. Output tables are printed in reverse order than is natural. This is because of a simple implementation of linked lists.
3. No valence tests are performed on molecules. For instance, a missing hydrogen in “ethane” CH<sub>2</sub>-CH<sub>3</sub> passes unnoticed.
4. Chirality is explicitly supported only for CH1E etc. in extended atom representation. Bad configurations around four-bonded tetrahedral atoms pass unnoticed.
5. If nbfixes for 1-4 interactions are present and **factor14=1** then 1-4 exceptions are always present even if it may happen in some cases that they are identical to normal interactions and some efficiency is lost.

### 7.2 Trouble shooting

Here we examine several possible sources of problems.

**Problem:** `blend` (UNIX, SGI) gives strange error messages which do not resemble anything described in this manual.

**Probable cause:** You run a system command of the same name. `blend` always prints a line like this:

```
! ----- MACSIMUS / BLEND 2.0k ----- (c) J.Kolafa 1993-2004 -----
```

**Solution:** Use system command `which blend` to determine which `blend` are you using. If this is not the correct `blend`, change your path (remember that if you want to run `blend` from the current directory, you must have the current directory `.` as the first item in your path) or rename `blend`.

---

**Problem:** I get error message “parameter file not found - try set BLENDPATH”

**Solution:** Set BLENDPATH (see Sect. 2.1). If it does not help, try to specify the parameter file name (with full path), see Sect. 2.2.2.

---

**Problem:** I get error message: “bad coordinate - endian?”

**Probable cause:** You have binary file created on a computer with different endianness, i.e., with different order of bytes (like Sun and PC).

**Solution:** Try option `-r` with negative argument (see Sect. 2.2.3).

---

**Problem:** I get error message: “atom coordinate out of range” while minimizing.

**Probable cause:** High-energy initial configuration, atoms overlap

**Solution:** Try to run more steps of the steepest descent method before switching to the conjugate gradients. This is done by giving option `-m` a negative argument, or by variable `sd` when running `blend` interactively.

---

**Problem:** I have changed `.che` file but `blend` ignores my changes.

**Problem:** I have changed `.che` file but `blend` suddenly reports strange errors.

**Probable cause:** `blend` reads `.mol` and/or `.plb` files generated in previous runs.

**Solution:** Try

```
blend -r2 species.che
```

Option `-r2` forces `blend` to use 2D configuration from `.che` file. `species.mol` file will be ignored because extension `.che` is specified.

---

**Problem:** While using scrolling, text on screen is badly justified and smashed together.

**Probable cause:** The number of columns on your window is not what expected (usually 80).

**Solution:** (1) use command `$Cnumber` to adjust the number of columns; (2) set correctly the environment variables COLUMNS and LINES; (3) resize your window to 80x24.

---

**Problem:** Missing lines when using scrolling.

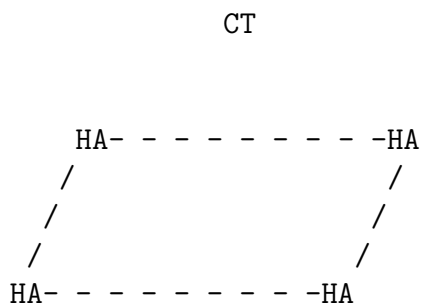
**Probable cause:** Some screens have that bad habit of eating ends of lines longer than number of characters/line which does not fit the scroll's assumptions.

**Solution:** (1) use command `$Cnumber` with large number; (2) set correctly the environment variables COLUMNS and LINES; (3) enlarge your window in x-direction.

---

**Problem:** I minimize a molecule from random initial configuration (or from rather complicated 2D screen coordinates) and I get unexpectedly high energy though the molecule configuration does not seem to be squeezed or locked.

**Probable cause:** Check arrangement of bonds around tetrahedral atoms. There is a local energy minimum consistent with the values of bond angles in which the four atoms form a square (instead of vertices of tetrahedron) and the "central" atom lies above or below this square:



(CT-HA bonds are not marked)

**Solution:** (1) Run `blend` interactively and enter `ran=1`; to randomize the positions; if it does not help, try higher values, but be careful. Hot key `:` works in a similar way. (2) Edit the problematic groups in the graphic environment (mark atom, press `m`, move the atom)

## 7.3 Frequently asked questions

### 7.3.1 Free molecules

Q: I want to add several water molecules (protons, ...) to a specified place on a protein.

A: I suppose you have already .mol and .plb files (obtained either via .che file, or using pdb). Run **blend** again interactively with **-e** option. Example:

```
blend -e -g -s myprotein
```

You will be prompted for editing commands. Type ? **Enter** to get brief help and then add atoms like in this example which places a water molecule close to atom O-HC:

```
af W-O OW -.834 O-HC 2
aa W-H1 HT .417 W-O
aa W-H2 HT .417 W-O
end
```

If you need to add more waters, try input from .edt file instead of typing the commands interactively.

Now you may need to re-optimize the whole structure again. Either type **keep=0** interactively, or re-run **blend** without **-e** option.

In fact, there is command **aw** available so that the above three command can be simplified to:

```
aw O-HC 2
```

### 7.3.2 Prevent molecules from evaporating

Q: I have a protein with several water molecules (or ions...) adsorbed. How to prevent them to 'evaporate' to infinity while minimizing?

A: Create the .jet file with artificial 'bonds' binding water to protein (See Sect. 3.2, option **-j**). Use **blend** with option **-j**. If you wish to have these forces (with much smaller force constant!) also in molecular dynamics, use **-j** option with **-o**, otherwise remove **-j** when the .ble file is generated by option **-o**.

### 7.3.3 One or more molecules?

Q: Is it better to split a 'molecule' consisting of parts not connected by bonds using option **-n** or treat it as one big 'molecule'?

A: Probably yes. But you must have the corresponding .gol file for **show** obtained WITHOUT option **-n-1** given to **blend**.

## Part II

Program ‘cook’ version 2.6



He who believes in evil spirits and ghosts is himself an evil spirit and a ghost (František Rachlík)

`cook` stands for a family of sequential and parallel programs for molecular dynamics of mixtures of molecules described by force fields based on site-site (Lennard-Jones etc.), Coulomb and bonded interaction. `cook` understands the description of molecules generated by `blend`.

# Chapter 8

## Overview

### 8.1 Features of cook

- Vibrating bonds or constraint dynamics to keep bond lengths constant (Lagrangian formalism or SHAKE)
- Free or periodic boundary conditions, rectangular box
- Linked-cell list method or all-pairs method for non-bonded interactions
- Ewald summation (highly optimized code) or cut-off electrostatic
- Attention paid to errors of integration as well as Ewald cutoff errors
- Convergence profiles of important quantities
- Measuring of many quantities, e.g.:
  - Electrostatic, internal, kinetic etc. energies
  - Virial pressure and pressure calculated by virtual volume change
  - Site-site correlation functions
  - Dihedral angle distribution
  - Bulk conductivity, autodiffusion coefficients
  - Shear viscosity
- Autocorrelation and block analysis of measured data
- Nose canonical thermostat, friction thermostat, Maxwell-Boltzmann thermostat.
- Friction-like isobaric ensemble
- Dipolar polarizability — original method with 2nd order predictor
- Parallelization — shared memory with Linux threads
- Batch and interactive modes

- Sophisticated input data unit including a calculator
- Scrolling windows emulation in interactive mode
- Flexible description of force field via program **blend**

## 8.2 History

10/1991 Constraint dynamics (Lagrangian and Hamiltonian formalism) \* Dept.Chem.  
\* Northwestern University \* Evanston \* Illinois \* USA

1/1992 Poly(ethylen oxide) \* Dept.Math. & Computer Sci. \* Odense University \*  
Denmark

4/1992 PEO V1.0, \* UTZCHT CSAV \* Praha 6 - Suchdol \* Czechoslovakia

5/1992 Bug fixes (V1.1) \* Odense

7/1992 Compatibility changes to the transputer parallel version (V1.2) \* Odense

8/1992 Serial version V2.0, parallel version V0.1 (see **ppeo.man**) \* Odense

3/1993 Easy implementation of different systems (V2.0) \* Institute of Chemical  
Processes \* Prague \* Czech Republic

10/1993 PROSIS/COOK Version C0.5 \* Odense

11/1994 Linked-cell list method version, free-boundary conditions version (C1.0) \*  
Odense

1/1995 PEO and COOK merged into 1 package \* Odense

4/1995 Polarizable dipoles \* Odense

6/1995 Isobaric ensemble supported. Cosinus shear stress to measure viscosity \*  
Odense

7/1995 Decoupled inter/intra-molecular friction thermostat \* Odense

2-10/1995 Support for shared memory parallel computers (Convex, SGI) \* Odense

11/1995 Playback and batch control improved (option **-number**) (**tcfg tprt** changed  
into **dt.prt dt.plb**; **dt.plb** instead of option **-y**) Packed convergence profile  
(and playback soon...) \* Evanston

7/1996 Better control over end-to-end distance, cross section, radius of gyration \*  
Evanston

6/1997 MPI parallelization

6-12/1999 New and extended polar version, Busing force field, axial polarizability \*  
Evanston, Prague

2000 Cluster analysis, conductivity \* Evanston, Prague

2001 CHARMM22, GROMOS96 and better manual – version number set to 2.0a. \* Evanston, Prague

2004 ANCHOR, WALL \* ICT Prague

2005 Rectangular box (not cube)

2005 Slab geometry, surface tension

2006 Widom insertion particle method

2008 Box scaling, pressure tensor (2.4a)

### 8.3 Compile-time versions of cook

The `cook` package allows customizing the executable according to various user requirements. This is accomplished by editing file `simopt.h` (see file `cook/generic/simopt.h` - lot of comments there!) and the makefile and recompiling (see Sect. 18.4).

Customizing of the most common versions is simplified by the *configurator*, script `configure.sh`, residing in the `cook` subdirectory. The projects created by `configure.sh` must be placed in subdirectories of `cook`. Example:

```
cd ~/macsimus/cook
configure.sh
```

The following major versions are available:

- EWALD Standard 3D periodic boundary conditions with Ewald summation to treat electrostatic interactions. See Sect. 11.2.
- NIBC Nearest-image boundary conditions, no cutoff, all pairs considered. Bug: cutoff corrections not calculated. Suitable for small systems only. Not tested recently.
- CUTELST Coulomb forces smoothly cut off to zero. Requested by `#define COULOMB 2` or 0—see `simopt.h`. See Sect. 15.3. Recommended executable name: `cookcut`.
- LINKCELL Instead of the “all pair” method to calculate the pair (*r*-space) sums, the linked-cell list method is used. Compatible with EWALD and CUTELST, cannot be combined with FREEBC or POLAR. Recommended executable name: `cooklew` (with Ewald), `cooklcut` (with CUTELST).
- FREEBC Free boundary conditions, Coulomb forces 1/*r*, all-pairs method to calculate pair interactions. Recommended executable name: `cookfree`.
- STARS The sign of electrostatic forces is changed so that gravity is simulated instead of charges. Normally with FREEBC. See Sect. 15.4. Recommended executable name: `stars`.

**WALL** Simulations at Lennard-Jones walls. See Sect. 15.8. Recommended executable name: `wall` etc. Not tested recently.

**SLAB** Slab geometry and z-density profiles measurement. Also forces needed to keep atoms in slabs. `SLAB=1` turns on some additional forces.

**GOLD** Simulation at vicinity of metal (ideal conductor) surface. **WALL** required. See Sect. 15.8. Not tested recently.

**POLAR** Polarizable dipoles supported. Can be combined with **FREEBC**. See Sect. 15.5.

**PARALLEL** At this moment, two versions parallelized by ‘`pthread`’ are available, (1) linked-cell list site-site interactions + Ewald and (2) real-space and Ewald running in separate threads. A MPI version was removed. (Other versions are under development, namely, to be rewritten from SGI to Linux.) See Sect. 10.

**MARK** Special to analyze pair energies. See Sect. 15.10. Not tested recently.

**ANCHOR** Fixes (anchors) one site and measures forces on it. See Sect. 15.9.

**PRESSURETENSOR** Direct measurements of the pressure tensor. See Sect. 15.6.

**SHAKE, VERLET** Several versions of the SHAKE algorithm and kinetic energy formula. See Sect. 11.3.1.

**site-site** The site-site potential (Lennard-Jones, exp-6, Busing, ...) is selected in the makefile (better said **metamake**). The respective code is in (from MACSIMUS home) `sim/XXX/sitesite.h` and `sim/XXX/sitesite.c`, where XXX is the name (e.g., `lj`, `exp6`, `busing12`...). The same code is used also by **blend**. The formats of the `ble`-file changes accordingly.

**LIBRARY** Library for energy and forces that can be called from other programs. See Sect. 18.5. Not tested recently.

In addition, a lot of compile-time options define “minor” versions, which may improve efficiency for particular systems and add/remove some extra features - see `cook/generic/simopt.h`.

## 8.4 Caveats and plans

See also `history.txt`

- Support for in-one-line or in-one-plane constraints is missing.
- Less memory-consuming method for erfc-like functions ( $r$ -space Ewald sums). This may even increase speed on systems with cache memory.
- Better I/O of configurations: read the **cooks** format, ASCII dump.
- Better load-balancing for parallel linked cell list method (dynamic slab widths as in the message-passing version)

## 8.5 Disclaimer

No warranty is provided on anything. If you do not like this program, just do not use it.

# Chapter 9

## Running cook

### 9.1 Where is

On some sites, particular executables (EWALD: `cookew`, FREEBC: `cookfree`, ...) may be in subdirectory `bin/` (of MACSIMUS home). See Sect. 18.4. Cf. Section ‘Where is’ in the manual for ‘blend’.

### 9.2 Synopsis

The command line to run any version of `cook` is:

```
cook [options] [sysname] simname [plbname.plb] [options]
```

If no parameter is given, a brief help on options is printed.

The parameters can be given in any order, the only exception is that sysname must precede simname.

Option `X` is switched on (i.e., set to value 1) by one of `-X` `-X+` `-X1` and switched off (cleared to value 0) by `-X-` or `-X0`. The options are case insensitive.

#### 9.2.1 File parameters

**sysname** Name of the system to simulate. If not given (which is deprecated), sysname=simname is assumed. Refers to file sysname.ble generated by `blend` and containing full description of the force field and, if simname.def does not exist, also to the definition file sysname.def

**simname** Simulation name. Names of files used in the run are obtained by appending extensions, see Sect. 9.2.3.

**plbname.plb** Optional name for the input playback file (used with `init="plb"` or `init<0`). If not given, plbname=simname. This parameter comes with extension `.plb` and may refer to file `plbname.vlb`.

## 9.2.2 Options

`-number` Overrides `no` for the 1st cycle (e.g., if input data are `no=3;;no=5;` and `-10` is specified, the result is the same as `no=10;;no=5;`).

`-@WHO@WHERE` Send e-mail to WHO@WHERE if the job finishes or crashes. Utility mail must be configured. (The old meaning of `-@` is now in the data as `No.occup`)

`-anumber` POLAR version: all polarizabilities are scaled by `number%`. The default is 100%. Useful for initializing the simulation.

`-b` Beep, progress, and stop option.

In the batch mode (no option `-s`):

`-bnumber` Checks for existence of `simname.stp` every `number`-th cycle. If found, finishes cycle and stops (as if `ctrl-C`).

`-b-number` Checks for existence of `simname.stp` every  $[100000 * \text{number} / (\text{No.s} * \text{noint}) + 1]$ -th cycle. The default is `-b-2` (stop in roughly a minute).

`-b0` Does not check for existence of `simname.stp`.

In the interactive mode (option `-s`):

`-b0` Does not beep

`-b1` Beeps (sends `\a` to `stderr`) when finished

`-b2` As above and after every ‘get data’ set processed (the default)

`-b3` As above and when SIGINT (`ctrl-C`) or SIGTERM caught

`-b4` As above and prints progress indicator to `stderr`.

`-b4` As above and prints progress indicator to `stderr`.

`-b-number` The same as `-bnumber`

`-c number` Method for correcting constraints and measuring constraint errors. See Sect. 11.3. Sum of flags; the default is `-c9 = 1+8`:

`-c1` This flag applies to the Lagrangian constraint dynamics only (irrelevant for SHAKE). The conjugate gradient method `conjgrad` is used to correct constraints after every integration step (default; denoted as `-c4` in versions prior V2.4a).

If not set, a SHAKE-like direct iteration method `Scorrect` is used to correct constraints of real configuration after every integration step (denoted as `-c3` in versions prior V2.4a).

`-c2` If set, constraint errors are measured before every integration step. Denoted as “constr err before” in statistics.

`-c4` If set, constraint errors are measured after integration step and before `Scorrect`. Meaningful with Lagrangian constraint dynamics and `Scorrect`. Denoted as “constr err mid” in statistics.



-c8 If set, constraint errors are measured after every integration step Denoted as “constr err after” in statistics (default).

-dnumber Check site-site distances.

-d1 once/‘get data’ set

-d2 once/cycle

-d3 once/step

The default is -d0 = never.

-enumber Number of items in the convergence profile. The default depends on compiling switches but can be reduced, or completely turned off by -e0.

-fnumber This option switches `cook` to a different mode. Instead of simulating, autodiffusion and structure factor are calculated from the recorded configurations. Generally it refers to reading a `simname.plb` with stride number; first and last frames are `init` and `no` (or option -number), respectively. See Sect. 14.6.

If CLUSTER, special data are placed in the def- and get-files after the first semicolon. (Normally, the get-file may contain more data sets separated by semicolons. Data after ; in a def-file are in normal runs ignored.) See Sect. 14.8.

-gnumber Input get-file is `simname.gnumber`. With -f, the output is `simname.Snumber`.

-hnumber Centers all atoms of given valence to its neighbors. To be typically used as -h4 to remove wrong pyramidal configurations of methane and other  $sp^3$  carbons, or similarly as -h6 for  $PF_6$ .

-inumber `ctrl-C` interrupt handler option. (`--EMX--`: `ESC` hit handler)

-i-1 Disable `ctrl-C` handler.

-i0 Interrupt and ask for the appropriate action, this is the default for the interactive run (see option -s) in serial versions. Not recommended for the PARALLEL version.

-i1 Finish cycle and read next set of data. This is the default in interactive run in PARALLEL versions.

-i2 Finish cycle, then save all and stop. This is the default in the batch run.

-i5 Finish step, then save all and stop. Recommended only if the next run will be with `init=2` because some measurements are affected by interrupting the measurement cycle.

WARNING: system-dependent and sometimes unexpected behavior. Can be disabled by deleting `#define SIG` in `main.c`. See Sect. 9.2.7.

-jnumber If number > 0, first number molecules are frozen (kept fixed and do not move). This option is intended for equilibrating water environment around one big macromolecule and similar. While initializing the configuration (see variable

`init`), the first number molecules do not rotate nor they move when MC (see variable MC) is selected.

If initializing configuration `init=5` is specified, and `number = nspec - 1`, this also causes that those lattice positions of species (`nspec-1`) which overlap (see `Emax`) with species `0... (nspec-2)` will be removed. A warning is printed and the next start of COOK should use the resulting (lower) number of molecules. Intended for immersing a big macromolecule into water prepared in advance. See Sect. 13.

BUGS: for `nspec>2` the first `nspec-1` molecules are placed at random lattice positions without checking possible mutual overlaps. This lattice site is not used for the last species.

In the POLAR versions, the dipoles in the fixed molecules are not polarizable.

The potential energy is not correct if option `-j` is used (differs by a constant dependent on the Ewald parameters, i.e., the Ewald test fails). Forces and thus the trajectory are OK

If `number < 0` then the initializer (`init>=3`) does not rotate first `|number|` molecules (formerly negative option `-q`). It is possible to use a big `-number` to denote all molecules (e.g., `-j-99999999`).

`-knumber` Force constant for keeping (“fixing”) selected sites in place. The units are  $\text{K}/\text{\AA}^2$  (not  $\text{kcal/mol}/\text{\AA}^2$ ). See Sect. 15.1.

`-lGFAVC` ASCII dump. GFAVC stand for digits denoting:

- C writes the configuration to file `SIMNAME.asc`
- V writes the velocities to file `SIMNAME.vel`
- A writes the accelerations to file `SIMNAME.acc`
- F writes the forces to file `SIMNAME.for`
- G writes the gradient of energy (numerical derivative) to file `SIMNAME.gra` (to compare with `SIMNAME.for`) SLOW!

The digits denote: `digit=0` do not write, `digit=1` denotes the maximum precision in the g-format, `digit=2..9`: number of decimal digits in the f-format.

EXAMPLE: `-11003` will dump the configuration (to 3 dec. digits) and forces (in max-precision g-format)

The six columns of the output file are:

- i Consecutive number
- mol Species number
- x,y,z Vector
- |vector| Its absolute value

`-morder` Integration method.

- m3, -m4, -m5, -m6 The order-value Gear predictor-corrector method is used with Lagrangian constraint dynamics (if constraints are present).
  - m2 Verlet integrator (with SHAKE if constraints are present). Requires `#define SHAKE` in `simopt.h`. This is the default.
  - m-2 Removed in version 2.4a, see `#define VERLET` instead
  - m1 Special: frames from simname.plb (which must have been created with option -yBIG) are read instead of simulating. The number of frames to read is given by variable `no`, the 1st frame to read is `-init` (default=1st frame in the plb file), the stride is `noint` (default=1=every frame); e.g., `init=-10 no=20 noint=2` reads frames 10,12,14,16,18,20. Useful if some measurements independent on simulations (calculating forces) were forgotten. Recommended usage: copy simname to a new name, symlink the plb-file, and run.
- m4 is generally recommended for complex molecules with many constraints (especially with constrained angles, hard bodies, etc.) as a balance between accuracy and stability problems of higher-order methods caused by some nearly-singular behavior of constraint dynamics and also noise introduced by cut-offs. Higher order may be better for very simple molecules.
- Verlet/SHAKE is recommended for simple constraints and for the POLAR version.
- n Write velocity playback (.vlb). WARNING: with Verlet/SHAKE, velocity is shifted by  $h/2$ .
  - onumber 0 Simulation is started without asking even if the lock file sysname.loc exists. Use with caution!
    - 1 If the lock file sysname.loc exists, the simulation does not start and an error message is printed. This is the default.
  - pKPC (for POLAR version: K,P,C are decimal digits )
  - pC (for nonpolar version) Selects various predictors.
- C With Verlet+SHAKE** (option -m2): C is the length of the predictor for calculating the velocities. Applies for the Nosé-Hoover thermostat, otherwise not used. The higher order, the better time-reversibility and total Hamiltonian conservation. C=0 is no extra history ( $v(t) \approx [r(t) - r(t - h)]/h$ ). The default is C=2.
- With Gear + Lagrangian constraint dynamics** (option -m > 2): Order of predictor for Lagrange multipliers (0=no prediction, default=9=value of -m). Affects efficiency only, not accuracy. The best results are probably obtained with the same order as Gear provided that parameter `eps` is optimized.
- P POLAR only:** the predictor or method order.
- 0 Car-Parrinello-like (extended Lagrangian) method. See Sect. 15.5.2.

- 1 No prediction (previous value).
  - 2 Always stable predictor-corrector. The predictor length is given by K, the default is K=2. This is the default.
  - 3,4 Higher-order predictors, may be useful with Gear integration.
  - K Predictor length. For P=2 (ASCP), the higher K, the better energy conservation.
- qnumber** Charges are rescaled by |number|%. The default is 100%. Negative number forces the configuration to be neutralized by adding the same charge to all charged atoms. Uncharged atoms are not charged.
- rorder** Deprecated option - may be removed/changed in future versions!
- Record configurations for further analysis (see, e.g., option **-f**). order is the maximum order of derivatives (it must not exceed the **-m** value). The configurations are recorded in files simname.1, simname.2, simname.3, etc., with the frequency dt.cfg ps. dt.cfg should be an integer multiple of **h\*noint**. The configurations are stored in float precision; negative order forces storing in the native **cook** real format, usually in double.
- Recommended values:
- r0** Do not record
  - r1** Record positions only (default)
  - r2** Record also velocities
- r-m** where m is the value of option **-m**: Stores fully restartable configuration.
- CAVEAT: Somehow duplicated by the playback, see option **-y**. The difference is that the **-r** function does not record the initial configuration (for **t=0**) and cannot be used more often than once in a cycle.
- snumber** This option sets the interactive mode: input is keyboard (instead of a get-file), output the screen (instead of a prt-file). If **SCR** has been #defined at compile time, you can use scrolling to watch the previous output. Type **\$?** from 'get data' to get help or see the **blend** guide for details. number is the buffer capacity in kbytes, missing number or number=1 means 31 kbytes (DOS) or 80 kbytes (other).
- t** Enables detailed runtime measurements (real time, i.e., it depends on the system load). In the PARALLEL versions sampling via **clock()** is also available; it reports critical portions of parallel times.
- (In case of compatibility problems, use compile-time option **-DCHEAPTIME**, and the time will be in 1 s resolution only. Even if the total time is long, there may be grid/interference errors if the resulting times are obtained using several cycles.)

- unumber** Bonds with the force constant  $K$  [in kcal/mol] less than number+0.5 will be vibrating, bonds with higher force constant will be constrained. (Artificial bonds written with  $K=0$  are always constrained. These are typically created by **blend** from bond angles with hydrogens and it does not make sense to combine constrained angles with vibrating bonds.) The default is **-u0**, i.e., all bonds are constrained. See the **blend** manual for details.
- vnumber** Verbosity level. Sum of powers of 2. The default is 3
- v0** Minimum verbosity. Only basic system+run info printed
  - v1** Runtime info, initialization protocols
  - v2** Brief statistics (was detailed prior V2.4a; use **staprt** for detailed statistics), message ‘playback written’, more initialization details; dihedral distribution and similar (if compiled so)
  - v4** Verbose details on site-site potentials and energy terms, copy of input **ble**-file, constraint optimization (SHAKE), detailed statistics: for debugging purposes
  - v8** POLAR: convergence
  - v16 = -v0x10** POLAR: file **.pol** with some summary
  - v32 = -v0x20** POLAR with **dV**: extensive files with induced dipoles:
    - .run.pol** Running induced dipole moments in program units (0.01175011 D)
    - .ex.pol** Exact (iterated) induced dipole moments in program units. NOTE: this is an average of the induced dipoles for **V+dV** and **V-dV**. **dV** thus must be small enough.
    - .err.pol** Errors in induced dipole moments in program units
  - v64 = -v0x40** Protocol on momentum and angular momentum and their setting to zero. See also variable **drift**
  - v128 = -v0x80** Raw dump of configuration, charges, and forces (incl. POLAR, after every evaluation, i.e., during iterations). Huge files of form **SIMNAME.#.dump** are created. To avoid problems if used unintentionally, **#** is limited to 0..9, then the program stops.
  - v1024** ASCII pseudo-graphs of radial distribution functions (obsolete)
- wnumber** Write configuration (simname.cfg).
- w0** Don’t write. Note that if several cycles (;;; in the data) are performed, **they are started from the same configuration** because a single **;** means restart from simname.cfg.
  - w1** Always write (default). Before the first write, a backup ((simname.cfg ) is made.
  - wnumber** As **-w1** and the available disk space is checked before selected files are written. If not enough space is available, it waits several minutes (more for the second time) and then checks again. The available space is compared to

a rather pessimistic estimate of space needed plus number kB added (to be even more pessimistic). -w2 should be enough unless your co-workers often fill the disk in a very greedy way. The algorithm is not 100% reliable, though. It relies on a system call to df; if this fails for any reason, it is considered as no space and the program keeps waiting. A warning is printed after space has appeared again; for apparent reasons, nothing is printed when the no-space condition is encountered.

Note: -w2 used to mean write only if the configuration has changed, but it became difficult to determine this condition so that this option was removed.

-xnumber (-1) Selects O-H and H-H Lennard-Jones terms in water models, see Sect. 15.2.

- 0 Turn off O-H and H-H Lennard-Jones terms. Note that the (strict) definitions of water models like SPC, TIP3, TIP4, etc. do not contain these terms.
- 1 As -x0 and use optimized code (not available for the LINKCELL version)
- 2 As -x-1 and disable doublecheck of water models. (This doublecheck was added in version 2.0m to warn users about previous TIP3P bug).
- 1 Keep O-H and H-H Lennard-Jones terms.

-ynumber Write playback files option.

number=0 Don't write (even if `dt.plb` is specified).

number>0 Write playback file(s), number is the number of molecules to write. The default is "big" which means that all molecules are written (in version prior 2.4g, this was -y-1). File names are always simname.plb (in version prior 2.4g, there were single-molecule files simname.p00, etc.). At the same time, `molcfg` is called to produce mol and gol files for `show` unless `init=0,1`.

number<0 As above but `molcfg` is not called. (Support for layers of water was removed; see `sim/old+misc/simils-plb.c` and `man/option-y.tex`).

-znumber Seed for random number generator. If -z is not specified, the seed is derived from time so that it is different for each run. Random numbers are needed for the configuration initializer and cross section measurements (certain modes only).

-[number Former box load option. Since V2.6h replaced by `load.n[]`, `load.N`, `load.L[]`, `load.tr` (see the data).

- number] (0) Former equalize masses (in %). Replaced by `equalize` in the data.

- number (2) POLAR only: parameter `k` (the predictor length) for the ASPC method [5]. The default is -2.

-number (0) POLAR only: (int)number/100 overrides the value of shell (Drude oscillator charge) from the ble-file, (number%100)/100 specifies value of `scf.omega` (can be overwritten by `get(scf.omega)` in the get data module). The default, (number%100)/100=0, will set the stability limit of ASPC.

- \number (0) PARALLEL only: The number of parallel threads (overrides environment variable NSLOTS). -\0 means NSLOTS.  
 With PARALLEL=1: A negative value sets the number of threads for the k-space part while the number of r-space (x-slab) threads is `No.cell[0]`.  
 With PARALLEL=1: 2 is expected unless NSLOTS=2.
- ]number (0) To show waters close to a big molecule (protein). -]-1 use all water sites (O,H, perhaps M; slower), -]is use only given water site (one of O,H,M).
- +number (0) Special for debugging with CHECKHEAP==2: AllocRange, see `gen/alloc.h` for details.

### 9.2.3 File extensions

Two names are specified at the command line. The system name (sysname) refers to sysname.ble generated by `blend` and containing full description of the force field. The simulation name (simname) refers to the specific MD run.

A list of extensions appended to simname follows. For instance, if simname=`crambin1` then file `crambin1.cp` contains the convergence profile.

- .1 .2 .3 ... Recorded configurations, see option `-r` and variable `tcfg`
- .anc ANCHOR: measured forces. See Sect. 15.9.
- .asc Ascii dump of the configuration, see option `-l`
- .cfg Configuration. Contains the whole configuration and some additional information.
- .cp Convergence profile. Record of total, potential, and bonded energies, pressure, temperature, and optionally more. Note that the term “convergence profile” here means just time dependence of quantities, not running averages of any kind.
- .cpa Ascii dump of (block-averaged) `.cp` file. Generated by `showcp`, in some special cases directly by `cook`.
- .cpi Input file of optional additional items to be recorded in the convergence profile (`.cp`) and statistics (`.sta`) Example:

```
! atom-atom distances will be recorded:
! atom1 atom2 [name]
  1      123   CC1 (this is comment, too)
  1      124   CC1_nbr
! additional variables will be recorded:
elst
LJ
+Lz
```

In the convergence profile only the first 4 letters of the name are used (e.g. CC1\_nbr is truncated to CC1\_). If the third column is missing, names ss1, ss2, ... are used.

If + is placed in front of the name or site-site pair, the variable will be also recorded in the statistics (this is redundant because this statistics can be always calculated by `showcp`).

The second form now accepts the following keywords:

- Epnc** Potential energy without cutoff corrections, in J/mol
- Ep0** Potential (bonded+LJ+elst+zero-energy) energy of first **No.first** molecules, in J/mol
- EpX** ? Cross potential energy (first **No.first** molecules vs. rest), in J/mol
- elst** Electrostatic energy of the configuration, in J/mol
- e10** Electrostatic energy of the first **No.first** molecules, in J/mol
- e1X** ? Cross electrostatic energy (first **No.first** molecules vs. rest), in J/mol
- LJ** ? Lennard-Jones energy, in J/mol
- LJ0** ? Lennard-Jones energy of first **No.first** molecules, in J/mol
- LJX** ? Cross Lennard-Jones energy (first **No.first** molecules vs. rest), in J/mol
- bond** Bonded terms (bonds, angles, dihedrals, impropers) (cf. Eintra), in J/mol
- bon0** Bonded terms for first **No.first** molecules, in J/mol
- Eext** Extended degrees of freedom energy (Nosé-Hoover, barostat; pot+kin), in J/mol
- U** Internal energy =  $E_{\text{pot}} + E_{\text{kin}} + \text{cutoff corrections}$ , in J/mol  
(It differs from the total energy, provided in the 1st column **Etot** of the convergence profile, so that it does not contain the extended degrees of freedom, **Eext**)
- Unc** Internal energy =  $E_{\text{pot}} + E_{\text{kin}}$  without cutoff corrections, in J/mol  
(It differs from the total energy, provided as the 1st column of the convergence profiles, so that it does not contain the extended degrees of freedom)
- H** Enthalpy =  $E_{\text{pot}} + E_{\text{kin}} + pV + \text{cutoff corrections}$ , in J/mol
- Hnc** Enthalpy without cutoff corrections, in J/mol
- fix** The potential keeping selected sites in place, in J/mol
- Pnc** Pressure without cutoff corrections, in Pa
- cmdr** Center-of-mass (to detect drift), in Å
- vdr** Center-of-mass velocity (to detect velocity drift), in Å/ps]
- andr** Angular velocity (to detect angular momentum drift) in free b.c., in 1/ps
- Tpo1** POLAR: for Lagrangian (Car-Parrinello-like) polarizability, kinetic energy of the additional degrees of freedom, in K



**pstd** POLAR: running error of induced dipoles (predicted minus corrected/iterated), standard deviation; statistics is called “polar one-step stderr”

**pmax** POLAR: running error of induced dipoles (predicted minus corrected/iterated), maximum; statistics is called “polar one-step maxerr”

**Pstd** POLAR: error of induced dipoles calculated in `measureP()` (see variables `dV`, `scf.epsx`), standard deviation; statistics is called “Polar stderr”. Note that `dV` must be small because (to spare one evaluation) a mean of  $V+dV$  and  $V-dV$  is used.

**Pmax** POLAR: as above, max error

**Pvxx,Pvyy,Pvzz,Pvyz,Pvzx,Pvxy** PRESSURETENSOR: Components of the virial part of the pressure tensor, in Pa

**Pkxx,Pkyz,Pkzz,Pkyz,Pkzx,Pkxy** PRESSURETENSOR: Components of the kinetic part of the pressure tensor (site-based)

**PKxx,PKyy,PKzz,PKyz,PKzx,PKxy** PRESSURETENSOR: Components of the kinetic part of the pressure tensor (molecule-based)

**Ptxx,Ptyy,Ptzz,Ptyz,Ptzx,Ptxy** PRESSURETENSOR: Total pressure tensor

**Lx,Ly,Lz** Sizes of the simulation box, in Å

**Jx,Jy,Jz** Current density (components of the vector) [Ampere/m<sup>2</sup>]

**NHxi** Nosé–Hoover degree of freedom  $\xi = \log(s)$ , dimensionless

**NHdx** Time derivative of Nosé–Hoover degree of freedom  $\dot{\xi}$ , in 1/ps

**zsh** slab center as determined by `autocenter`; in the units of  $L_z$

**Wc1**  $dG/d\sigma$ , reversible work per unit interface area, in K/Å<sup>2</sup>

**clz0,clz1** centers of cleaving walls, in the units of  $L_z$

**clf0,clf1** forces to the cleaving walls, in K/Å

Variable `No.first` = number of molecules in the first group should be in input data.

WARNING: The default is 0 (even if the `cp`-file is present).

Notes:

see `sim/simmeas.c`, variable `CPItab`; it is quite easy to add a new variable of type double.

`plot` (called by `showcp`) supports max 26 columns; to show more columns, use `mergetab`

**.cpz** Convergence profile in packed format (see variable `CPnbit`). Use utility `cppak` to convert it to `.cp` (and vice versa). Utility `showcp` understands the `cpz`-files as well so that the analysis of convergence profiles is transparent.

**.def** System definition (or defaults). If `simname.def` is not found, `sysname.def` is tried. The file should contain 1 set of input data (see Sect. 9.2.5). Note that numbers of molecules (species) `N[*]` can be given here, but not in `.get` file or by keyboard input. In addition, this file is read twice so that you cannot modify

the program default here (e.g., the cook default is `h=0.001`, but `h*=2` in this file would lead to `h=0.004`).

`.dih` DIHedral angle distributions, binary.

`.dia` DIhedral angle distributions, Ascii dump.

`.dcp` Dihedrals Convergence Profile. The dihedral values, in degrees, are printed every cycle to `simname.dcp` as ASCII. Which dihedrals are dumped is selected by `simname.ddh` and variable `dih.dcp` in input data. Must be compiled with `DIHHIST=-1`, applies to `species=0` only.

`.ddh`

`.mar` Define DiHedrals (`species=0` only). Must be compiled with `DIHHIST=-1`. Selects dihedrals to dump to `simname.dcp` and record (in `simname.dih` and `.dia`). If does not exist, all dihedrals are dumped (lot of data!). File of lines as, e.g.:

```
1 156 phi = -154.04 C5 - N6 - Ca10 - C15
2 145 psi = -124.77 N6 - Ca10 - C15 - N16
0 141 omega = -177.79 Ca10 - C15 - N16 - Ca20
```

The first column is irrelevant provided that it is nonzero (lines with 0 are ignored), so are lines beginning by '!'. The second column contains the dihedral number, as from the `sysname.ble` file, numbered from 1 to `ndihedrals`. The rest of the line is irrelevant. For the peptide backbone, angles phi,psi, the output of `ramachan` (the `sysname.mar` file) has exactly the correct format and can be renamed to `simname.ddh`. (`ramachan` uses the ble-file to derive this information).

`.dpr` Density profile, binary. Requires SLAB defined.

`.fix` To fix certain atoms. See Sect. 15.1.

`.for` Ascii dump of the forces, see option `-l`

`.g` For `.g` extension, see `rdfg` in the utilities manual.

`.g1, .g2, ...` Alternate names of input files (instead of `simname.get`). See options `-g` and `-f` (see Sect. 9.2.2).

`.get` Input data. `simname.get` contains one or more sets of input data (see Sect. 9.2.5) ended by 'quit=1;' or EOF. Irrelevant if `-k` or `-s` are specified (input from keyboard).

`.loc` A lock-file to prevent running two instances of `cook` with the same `simname`. Normally removed when `cook` finishes by any controlled way (incl. ERRORS and `ctrl-C`). Remains if `cook` (or computer) crashes or is killed by `kill -9`. It should be then removed before restarting `cook`.

`.p00 .p01 .p02 ...` Playback files. `simname.p00` is the file for the first molecule, etc. See also option `-y`. REMOVED since version 2.4g.

**.pol** POLAR: Summary info, requires option **-v16**

**.run.pol**

**.ex.pol**

**.err.pol** POLAR: Extensive tables of induced dipoles, see option **-v32**

**.prt** Output protocol. Irrelevant if **-d** or **-s** are specified (output to display), changed to **.sfd** if option **-f** is given

**.rdf** Radial distribution function(s), binary, see **rdfg** in the utilities manual.

**.s-s** List of atom types and groups for measuring radial distributions functions. See Sect. 14.7.

**.S1, .S2, ...** Alternate names of protocol files. See options **-g** and **-f** (see Sect. 9.2.2).

**.sf** Structure factor, see option **-f**. Note that the k-vectors are ‘circular’, i.e., they mean a number of waves in 1 Å multiplied by  $2\pi$ .

**.sfd** Output protocol with option **-f** (Structure Factor and autoDiffusion)

**.sta** Statistics. Contains measurements recorded for autocorrelation and error analysis (module **statics**)

**.stp** See option **-b**

**.xxx** Backups. Most of output files (**.cfg**, **.sta**, **.rdf**, **.dih**...) are backed up before writing; this is done every data set. The **.prt** file is backed up once when **cook** is started. Files **.cp**, **.plb**, **.pol** ... are never backed up—they are appended. Backup names are created by appending **.** BUG: not clear what happens for DOS...

**.wid** Results for the Widom method (WALL only).

**.z** Density profiles in ASCII, as generated from **.dpr**.

## 9.2.4 Program flow

(out-of-date)

- Options are analyzed.
- Files sysname.ble and sysname.def (or simname.def) are read and force field tables are constructed.
- A set of input data (each set is ended by **;**) is read from simname.get or from console. If **quit=1** is specified, or EOF is reached, the program stops.
- if not option **-f** nor **-m1** (normal mode) Either the initial configuration is generated (if **init**>=3) or it is read from file simname.cfg (if **init**<=2)

if option `-f` or `-m1` (special mode) Configurations (`simname.plb` or `simname.1`, `simname.2`, ..., are read, quantities (autodiffusion and structure factor for `-f`, other selected for `-m1`) are calculated, and then the program stops.

5. EWALD only: Unless `e1.test=0` or `e1.test=-10`, the control switches to another module that tests the Ewald summation and sets the parameters. This module accepts another data (see Sect. 11.2) but one of them is `e1.test`: once `e1.test=0` is specified, this module is abandoned and the control continues by the next step.
6. The specified number of steps (given by variables `no` and `noint`) is performed unless interrupted by pressing `ctrl-C`.
7. The configuration is stored to file `simname.cfg`. Files with statistics, convergence profile, etc., are closed. The protocol file `.prt` is flushed.
8. Control continues by step 4.

## 9.2.5 Input data

The input data are in `get data` format that has been described in details in the manual for `blend`. Briefly, it consists of a set of assignments ended by a semicolon; expressions can be written at the right hand sides of the assignments. Instead of using integer switches, a string with a mnemonic value can be used, e.g., `thermostat="Nose"` is the same as `thermostat=2`.

The following list contains all variables that can be used either in `sysname.def` (one set of data containing the defaults), or in `simname.get` or interactive input from keyboard. The values in parentheses are the default values, “strings” are the mnemonic values of switches. In addition, `sysname.def` is parsed so that you cannot modify the program default here (e.g., the cook default is `h=0.001`, but `h*=2` in `sysname.def` file would lead to `h=0.004`). Example (of `simname.get`):

```
thermostat="friction"
no=10 tau.T=0.01;           ! 10 cycles of fast cooling
no=20 tau.T*=5;             ! 20 cycles of 5*slower cooling
```

Note: in case of errors as e.g.:

```
ERROR data:bad identifier or syntax
ERROR data:bad number or expression
```

it may be difficult to say where the error occurred. Command `?` in the input data may help. Note also that there may be wrong data in the `ble`-file. E.g., `WARNING` in the `ble`-file prevents its correct reading.

Any declared variable but array can be used on the right-hand side in an expression. To enable calculations, there are the following auxiliary variables declared:

```
double a,b,c,aux,x,y,z;
int i,j,k,n;
```

In addition, `pi =  $\pi$`  is available.

Example:

```
x=0.01           ! auxiliary, to mean cycle in ps
noint=6          ! steps/cycle
h=x/0.001        ! timestep in ps
no=20/(h*noint)+0.5 ! simulation length 20 ps
```

Note that `no` is integer while expressions are calculated in double, in conversion to integer, the result is rounded down (truncated), and that is why `+0.5` is added.

`box.rmin` (0) For LINKCELL only: all coordinates are normalized to be `> box.rmin`. (Without LINKCELL always `> 0`). This is to prevent unnormalized configurations with Gear or box change. The default is `box.rmin = 0 = automatic setup` (recommended). `box.rmin < 0` forces `box.rmin = 0` even with Gear or box change (not recommended).

`bulkmodulus` (0) An estimate of bulk modulus (inverse compressibility) in Pa. It is used in the Berendsen-like NPT simulation (see `tau.P`), van der Waals radius fitting and similar. If not set, an ideal gas approximation is adopted. Hint: for water `bulkmodulus=2.2e9`. (No bulk modulus applies for the MTK-style barostat, `thermostat="NPT"`)

`cache` (64) This option rearranges the pair (*r*-space) sums. Void with the LINKCELL version.

`cache=0` Recursive triangulation (fractal) algorithm, efficient for large systems of polyatomic molecules (with long cutoff, otherwise LINKCELL may work better).

`cache=1` The simplest pair forces in 2 nested loops, good for small systems.

`cache>1` The pair (*r*-space) sum is rearranged into pair sum of blocks over `cache × cache` molecules. In case of a simulation with a lot of small molecules, the program runs faster because the number of cache-memory transfers is decreased. The typical speed gain is 20% on x86 for 1000 water molecules.

`center.K` (0) FREEBC only: Central force constant (see below). The center is (0,0,0).

`center.K[*]` (0) Periodic b.c.: Central force constants (separately in x,y,z coordinates). The force is to the center of the box.

`center.r0` (0) FREEBC only: radius of zero force.

`center.r0[*]` (0) FREEBC only: offsets from the center of the box.

`center.r0>=0`

$$\vec{f} = \begin{cases} -\text{center.K}(r - \text{center.r0}), & \text{for } r > \text{center.r0} \\ 0, & \text{for } r \leq \text{center.r0} \end{cases}$$

is added to the equations of motion for each atom (irrespective of its mass). `r` is the distance from the coordinate center, `center.K` is in the internal program units, i.e., in K. Similar formulas apply in the periodic version for each coordinate (with respect to the box center).

If `center.r0=0`, this is the harmonic oscillator: A free molecule will exhibit harmonic vibrations with period

$$\tau = c(m/n/\text{center.K})^{1/2},$$

where  $m$  is the mass in g/mol,  $n$  is the number of sites, and  $c = 2\pi \cdot 0.776 \text{ ps} = 4.88 \text{ ps}$ .

Example: for water, `center.K=1`,  $m = 18 \text{ g/mol}$ ,  $n = 3$ , and thus the period is approximately 12ps.

These forces are intended with nonzero `center.r0` to keep molecules in the selected area; outside this area, accelerations acting to molecules of different masses differ and motion is affected by a torque. Consider also `center.cmK` below.

`center.r0<0` This paragraph should be updated:

Special for MC initializing the in FREEBC version. `-jnmol` should be specified, too. Then a special ‘sticky’ potential of the form  $\text{center.r0} * (\text{mindist}/\text{center.r0})^2$  is added to all molecule-molecule pairs (molecules of indices  $< \text{nmol}$ , molecules of indices  $\geq \text{nmol}$ ), where `mindist` is the minimum atom-atom distance. Typical usage is for protein (`-j1`) and water: water molecules will find places in a shell (layer) of average thickness `center.r0` around the protein. There is no water-water sticky force.

The algorithm of inserting waters works as follows:

- gaussian random shooting of molecules to give total density of the cloud roughly `initrho`
- if inserting fails (`prob<pins`), `initrho` is slightly decreased and `number` is increased
- MC uses the same potential as the initial shooting but MD does not
- `pins<0` should not be used

`center.cmK[*]` (0) A central force (pointing to `rc=(0,0,0)` in case of FREEBC, or to `rc=(Lx/2,Ly/2,Lz/2)` = center of the box for periodic b.c.) is added to all particles. The potential of the force is

$$\text{center.cmK} * (\text{CM} - \text{rc})^2$$

where CM is the center-of-mass. This force causes the whole configuration (typically the slab for `center.cmK[2]>0`) to move to the center of the box, but there is no force to keep the slab together. The force applies to the first

`center.cmn` molecules only (excl. those fixed by option `-j`). Centering and removing velocity drift should be turned off for the selected coordinates (see `drift`); e.g., `drift&4` and `drift&32` should be zero if `center.cmK[2]>0`). The typical usage is to keep a configuration (slab) in place if other forces (fixed atoms) are present. A suitable force constant `cmK` may be from 100 to 10000. The correlation time in ps is  $\tau = \sqrt{\text{mass}/2/\text{cmK}} = 0.7755 \cdot \sqrt{M/\text{cmK}}$ , the period is  $2 \cdot \pi \cdot \tau$ . (The mass is in program units;  $\text{mass} = M \cdot 0.8314472$ , where  $M$  is the total mass of all molecules in g/mol.)

`center.cmn` (0) Number of molecules to which central force (see `center.cmK`) is added. A number greater than the number of molecules  $N$  is equivalent to  $N$ .

conserved (-1) Number of conserved degrees of freedom (momenta and angular momenta; not energy). If

conserved=-1 (default), it is determined from `drift`.

`corr` (3+16) Selects which corrections will be included by summing up the following flags.

- 1 Cutoff corrections are included in the final results (pressure, energy). (Some results are still recorded without these corrections.) If this bit is unset, the cutoff corrections are not included, they are only calculated and reported so that they can be added later; however, they are not available for the barostat (flag 2) not slab calculations.
- 2 Cutoff corrections are included in the barostat. Turning off this feature is useful for maintaining a selected pressure component as  $P_{zz}$ .
- 4 Finite-size correction  $N_{f0}/N_f$  (see (15.5)) is included (default since V2.6m). This applies to the atom-based pressure tensor (standard calculation) and atom-based virtual volume change calculation. The value can be combined with 1 or 2. Not suitable for NPT.
- 8 As above, except that the degrees of freedom are molecule-based (w.r.t centers of mass) Suitable for pressure calculated via molecule-based rescaling. This value can be combined with 1 or 2.
- 16 Finite-size correction is set unless a barostat (`thermostat="NPT"`) is used.

NOTES: New in V2.6m. These corrections are small for bulk liquids and solids but important for gases. Some finite-size effects  $\approx C/N$  are still present, but these corrections decrease substantially the constant  $C$ .

`CPnbit` (0) If  $1 < \text{CPnbit} < 24$  ( $1 < \text{CPnbit} < 16$  for DOS) is given, packed file `.cpz` will be created for the convergence profile instead of normal `.cp` one. `CPnbit` is the number of bits for coding the min-max range of each recorded variable (i.e., there are  $2^{\text{CPnbit}}$  levels). Use utility `cppak` to convert it to `.cp` (and vice versa). Recommended value is `CPnbit=12`, the file is then packed typically by 70%. Don't use with small `no`. Utility `showcp` understands the `cpz`-files as well so that the analysis of convergence profiles is transparent.

**cutoff** (-1) Real-space cutoff in A (irrelevant for FREEBC version [???]). **cutoff** must not be shorter than **LJcutoff** and longer than half the box size (for the LINKCELL version this limit is weaker). **cutoff=0** means that cutoff will be  $L/2$ , where  $L$  is the final box size (determined from density **rho**). For EWALD, **cutoff<0** means that cutoff will be selected automatically to a certain compromise value. **cutoff** also defines the range for calculating the radial distribution functions (this does apply also for the FREEBC version) unless modified by **rdf.cutoff**.

**slab.grid** (0) SLAB version only (see Sect. 15.7). The grid of the  $z$ -profile in the number of points per 1 A (if **slab.zmax** is specified) or  $L_z$  (rescaled to 1). May be a real number (**denseprof.grid=0.5** is a histogram bin width of 2).  
(Called **denseprof.grid** in versions 2.6h and older.).

**slab.mode** (0) If set (**slab.mode=1**), the system is assumed to be in the slab geometry and surface tension is calculated from previously recorded components of pressure tensor and virial pressure. Slab-based cutoff corrections are calculated from the  $z$ -density profile.

If **slab.mode&2**, and **dV** and **rescale** are set appropriately, the test area method is used instead of virial pressure (see Sect. 15.7).

(Called **denseprof.slab** in versions 2.6h and older.).

**slab.sp** (big number = no centering) Automatic slab centering for the measurements of the  $z$ -density profiles. It affects the measurement only, not trajectory.

**slab.sp**  $\geq 0$  specifies the species used for centering.

**slab.sp**  $< 0$  means that all species from number 0 to  $|\mathbf{slab.sp}|$  (incl.) are used. The centering is based on the best fit of function  $A \cos(2\pi z_i/L_z + \phi)$  (actually the first sin and cos components of the Fourier series are calculated.)

(Called **denseprof.sp** in versions 2.6h and older.).

**slab.zmax** (0) Max.  $z$  recorded. NEW: **denseprof.zmax=0** means the density profile calculated in  $[0, L_z]$  rescaled to  $[0, 1]$ .

(Called **denseprof.zmax** in versions 2.6h and older.).

**dih.grid** (0) Grid for measuring the dihedral angle distributions. See also **#define DIHHIST** in **simopt.h**. The full angle of 360 deg is divided into **dih.grid** subintervals. **dih.grid=0** turns off measuring. Cannot be combined with **dih.res**. WARNING: in versions 2.0h and older, angle 180 was divided into **dih.grid** subintervals and the sign of the dihedral angle was not honored.

**dih.res** (0) Resolution for measuring the dihedral angle distributions, in degrees. Cannot be combined with **dih.grid**.

**dih.cp** (-1) -1 Average of all gauche or cis counts recorded in the convergence profile

>0 Dihedral # **dih.cp** is recorded. (For the numbers, see the output .prt file)  
NOTE: from V2.0j, dihedrals (and **dih.cp**) are numbered from 1

**dih.dcp** (0) Must be set (**dih.dcp=1**) to enable dihedral dump. See file extensions .dcp and .ddh for details. DIHHIST=-1 required, species=0 only.



**drift** (4096) This variable affects the way how the momentum, center of mass, and angular momentum are corrected for numerical errors. In addition, number of conserved degrees of freedom is derived from **drift** if variable **conserved=-1**. It is a sum of:

**drift=1** Place the x coordinate of the center of mass to center (FREEBC) or box center (periodic b.c.)

**drift=2** Place the y coordinate of the center of mass to center (FREEBC) or box center (periodic b.c.)

**drift=4** Place the z coordinate of the center of mass to center (FREEBC) or box center (periodic b.c.)

**drift=8** Correct the x-component of the linear momentum (velocity drift)

**drift=16** Correct the y-component of the linear momentum (velocity drift)

**drift=32** Correct the z-component of the linear momentum (velocity drift)

**drift=64** Correct the x-component of the angular momentum (rotation)

**drift=128** Correct the y-component of the angular momentum (rotation)

**drift=256** Correct the z-component of the angular momentum (rotation)

**(drift&1536)=0** Corrections will be performed at start (init/load) and every cycle (default)

**(drift&1536)=512** Corrections will be performed at start (init/load) only

**(drift&1536)=1024** Corrections will be performed at start (init/load) and every step

**(drift&1536)=1536** Corrections will be performed at start (init/load) and save

**drift=2048** For molecules with 'dependants' (like site M in TIP4P water), the dependant positions are recalculated after every cycle (of **noint** steps). The default is that the dependants are calculated when necessary only (=before force calculation and saving cfg- or plb-files).

**drift=4096** Requests automatic determination of all components 1..256. This is the default. However, it may fail at certain combinations (like configuration anchored by a few atoms, combination of anchoring with walls, etc.)

**drmax** (0.125) Maximum displacement allowed in one integration step. Positive: applies with **init**>=3 (initialization) only, irrelevant for **init**<3. Negative: **|drmax|** applies unconditionally. Useful while initializing when forces are large and would cause the integrator to crash. Equilibration with this option turned off (**drmax=0** or **|drmax|** large enough) should follow. Should not be used in productive runs! For compounds containing hydrogen at ambient temperatures, **drmax=0.125** works fine. If **drmax** is used, **E<sub>max</sub>** can be larger (about 10000, but usually not more than 100000), which makes the initializer faster. WARNING: this was opposite in old versions, and this manual was wrong.

**dt.prt** (0.001) Frequency of printing protocol (in ps). **dt.prt=0** switches off printing, **dt.prt=tiny** value causes every cycle to be printed.

**dt.cfg** (0) Frequency of writing configurations to files `simname.1`, `simname.2`, ..., in ps. **dt.cfg=0** switches off the function, so does **-r0**. **dt.cfg** should be an integer multiple of **h\*noint**.

**dt.plb** (0) How often to write playback, in ps. See also option **-y** (the default is to record the whole configuration). **dt.plb** should be an integer multiple of the timestep **h**, otherwise the configurations will not be written in regular time intervals. **dt.plb=0**: writes once after every sweep (data set ended by **;**) finishes (or is interrupted). In both cases the initial configuration (at  $t=0$ ) is included (even if  $t=0$  is a consequence of **init=2**).

**dV** (0) Fool-proof independent pressure calculation by virtual volume change, also for area-change for surface tension [15.7](#). For debugging and checking purposes—normally the virial of force as calculated in **cook** from all pair forces is more accurate. Uses the formula:

$$P_{dV} = kT\rho - \langle dU/dV \rangle$$

where the derivative in the ensemble average is calculated numerically by scaling the volume  $\exp(dV)$  and  $\exp(-dV)$  times and taking the difference.

Turned off by **dV=0** (default).

With **rescale="CM"** (**rescale&8** set), selects a molecule-based calculation: Positions of centers of mass of the molecules are scaled by a factor given by the cube root of the volume ratio. The shape and size of the molecules are unchanged. In the above formula,  $\rho$  is the molecule number density and  $kT\rho$  becomes  $2E_{\text{kin-inter}}/3V$ , where  $E_{\text{kin-inter}}$  is the translational kinetic energy of whole molecules (of their centers of mass). Can be used also with constraint dynamics, but fails if the molecules are of sizes larger or comparable with the box size/2.

If **rescale&8** is not set, atom-based calculations are selected. Positions of all sites are scaled by a factor given by the cube root of the volume ratio. In the above formula,  $\rho$  is the atom number density and  $kT\rho$  becomes  $2E_{\text{kin}}/3V$ . Should not be used with constraint dynamics, i.e., **cook -u99999** should be used.

Reasonable **dV** is around 0.001, according to the accuracy of calculations (Ewald with truncated potential needs larger **dV**). The pressure, called **PdVa** (atom-based) or **PdVm** (molecule-based), is recorded in column 4 in the convergence profile.

For Ewald summation, the atom-based pressure **PdVa** is the same as the true virial pressure with small numeric deviation caused by finiteness of **dV** and sampling at different time. The ensemble average of the molecule-based pressure **PdVm** is also the same (irrespective of the system size).

For cutoff electrostatic, see Sect. [15.3](#), this is no longer true because of the truncate-and-shift errors. However, both pressures are the same in the thermodynamic limit and large **cutoff**.

For POLAR, see **scf.epsx** determining the accuracy of the self-field. **dV** should be relatively large, perhaps up to **dV=0.001**, even though this may lead to systematic pressure error. Note that the pressure error is proportional to  $dV^2$

because a second order difference method is used to calculate the energy over volume derivative.

For POLAR, the accurate (=iterated to precision `scf.epsx`) induced dipoles are compared to those generated by the integrator (predictor-corrector). However, to spare, the reference induced dipoles are calculated as average of the values for volumes  $V+dV$  and  $V-dV$  so that  $dV$  must be small enough.

Other boundary conditions (NIBC) have not been tested.

Bug: simple implementation, requires by one evaluation of forces more than necessary.

**E (0)** Total (potential+kinetic) energy, i.e., the value of the Hamiltonian, to be kept constant, in Kelvin. If the Nose ensemble is used, then the extended Hamiltonian. If the isobaric ensemble is used, then the enthalpy. See also `tau.E`

**Eelst[\*]** (0) External electrostatic field  $\vec{E} = (E_x, E_y, E_z) \equiv (E_0, E_1, E_2)$ , in V/m. It is recommended to use only the  $z$ -part, e.g.,  $E_z = \text{Eelst}[2] = 3e7$ .

**For dipolar systems:**

If `Eelst[i]` is set,  $i = 0, 1, 2$ , the component `M[i]` instead of total dipole moment  $|M|$  is reported in the convergence profile.

MACSIMUS reports `Mz` (or `M[2]`) in “program dipole units” (1 p.u. = 0.0117501077 D = 3.919414e-32 Cm). The dielectric constant (relative permittivity) is then computed from averaged  $M_z = Mz$ , see Appendix 30, eq. (30.3).

**For ionic systems:**

Can be used for NEMD determination of conductivity. To measure the electric current density, you must specify `lag.cond` (e.g., `lag.cond=lag.err`). Note that energy conservation is lost. MACSIMUS reports the current density in A/m<sup>2</sup> so that the conductivity in S/m is simply  $\kappa = J_z/E_z$ .

**el.alpha** (0.2 [EWALD], 0.7 [CUTELST]) EWALD version: The separation parameter, in 1/Å. See also `el.test`. It is set automatically if `el.test=-10`.

FREEBC: irrelevant.

CUTELST version: the cutoff is smooth in the interval `[alpha*cutoff,cutoff]`, `alpha=0.7` is recommended. (NOTE: larger values, as e.g. 0.9, cause artifact angular correlations for small molecules as e.g. water).

**el.centroid** (0) This option make sense only for ionic systems (charged molecules). If set to 1, the dipole moments of ions are calculated with respect to charge centroid (center of squared partial charges). The total dipole moment of the simulation cell then is not a sum of the partial dipole moments. (The total dipole moment calculate during the Ewald summation is anyway.) If set to 0 (default), no transformation is made; the dipole moments of individual moment do not make sense. (To be changed in future.)

`el.diag` (0) `el.diag=1` Calculate and include to the electrostatic energy the diagonal  $k$ -space correction [20]. Useful for molten salts.

`el.diag=0` Calculate, but do not include the diagonal correction for Ewald electrostatic energy [20].

`el.diag=-1` Do not calculate the diagonal correction at all.

`el.diff` (0.05) Relative error of box size change (any coordinate) before a warning “\*\*\* box has changed by  $>el.diff$  (log-scale) and  $k$ -vectors remain unchanged” is printed. Note that the number of  $k$ -vectors is not adjusted with the box because it would lead to a jump in energy.

`el.epsinf` (3e33 for Ewald, 1 for cutoff electrostatics) The relative dielectric constant  $\epsilon'_r$  of surrounding continuum for the Ewald summation. `el.epsinf=-1` sets the dipole slab correction [18]. To be used for the slab geometry.

`el.epsk` (0.5) EWALD only: Accuracy of  $k$ -space Ewald sums, in  $K/\text{\AA}$ . `el.epsk` is the maximum allowed  $k$ -space cutoff error in force on one charge. See Sect. 11.2. (Prior 2.4e called only `epsk`)

`el.epsr` (0.05) EWALD or CUTELST: Accuracy of  $r$ -space Ewald sums (see Sect. 11.2) or approximation of  $1/r$  in cutoff electrostatics (see Sect. 15.3), in  $K/\text{\AA}$ . `el.epsr` is the maximum allowed cutoff error in force on one charge. (Prior 2.4e called only `epsr`)

`el.grid` (256 for COULOMB=-2, 512 otherwise) Grid to calculate functions needed for real-space Ewald sums and also the splines for CUTELST version. [The default is 64 for the 16-bit Turbo C/DOS version.] Not available for FREEBC version. WARNING: in special cases, a small `el.grid` may cause inaccuracies. For water the error in pressure caused by intramolecular Ewald  $r$ -space terms calculated via splines is  $2e13/el.grid^3$  Pa; thus, at least `el.grid=1024` is needed. Similarly the surface tension error is about  $200000/el.grid^3$  N/m. With Ewald use COULOMB=-2 to use a more accurate `erfc` for intramolecular terms. This is not possible with LINKCELL, thus, `el.grid=1024` is the minimum. (Called `erfcgrid` prior 2.4e)

`el.kappa` (0.2 [Ewald], 0 [CUTELST]) The reciprocal space cutoff parameter in  $[1/\text{\AA}]$ : maximum integer  $k$ -vectors are  $K_x=el.kappa*L[0]$ ,  $K_y=el.kappa*L[1]$ ,  $K_z=el.kappa*L[2]$  in the respective directions. More precisely, the absolute value of vector  $(K_x/L_x, K_y/L_y, K_z/L_z)$ ,  $\vec{K} \in Z^3$ , is bound by `el.kappa`. Note that these  $k$ -vectors mean the number of waves in a unit of length (1  $\text{\AA}$ ). The  $k$ -vectors used in the structure factor calculations are multiplied by  $2\pi$ . See also `el.test`. For the CUTELST version, `el.kappa=0` should be used. HISTORY: in old cubic-box versions, dimensionless parameter  $K=el.kappa*L$  was used. In non-rectangular-box versions prior 2.4e, this  $K$  meant  $K_z$  and  $K_x, K_y$  were scale accordingly.

**el.minqq** (1) Minimum charge-charge distance, in Å. This value affects only the reported error for the *r*-space functions. See Sect. 11.6, for details.

(Called **minqq** prior version 2.4e)

**el.sf** (1) Applies for **cook\* -f** (re-reads configurations and calculates structure factor, diffusivity, etc.).

**el.sf=1** : Calculate a sphericalized structure factor (for a cubic box only)

**el.sf=3** : Calculate the full 3D structure factor.

**el.test** (-10) EWALD only: Unless **el.test=0** or **el.test=-10**, the control switches to a module that sets and tests the Ewald parameters. See Sect. 11.2.

The default value **el.test=-10** means automatic selection of **el.alpha** and **el.kappa** based on default accuracies **el.epsk** and **el.epsr**.

**el.test=0** means no automatic selection (the values of **el.alpha** and **el.kappa** remain unchanged).

**E<sub>max</sub>** (10000) The energy limit for inserting one molecule when the configuration is initialized (**init>=3**), in Kelvin. See also **drmax**.

**eps** (1e-6) Accuracy for calculating the Lagrange multipliers, See Sect. 11.3. With option **-1**, the step for numerical gradient. (default was 1e-5 prior V 2.0d)

**epsc** (1e-6) Accuracy of correcting constraints, See Sect. 11.3. The default is changed to **epsc=0.05** for conjugate gradient method of correcting constraints (unset bit 1 of **-c**). (default was 1e-5 prior V 2.0d) Special: For **|SHAKE|=1**, **epsc>=1** determines a constant number of iterations; **omegac** must be set by hand.

**eps1c** (1e-5) LINKCELL only: accuracy with which any site should be in the basic cell. Negative: no checking at all. The sites may go slightly off the basic cell only if the box is rescaled, really or virtually (cf. **tau.rho**, **tau.P**, **dV**), but this error should be small. If a site off the box is detected, warning is printed and the limit is doubled; if it grows over 0.1, this is an error.

**epsp, epspx, epspq** Changed into **scf.eps, scf.epsx, scf.epsq** – see there.

**equalize** (0) Redistribute masses of atoms. 0 (the default) = use the original masses. 1 = masses of all atoms in a molecule are equal. Equalizing masses (especially of hydrogens) enables longer timesteps, however, kinetic quantities are affected. E.g., while **h=0.001** ps is recommended for water (with normal hydrogens), **h=0.002** ps is acceptable if the masses are equalized. Dependents are not equalized (if there are dependants with mass, it is distributed to nearby sites.) (Formerly option **-j**)

CAVEAT: Since **equalize** is read from data before the **ble**-file is read, not all variables are available to be used in formulas. The following variables are available: **no**, **noint**, **h**, **E**, **rho**, **L[]**, **tau.T**, **T** Examples of data:

```
noint=5 h=.01/noint equalize=0.8*(h>0.0015) ! allowed, equalize=0.8
center.K=5 equalize=center.K>1 ! equalize=0: WRONG - no error reported
```

**h** (0.001) The integration step in ps. Hint: this is a useful trick to set **h**:

```
noint=8 h=.01/noint
```

**init** (0="continue") The initial key:

- 0 "continue" or "cont" Continue run from the point stored on disk, all measurements the convergence profile and playback will be appended. This is the default for every cycle. Some measurements like CPU times are not stored and are initialized for each run.
  - 1 "append" As above but the measurements are initialized; only the convergence profile and the playback files will be appended.
  - 2 "start" As above but also the convergence profile will be initialized.
  - 3 "random" Everything will be initialized included the configuration. Random shooting algorithm used. See also MC. FREEBC: the first molecule (molecule 0) is always centered, other molecules are shot according to Gaussian distribution. See Sect. 13.
  - 4 "slab" Cheap way to make a liquid slab. The same as **init=3="random"** with a bias to the center of the box (in the z-direction). The initial slab is quite diluted, but it condenses. The box should be longer in the z-direction and the initial cooling should be intensive (otherwise the slab expands instead of condensing to the center). Tested for water with **thermostat="Berendsen"** **tau.T=0.1**.  
(In versions prior 2.4n **init=4="rawrandom"** meant not correcting constraints of input configurations.)
  - 5 "crystal" or "lattice" Everything will be initialized included the configuration. Initial configuration is a regular lattice. See **pins** for available crystal structures.
  - 10 "asc" Reads configuration from file simname.asc (see option -1 for its format.) incl. box size.  
This option is not not equivalent to standard read of binary .cfg file (**init=0,1,2**). Some information is lost, most load options not available).
  - 11 As above + also velocities are read from simname.vel.
  - 12 As above + also accelerations are read from simname.acc. Useless for normal runs (because accelerations are calculated), usable for debugging Ewald summation. In older versions requested by **init=20**.
- init<0** Reads playback, frame |**init**|, then as if **init=2**. Symbolically **init=-1="plb"**. Velocities are assigned randomly (as if **initvel=big\_number**). If the plb-file contains zero box size (any of Lx,Ly,Lz), the box size is copied from cook initialization (given by values of **rho** and **L[]**). If the box size in the plb-file is nonzero, it overrides the cook initial values; however, some initializations (Ewald, cutoff corrections) are kept from cook initializations. Thus, it is recommended to save the configuration (after **no=0**) and then continue or restart cook.
- (Since version 2.6b it is possible to rewrite the plb-file after reading. Utilities **plb2cfg** and **cfg2plb** can be also used for mutual conversion.)

999 "convert" Obsolete. Reads simname.1, simname.2 ... simname.no and writes frames to the playback file(s); nothing is calculated. The last configuration treated has number **no**

**initrho** (0) The initial density in kg/m<sup>3</sup>. Applies for initializing the configuration (**init**>=3). The value of 0 (default) means that **initrho**=**rho**.

**initvel** (0) Re-assign the velocities of the first **initvel** molecules according to the Maxwell-Boltzmann distribution with current **T**. It is done just once (**initvel**=0 is set after assignment). (Formerly option **-j** with negative arg).

**key** (0) A "command" to execute (immediately, i.e., not after **;**):

0="run" No command. This is the default which is re-set after any other command is finished (This means that a simulation is (re)started after data input is ended by **;**.)

1="sort+x"

2="sort+y"

3="sort+z"

-1="sort-x"

-2="sort-y"

-3="sort-z" The molecules (of each species separately) are sorted according to increasing values of given coordinate of the center-of-mass; the versions with + in the increasing order, version with - with the decreasing order. May be useful, e.g., for selecting some molecules.

4="cp" Show the convergence profiles: calls **showcp -99**

5="show" Show the trajectory: calls **show**

6="rdf" Show the rdfs: calls **rdfg pu**

7="cn" Show the coordination numbers: calls **rdfg u;plot :1:4 NAME.\*.\*.g**

8="shell" Calls the shell

9="quit"="exit" Exits cook

**L[\*]** (0) Box sizes (3D array). If **rho**=0 is set, these values are taken literally and are used in configuration initializer as well as target size (if **tau.rho** is specified). With nonzero **rho**, the box is rescaled to reach given **rho** (i.e., **L[\*]** specifies only ratio **L[0]:L[1]:L[2]**). If one, two or three of **L[]** are missing (i.e., = 0), and **rho** is given (> 0), the missing **L[]**s are calculated to reach given **rho**; if more than one **L[]** are, they are the same. E.g., if all **L[\*]** are missing and **rho** is given, a cubic simulation cell is created.

**L[]** and **rho** are ignored if the configuration is read (**init**<=2 from file and **tau.rho**=0 is specified. This default behavior can be changed by variable **load.L[]**.

**LJcutoff** (-3) Cutoff for the Lennard-Jones interactions. If negative, then a multiple of Lennard-Jones **sigma**'s

**lag.err** (32) Lag for the statistical analysis of time-correlated data, in the number of cycles. In every cycle, one item of data is recorded. See Sect. 14, for details. (One cycle is **noint** integrations steps). Applies only to certain more important variables.

**lag.n** (14) Blocking (sub-averaging) by the factor of 2 to the block length of  $2^{lag.n}$ .

**lag.v** (0) Lag for the velocity-velocity time autocorrelation functions. Active for `#define RGYR` only. From these functions the diffusivities can be calculated. (There is an equivalent method based on the Einstein relation, see `plb2diff.c 22.22`.) WARNING: slow for too long **lag.v** and/or too many molecules **lag.nv**.

**lag.dim** (3) Number of coordinates ( $v_x, v_y, v_z$ ) recorded.

**lag.nv** (big number) Velocity-velocity time autocorrelation functions recorded for **lag.nv** molecules.

**lag.cond** (0) Lag for the time autocorrelation functions of the current density. From these functions the conductivity can be calculated. (There is an equivalent method based on the Einstein relation, see `plb2diff.c 22.22`.) The names of variables are **jx,jy,jz** (can be specified in `SIMNAME.cpi`)

**lag.visc** (0) Lag for the time autocorrelation functions of the off-diagonal components of the pressure tensor. From these functions the viscosity can be calculated.

**load.L[\*]** (0,0,0) **load.L[i =0]** The loaded box size is used (i=0,1,2).

**load.L[i =1]** **L[i]** is taken from data only if larger (so that no overlap may occur)

**load.L[i =2]** **L[i]** is taken from data only if smaller (i.e., only if the box shrinks, which may result in overlaps).

**load.L[i =3]** **L[i]** is taken from data unconditionally (the box may shrink or swell)

“**L[i]** is taken from data” means it is directly specified or calculated from **rho** and **N[]**. Example: You have a periodic cubic box of size  $31 \times 31 \times 31$  Å and want to use it as an initial configuration for a slab of size  $30 \times 30 \times 75$  Å: while in the input data you should have

```
drift=4+8+16+32    ! center slab in z, remove drift in x,y,z (8+16+32)
el.epsinf=-1       ! slab dipole correction - Ewald only
L[0]=30 L[1]=30 L[2]=75 rho=0 ! box size fixed
load.L[2]=1        ! L[2]=75 will be used
tau.rho=1          ! how fast to reach the box
```

**L[0]=30 L[1]=30** will remain original, but will slowly be reached because of **tau.rho=1**

HINT: if you wish to recenter the configuration, use array **shift[]** in the input data.



`load.N` (0) Change the number of molecules. Not good for mixture components.

`load.N=0` Default: If the specified (in the def- or ble-files) number of molecules does not match the loaded one (from the cfg-file), this is an error and the program stops.

`load.N=1` If the specified number of molecules > the loaded one, the missing molecules are inserted randomly; the opposite inequality is an error. It is recommended to use `pins=0` to avoid box scaling.

`load.N=2` If the specified number of molecules < the loaded one, the overflow molecules are omitted; the opposite inequality is an error.

`load.N=3` Either of `load.N=1,2` applies.

`load.n[*]` (1,1,1) Periodically repeat the cell while loading. Useful for simulating fluids of small molecules. To use this feature, prepare first a small configuration of the same name (unless you use a plb-file: see variable `init<0`). Then prepare a def-file of the new configuration (its `N[]` should contain the NEW [large] numbers of molecules) and run with `load.n[]` and `init=2` in the input data. It is then recommended to randomize velocities by specifying `initvel=999999`, otherwise all the copies will develop in the same way (they are identical incl. velocities) until tiny rounding errors cause the trajectories to diverge. See also utility `plbrePLICATE`.

`load.reverse` (0="no") If `load.reverse=1` or `load.reverse="yes"` the configuration is loaded with reversed endian. `cook` must be compiled with `-DReverse`. Now (2008) rarely needed. Former `loadreversed`.

`load.tr` (0) Specifies a transformation (swapping, rotation) of the initial box while loading.

1="xy" x <-> y

2="yz" y <-> z

3="zx"="xz" x <-> z

4="zyx" z -> y -> x -> z

5="xyz" z -> x -> y -> z

WARNING: may interfere with `L` specified in the data. Example: to immerse a crystal of NaCl into water, prepare the crystal (use order of species Na Cl water; e.g., `N[0]=108`, `N[1]=108`), add water (e.g., `N[2]=800`), then specify a larger box size in the data (may be done indirectly via `rho=1050`) and run `cook` with `load.L[0]=1 load.L[1]=1 load.L[2]=1 load.N=1`. The original (smaller) crystal of NaCl will be placed into a large box and surrounded by water molecules.

`maxscale` (1.03) Maximum allowed scaling of box or R. If larger scaling is requested, warning is printed and scaling is reduced to range `[1/maxscale,maxscale]`.

**MC** (0="no") WARNING: deprecated unless special requirements. It is recommended to use MD minimization with **drmax** turned on, and large **E<sub>max</sub>**.

For **init=3** only. Adds MC Metropolis Monte Carlo sweeps (attempted moves for every molecule) at temperature **T** after initializing the configuration. Pressing **^C** (and selecting 1, see option **-i**) interrupts MC and (after finishing a sweep) continues by MD. **MC=-1="forever"** selects running MC until all pair energies fall below **E<sub>max</sub>** (or until **^C**). In addition, molecules are not checked for mutual overlaps (given by the energy limit **E<sub>max</sub>**) when they are inserted; however, if option **-j<sub>number</sub>** is specified, then overlaps with first number molecules are checked. One MC step is with probability 1/3 a shot to a random place, with probability 1/3 a random displacement (the length is adjusted during the run) and with probability 1/3 a random rotation (the angle is adjusted during the run). Note that the *k*-space Ewald contribution (if any) is not included in the MC energy.

**FREEBC**: central force (see **center.K**) is included into the MC energy.

WARNING: uses a table of pair energies and therefore requires a lot of memory for many small molecules.

**mirror** (0) For **mirror=1**, mirror inversion is added to a random orientation of a molecule. Applies for the Widom insertion and random initial configuration.

**N[\*]** Number of molecules. Available in simname.def only. The format is:

**N[species number]=number of molecules**

The number of molecules passed from **blend** (option **-n**) and written to **sysname.ble** can be thus changed. Note that species are numbered from zero: the 'first' species is referred to by **N[0]**, e.g., **N[0]=500** to simulate 500 molecules.

**newt** (-3e33) REMOVED since V2.3a. Now **init=2** sets **t=0**.

**no** (10000) Number of cycles to be performed. Note that you can interrupt the calculations before **no** cycles gracefully by ctrl-C or signaling **kill -2**. See also option **-number**.

**No.cell[\*]** (0) Only for LINKCELL version: number of subdivisions of the simulation box in each dimension. Default **No.cell=0** means automatic selection according to **No.per<sub>cell</sub>**.

**No.first** (0) Energy of **No.first** molecules (without Ewald *k*-space part, if any) used instead of **E<sub>in</sub>** in the convergence profile. When **No.first=0** (default) then the 'intramolecular' energy of all molecules is recorded. Not supported for LINKCELL.

**No.occup** (0) Applies to the LINKCELL version only: Histogram of cell occupancies is calculated and printed in every step. If **No.occup>1**, then max number atoms in a cell are recorded, for **No.occup=1** certain default is chosen. For debugging purposes.

**No.percell** (5) Only for LINKCELL version: Calculates **No.cell[\*]** so that there are about **No.percell** sites per each cell and the cells are (approximately if not possible exactly) cubical.

**noint** (10) Number of steps per one cycle. There are (**noint**-1) steps performed without measurements, and 1 step with the measurements included. Note that measurements are quite expensive so that do not use too low **noint**.

**nomax** (0) The simulation stops if the total number of measured cycles reaches or exceeds **nomax**. **nomax**=0 turns off the check (the simulation runs until all data sets are processed or **quit**=1 or EOF is specified). See also **stop**.

**norm** (0) REMOVED: see **drift** and **conserved**.

**omega** (0) Not implemented in the current version! The relaxation parameter for the **directiter** method for calculating the Lagrange multipliers.

**omegac** (-1.2) The relaxation parameter (mixing iteration parameter) for the **Scorrect** method for correcting constraints or for SHAKE. If negative, automatic optimization is performed (use option **-v&4** to get detailed protocol). See Sect. 11.3, for details.

**omegap** See **scf.omega**

**over14** (3) LINKCELL version only: The linked-cell list algorithm searches for 1-2, 1-3, and 1-4 exceptions only in a certain radius around sites. This radius is dynamically adjusted during the run. **over14** is the factor by which the estimate of the change (between MD steps) of the max 1-2, 1-3, 1-4 distance is multiplied. (It has more effects, see **lcforce.c** for details). If **over14**=0, **cutoff** is used for this distance (safe but slow). Negative **over14**<-1 selects the debug mode: also prints distances and speeds of their changes. **over14**=3 is safe enough in most cases. If the algorithm runs close to its safety limits, a warning is printed, and **over14** is automatically increased. If this happens too often (let us say more than twice during a moderately long run), use larger **over14**.

**P** (1e5) Pressure to be kept constant (if **tau.P** is nonzero), in Pascal.

**pins** (0.01) **init=3,4** 0<**pins**<1 Lowest acceptable probability for inserting molecules while the configuration is initialized. If the insertion probability falls below **pins**, the density is decreased (box size increased) and a new attempt to fill the simulation box with molecules is made. Note: different in versions prior Sept 96.

**init=3,4** -1<**pins**<0 As above with **|pins|**; in addition, if the insertion probability is higher than **sqrt(|pins|)**, a new attempt is made with a higher density. Thus, after several iterations, a density with an insertion probability in (**|pins|**,**sqrt(|pins|)**) is found.

**init=3,4 pins=0** Does not change density while initializing, i.e., continues in shooting molecules even if very unsuccessful; it's user's responsibility to choose a suitable **initrho**.

`init="crystal" pins="auto"`  
`init=5 pins=0` Automatic selection of the lattice (minimum number of vacations)  
`init="crystal" pins="sc"`  
`init=5 pins=1` Simple cubic lattice (with possible vacations at random positions)  
`init="crystal" pins="bcc"`  
`init=5 pins=2` Body-centered lattice (with possible vacations)  
`init="crystal" pins="fcc"`  
`init=5 pins=3` Face-centered lattice (with possible vacations)  
`init="crystal" pins>3`  
`init=5 pins>3` Structure contained in file with name `cfg $\underline{ns}$ . $\underline{pins}$` , where  $\underline{ns}$  is the number of sites per molecule and  $\underline{pins}$  the number of molecules, will be used and, if necessary, periodically copied; if the total number of molecules is not an `integer^3` multiple of  $\underline{pins}$ , then vacations will be present. The box size will be derived from the box size stored in the file. The configuration file can be created using option `-l` by `cook`, it only must be renamed. May be combined with option `-j`

`quit` (`0="no"`) Deprecated: quit `cook`, start a shell, debug. Use `key` instead.

- 1 `"yes"` Quits the program `cook`. The same as if EOF is encountered (or ctrl-D from keyboard).
- 1 `"debug"` Debug mode: enables printing distances of selected sites.
- 2 `"paste"` Paste molecule: enables replacing a configuration of a selected molecule by data given on input (3-column ascii x y z format)
- 3 As -1 and -2 simultaneously.
- 4 `"shell"` Starts a shell (UNIX: `csh`, OS/2: `cmd.exe`, DOS: `command.com`). After exiting the shell, the control returns back to 'get data'.

`rescale` (`15="xyzCM"`) Mode for box rescaling used by a barostat (see `tau.P`), density initializer (see `tau.rho`), and virtual volume (area) method (see `dV`). Sum of powers of 2:

- 1  $x$  coordinates are rescaled; "`x`" in the mnemonics below
- 2  $y$  coordinates are rescaled; "`y`" in the mnemonics below
- 4  $z$  coordinates are rescaled; "`z`" in the mnemonics below
- 8 rescaling is based on molecular center-of-mass (default), "`CM`" in the mnemonics below
- 16 Pressure tensor components are calculated and used separately (`#define PRESSURETENSOR=3` required). All specified dimensions of the box fluctuate independently. Denoted by uppercase "`X`", "`Y`", "`Z`" in the mnemonics below. Useful for crystals, slab version, etc. E.g.,

`rescale="ZCM" tau.P=100` will maintain  $P_{zz} + P_{\text{cutoff corr.}}/3 = P$  by changing  $L_z$  only.

If this flag is not set, the isotropic pressure is used; consequently, all specified dimensions of the box fluctuate synchronously (their aspect ratio is kept, a cube is always a cube). Should be used for homogeneous fluids.

32  $x$  and  $y$  scalings are the same; "xx" or "XX" in the mnemonics below.

64 All  $x, y, z$  scalings are the same; "xxx" or "XXX" in the mnemonics below.

In principle, 1+2+4 and 1+2+4+16+64 denote the same uniform box rescaling. In the latter case, a pressure tensor components are used instead of the virial pressure. The results are almost the same with Ewald summation, but there is a noticeable difference for cutoff electrostatics.

Examples:

- `rescale="zCM"` or `rescale=12` selects molecule-based rescaling (good for small molecules) of  $z$ -coordinates only
- `rescale="CM"="xyzCM"=15` selects molecule-based rescaling (all coordinates).

The full list of available mnemonic codes is:

"CM",  
"x", "xCM", "y", "yCM", "xy", "xyCM", "z", "zCM",  
"xz", "xzCM", "yz", "yzCM", "xyz", "xyzCM",  
"xx", "xxCM", "X", "XCM", "Y", "YCM",  
"XY", "XYCM", "Z", "ZCM", "XZ", "XZCM", "YZ", "YZCM",  
"XYZ", "XYZCM", "XX", "XXCM", "XXZ", "XXZCM", "XXX", "XXXCM"

**rho** (1000) The density in  $\text{kg/m}^3$ . If `tau.rho` is specified, this density will be reached and kept constant. The box is rescaled so that its shape (`L[0]:L[1]:L[2]`) is the same. See also `L[*]`. If one or two of `L[]` are missing, and `rho` is given, the remaining `L[]` are calculated.

**rdf** (**structure**) Variables controlling measurements of the radial distribution functions (`rdf.grid`, `rdf.cutoff`, `rdf.onefour`), See Sect. 14.7.

**rg.cp** (0) `rg.cp >= 0` Radius of gyration and end-to-end distance of molecule number `rg.cp` recorded in convergence profile. `rg.cp=0` is recommended for e.g. a protein in water: only Rgyr of the protein will be recorded.

`rg.cp = -1` Averaged radius of gyration and end-to-end distance over all molecules recorded in convergence profile. `rg.cp=-1` is recommended for bulk pure molecular fluid.

The value of `rg.cp` does not affect statistics which is recorded for each species separately (if possible by the values of `rg.end`)

**rg.end[0]** (0) Head atom for measuring the "end-to-end" distance. Negative value means `ns+rg.end[0]`, where `ns` is the number of sites. For instance, `rg.end[0]=-1` is the last atom in the molecule.

`rg.end[1 (-1)]` Tail atom for measuring the “end-to-end” distance. See above.

`scf.eps` (1) POLAR only: Accuracy of induced dipoles. Negative integer `scf.eps` means a constant number of iterations. (Called `eps` for  $V < 2.6f$ ). See Sect. 15.5.

`scf.epsx` (0) POLAR only: Accuracy of induced dipoles for  $P$  measured by virtual volume change and similar testing purposes. The default, `scf.epsx=0`, implies certain value which can be often reasonable, but generally it is recommended to try several values to resolve a speed/accuracy tradeoff. The optimum `scf.epsx=0` depends on  $dV$  and Ewald setting. For predicted force field and one iteration/step, one should have `scf.epsx`  $\ll$  `scf.eps`. For integration methods with full iteration of the self-field, both `scf.eps` and `scf.epsx` should be small enough.

Since this method calculates the numerical derivative via a difference, it is necessary to ensure high accuracy of the self field. See also variable  $dV$ .

(Called `epspx` for  $V < 2.6f$ ).

`scf.epsq` (0.8) POLAR only:

$0 < \text{scf.epsq} < 1$  `scf.eps*(1-scf.epsq)` will be used as the actual error limit EPSP in the 1st step, instead of `scf.eps`. In subsequent steps,

$$\text{EPSP} := \text{EPSP} * \text{scf.epsq} + \text{scf.eps} * (1 - \text{scf.epsq})$$

is performed after every step so that after some time `scf.eps` is reached. After every data set (ended by ;) read, `scf.epsq` is made negative and this function is turned off. This is because the history for the predictor IS saved across batches.

`scf.epsq <= 0` turns off this function and `EPSP=scf.eps` is always used.

`scf.epsq >= 1` is invalid

`scf.maxit` (30) (POLAR only) Maximum number of iterations to calculate the induced dipoles. Applies both to the simulation and virtual volume change. If not enough to reach the predefined accuracy, it is doubled with verbose output on. (To use a constant number of iterations, use negative `scf.eps`)

`scf.omega` (POLAR only) The mixing iteration parameter for iterations of the self-consistent field during simulation. Can be also given in % as option `-^`. (Called `omegap` for  $V < 2.6f$ ).

`scf.omegax` (0.95) The mixing iteration parameter for iterations of the self-consistent field for virtual volume change. The optimum values are slightly below unity.

`shear` For measuring the shear viscosity: Only if compiled with `#define SHEAR`. See Sect. 15.11.

`shift[*]` The whole configuration is shifted by this 3D vector (before the first MD step). This vector is then set to zero so that no shift is performed in the next data set (unless specified again). The maximum allowed shift in periodic b.c. is  $\pm L$  (no check!).

SI (1="yes") 0="no" Energies (e.g., potential energy E<sub>pot</sub> and internal energy E<sub>int</sub>), are given in kcal/mol. Applies to both statistics and convergence profile

1="yes" Energies are given in J/mol. Applies to both statistics and convergence profile

WARNING: do not change SI during the run!

sort (0="none") DEPRECATED: use **key** instead.

If one of 1="x", 2="y", 3="z" is specified, the molecules (of each species separately) are sorted according to increasing values of given coordinate of the center-of-mass. If one of 1="-x", 2="-y", 3="-z" is specified, the sort order is decreasing. Sorting is performed once before writing a cfg-file (or asc), then **sort=0** is set. May be useful, e.g., for selecting some molecules.

The following forces are active in the SLAB version only: (in version 2.6h called center.\*)

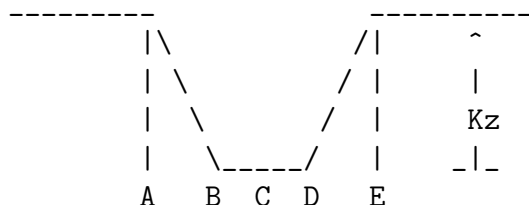
**slab.n[\*]** (0) SLAB: Number of molecules selected for adding forces in the z-direction.

**slab.Kz[\*]** (0) SLAB: Harmonic potential constant in the z-direction

**slab.z[\*]** (0) SLAB: The center of the force, with respect to the box center in the z-direction ( $L[2]/2$ )

**slab.z0[\*]** (0) SLAB only: The added energy term is zero for  $z \in [z-z_0, z+z_0]$ . (NOTE: in older versions, this was called **slab.r0**)

**slab.z1[\*]** (0) With SLAB=1 only (set **#define SLAB 1** in **simopt.h**):  
A 'bias function' is added:



This function (actually quadratically smoothed) is selected by **slab.z1!=0**, where  $|BC|=|CD|=\text{slab.z0}$  and  $|AC|=|CE|=\text{slab.z1}$ , and **slab.Kz** = height of the outside part from bottom (bottom=potential 0).

Note: with **slab.Kz<0**, and a slab placed at B..D, there are molecules with a higher probability  $\propto \exp(-\text{slab.Kz}/RT)$  outside the slab. Note that the meaning of **slab.Kz** differs from the standard (U-shape) case for **slab.z1=0**.

**slab.ns[\*]** (0) SLAB: Affected sites are: **slab.ns[\*]=0** all, **slab.ns[\*]>0** only first **slab.ns[\*]** sites, **slab.ns[\*]<0** only last  $|\text{slab.ns[*]}|$  sites. Example:

```

slab.n[0]=800      slab.n[1]=200
slab.Kz[0]=100     slab.Kz[1]=100
slab.z[0]=20       slab.z[1]=-20
slab.z0[0]=10      slab.z0[1]=2
                   slab.ns[1]=2

```

Molecules 1..800 are kept in a slab centered at  $z=L_z/2+20$  and  $20=10+10$  wide. First 2 sites of molecules 801..1000 are kept in a slab centered at  $z=L_z/2-20$  and 2 wide.

**slab.T** (-1) Separate thermostat for particles with  $z$ -coordinates in the range  $[\text{slab.Tz0}*L_z, \text{slab.Tz1}*L_z]$ . Requires Verlet/SHAKE and Andersen/Maxwell thermostat (incl. center-of-mass versions). For instance, to partly melt a crystal in a slab, you may try  $\text{slab.T} \approx 2T$  and  $\text{tau.T}$  in the range 0.1 to 1.

CAVEAT: Not fully implemented with the Berendsen thermostat. The algorithm just changes the velocity scaling factors in both slabs using the global temperature (it does not determine the temperatures separately in both slabs). In turn, if the heat conduction is too slow (compared to  $\text{tau.T}$ ), the difference of both temperatures grows to more than  $T$  vs.  $\text{slab.T}$ . Nevertheless, even this partial algorithm may be useful. For instance, to melt a crystal in a slab, you may try  $\text{tau.T} \approx 2T$  and  $\text{tau.T}=1$  or more (depending on the system size) and watch the melting process.

**slab.Tz0** (0.25)

**slab.Tz1** (0.75) Defines range of  $z$ -coordinates (in the units of  $L_z$ ) to apply a separate thermostat (see **slab.T**).

**stop** (0) Stop simulation if the time since (re)start, i.e., last **init** $\geq 2$ , exceeds **stop** ps. Note that **nomax** applies to the number of measurement, i.e., also last **init** $=1$ . If **init** $=1$  is not used and the cycle length does not change (see **h** and **noint**), then **nomax** and **stop** $=\text{nomax}*\text{noint}*h$  are equivalent.

**T** (300) Temperature for the thermostat in K.  $\text{tau.T}$  is the correlation time to keep the temperature constant

$T$  is also the approximate initial kinetic temperature (**init** $\geq 3$ ).

**tau.E** (0) The correlation time to keep the energy (Hamiltonian, extended Hamiltonian, or the enthalpy) constant, in ps.  $1/\text{tau.E}$  is the friction coefficient. **tau.E** $=0$  turns off the function (infinite correlation time or zero friction coefficient). Cannot be combined with any thermostat. Implemented via velocity rescaling once a cycle.

**tau.i** (0) See **tau.R**.

**tau.P** (0) Time constant for a barostat. If **thermostat** $="NPT"$ , this is the approximated oscillation time of the MTK-style barostat, otherwise the exponential correlation time of the Berendsen (friction)-style barostat (see Sect. 12.2). If **tau.P** is combined with **tau.rho**, **tau.P** is set to zero. The



Berendsen-like barostat uses the ideal-gas bulk modulus unless `bulkmodulus` is set (see there).

`tau.P=0` Isochoric ensemble.

`tau.P>0` Box is rescaled before every step, the scaling factor is calculated at the end of the preceding cycle and is kept the same for all steps in the cycle (new version, should be more stable, though biased)

`tau.P<0` Box is rescaled at the end of every cycle using the current value of virial pressure (this is the old version)

If `dV=0` (default), the virial pressure calculated directly for pair forces is used. However, for nonzero `dV` the pressure calculated by virtual volume change is used instead. See variable `dV` for details.

With `PRESSURETENSOR=3` (or more) and appropriate value of `rescale`), applies to diagonal components of the pressure tensor independently in each coordinate (suitable for crystals). Virtual volume/area is not supported.

`tau.R (0)` Support for adjusting atom size according to defined density (pressure in NVT). Works much like NPT, but instead of box size, parameter `RvdW` ( $R_{\min}$ ) of site type `tau.i` is changed, where `tau.i` is the number (counted from 0) of the site in the table `Lennard-Jones` or similar in the ble-file; it is NOT the site number printed (which is copied from the par-file). `tau.R` should be quite long, at least 100 for liquid. No cross-term in mixtures nor 1–4 terms are changed, so that this feature is not good for mixtures.

The current value of `RvdW` is stored in the cfg-file for possible restart, and replaces the value from the force field (ble-file) only when nonzero `tau.R` is specified; however, if there is zero in the cfg-file, the value from the ble-file is used. For `tau.R=0` the value possibly stored in the cfg-file is ignored, and `RvdW=0` is set for subsequent write.

`tau.rho (0)` The correlation time to reach the desired density. The default `tau.rho=0` means in fact infinity, i.e., the box size (density) does not change. Time needed to change density `initrho` to `rho` is  $\text{abs}(\ln(\rho/\text{initrho})) * \text{tau.rho}$ . The number of cycles needed is then  $\text{no}=1+\text{abs}(\ln(\rho/\text{initrho})) * \text{tau.rho}/(\text{noint} * h)$ ; this formula can be directly written to input data.

`tau.T (0)` The correlation time for the thermostat. See `thermostat` and see Sect. 12.1. Recommended values are 0.2–5 ps, in most cases 0.5–1. For simulated annealing (minimizing energy as  $T \rightarrow 0$ ), much longer times are needed to cool slowly to escape from local minima. However, close to the minimum, shorter times (comparable to the periods of typical motions) are more efficient; e.g., for water ice, the fastest relaxation to zero is obtained with `tau.T=0.12`.

For extreme rates of cooling or heating using friction thermostat, the friction term is bounded by  $\pm 1/\text{tau.T}$ . Therefore `T=0` can be safely used.

`thermostat` The selection of thermostat. See Sect. 12.1.

- 0 "none" No thermostat—the MD NVE ensemble.
  - 1 "friction"
  - 1 "Berendsen" Simple friction thermostat, see variables `tau.T` and `T`.
  - 2 "Nose" The Nose canonical ensemble.
  - 3 "Andersen" Velocities of randomly chosen atoms are replaced by those drawn from the Maxwell-Boltzmann distribution. The mean time of updating the same atom is `tau.T`. Not good with constrained dynamics.
  - 4 "Maxwell" Velocities of all atoms are replaced by those drawn from the Maxwell-Boltzmann distribution in the regular period of `tau.T`. Not good with constrained dynamics.
  - 5 "AndersenCM" Velocities of randomly chosen molecules (their centers of mass) are replaced by those drawn from the Maxwell-Boltzmann distribution. The mean time of updating the same atom is `tau.T`. May be used with constrained dynamics. Note that the actual relaxation time is longer because energy must be equipartioned with internal degrees of freedom which may be slow. Not suitable for fast initial cooling/heating.
  - 6 "MaxwellCM" As above, in regular period of `tau.T`.
  - 7 "Langevin" Langevin thermostat for atoms. Not good with constrained dynamics. Perturbs the Gear predictor, therefore only long enough `tau.T` are acceptable.
  - 8 "LangevinCM" Langevin thermostat for molecules (centers of mass). Can be used with constrained dynamics. Perturbs the Gear predictor, therefore only long enough `tau.T` are acceptable.
  - 11 "BerendsenCM" The friction thermostat applies for the translational motions of the centers of mass only.
  - 12 "BerendsenCMIN" The friction thermostat applies for the intramolecular (vibrational+rotational) motions only.
  - 13 "frictions" The translational and intramolecular+rotational degrees of freedom are thermostated separately. See also `T_tr_in`.
  - 22 "NPT" NPT ensemble: thermostat + barostat [32], via velocity predictor (see Nose) and box predictor (for constraints). See Sect. 12.2
- `T_tr_in` (1) Applies to `thermostat=13="frictions"` only. The translational and intramolecular+rotational temperatures to be kept constant are different with ratio  $T_{tr}/T_{in}=T_{tr\_in}$ . The total kinetic temperature is kept to `T`.
- `xs.mode` (0) Only if compiled with `#define XSECTION`: method of calculating the cross sections.
- `xs.mode=0` Uses 32 sample projection directions for each measurement. The 32 vectors correspond to vertices of regular dodecahedron and inscribed icosahedron with weights giving the Gaussian integration formula (see Strout: Multidimensional Integrals). The orientation of the polyhedron is random.

**xs.mode=1** One random projection direction per measurement.

**xs.mode=2** Three directions measured separately: (x+y),(x-y), and z. Suitable for testing possible orientational order caused by shear stress.

**xs.mode=3** Three directions measured separately: x,y,z. Suitable for monitoring melting a crystal and similar.

**xs.mode=4** Three sample orthogonal projection directions for each measurement, rotated by a random orientational matrix as in **xs.mode=0**

Note that modes 0,1,4 give the angle-averaged cross-section while modes 2,3 give cross-sections in specified directions.

**xs.grid (10)** Grid for calculating the cross-section, in number of points per 1Å. The method places the molecule on a rectangular mesh, projects all atoms and counts the number of mesh points. Note that there is a limit for the size of this mesh (see **xs.sizelimit**), the default is enough with **xs.grid=10** for most cases but choosing a finer mesh could cause exceeding this limit.

**xs.freq (0)** How often to measure the cross section, in cycles (**1cycle=no**int steps). (**xs.freq=1** is thus the same frequency of measurements as for most other quantities as internal energy, pressure, rdf.) Negative **xs.freq** means also printing the value(s) of cross section.

**xs.Rscale (2<sup>-1/6</sup>)** For cross section measurements, the atoms are represented by spheres of radii given by **xs.Rscale\*(RvdW+xs.Rvdw)**, where **RvdW** is the van der Waals radius of the atom being measured and **xs.RvdW** of the testing particle in a scattering experiment. The default is the same as using half Lennard-Jones sigma. (NOTE: old name = **xs.scaleR**).

**xs.RvdW (1.4)** Van der Waals radius of the testing particle in a scattering experiment. 1.4 corresponds to Helium.

**xs.cp (0)** **xs.cp >= 0** Cross section of molecule # **xs.cp** is recorded in convergence profile. **xs.cp=0** is recommended for e.g. protein in water: only cross section of protein will be recorded.

**xs.cp = -1** Averaged cross section over all molecules recorded in convergence profile. **xs.cp=-1** is recommended for bulk pure molecular fluid, e.g. when viscosity is measured (with **xs.mode=2**).

**xs.ns (-1)** **xs.ns >= 2** Number of sites (from the 1st one) of the molecule used to calculate the cross section.

**xs.ns=-1** All sites will be used

**xs.sizelimit (67108864)** Max number of bytes that can be used for the cross-section grid. (This is the upper limit, only necessary memory is allocated dynamically.) If exceeded, a warning is printed and outer parts of the molecule are ignored; the parts which are farthest from the origin are more likely to be cut off. Thus, if this happens for one molecule in free boundary conditions (FREEBC) which contains parts which can be released and fly away (like water molecules), these water molecules will be simply ignored.

wall (structure) WALL (and GOLD) version: See Sect. 15.8.

## 9.2.6 Interactive and batch control

cook runs in both batch and interactive modes. To turn the interactive mode on use option `-s`. This also enables scrolling (if compiled so).

The ‘get data’ input module, the use of scrolling and error handling are the same as in program `blend` — see the manual for `blend`!

## 9.2.7 Interrupt

If not turned off by option `-i-1`, pressing `ctrl-C` (or signalling `kill -2` or `kill -INT`, on some systems also `kill -15` or `kill -TERM`) causes the program to be interrupted and you can select (in interactive mode and a serial version from keyboard, in PARALLEL version and/or batch mode by option `-i`):

- 0 = (r)esume: continue calculations Continue running as without interrupt.
- 1 = (i)nterrupt: finish cycle then save all and read next data In interactive mode just the running cycle will be finished and you will be prompted for other data. Not so useful for the batch mode because the data are read from a file.
- 2 = (.)stop: finish cycle then save all and stop Stop cook gracefully (both in interactive and batch modes). Default for the batch mode.
- 5 = (j)ump to stop: finish step then save all and stop (incomplete cycle!) Stop cook less gracefully. The configuration will be saved but measurements will not be recorded in proper intervals (one per a cycle). To be used if you want to start cook again with `init=2`.
- 1= (e)xit immediately (nothing saved!) Kill cook.
- 9 = (s)croll (then type ? for help) Enter scrolling; once prompt \$ appears, type a scroll command or ? for help.

NOTE: You can disable the `ctrl-C` handler totally by deleting `#define SIG` from `main.c`. See comments there!

# Chapter 10

## Parallelization

MPI support was removed at cook V 2.5a. Now there are two shared memory versions implemented using POSIX threads (“pthreads”).

### 10.1 Compiling

The version is determined by compile-time switch `PARALLEL` (`#define PARALLEL`) in `simopt.h`. Several options are possible (see below). The pthread library is linked by option `-lpthread`.

Script `configure.sh` takes care about all settings.

### 10.2 Running

The number of threads is given by environment variable `NSLOTS`; this can be overridden by option `-N`. See more below.

If cook is submitted using SGE `qsub`, the number of requested cores (“slots”) is specified by parameter `-pe shm NUMBER`; it is passed in the form of environment variable `NSLOTS` to the node. Example:

```
qsub -cwd -b y -q sq-8-16 -pe shm 2 ./cookewslcp0P2 polwater slab -t
```

### 10.3 Linked-cell list and Ewald parallelized

`#define PARALLEL 1`: This version is suitable for simulation of large periodic systems.

The Ewald  $k$ -space part is parallelized in the natural way by splitting the sums over charged sites into parallel threads.

The linked cell list method is parallelized by slabs (2D arrays of cells) in the x-direction. There are `No.cell[0]` such slabs; it should be an integer multiple of the number of threads, otherwise a bad load-balancing occurs.

If you are running cook on your own machine, it may be advantageous to use negative option `-\`. It will split the Ewald  $k$ -space part into given number of threads while the  $r$ -space sums into `No.cell[0]` threads (more than processors).

## 10.4 Ewald $k$ -space and $r$ -space running in parallel

`#define PARALLEL 2` : This version is suitable for smaller systems. It uses two processors, in one the Ewald  $k$ -space part is calculated and in the other and the other the  $r$ -space part (of Ewald plus Lennard-Jones). It is good to set the Ewald parameters (in the standard situation for `e1.test=-10` this means trying several values of `cutoff`) so that the time spent by both parts is the same, otherwise load-balancing will be lost. The  $r$ -space part may be either all-pair or linked-cell.

## 10.5 Pair sums for a single big molecule parallelized

WARNING: not implemented now

`#define PARALLEL 3` : This version is suitable especially for 1 big macromolecule in free boundary conditions (version FREEBC). All pairs in the pair-sum are considered and parallelized as in the following scheme:

```

-----
1-0

2-0 2-1
                                Thread 1
3-0 3-1 3-2

4-0 4-1 4-2 4-3
-----
5-0 5-1 5-2 5-3 5-4
                                Thread 2
6-0 6-1 6-2 6-3 6-4 6-5
-----
7-0 7-1 7-2 7-3 7-4 7-5 7-6  Thread 3
-----

```

The amount of "lines" is determined in such a way that the amount of work is approximately the same for each thread. The algorithm allocates a copy of forces (though not of the full length in all cases) for each thread.

BUG: measurements of radial distribution functions and dihedral angle distributions are not supported.

# Chapter 11

## How to set parameters

There are many parameters that control the accuracy and efficiency. Most of them have reasonable defaults that will work at least fairly well in most typical cases. But one is never watchful enough. You must be able to recognize that something is getting wrong, so do not skip this chapter completely!

### 11.1 Accuracy

From the technical point of view, there are four sources of inaccuracies in the generated configurations:

- Integration errors. They are controlled by the integration method used (option `-m`) and the timestep `h`.
- Cutoff errors. These (in periodic b.c.) consist of the site-site (Lennard-Jones) cutoff errors and electrostatic cutoff errors. For Ewald summation, the latter consist of the  $k$ -space and  $r$ -space errors.
- Constraint errors. These are caused by different phenomena and must be corrected to some low value.
- Inaccurate calculation of charge-charge interactions ( $r$ -space sums or CUTELST). Normally negligible.
- Inaccurate ensemble. E.g., serious problems may arise when the friction thermostat with short correlation time is used for small molecules (they will rotate faster than corresponds to the temperature).

From the physical point of view, there may be many reasons why things go wrong. A few typical reasons follow:

- Long correlation times of crossing large energy barriers. For instance a convergence to equilibrium between cis and trans conformations may be slow.

- Small coupling between different degrees of freedom, e.g., slow coupling between fast bond vibrations and translations/rotations. That is why constrained bonds may be more accurate/efficient than simulating a fully flexible model. Slow convergence of Nose thermostat is of similar nature.
- Finite size effects. Especially important for systems close to a critical point. Increasing system size not only slows down the simulation algorithm but also increases the correlation times.

The following quantities serve to observe inaccurate sampling of the phase space:

- Conservation of integrals of motion. The most important is the total energy (Hamiltonian), in free b.c. the angular momentum
- Translational and intramolecular (+rotational) temperatures (called Ttr and Tin) can reveal a possible wrong equipartition (e.g., the “flying ice cube problem” of the Berendsen thermostat)
- Errors of constraints
- (POLAR only) Errors in self-consistent field

### 11.1.1 Errors of constraints

The following quantities are available to observe the accuracy of the constraints:

`cerr.r1` Error of constraints of predictor; reported as “`constr err 1`” in statistics.

`cerr.r2` Error of constraints after integration step.

`cerr.r3` Error of constraints (of true configuration, i.e., after correcting constraints).

`cerr.v1` Error of velocity constraints of predictor; reported as “`v constr err 1`” in statistics.

`cerr.v2` Error of velocity constraints after integration step

`cerr.v3` Error of velocity constraints (of true configuration)

All these quantities are normalized to be dimensionless.

Generally, `cerr.r1` should be several times higher than `cerr.r2` which should be several times higher than `cerr.r3` and similarly for `cerr.v`’s (with the exception of `cerr.v1/cerr.v2` which is rather high anyway). If all `cerr.r`’s are comparable (ratios less than 2), the constraint dynamics is solved too inaccurately and `epsc` and/or `eps` should be decreased. If `cerr.r2/cerr.r3` or `cerr.v2/cerr.v3` is too high (say,  $> 100$ ), `epsc` is unnecessarily low and efficiency is lost.



### 11.1.2 Energy conservation

The total energy should be in principle constant. The exception is the friction thermostat (and friction-like isobaric ensemble and similar) which spoils in principle the energy conservation — even if you wish to use the friction thermostat, try once a while a short microcanonical run to check the energy conservation! But even in the true microcanonical ensemble, various inaccuracies cause both statistical fluctuations and systematic secular drift in the total energy (the Hamiltonian).

1. Integration errors cause cooling (energy decrease) for the 4-value Gear method for 2nd-order equations which is the case of `cook` with the default value of option `-m`. In other cases they cause heating.
2. Cutoff errors (i.e., small jumps or peaks in the potentials and forces) cause heating (energy increase)
3. Inaccurate constraint dynamics causes heating

The first reason usually is (and should be) most important.

For the Nose canonical ensemble the drift does not matter because it means only some rescaling. The drift, of course, must not be too high: the good criterion is not the drift during the entire (possibly very long) simulation but the drift during a typical correlation time (say, 1ps) which should be set according to the demanded accuracy. The same holds true for the statistical fluctuations of the total energy.

The situation is not so simple for the microcanonical ensemble where energy drift causes cooling or heating. To avoid this, set `tau.E` to a typical correlation time (at least 1ps) and energy `E` will be conserved; the difference of `E-<measured Etot>` as well as `sqrt(Var Etot)` (both quantities are measured and statistically analyzed) is then the equivalent measure of the quality of the simulations.

The problems are usually smaller for the Verlet integrator (with SHAKE for constraints) which is time reversible so that (if there are no other sources of errors) there is no energy drift. In addition, the Verlet integration can be treated as exact integration of a perturbed Hamiltonian so that the errors are bound. (To be exact, this holds true only for certain class of continuous potentials with derivatives. Typical potentials like Lennard-Jones with singularities for particle overlaps exhibit energy increase. This phenomenon is for typical timesteps usually negligible.)

### 11.1.3 Self-consistent field accuracy

Applies to POLAR version only. The reported errors, `selffield maxerr` (maximum found in the configuration) and `selffield stderr` (standard deviation) of induced dipole moments are in the program units (one program unit is 0.01175 Debye, or see `units.h` for details). See also the explanation of `scf.eps` variable and paper [3].

## 11.2 How to set Ewald parameters $\alpha$ and $\kappa$

Both  $\alpha = \text{el.alpha}$  and  $\kappa = \text{el.kappa}$  depend on the r-space cutoff (`cutoff`). It must not exceed half the box size  $L/2$  (to be precise, a tiny overflow is accepted; in addition `cutoff` may exceed half box if the linked cell list method is used, but this method becomes efficient for large systems so that using `cutoff > L/2` is not recommended. For small systems (say, number of sites  $< 2000$ ), half the box size is the optimum value. For larger systems a lower value is optimum. If you select `cutoff < 0` (this is the default), `cook` calculates a value that should work fairly well in typical cases (it uses the final density `rho` to calculate the final box size). But if you want to optimize the run, you should try several values of `cutoff` and measure the time.

### 11.2.1 Simple way

It follows from the formulas in Appendix 25 that the main term of the r-space error is  $\text{erfc}(\alpha \text{cutoff}) \propto \exp[-(\alpha \text{cutoff})^2]$  whereas the k-space error is  $\exp[-(\pi \kappa / \alpha)^2]$ . Let  $\epsilon$  be the desired accuracy. Then the estimated parameters are

$$\alpha = \frac{\sqrt{-\ln \epsilon}}{\text{cutoff}}, \quad \kappa = \frac{\alpha}{\pi} \sqrt{-\ln \epsilon}.$$

Usually  $\epsilon = \exp(-\pi^2) \approx 5.2 \times 10^{-5}$  is a reasonable choice. Then

$$\kappa = \alpha = \pi / \text{cutoff}$$

Example of code:

```
el.test=0           ! no automatic setting of alpha,kappa
el.alpha=pi/cutoff  ! simple estimate
el.kappa=el.alpha   ! simple estimate
```

### 11.2.2 More accurate way

As the default (parameter `el.test=-10`), `cook` implements approximate formulas [20] based on the assumption that charges in the system are distributed randomly. The value of the  $r$ -space `cutoff` is here a free parameter and `el.alpha` and `el.kappa` are calculated to satisfy the accuracy requirements given by errors `el.epsr` and `el.epsk`.

But be aware that *approximate* error formulas are used. They tend to overestimate the  $k$ -space cutoff errors for large systems without free charges (i.e., if all charges are parts of small electroneutral groups). In addition, the  $k$ -space cutoff errors are both positive and negative and do not cause any violation of energy conservation. Hence, `el.epsk` may be several times higher than `el.epsr`.

### 11.2.3 Most accurate way

Since only approximate formulas are used for the cutoff errors, I recommend to check them against the actual errors. This is done in a special module that is entered by specifying `el.test` different from 0 and -10. The following data are available:

`el.alpha` As in the main get data module; in addition, `el.alpha<=0` restores the original value valid before the Ewald testing module has been entered.

`el.kappa` As in the main get data module; in addition, `el.alpha<=0` restores the original value valid before the Ewald testing module has been entered.

`cutoff` As in the main get data module; must be positive.

`eps` As in the main get data module; the original value is restored automatically when the Ewald testing module is left.

`el.epsk` As in the main get data module

`el.epsr` As in the main get data module

`el.grid` As in the main get data module

`el.test` `el.test=0` Leaves Ewald testing module Ewaldtest

`el.test=1` Calculates forces and energy and, if reference is set, errors

`el.test=2` Calculates forces and energy and store them as a reference for evaluating errors in next steps, sets `el.test=1`

`el.test=-1` Calculates `el.alpha` and `el.kappa` from `cutoff` `el.epsr` `el.epsk` and performs `el.test=1`

`el.test=-2` Calculates `el.alpha` and `el.kappa` from `cutoff` `el.epsr` `el.epsk`, performs `el.test=2` and sets `el.test=-1`

`el.test=-3` Calculates `el.alpha` and `el.kappa` from `cutoff` `el.epsr` `el.epsk` and waits for further data

`el.test=-10` Calculates `el.alpha` and `el.kappa` from `cutoff` `el.epsr` `el.epsk`, works silently and quits Ewaldtest afterwards.

`el.minqq` As in the main get data module

`No.cell` As in the main get data module

`quit` `quit=1` quits the program immediately.

Testing should not be performed using the initial configuration which is very artificial — it should be at least a little bit condensed and equilibrated. An example follows:

```

eps=1e-9      ! sufficiently high accuracy for calculating the Lagrange
               ! multipliers (not to introduce other inaccuracies!)
epsc=1e-9     ! sufficiently high accuracy for calculating constraints
LJcutoff=6    ! cutoff in the module should not fall below cutoff
el.test=-3 ; ! switch to the Ewaldtest routine (step 7. of 'program flow'
               ! above) and select automatic setting of alpha and K
               !
el.epsr=1e-6 ! sufficiently high accuracy for real-space errors
el.epsk=1e-6 ! sufficiently high accuracy for k-space errors

```

```

;          ! end of data, values of alpha and K will be evaluated
          !
el.test=2; ! accurate forces and energies are evaluated and stored as
          ! "reference" benchmark values
          ! NOTE: el.test=-2 is the same as el.test=-3, then el.test=2
          !
el.alpha=0 ! the original values of alpha and kappa are recovered
el.kappa=0 ! (otherwise the newly set are kept)
          ! (eps is restored automatically when leaving Ewaldtest)
el.test=1; ! and with these {alpha,K} the forces, energy and the errors
          ! are evaluated. These are "cumulative errors" (over all
          ! charges in the system) and you thus should compare them
          ! to the estimates of the cumulative errors. This step
          ! may be repeated with different {alpha,K}.
          !
el.test=0; ! abandon Ewaldtest: the program continues by step 5. of
          ! 'program flow'
          ! (but note that here we changed eps,epsc,LJcutoff!)

```

## 11.3 How to set accuracy for constraint dynamics

The constraint dynamics comes in two versions, one using Lagrangian multipliers for calculating the constraint forces, and the SHAKE algorithm. The SHAKE algorithm is based on the second-order Verlet integration method while the multiplier method allows higher-order integrators (see option `-m`). The multiplier method requires a step correcting the constraints.

### 11.3.1 The SHAKE algorithm with Verlet integration

Several versions of SHAKE are available differing marginally in efficiency for different systems.

`undefined` Shake is not implemented (only Gear+Lagrangian constraint dynamics is available)

`#define SHAKE 1` Simple update in sweeps. This is the default in `configure.sh`

`#define SHAKE 2` Info on moved sites is kept and only bonds of sites that have moved in previous step are updated. May be slightly more efficient in some cases

`#define SHAKE -1`

`#define SHAKE -2` As above and a more complicated algorithm taking into account the angle between the old and new constraint is used. Some more tests are added.

The velocity algorithm comes in four compile-time flavors. The trajectory is the same and given by the Verlet algorithm with SHAKE. The velocity algorithm affects the

kinetic energy and kinetic part of the pressure tensor. It does not affect the .vlb file which always corresponds to VERLET=0 (shifted by  $h/2$ ). The changes in speed are marginal.

```
#define VERLET 0 The simplest, fastest, and least accurate  $O(h)$  version,  $v(t) = [r(t+h) - r(t)]/h$ . Essentially, velocity is shifted by  $h/2$  from positions. Correct averaged energy of the harmonic oscillator. Good enough with friction (Berendsen) thermostat. (Added in V2.4a)
```

```
#define VERLET 1 Compromised speed, velocity accurate to  $O(h^2)$ :  $v(t) = [r(t+h) - r(t-h)]/(2h)$ . Averaged energy of harmonic oscillator has error  $O(h^2)$ . (Added in V2.4a) This is equivalent to the velocity Verlet algorithm.
```

```
#define VERLET 2 Best energy conservation (exact for harmonic oscillator but with  $O(h^2)$  error), slowest,  $v(t)^2 = [r(t) - r(t-h)]/h \cdot [r(t+h) - r(t)]/h$ . (The off-diagonal components of the pressure tensor are approximated as average of both possible  $h/2$ -shifted terms).
```

```
#define VERLET 3 Compromised speed, velocity and energy conservation accurate to  $O(h^2)$  (slightly worse than for VERLET=1), but with correct averaged energy of harmonic oscillator. Energy and pressure tensor components are averages of both shifted values  $v(t) = [r(t+h) - r(t)]/h$  and  $v(t) = [r(t) - r(t-h)]/h$ . This is the recommended default.
```

SHAKE is controlled by two parameters, **omegac** and **epsc**. The first one is the relaxation (mixing iteration) parameter: **omegac=1** means direct iterations, **omegac>1** overrelaxation, and **omegac<1** underrelaxation (mixing with previous iteration). Recommended for typical molecules are values around **omegac=1.2**. ANCHOR version (see Sect. 15.9) may require **omegac<1**. Parameter **epsc** is the required relative accuracy of constrained bonds.

Since V2.4a, negative value of **omegac=-1.2** causes optimization, separately for each species. The optimization starts from the absolute value, but is not stored (is done again even if **init=0** is selected). Use option **-v&4** to get detailed protocol on the process of optimization.

The numbers of needed SHAKE iterations are recorded and statistically analyzed in variable **nitc** and reported as **corr constr it** separately for the 1st molecule and averaged rest.

### 11.3.2 Constraint forces by Lagrangian multipliers

The conjugate gradient method is used to calculate Lagrange multipliers and then the constraint forces. The algorithm is described in **constrd.c**. The method is controlled by one parameter, **eps**, which is the maximum relative (dimensionless) error. If **eps** is too high, you will get into troubles with correcting constraints (see Sect. 11.1.1).

Another guide to optimization is the number of iterations: **nit**, or "Lagr mult it", (the number of iterations of the conjugate gradient method needed to calculate the

Lagrange multipliers) should not exceed (too much) the number of bonds; if it does, `eps` is unnecessarily small (or there is something seriously wrong like bad cutoffs etc.).

### 11.3.3 Correcting constraints

Integration and cutoff errors and inaccurately calculated Lagrange multipliers cause errors in constraints. It is necessary to correct these errors, otherwise they cumulate and the algorithm explodes. The constraint errors are — and must be — corrected every integration step.

Two methods are available to correct the constraints.

**Scorrect** is a SHAKE-like iteration method. The inverse-mass weighting ensures that the total momentum is conserved (within machine precision). This method is selected by unset bit 1 in option `-c` (was `-c3` prior V2.4a). It is controlled by two parameters, `epsc` and `omegac`. `epsc` is the maximum relative error of the constraints and `omegac` is the relaxation parameter: `omegac=1` are pure SHAKE-like iterations. Usually, small overrelaxation (`omegac=1.1–1.2`) is more efficient. The number of iterations is recorded and printed (separately for the 1st molecule and average for the rest). **Scorrect** is more efficient than the following method for simple constraints provided that good values of the parameters are set. It may become inefficient for molecules with many small cycles and/or bond angle constraints.

**conjgrad** uses the same conjugate gradient method and the same (inverse-mass weighted) matrix that has been used for calculating Lagrange multipliers for the constraint dynamics, hence, the total momentum is conserved as above. The method is selected by bit 1 in option `-c` (`-c4` prior V2.4a). It works fairly well for both simple and complex constraints. It is controlled by one parameter `epsc` and it seems that the value `epsc=0.05` (default) is OK if the order of multiplier predictor is the same as of the method (see options `-m` and `-p`). However, it requires more memory than **Scorrect** above.

For as complex systems as proteins a pessimistic choice of `eps`, `epsc`, or `omegac` affects the efficiency only slightly because calculating the force field is most time consuming anyway. See the discussion on `cerr` above.

Note that for **conjgrad** the conjugate gradient method in the Lagrange formalism is called twice (once for length constraints and once for velocity constraints).

For **Scorrect**, the number of iterations depends on `omegac`. If it is too high, **conjgrad** should be used instead of **Scorrect**.

Summary:

- `eps` about  $1e-5..1e-6$  should work (for `-m4` and `-p4`)
- use **Scorrect**, `epsc=eps` and `omegac=1.2` for simple molecules
- use **conjgrad** and `epsc=0.05` for complex molecules (`omegac` is irrelevant)
- **conjgrad** requires more memory than **Scorrect**. If **Scorrect** is to be used also for complex molecules, it is recommended to adjust `omegac` to get the minimum

value of the number of iterations. (This is automated since V2.4a)

### 11.3.4 Dependants

A dependant is a massless site which is calculated from other sites (parents). The configuration of the parents must be rigid (constrained). Forces on dependants are distributed back to parents before constraint dynamics.

There are two types of dependants:

- M “Middle” dependants are a linear combination of the parents. The forces are distributed back using the same coefficients. Example: site M in the TIP4P water model.
- L (New in cook V2.5f): “Lone” dependants are based on three atoms, but they lie out of the plane defined by these three atoms. Example: site L in ST2 or NE6 water models.

There may be several mechanically separated dependants (of both types) in one molecule.

Data for the dependant mechanics are collected in table “dependant” in the ble-file. The “Middle” dependants are calculated by blend. For water models, utility `waterdep` is available.

#### Lone dependant mechanics

Let us denote the positions of the three parents as  $\vec{r}_a$ ,  $a = 1, 2, 3$ . The calculations need a local orthonormal coordinate system  $(\hat{x}, \hat{y}, \hat{z})$ , where vectors  $\hat{x}, \hat{y}$  are in the plane of the parents and  $\hat{z}$  is perpendicular:

$$\begin{aligned}\hat{x} &= \sum_a x_a \vec{r}_a & (\sum_a x_a = 0), \\ \hat{y} &= \sum_a y_a \vec{r}_a & (\sum_a y_a = 0), \\ \hat{z} &= \hat{x} \times \hat{y}.\end{aligned}$$

Constants  $x_a, y_a$  are defined in the ble-file, table “dependants”.

The dependant position is

$$\vec{r} = \sum_a w_a \vec{r}_a + w_z \hat{z} \quad (\sum_a w_a = 1) \quad (11.1)$$

where again  $w_a$  and  $w_z$  are defined in the table “dependants”.

To distribute force  $f$  on the dependant back to the parents, we first calculate its components in the local coordinate system,

$$f_x = \hat{x} \cdot \vec{f}, \quad f_y = \hat{y} \cdot \vec{f}, \quad f_z = \hat{z} \cdot \vec{f}.$$

The force on parent  $a$ ,  $a = 1, 2, 3$ , is

$$\vec{f}_a = w_a \vec{f} + (t_{x,a} f_x + t_{y,a} f_y) \hat{z}.$$

The ‘torque constants’  $t_{x,a}, t_{y,a}$  are again defined in the table “dependants”.

## Pressure tensor calculation

The force-calculating module also calculates the virial of force and optionally also components of the pressure tensor, see Sect. 15.6. For the “Lone” dependants, a correction must be made because of force redistribution,

$$V\Delta\vec{P} = \sum_a \vec{f}_a(\vec{r}_a - \vec{r}) = \sum_a (t_{x,a}f_x + t_{y,a}f_y)\hat{z}(\vec{r}_a - \vec{r}) - w_z\vec{f}\hat{z}$$

where we used (11.1) to simplify the expression. The virial of force correction is the trace,

$$V\text{Tr}\Delta\vec{P} = \sum_a (t_{x,a}f_x + t_{y,a}f_y)\hat{z} \cdot (\vec{r}_a - \vec{r}) - w_z f_z$$

Note that the above corrections are zero for the “Middle” dependants.

## 11.4 Site-site potential cutoff

Any site-site potential (but charge-charge), as typically the Lennard-Jones potential

$$u_{\text{LJ}}(r) = 4\epsilon \left[ (\sigma/r)^{12} - (\sigma/r)^6 \right] = \epsilon \left[ (R_{\text{vdW}}/r)^{12} - 2(R_{\text{vdW}}/r)^6 \right]$$

is approximated by:

$$u(r) = \begin{cases} u_{\text{LJ}}(r) & \text{for } r < C_1 \\ A(r^2 - C_2^2)^2 & \text{for } C_1 < r < C_2 \\ 0 & \text{for } C_2 < r \end{cases}$$

where  $C_1 = \text{C1}$  is calculated from  $\text{C2}=\text{LJcutoff}$  so that the forces are continuous. Approximately  $\text{C1}=0.775*\text{C2}$ . Cutoff corrections are calculated by numerical integration (hence, any potential can be easily implemented) using the assumption that  $g(r)=1$  for  $r>\text{C1}$  and are included in the final values of energy and pressure (not for FREEBC). Too short  $\text{LJcutoff}$  not only causes systematic errors in energy and pressure (because the cutoff corrections are only approximate) but also may violate energy conservation because the potential has jumps in derivatives of forces. Note that the correction in energy is  $\text{En.corr}/V$  and in pressure  $\text{En.corr}/V^2$ , where  $\text{En.corr}$  is a single correction value calculated.

About 3 LJ sigma for  $\text{LJcutoff}$  seems to be sufficient for liquid phase (but not for fluids close to the critical point!). This is the default:  $\text{LJcutoff}=-3$  in input data means that the real cutoff for each pair of interacting sites will be just 3 times their LJ sigma. For small systems  $\text{cutoff}=\text{LJcutoff}=\text{L}/2$  is OK. (L is the smallest box size).

## 11.5 The timestep

The errors (total energy drift and fluctuation, with caution also  $\text{cerr}$ 's) are proportional to higher powers of the timestep  $h$  (with the exception of the Verlet drift, see Sect. 11.1.2). If they do not scale with these higher powers once  $h$  is changed it



means that other errors (e.g.,  $r$ -space cutoff errors) spoil the energy and constraint conservation.

For Ewald, the  $r$ -space cutoff errors (see Sect. 11.2) should be comparable with the integration errors. Check it: an increase of `el.epsr` (or explicitly decrease `cutoff` or increase `el.alpha`) or an increase of `h` should worsen the conservation of energy while a decrease of `el.epsr` or decrease of `h` should not significantly improve it.

The optimum timestep size generally depends on the complexity of molecules and the temperature and pressure. Good `h` in normal conditions is around 0.002 ps if there are no explicit hydrogens but `h=0.001` if there are hydrogens; the latter pessimistic value is the default. Vibrating bonds (see option `-u`), especially for hydrogens, require `h=0.0005`.

## 11.6 Functions for $r$ -space Ewald sums

The following functions are needed to calculate the  $r$ -space part of Ewald sums:

$$\begin{aligned}\text{eru}(x) &= \alpha e(\alpha\sqrt{x}) \\ \text{erd}(x) &= -\alpha^3 e'(\alpha\sqrt{x})/(\alpha\sqrt{x})\end{aligned}$$

where

$$\begin{aligned}e(y) &= \frac{\text{erfc}(y)}{y} \\ e'(y) &= \frac{de(y)}{dy} \\ \text{erfc}(y) &= \frac{2}{\sqrt{\pi}} \int_y^\infty \exp(-t^2) dt\end{aligned}$$

They are approximated by hyperbolic splines (functions  $A + B/(x + C)$ , no jumps in the derivative). These simple functions are extremely fast to evaluate, however, large tables are required to reach a good accuracy. The accuracy is controlled by `el.grid` which means that unity in the argument (= squared distance) is divided into `el.grid` subintervals. Max error occurs at the minimum distance `el.minqq` (the value of this variable affects the reported accuracy only, you can call `eru` and `erd` for any distance from 0 to `cutoff`<sup>2</sup>). By default, `cook` has `el.minqq=1.0`. Since this minimum distance occurs normally for constrained bonds, the error in `eru/erd` in `el.minqq` is constant too and cannot affect energy conservation. A minimum intermolecular distance is longer, errors lower but more relevant.

If requested so by `#define COULOMB -2` (former EXACTERFC) in `simopt.h`, `cook` uses a more accurate `erfc` formula for 1–2 and 1–3 distances (and auxiliary charges in POLAR version) so that larger `el.minqq` applies. Also POLAR implies `COULOMB=-2`.

Low `el.grid` may include some systematic errors into calculations because of large errors for short distances (note that the  $r$ -space terms must be evaluated also for bonded atoms). High `el.grid` may slow down the calculations (consider the cache

memory!). The default 256 should be OK in most cases. In the 16-bit Turbo C/DOS version, lower values must be used because otherwise the table exceeds the 64KB limit.

For ambient TIP4P water, a typical error with the default `e1.grid=256` caused by omitting `COULOMB=-2` (e.g., in LINKCELL) is 10kPa in pressure and 1J/mol in energy per particle. These errors are constant (for the same trajectory). A trajectory is not affected because all imprecise forces are constrained and are removed.

# Chapter 12

## NVT and NPT ensembles

### 12.1 Constant temperature simulations

The thermostat is turned on by selecting variable `thermostat`. The value of `tau.T` is then the typical correlation time to keep the temperature constant, irrespective of the method.

There are three kinds of thermostats available, the friction (Berendsen) thermostats, the Nosé–Hoover canonical ensemble, and the Maxwell-Boltzmann thermostats.

Special effects are obtained by decoupled inter- and intramolecular friction thermostats, selected by non-zero `thermostat>10`.

#### 12.1.1 The friction thermostat

The friction method, attributed to Berendsen, is based on the idea of velocity rescaling. In its crudest version the velocities are rescaled in every step so that the desired kinetic temperature is recovered. It is rather suspicious because the trajectories are affected. It can be estimated that the kinetic temperature changes in one step by the order of  $\sqrt{h/\text{sqrt}(N)}$  and hence

1. the influence to individual trajectories is lower if  $N$  increases
2. this ‘primitive’ method may be used only with low-order integrator like Verlet Leap-frog otherwise the order of the method decreases.

Point 2. can be overcome by writing the (approximately) equivalent differential equations with a friction term:

$$\ddot{\vec{r}}_i = \frac{\vec{f}_i}{m_i} - \frac{1}{2\tau_T} \dot{\vec{r}}_i \ln \frac{T_{\text{kin}}}{T}$$

where

$$T_{\text{kin}} = \frac{1}{fkT} \sum_i m_i \dot{r}_i^2$$

and

$f$     number of degrees of freedom

$T$     desired temperature

$\tau_T = \text{tau.T}$  = typical time

For  $\text{tau.T}=\hbar$  the crude method is approximately recovered. In order not to spoil trajectories,  $\text{tau.T}$  should be comparable to the typical correlation time in the system and much longer than  $\hbar$ !

The above formulation can be used also for constraint dynamics.

Advantages of the friction thermostat:

- Exponential convergence, for reasonable choices of  $\text{tau.T}$  usually fast.
- Simplicity, suitable also for Verlet/SHAKE

Disadvantages:

- Artifacts for rotating molecules not sufficiently coupled to the rest of the system. In this case, the rotational degrees of freedom are systematically hotter. Thus, problems are expected for fluids of small and weakly interacting molecules (hydrocarbons) while large molecules and with care ( $\text{tau.T} \geq 1$ ) also water may be OK. Note: this affects also one molecule in free boundary conditions, however, the system sets the angular momentum to zero. These artifacts can be observed via variables  $T_{tr}$  and  $T_{in}$ , which should be the same if the system is well equipartitioned. (They may both differ from the kinetic temperature  $T_{kin}$  by not more than  $1/f$ , where  $f$  = number of degrees of freedom). In old versions,  $T_{tr}$  was called  $T_{kin \text{ inter}}$  or  $T_{mol}$ ,  $T_{in}$  was called  $T_{kin \text{ intra}}$ . In addition, the kinetic temperature was simply  $T$
- Does not generate the canonical ensemble. If  $\text{tau.T}$  is long enough, then some distribution with  $T = \langle T_{kin} \rangle$  is obtained, the fluctuations, however, differ from the canonical values (heat capacity cannot be obtained from fluctuations of internal energy). If  $\text{tau.T}$  is not long enough, the results are undefined.
- *The energy is no longer constant!* This strong test that our simulation is OK is lost.

I would like to stress that the above method is something different than the method used to *correct* energy drift introduced by the integrator (by using  $\text{tau.E}$ ) because errors of  $h^2$  (order of the method) are corrected. The velocities are rescaled by values that are of the order of the integration errors that occur anyway so that no additional errors are introduced. The energy conservation criterion is replaced by the difference between  $E$  and measured total energy but it is not lost.

The friction thermostat is selected by `thermostat=1` or `thermostat="friction"` or `thermostat="Berendsen"`.

### 12.1.2 Decoupled translational and intramolecular thermostats

The translational thermostat affects only motions of molecular centers-of-mass, the intramolecular thermostat affects only relative motions with respect to the center of mass (incl. rotations).

The decoupled thermostats should be used for smaller compact molecules with intramolecular motions weakly coupled with the rest of the system. Be aware that the improvement is to some extent only optical because the coupling remains weak. You will get the correct average kinetic temperature, but the distribution of intramolecular kinetic energy may be incorrect.

[[[To be checked again: Intermolecular degrees of freedom contain only translational moves of centers of mass BUT no correction (-1) for energy conservation if no Nose thermostat is used. This introduces certain inconsistency: The expectation value of the (global) kinetic temperature (which is used by the friction thermostat) differs from ‘intermolecular’ and ‘intramolecular’ temperatures by the factor of the order of 1/(degrees\_of\_freedom), though ‘intermolecular’ and ‘intramolecular’ temperatures should be the same. This error is inherent to approximating the true canonical ensemble by the friction-thermostat  $T=\text{const}$  ensemble. ]]]

### 12.1.3 The Nosé–Hoover canonical ensemble

The second approach by Nose [7, 33] adds one additional degree of freedom to the system in such a sophisticated way that (provided that the system is ergodic) the true canonical ensemble is obtained.

The equations of motion follow. This form is directly used in the program (only constraint dynamics has been added). It was obtained from [7] by using his ‘time transformation’ and  $\xi = \ln s$  instead of  $s$ , [8, 33].

$$\ddot{\vec{r}}_i = \frac{\vec{f}_i}{m_i} - \dot{\vec{r}}_i \dot{\xi} \quad (12.1)$$

$$\ddot{\xi} = \frac{1}{\tau_T^2} \left( \frac{T_{\text{kin}}}{T} - 1 \right) \quad (12.2)$$

Here,  $\xi$  is the additional degree of freedom. Note that  $f = 3N - 3$  for  $N$  atoms in periodic boundary conditions: there are  $3N + 1$  degrees of freedom ( $\xi$  included) and 4 constraints, 3 for conserved total momentum and 1 for the (generalized) Hamiltonian; for constraint dynamics the number of constraints has to be subtracted.

If  $f_i/m_i$  is omitted and the equations are linearized, we get a harmonic oscillator of the form (schematically)

$$\ddot{v} = -\frac{2}{\tau^2}(v - v_0)$$

I wrote “If  $f_i/m_i$  is omitted” and you may ask whether this is a good approximation. Unfortunately, it is—note that absolute and linear terms in the forces cannot change the oscillatory behavior and small varying positive/negative contributions behave like a

noise. The more complex and denser liquid, the higher coupling between the auxiliary degree of freedom  $s$  and the system is provided, and the faster convergence occurs.

The Nose method is unsuitable for fluids at low densities and for soft potentials like exp-6.

The typical convergence profile of the kinetic temperature in the Nose simulation is:



The typical length of one cycle is `tau.T`. The time to get a good canonical distribution is longer and depends on `tau.T` in a way which is not straightforward – the convergence profile should be observed (using [21.1](#)).

The convergence problems occur also when starting from unequilibrated state. The initial oscillations of `Tkin` may be as high as 1:10 and it takes a long time to damp them. It is recommended to use another thermostat first.

The Nose thermostat can be also used for calculating one molecule in gas phase using the `cookfree` version. (The molecule must not be too simple to guarantee ergodicity).

The Nosé-Hoover thermostat is selected by `thermostat=2` or `thermostat="Nose"`. The implementation is straightforward for the Gear integrator. With the Verlet integrator, there is a problem that the standard Verlet method with SHAKE does not allow velocities in the right-hand side. MACSIMUS solves this problem by using predicted velocities (of both particles and  $\dot{\xi}$ )<sup>1</sup> The predictor length is selected by option `-pC`; the default is `C=2`, which gives time reversibility of  $\mathcal{O}(h^7)$ . See Sect. [27](#).

### 12.1.4 Maxwell-Boltzmann thermostat

The third method, suitable for systems without constraints (see option `-u`), is to replace once a while the velocities by new ones drawn from the Maxwell-Boltzmann distribution. It can be done in two ways: periodically (in time intervals given by `tau.T`, all velocities are replaced: `thermostat="Maxwell"`), or to update randomly chosen atoms (`thermostat="Andersen"`). Advantages

- Fast convergence.
- True canonical ensemble

Disadvantages:

- Problematic for constraints and Gear integration

---

<sup>1</sup>Other methods include iterations to obtain self-consistency in velocities and RATTLE, methods based on the Trotter decompositions[\[33\]](#).

- Spoils time development of trajectories
- The Andersen thermostat erases any energy conservation information

This thermostat is fully and correctly implemented for vibrating bonds and Verlet/SHAKE only. For weak coupling, (long `tau.T`), it can be used, with some loss of accuracy, for constrained bonds or Gear integration. It works worse for constrained bonds and Gear integration (Lagrangian dynamics).

The same effect as the the Maxwell thermostat, but executed just once, can be obtained by specifying `initvel`.

### 12.1.5 Langevin thermostat

Random force and friction are added to the equations of motion,

$$\ddot{x} = -\frac{\dot{x}}{\tau_T} + \sqrt{\frac{2kT}{m\tau_T}}X$$

where  $X$  is uncorrelated Gaussian random noise,  $\langle X(t)X(t') \rangle = \delta(t - t')$ ,  $x$  stands for one coordinate (of atom or center of mass), and  $m$  is the atom or molecule mass. The time constant  $\tau_T = \text{tau.T}$  in the data (in ps). The random force is implemented (with respect to time step  $h$ ) as normalized Gaussian random number divided by  $\sqrt{h}$ .

### 12.1.6 Which thermostat

For easy systems there is no difference between thermostats. Problems start with systems with fast and slow motions not coupled together: e.g., the full-atom model of methane. When the C-H potentials are harmonic springs, then the energy transfer between the fast C-H vibrations and other degrees of freedom is very slow (and the integration errors accumulate to violate the thermal equilibrium between different degrees of freedom). Then, the Maxwell-Boltzmann thermostat is recommended because it samples all degrees of freedom. In less problematic cases (e.g., constrained bonds but not angles) the decoupled thermostats may be a choice. In all these cases the Nose and simple friction thermostats will be inefficient. Sometimes it may help to start with the Maxwell-Boltzmann thermostat for fast equilibration and continue by, e.g., Nose or friction (the latter with a long `tau.T` to avoid its artifacts). In any case, it is strongly recommended to monitor the ‘internal’ and ‘external’ (translational) temperatures.

## 12.2 Constant pressure simulations

Nonzero value of `tau.P` allows for a variable box size. The barostat pressure to keep is given by variable `P`. See also `rescale` and `bulkmodulus`.

### 12.2.1 Friction (Berendsen) barostats

Unless `thermostat="NPT"`, the barostat is a simple Berendsen-style one. Pressure is calculated once every cycle (`noint` steps) and either the configuration is rescaled once after every cycle (for `tau.P<0`) or after every step by a proportionally smaller value (for `tau.P>0`).

Unless `bulkmodulus` is set, the friction term is derived using ideal-gas compressibility (more exactly with the number of degrees of freedom instead of the number of particles). In dense systems the compressibility is several times smaller (16 times for water, roughly 100 times for ambient-temperature solids), therefore the values of `tau.P` must be increased in the same proportion; `tau.P=10` to 1000 is a useful range. Alternatively, variable `bulkmodulus` can be set; `tau.P` is then a realistic relaxation time. This (realistic) `tau.P` should be several times longer than `tau.T` anyway. Setting `bulkmodulus` is recommended for solids at very low temperatures where the ideal gas approximation is very bad indeed; the bulk modulus of solids is of the order of 1e10 Pa.

### 12.2.2 MTK thermostat and barostat

The Martyna-Tobias-Klein (MTK) barostat [32], providing the true NPT ensemble corrected for momentum conservation in MD, is selected by `thermostat="NPT"`. The pressure tensor (virial pressure in the isotropic case) is needed and calculated at every step.

The periodic simulation box with all three box sides fluctuating independently is described by tensor  $\overleftrightarrow{\lambda} = \ln(\overleftrightarrow{L})$  (by components). We consider only diagonal components, i.e., a rectangular box, however, we keep the tensor notation for consistency.

$$\dot{\vec{r}}_i = \vec{v}_i + \dot{\overleftrightarrow{\lambda}} \cdot \vec{r}_i \quad (12.3)$$

$$\overleftrightarrow{L} = \exp(\overleftrightarrow{\lambda}), \text{ by components} \quad (12.4)$$

$$V = L_x L_y L_z \quad (12.5)$$

$$f_i = -\frac{\partial U}{\partial \vec{r}_i} \quad (12.6)$$

$$\dot{\vec{v}}_i = \frac{\vec{f}_i}{m_i} - \left[ \left( \dot{\xi} + \frac{\text{Tr} \dot{\overleftrightarrow{\lambda}}}{f} \right) \overleftrightarrow{I} + \dot{\overleftrightarrow{\lambda}} \right] \cdot \vec{v}_i \quad (12.7)$$

$$\overleftrightarrow{E}_{\text{kin}} = \frac{1}{2} \sum_i m_i \vec{v}_i \vec{v}_i, \quad E_{\text{kin}} = \text{Tr}(\overleftrightarrow{E}_{\text{kin}}) \quad (12.8)$$

$$\ddot{\xi} = \frac{2E_{\text{kin}} + \sum_{\alpha} M_{P,\alpha} \dot{\lambda}_{\alpha\alpha}^2 - (f + f_+)kT}{M_T} \quad (12.9)$$

$$\overleftrightarrow{P}_{\text{cfg}} = \frac{1}{V} \left( 2\overleftrightarrow{E}_{\text{kin}} + \sum_i \vec{r}_i \vec{f}_i + \dots \right) - P_{\text{corr}} \overleftrightarrow{I} \quad (12.10)$$

$$\ddot{\overleftrightarrow{\lambda}} = \frac{1}{M_P} \left[ V(\overleftrightarrow{P}_{\text{cfg}} - P \overleftrightarrow{I}) + \frac{2E_{\text{kin}}}{f} \overleftrightarrow{I} \right] - \dot{\xi} \dot{\overleftrightarrow{\lambda}} \quad (12.11)$$



where  $f_+$  is the number of independently fluctuating box sides and  $\vec{\vec{I}}$  is the unity tensor. Note also the different definitions of  $E_{\text{kin}} = \text{Tr}(\vec{\vec{E}}_{\text{kin}})$  and  $P_{\text{cfg}} = \text{Tr}(\vec{\vec{P}}_{\text{cfg}})/3$ .

The ellipsis in (12.10) stands for other terms (constraint forces, polarizable terms, etc.; see Sect. 15.6). The cutoff correction is assumed to be isotropic (see Sect. 11.4):

$$P_{\text{corr}} = -\frac{\partial U}{\partial V} = \frac{\text{En.corr}}{V^2}$$

The masses of the extended degrees of freedom are related to the thermostat and barostat correlation times (inverse circular frequencies in the ideal gas approximation) via

$$M_T = f k T \tau_T^2 \quad (12.12)$$

$$M_P = f_P (f + 3) k T \tau_P^2 \quad (12.13)$$

where  $f_P$  is the number of fluctuating (not constant) box sides. (Note the difference between  $f_P$  and  $f_+$ . E.g., if  $x$  and  $y$  fluctuate synchronously and  $z$  is independent, then  $f_P = 3$  and  $f_+ = 2$ .)

MACSIMUS uses the real coordinates throughout, consequently the Verlet-based implementation of Eq. (12.3) transforms into scaling by  $\vec{\vec{L}}(t+h)/\vec{\vec{L}}(t)$  after a step. Velocities in (12.3) are thus “with respect to the background”.<sup>2</sup>

These equations do contain a correction to the conserved total momentum [32] (term  $\text{Tr} \dot{\vec{\vec{\lambda}}}/f$  in (12.7) or  $3\dot{\vec{\vec{\lambda}}}/f$  in (12.18)) so that  $E_{\text{kin}}$  should not be rescaled by  $(f+3)/f$ , cf. (15.5).<sup>3</sup> A warning is printed on incorrect use.

The MACSIMUS implementation uses a time-reversible velocity predictor (see Sect. 27). This is not sufficient if there are constraints treated by SHAKE because the scaling (box size) is needed to know the bond lengths at time  $t+h$ . A sufficiently accurate workaround predicts the value of  $\lambda(t+h)$  as

$$\vec{\vec{\lambda}}(t+h) = \vec{\vec{\lambda}}(t) + h \dot{\vec{\vec{\lambda}}}(t+h/2)$$

with a second-order predictor for the difference

$$\dot{\vec{\vec{\lambda}}}^{\text{pred}}(t+h/2) = 2\dot{\vec{\vec{\lambda}}}(t-h/2) - \dot{\vec{\vec{\lambda}}}(t-3h/2).$$

The Verlet-based values of  $\vec{r}_{ij}$  to be constrained are then multiplied<sup>4</sup> by the predicted anisotropic scaling  $\exp[h\dot{\vec{\vec{\lambda}}}^{\text{pred}}(t+h/2)]$ . After  $\vec{\vec{\lambda}}(t+h)$  is calculated by Verlet, the

---

<sup>2</sup>A “cleaner” solution is to work with the scaled coordinates. Then instead of (12.14) and (12.18) we have

$$\ddot{\vec{\rho}}_i = \frac{\vec{f}_i}{L m_i} - \left[ \dot{\xi} + \left( 2 + \frac{3}{f} \right) \dot{\lambda} \right] \dot{\vec{\rho}}_i$$

with  $\vec{r}_i = L \vec{\rho}$  and  $\vec{v}_i = L \dot{\vec{\rho}}$ .

<sup>3</sup>In the case of this correction added anyway,  $3/f$  in (12.18) would be replaced by  $6/f$ , so in fact the correction would be applied twice.

<sup>4</sup>Equivalently, the target bond lengths may be rescaled by  $1/\exp[h\dot{\vec{\vec{\lambda}}}^{\text{pred}}(t+h/2)]$

configuration is rescaled by  $\exp[\vec{\lambda}(t+h) - \vec{\lambda}(t)]$ . The constraints are a subject of error of the order of  $h^2 \ddot{\lambda}(t) \propto (h/\tau_P)^2$ .

General independent scaling of all box sides is selected by `rescale="XYZ"`, scaling of  $x$  and  $y$  (keeping  $L_z$  constant) by `rescale="XY"`, etc.

If a synchronous scaling, i.e., constant ratio  $\lambda_{xx} : \lambda_{yy}$  or  $\lambda_{xx} : \lambda_{yy} : \lambda_{zz}$ , is requested (by `rescale="XX"` or `rescale="XXX"`), the value of  $f_+$  changes to 2 or 1, and the second derivative in (12.11) is modified by

$$\ddot{\lambda}_{xx} = \ddot{\lambda}_{yy} := \frac{\ddot{\lambda}_{xx} + \ddot{\lambda}_{yy}}{2} \quad \text{or} \quad \ddot{\lambda}_{xx} = \ddot{\lambda}_{yy} = \ddot{\lambda}_{zz} := \frac{\ddot{\lambda}_{xx} + \ddot{\lambda}_{yy} + \ddot{\lambda}_{zz}}{3} = \frac{\text{Tr} \ddot{\vec{\lambda}}}{3},$$

respectively.

The conserved total energy is

$$E_{\text{tot}} = E_{\text{kin}} + \frac{M_T}{2} \xi^2 + \frac{M_P}{2} (\dot{\vec{\lambda}})^2 + U + PV + (f + f_+) kT \xi$$

It can be easily proven by taking  $\dot{E}_{\text{tot}}$ . Symbol  $(\dot{\vec{\lambda}})^2 = \dot{\lambda}_{xx}^2 + \dot{\lambda}_{yy}^2 + \dot{\lambda}_{zz}^2$ .

For NPT in a cubic box there is simply  $L = \exp(\lambda)$ ,  $\lambda = \text{Tr} \vec{\lambda}$ . This is selected by `rescale="xyz"`. This option differs from `rescale="XXX"` because the isotropic pressure (not pressure tensor) is used. This is faster (if `PRESSURETENSOR=3` is not configured), however, for cutoff electrostatic less accurate. Real positions are  $\vec{r}_i = L \vec{\rho}$ , velocities with respect to the “background” are  $\vec{v}_i = L \dot{\vec{\rho}}$  (i.e., not dragged with if the whole box shrinks/swells), real velocities including the “drag” caused by fluctuating box are  $\vec{v}_i$ . Number of degrees of freedom =  $f$ ; for simple fluid  $f = 3N - 3$  (minus 3 for periodic b.c.).

For isotropic fluctuations we have (in the algorithm order):

$$\dot{\vec{r}}_i = \vec{v}_i + \dot{\vec{\lambda}} \vec{r}_i \quad (12.14)$$

$$L = \exp(\lambda) \quad (12.15)$$

$$V = L^3 \quad (12.16)$$

$$f_i = - \frac{\partial U}{\partial \vec{r}_i} \quad (12.17)$$

$$\dot{\vec{v}}_i = \frac{\vec{f}_i}{m_i} - \left[ \dot{\xi} + \left( 1 + \frac{3}{f} \right) \dot{\lambda} \right] \vec{v}_i \quad (12.18)$$

$$E_{\text{kin}} = \frac{1}{2} \sum_i m_i \vec{v}_i^2 \quad (12.19)$$

$$\ddot{\xi} = \frac{2E_{\text{kin}} + 3M_P \dot{\lambda}^2 - (f+1)kT}{M_T} \quad (12.20)$$

$$P_{\text{cfg}} = \frac{1}{3V} \left( 2E_{\text{kin}} + \sum_i \vec{r}_i \cdot \vec{f}_i + \dots \right) - P_{\text{corr}} \quad (12.21)$$

$$\ddot{\lambda} = \frac{1}{M_P} \left[ V(P_{\text{cfg}} - P) + \frac{2E_{\text{kin}}}{f} \right] - \dot{\xi} \dot{\lambda} \quad (12.22)$$

# Chapter 13

## Initial configuration

There are several ways how to generate initial configurations. Small molecules can be placed in vertices of a lattice, bigger either to enlarged box (to be compressed later) or shot to an (enlarged) lattice at random. A small lattice with a periodic configuration can be repeated several times. A problem of immersing a big molecule into a liquid (of small molecules) can be solved by removing overlapping small molecules.

### 13.1 Random-shooting algorithm

If `init=3` or `init="random"` is specified in input data, the initial configuration is generated by the following algorithm suitable especially for small or moderately large molecules.

1. For given initial density (variable `initrho`), an attempt is made to fill the simulation box by randomly distributed and rotated molecules. Molecules are inserted one-by-one and there is an energy limit (variable `Emax`); if it is exceeded, the insertion of the molecule is rejected.
2. If step 1. is not successful (i.e., if the insertion probability exceeds the limit of `pins`), the density is lowered and step 1. is repeated. Ignored if `pins=0`.
3. If `pins<0` only: If 1. is successful, however, the system is too dilute (i.e., the insertion probability of step 1. has been greater than `sqrt(|pins|)`), step 1. is repeated with a higher density.
4. Velocities are assigned (approximately according to temperature `T`) and both velocity and length constraints are adjusted.

A configuration obtained by this algorithm can be dilute and should be concentrated. The final density is named `rho` (and is in  $\text{kg/m}^3$ ). The speed of changing the density is given by typical time `tau.rho` (in ps). Recommended values are 0.1–1 ps; too low values may cause a crash. If you meet problems, we recommended to run first several steps with `tau.rho=0` (this means in fact infinity, i.e., changing density switched off). Use the friction thermostat to cool the heat of compression. After reaching the required density, the system must be, of course, equilibrated.

## 13.2 Crystal initial configuration

Alternatively, small molecules may be placed at vertices of a regular lattice which is selected by `init=5`. The lattice type is defined by variable `pins` either explicitly (values 1,2,3) or left to program's decision (`pins=0`) to obtain lattice with minimum number of vacancies. There is no overlap check and it is user's responsibility to choose, if necessary, density low enough not to have overlapping molecules; see also option `-q` which offers possibility to turn off random rotation of molecules.

If `pins>3` then the configuration is obtained by repeating an elementary cubic cell containing a prepared configuration of `pins` molecules; see the explanation of `pins` for details.

## 13.3 Immersing a large solute into solvent

Rapid changes in density can destroy a structure of a big macromolecule. For building an initial configuration consisting of (for example) a big protein in water, there are two possibilities:

### METHOD 1

1. Prepare the `.ble`-file with the macromolecule as the first molecular parameter given to `blend`. Thus, it has index 0.
2. Select `cook` option `-j1` to freeze the atom positions of the protein.
3. Run the initializer `init=3` with `initrho=rho` (or only slightly lower if necessary) and very high `Emax` and `pins=0` (or so low so that the initializer succeeds);
4. Use the friction thermostat (set `tau.T` and `T`). You may have to specify a rather low timestep `h` if the initial `Emax` was very high. In normal cases, `tau.T` should not be (much) lower than the typical relaxation time in water which is about 1ps; you may have to use longer values for complex proteins with cavities.
5. Run `cook` again without option `-j`.

### METHOD 2

1. Prepare the `.ble`-file with the macromolecule as the first molecular parameter given to `blend`. Thus, it has index 0, and the solvent has index 1.
2. Prepare independently a configuration (may be small, but normally at least about 50 molecules) of the solvent (water) and export it using `-l` option. Your protein will be immersed into this configuration (possibly periodically repeated) so that you must precalculate the appropriate size of the system to allow this.
3. Rename the ascii dump to `cfgns.pins`; e.g., `cook/cfg3.50` contains such configuration of 50 TIPS water molecules.

4. Run `cook` with option `-j1` to freeze the positions of the protein.
5. Run the initializer `init=5` with `pins` equal the number of molecules from step 1. `Emax` should be quite high.
6. Some of the solvent molecules are removed. Record the changed numbers of molecules and edit the `.def` file accordingly.
7. Run `cook` again without option `-j`. As in METHOD 2, equilibrate, preferably first with the friction thermostat.

# Chapter 14

## Measurements

### 14.1 Units of measurements

I'm sorry, but three systems of units of measurements are used by program `cook`:

- File `sysname.ble` defining the force field uses kcal/mol as the energy unit and Å as the length unit because these units are used by CHARMM. See the documentation of `blend` for details.
- Internally, `cook` uses Ångström to measure lengths, ps to measure time, Kelvin to measure temperature and  $k \cdot K$  (where  $k$  is the Boltzmann constant) to measure energy (loosely, energy is measured in Kelvins). See `units.h` for details.
- `cook` communicates with the user in SI units: density in kg/m<sup>3</sup>, pressure in Pa, energy in J/mol; however, energy is often given in Kelvins (I have a better feeling on Boltzmann probabilities if the Boltzmann constant is defined as unity).

### 14.2 Convergence profile

The convergence profiles are important for observing whether the system is in equilibrium. Many numerical or physical problems are also discernible from convergence profiles. Note that the term “convergence profile” here means just time dependence of quantities, not running averages of any kind.

`cook` records the following five quantities every cycle:

**Etot** Total energy (Hamiltonian). It should differ only slightly from a constant. If `tau.E` is zero, it may have a secular drift and fluctuations, if `tau.E` is nonzero, it should have (after reaching equilibrium value) small fluctuations. The total energy should never have jumps (followed by relaxation if `tau.E` is nonzero). See Sect. 11.1.2, for details.

**Tkin** Instantaneous (kinetic) temperature. In the microcanonical ensemble it should just fluctuates around certain equilibrium value. In the Nose ensemble it consists

of harmonic oscillations with the period given by `tau.T`, perturbed by statistical noise. The coupling between the Nose harmonic oscillations and the noise should be sufficient, otherwise the canonical ensemble is not accurately sampled and quantities like specific heat (given by energy fluctuations) are not accurate. In older versions, it was called only `T`.

**Epot** Total potential energy. Similar observations like for **Tkin** apply.

**Eintra** Intramolecular non-bonded energy. Sometimes, coupling between the intramolecular degrees of freedom (namely the dihedral torsions) and intermolecular ones is weak. This is seen from long relaxation or correlation times of **Eintra**.

**P** Pressure (if not `tau.P`) or density (if `tau.P`) Similar observations like for **Tkin** apply.

**more...** The format has been updated. Further items depend on compilation `#defines` (in file `simopt.h`). Run program `showcp` to get more info!

## 14.3 Analysis of statistical errors

Let  $Q$  be an arbitrary quantity measured in the simulations and  $Q_i, i = 1..n$ , its values obtained in consecutive measurements. Then the expected error of the average  $\langle Q \rangle$  is given (under some assumptions) by the formula

$$\delta Q = \sqrt{\text{Var}Q(1 + 2\tau)}$$

where the (discrete) correlation time is defined by:

$$\tau = \sum_{t=1}^{\infty} c_t$$

and the autocorrelation coefficients are

$$c_t = \frac{\text{Cov}(Q_{i+t}, Q_i)}{\text{Var}Q}, \quad \text{Cov}(Q_{i+t}, Q_i) = \langle Q_{i+t}Q_i \rangle - \langle Q_{i+t} \rangle \langle Q_i \rangle, \quad \text{Var}Q = \text{Cov}(Q_i, Q_i)$$

$c_t$  should go to zero and the sum for  $\tau$  should converge (if it does not or does slowly, another methods should be used). In practice, the sum for  $\tau$  is truncated at a certain  $t = t_{\max}$ ; if  $t_{\max}$  is too low,  $\tau$  is usually underestimated; if  $t = t_{\max}$  is too high then the inaccuracies in  $c_t$  may cause too large errors in  $\tau$ .

It is possible to extrapolate  $c_t$  to infinity, usually by  $c_t = A \exp(-Bt)$  (one bottleneck process or barrier) or  $c_t = At^{-3/2}$  (hydrodynamic tail) and thus to correct  $\tau$ .

Another way is to use blocking or sub-averages and the same method with only few first autocorrelation coefficients, or perhaps none (just blocking)m but it is better to include at least  $c_1$ .

To understand the output of `statics.c`, let us consider the following example produced by testing program `staticst.c`:

```

-----
r               No = 100000      range = <0.78591037,4.2648801> = 3.4789698
Mean = 2.50087886      Var = 0.232787747      StDev0 = 1.5257e-03
_t__c[t]_1+2tau__StDev_ _t__c[t]_1+2tau__StDev_ _t__c[t]_1+2tau__StDev_
  1 0.8020 2.604 2.5e-03  2 0.6440 3.892 3.0e-03  3 0.5170 4.926 3.4e-03
  4 0.4143 5.755 3.7e-03  5 0.3311 6.417 3.9e-03  6 0.2654 6.948 4.0e-03
* 1 0.7225 2.445 3.2e-03  2 0.4652 3.375 3.8e-03  3 0.2981 3.972 4.1e-03
* 1 0.5652 2.130 3.9e-03  2 0.2402 2.611 4.3e-03  3 0.1096 2.830 4.5e-03
* 1 0.3662 1.732 4.4e-03  2 0.0763 1.885 4.6e-03  3 0.0114 1.908 4.7e-03
-----

```

1+2tau and StDev that appear in the table use truncated sums up to the lags of t. The first line marked by \* refers to the analysis of data averaged in blocks of length 2, the second line marked by \* in blocks of length 4, etc. The result (rounded) is  $\langle r \rangle = 2.501 \pm 0.005$ , to be compared with the following exact results for the example:

```

<r> = 2.5
standard deviation of <r> = 4.564e-03
Var r = 25/108 = 0.2315

```

NOTE: in newer versions, a summary determined from maximum blocked StDev with  $c_1$  included is printed.

Try to run `staticst.c` several times for different  $n$ 's to see how the method works!

See also utility `autocorr`, [21.10](#).

## 14.4 Kinetic quantities from equilibrium molecular dynamics

### 14.4.1 Diffusivity

#### Green-Kubo

Diffusivity can be calculated from the velocity-velocity time autocorrelation function (more precisely the covariance) using the Green-Kubo formula

$$D_i = \int_0^\infty \langle \dot{x}_i^{\text{CM}}(0) \dot{x}_i^{\text{CM}}(t) \rangle, \quad (14.1)$$

where  $x_i^{\text{CM}}(t)$  is the x-coordinate of the center-of-mass of molecule  $i$ . MACSIMUS offers a direct, but (sometimes) costly, calculation of the covariance  $\langle \dot{x}_i^{\text{CM}}(0) \dot{x}_i^{\text{CM}}(t) \rangle$  by setting variable `lag.v` (time range in the units of `noint*h`), see also `lag.dim` (1 to record the sum over coordinates x,y,z or 3 to record the all separately) and `lag.nv` (number of molecules). By averaging these covariances (over all molecules of the same species and all three coordinates) and integrating them (use utility `runint`), the diffusivity can be obtained.

The program unit of diffusivity is  $\text{\AA}^2/\text{ps} = 10^{-8} \text{ m}^2/\text{s} = 10^{-4} \text{ cm}^2/\text{s}$ .



## Einstein

It is usually more efficient to calculate the diffusivity from recorded configurations using the Einstein relation. The mean square displacement

$$\text{msd}(t) = \frac{1}{6} \frac{\sum_i \left\{ m_i [\vec{r}_i(t) - \vec{r}_i(0)] \right\}^2}{\left( \sum_i m_i \right)^2} \stackrel{t \rightarrow \infty}{\equiv} \text{const} + D_i t, \quad (14.2)$$

where the sums are over all sites of one molecule.

We use the algorithm that follows the path of the particles (and can calculate  $D_i$  over overlapping boxes), see `p1b2diff` below. Thus we obtain the correct 3D autodiffusion coefficient even if the particles travel more than one half of the periodic simulation box.

## Finite-size effects

The diffusivity obtained by either method is subject of large finite-size errors [30]. These errors are caused by the periodic images traveling in the same direction. The MD diffusivity is thus underestimated. The corrected diffusivity is [30]

$$D = D_{\text{MD}} + \frac{2.873kT}{6\pi\eta L} \quad (14.3)$$

where  $\eta$  is the shear viscosity. This formula applies to cubic boxes  $L^3$  only.

Recently, we tested the above correction for argon and water [31]:

```

Ar
^^
EvdW=-0.2380684 kcal/mol, RvdW=1.910992 A
T=143.76 (T*=1.2)
rho=1344.2582 kg/m3 (rho*=0.8)

viscosity (Green-Kubo): eta=0.00017543 Pa.s~~~~~
D is in 1e-9 m^2/s
Dcorr = Dsim + 2.837*k*T / (6*pi*eta*L)

=====
N method tau/ps  Dsim  stderr Dcorr
-----
250  B  0.2      4.217 0.019 4.954
250  B  1        4.229 0.022 4.966
250  N  0.2      4.210 0.021 4.947
250  N  1        4.220 0.022 4.957
2000 B  0.2      4.560 0.012 4.928
2000 B  1        4.567 0.011 4.935
2000 N  0.2      4.568 0.013 4.936
2000 N  1        4.578 0.010 4.947
=====

SPCE water
T=298.15 K
viscosity (N=250): 0.00058(6) Pa.s
L=19.575161 A (N=250)
=====

2000: L=46.21296 A
250: L=23.10648 A
N=Nose+Gear(+ Lagrange constraint dynamics)
B=Berendsen(+Shake)

```

## 14.4.2 Conductivity

Partial molar conductivity is proportional to the diffusivity

$$\lambda_n = \frac{z_n^2 e^2}{kT} D_n = \frac{z_n^2 F^2}{RT} D_n \quad (14.4)$$

where  $z$  = charge in  $e$ ,  $F$  = Faraday constant,  $R$  = gas constant, and  $V_m$  = molar volume. Equivalently we may define the partial conductivity as

$$\kappa_n = \frac{z^2 e^2 N_n}{kTV} D_n = \frac{z^2 F^2 c_n}{RT} D_n \quad (14.5)$$

where  $N_n$  is the number of molecules of species  $n$  and  $c_n$  is the concentration (in mol/dm<sup>3</sup>). In program units ( $k = 1$ ) this formula becomes

$$\kappa_n = \frac{N_n D_n z_n^2}{TV}, \quad (14.6)$$

where the program unit of conductivity is 1.11265006e+10 S/m.

The Green-Kubo formula for the (total) conductivity is

$$\kappa = \frac{V}{kT} \int_0^\infty \langle J_x(0) \cdot J_x(t) \rangle \quad (14.7)$$

and the same in the y and z directions (three independent values). The current density  $\vec{J}$  is

$$\vec{J} = \frac{1}{V} \sum_i q_i \dot{\vec{r}}_i \quad (14.8)$$

Three covariances  $\langle J_a(0)J_a(t) \rangle$ ,  $a \in \{x, y, z\}$ , are calculated by cook for `lag.cond` set. In addition, a guide how to finish the calculations is printed.

For a successful calculation, `DT=h*noint` must be fine enough and the lag `lag.cond` long enough, which is best checked using graphs of covariances and their running integrals.

The corresponding Einstein relation (derived from the current, not current density) reads as

$$\text{mscd}(t) = \frac{1}{6} \left\{ \sum_i q_i [\vec{r}_i(t) - \vec{r}_i(0)] \right\}^2 \stackrel{t \rightarrow \infty}{\equiv} \text{const} + kTV\kappa t \quad (14.9)$$

Again, `plb2diff` can be used to calculate mscd and in turn the partial and total conductivities.

In the non-equilibrium molecular dynamics, the conductivity is given by

$$\kappa = \frac{J_z}{E_z}$$

where we assume that only the z-component of the electrostatic field,  $E_z = \mathbf{Eelst}[2]$  (in V/m), was set. Since the  $J_z = \mathbf{Jz}$  [A/m2] is reported in SI units,  $\kappa$  will be in S/m.

### 14.4.3 Viscosity

The Green-Kubo equation for viscosity is

$$\eta = \frac{V}{kT} \int_0^\infty \langle P_{ab}(t)P_{ab}(0) \rangle, \quad a \neq b \quad (14.10)$$

where the pressure tensor components are for pairwise interactions given by (15.2). Three covariances are calculated by cook for `lag.visc` set. `cook` prints a guide, too, and similar rules as for the conductivity apply. A support for off-diagonal pressure tensor components is needed (see Sect. 8.3).

Note: it is (probably) not possible to calculate viscosity from the Einstein relation obtained by the following “naive integration” of the Green-Kubo formula (see [23], p. 86+).

$$P_{ab} = \frac{1}{V} \left( \sum_{i=1}^N \frac{p_{i,a}p_{i,b}}{m_i} + \sum_{i=1}^N r_{i,a}f_{i,b} \right) = \frac{1}{V} \left( \sum_{i=1}^N \frac{p_{i,a}p_{i,b}}{m_i} + \sum_{i < j}^N r_{ij,a}f_{ij,b} \right)$$

$$2t\eta = \frac{V}{kT} \langle [L_{ab}(t) - L_{ab}(0)]^2 \rangle$$

where

$$L_{ab} = \frac{1}{V} \sum_i r_{i,a}p_{i,b}$$

## 14.5 Kinetic quantities from the Einstein relations

UPGRADE 6/00: conductivity and autodiffusion measurements by Einstein relation  
See `plb2diff.c` [22.22](#) and `sfdx.c`

NOTE: The Green-Kubo-based conductivity is usually more precise, but the grid has to be short enough, sometimes  $2 \cdot h$ , and the lag long enough, which may be a problem for one-particle diffusivities. The analysis may be rather subjective because a graph has to be analyzed. The Einstein version is very similar of the same lag is used. It becomes safer (less systematic error) but less accurate (larger statistical error) as the lag increases.

### 14.5.1 Requirements

- Periodic b.c.
- NVT or NPT ensemble (for NPT, see below)
- Stored configurations:
  - either in a plb-file (`simname.plb`, generated by `cook -y-1`)
  - or `simname.1,simname.2,...` (generated by `cook -r`)

### 14.5.2 Usage

Re-run cook:

- either with option `-f#` (to read `simname.plb`)
- or `-f-#` (to read `simname.1,...`), `#` stands for the stride

Input data are:

`init` First frame (stored configurations) read

`no` Last frame read (hint: use `plbinfo simname.plb` to get the max frame!)

Example:

```
cook* -f2 xxx
```

with `init=5` and `no=9` will read frames 5,7,9 from `xxx.plb`

WARNING: in versions older than about 9/2000, `-f#` denoted the 1st frame read, and `noint` was the stride.

### 14.5.3 Results

simname.sfd Protocol (is the screen if option **-s**)

simname.m.cp Mean square displacements of the center of mass:

COLUMN 1 mean square displacement for species 0:

$$\text{msd} = \frac{1}{6 N_0} \sum_n \left\{ \sum_i m_i [r_i(t) - r_i(0)]^2 \right\} / \left\{ \sum_i m_i \right\}$$

where  $m_i$  = mass of site  $i$ ,  $\sum_i$  is over all sites of species 0,  $\sum_n$  is over all molecules of species 0,  $N_0 = \sum_n 1$  = number of molecules of species 0

COLUMN 2 as above, species 1

...

COLUMN  $n_{\text{spec}}+1$  The sum over the whole simulation box.

$$\text{mdif} = - \frac{1}{6} \sum_i m_i [r_i(t) - r_i(0)]^2$$

Note that  $\text{mdif}=0$  but rounding errors because of momentum conservation!

simname.q.cp mean square charge displacement:

COLUMN 1 cumulative mean square charge displacement for species 0:

$$\text{mscd} = - \frac{1}{6} \sum_n \left\{ \sum_i q_i [r_i(t) - r_i(0)]^2 \right\}$$

where  $q_i$  = charge of site  $i$ ,  $\sum_i$  is over all sites of species 0,  $\sum_n$  is over all molecules of species 0,

NOTE:  $\text{mscd}$  is SUM while  $\text{msd}$  is AVERAGE (divided by  $N_0$ )

COLUMN 2 as above, species 1

...

COLUMN  $n_{\text{spec}}+1$  total mean square charge displacement:

$$\text{mscd} = - \frac{1}{6} \sum_i q_i [r_i(t) - r_i(0)]^2$$

where the sum is now over the whole simulation box. This is related to the conductivity, see below.

### 14.5.4 Analysis of results

See also `plb2diff` which makes time averages and automates the following algorithm.

## DIFFUSION COEFFICIENTS Run:

```
showcp -p SIMNAME.q.cp
```

and estimate the time derivatives

$$D = d \text{ msd}(t) / dt$$

Hints:

- Frames are separated by dt.plb\*# or dt.cfg\*# ps (# denotes the -f option, -f# or -f-#).
- Several first data should be skipped
- Estimate the slope from the plots or by linear regression of; the linear regression should use weight 1/t because the error of msd(t) is proportional to sqrt(t).
- And all this should be done for several blocks to have statistics!

D is the diffusion coefficient of given species (column 1 = species 0) in program units [A<sup>2</sup>/ps]; in usual units it holds:

- D\*1e-8 = diffusion coefficient in m<sup>2</sup>/s
- D\*1e-4 = diffusion coefficient in cm<sup>2</sup>/s

Make also sure that the last column in simname.q.cp (**mdif**) is small.

NOTES: 1st 2 columns of simname.q.cpa and the plots are incorrectly labeled as Etot and Tkin instead of 0 and 1. In addition, the last column in simname.q.cpa just repeats the 1st column, thus the “last column in simname.q.cp” is actually the second-last in simname.q.cpa.

## CONDUCTIVITY Run:

```
showcp -p SIMNAME.q.cp
```

and the LAST graph (named **cond**) should be approximately linearly increasing function of time, **cond(t)**; of course, it is not and averaging should be done over different runs or blocks of one run! The conductivity (per unite volume) in program units is:

$$\kappa = \frac{1}{TV} \frac{d\text{cond}(t)}{dt} \quad (14.11)$$

where  $T$  = average temperature in K and  $V$  is the volume of the simulation box. In MACSIMUS units:

```
kappa*111.26502 = conductivity in S/m  
kappa*1.1126502 = conductivity in S/cm
```

**PARTIAL CONDUCTIVITIES**  $\kappa_n$  are obtained from columns 0,1,...nspec-1 in the same way. Note that the total bulk conductivity is a sum of the partial conductivities only if the ions are uncorrelated which holds true in special cases and only approximately. It holds

$$\kappa_n = \frac{D_n z^2 F^2}{RTV_m}, \quad (14.12)$$

where  $z$  = charge in  $e$ ,  $F$  = Faraday constant,  $R$  = gas constant, and  $V_m$  = molar volume, which in program units ( $k = 1$ ) becomes

$$\kappa_n = \frac{N_n D_n q_n^2}{TV}, \quad (14.13)$$

where  $N_n$  = number of molecules of species  $n$ ,  $V$  = volume of the simulation box,  $q_n$  = charge of species  $n$ .

Note that for monoatomic ions,  $\text{mscd} = N_n q^2 \text{msd}$ . For molecules generally  $\text{mscd}(t) \neq N_n q^2 \text{msd}(t)$  although the time derivatives should be the same (but statistical noise).

**FINAL NOTES** See also `util/plb2diff.c`, [22.22](#)!

For NPT ensemble (`tau.P` is nonzero), `cook` produces scaled (by powers of  $V^{(1/3)}$ ) quantities and the above factors should be recalculated! However, `plb2diff` does it for you.

## 14.6 Structure factor

### 14.6.1 Structure factor for pure simple fluids

The structure factor is in principle the Fourier transform of a radial distribution function, see Sect. [31](#). Here, we use the atomic masses to weight different atoms. We define a  $k$ -vector by  $\vec{k} = \vec{n}/\lambda$ , where  $\lambda$  is the wavelength and  $\vec{n}$ ,  $|\vec{n}| = 1$ , its direction. (This  $k$ -vector is common in crystallography, whereas in physics a ‘circular’  $k$ -vector is more common,  $2\pi\vec{n}/\lambda$ . In `cook` V2.6h and older, the circular definition was used, from V2.6i the above ‘crystallographic’ definition is used. The definition is noted in the output files.) In a periodic box only some  $k$ -vectors are available,  $\vec{k} = \vec{k}^i/\vec{L}$ , where ‘division of vectors’ is defined in [\(25.7\)](#) and  $\vec{k}^i$  is an integer vector. The structure factor of a simple fluid is:

$$S(\vec{k}) = \frac{1}{N} |Q(\vec{k})|^2, \quad Q(\vec{k}) = \sum_j \exp[-2\pi i \vec{k} \cdot \vec{r}_j]$$

and the sum is over all atoms  $j$ .

The structure factor is used to detect crystallization and glassy state of liquids.

## 14.6.2 Structure factor for mixtures

### Theory

A sensitivity of atoms to scattering is described by weights, in neutron diffraction ‘coherent scattering lengths’  $b_j$ . They are generally complex numbers (LIMITATION: MACSIMUS supports only real values). The mixture structure factor is then

$$S(k) = 1 + N \frac{\langle |Q(\vec{k})|^2 \rangle - \sum_j b_j^2}{\left(\sum_j b_j\right)^2} \quad (14.14)$$

where

$$Q(\vec{k}) = \sum_j b_j \exp[-2\pi i \vec{k} \cdot \vec{r}_j]$$

and the sums are over atoms.

### Implementation

The scattering lengths  $b_I$  should be written ‘by hand’ to the ble-file to the table of pair potential parameters (Lennard-Jones, Busing-12, ...) after the last column. (If this column is missing, atomic masses  $m_I$  are used instead, however, do NOT write  $b_I$  instead of masses to the 2nd column!)

Cook has to be run again with option **-f** (see there) to calculate the structure factor from stored configurations. Variables **init** and **no** are the first and last frames processed, respectively. Variable **e1.kappa** is the maximum k-vector calculated.

### Sphericalized structure factor

The structure factor of fluids is isotropic. A sphericalized structure factor is

$$S(k) = \sum_{k=|\vec{k}|} S(\vec{k}) / \sum_{k=|\vec{k}|} 1$$

LIMITATIONS: MACSIMUS can calculate a sphericalized structure factor for a cubic simulation cell only.

Sphericalization is requested **e1.sf=1** in the data (while running cook with option **-f**). **e1.sf=3** in the data specifies the full 3D structure factor.

It is not possible to calculate both sphericalized and 3D structure factors at the same run. Note that a sphericalized structure factor can be obtained from the 3D one provided that the numbers of vectors  $\vec{k}$  giving the same  $k$  are known (the last column, cf. utility **avdata**).

See also utility **sfourier**.



## 14.7 Radial distribution functions

If `rdf.grid` is selected, the program measures the site-site correlation functions, also called radial distribution function (RDF). The results are stored in binary file with extension `.rdf` and may be printed and viewed using program `rdffg`. Site types used for RDF are listed in table `Lennard-Jones` (or similar according to the force field used) in the `ble-file` in 2nd column (denoted as `atom`). By default there is one RDF for each pair of site types even if this site type appears several times in a molecule (molecules).

WARNING: program `rdffg` (and also `harmg`) uses for site-site correlation functions formulas giving ‘incorrect’ limits

$$\lim_{r \rightarrow \infty} g_{IJ} = 1 \quad (14.15)$$

and not the correct (conforming the NVT definition)

$$\lim_{r \rightarrow \infty} g_{IJ} = \begin{cases} 1 - 1/N_I & \text{for } I = J \\ 1 & \text{for } I \neq J \end{cases} \quad (14.16)$$

Formula (14.15) is probably a better approximation of the thermodynamic limit, however, in certain cases (e.g., when integrals of RDFs are calculated), (14.16) should be preferred. In most cases the difference between (14.15) and (14.16) is irrelevant.

Options for RDF are controlled by variables and by file `simname.s-s`.

`rdf.grid` (0) Grid for calculating the site-site radial distribution functions, in  $1/\text{\AA}$  (i.e., the number of histogram bins per 1  $\text{\AA}$ ). If negative, there is one summary (equally weighted) function for all combinations of site types. If positive, there is a separate function measured for each combination of site types (if `simname.s-s` is missing) or only selected site-site pairs are measured (if `simname.s-s` exists, see below). If `rdf.grid=0`, the radial distribution functions are not measured.

`rdf.cutoff` (0) Max range for which the site-site radial distribution functions are measured, in  $\text{\AA}$ . If `rdf.cutoff=0` then the value of `cutoff` is assumed (default).

`rdf.onefour` (1) Controls whether 1-4 (or 1-5 if `distance14=4` in the `ble-file`) are to be included to measured radial distribution functions. E.g., in a united-atom model of butane, CH3-CH2-CH2-CH3, there is a peak on the CH3-CH3 radial distribution function caused by intramolecular CH3-CH3 pairs. Setting `rdf.onefour=0` will eliminate this intramolecular peak.

More options can be specified in file `simname.s-s`. Data in this file are active if `rdf.grid>0`.

Site types are listed in table `Lennard-Jones` (or similar according to the force field used) in the 2nd column (denoted as `atom`).

File format:

```
! define group of atoms of given TYPE:
ID TYPE SPECIES.ATOM [SPECIES.ATOM ...]
```

```

ID ...
! include all pairs of types and stop reading TYPE1-TYPE2 data:
*
! include RDF of TYPE1-TYPE2:
TYPE1 TYPE2
...

```

where

**ID** Lowercase letter or a decimal digit. The full group identifier is a concatenation ID+TYPE (e.g., aCH3). It is not possible to have different atom types in one group.

**TYPE** Atom type, see above

**SPECIES** Species number (numbered from 0 in the order given to **blend** and used in the ble-file)

**ATOM** Atom (site) number, see the mol-file or table sites (in the corresponding species section) of ble-file.

One group of atoms should correspond equivalent (with respect to symmetry) atoms. All atoms of the same type not listed in any group comprise a ‘default’ group (with identifier = **TYPE** without prefix).

Examples of simname.s-s for a simulation of a united-atom model of pentane CH3E-CH2E-CH2E-CH2E-CH3E (species 0) follow.

To measure only RDF of CH3E-CH2E and CH3E-CH3E pairs:

```

!type type
CH3E CH2E
CH3E CH3E

```

To measure all RDF’s so that the central CH2 is distinguished, use:

```

c CH2E 0.2
*

```

If, in addition, individual atom pairs are to be selected, the ID’s are prepended, e.g.:

```

c CH2E 0.2
CH2E CH2E
CH2E cCH2E
cCH2E cCH2E

```

## 14.8 Cluster (oligomer) analysis and bond kinetics

Let us consider bulk fluid simulation of atoms (ions) (or small molecules – not fully implemented) interacting via strong attractions so that it is reasonable to say that two atoms are bonded if their distance is less than certain limit. Typical example is molten salt like  $\text{AlCl}_3$  consisting of individual ions. In the melt, the ions are bound to relatively very stable molecules like calculates the clusters (molecules).

### 14.8.1 Cluster overview

1. Reads stored configurations (simname.plb or simname.1..., see option **-f**)
2. Calculates bonds between molecules (species), bonds are defined by a distance criterion between selected sites (atoms) in molecules
3. Analyzes connectivity of molecules and splits each configuration into clusters
4. Makes statistical analysis of clusters. Clusters are distinguished by size, stoichiometry, and topology (with the exception of very large clusters)
5. Calculates time development of selected clusters, prints summary of all clusters in all configurations

### 14.8.2 Compilation and synopsis

`cook` must be compiled with `#define CLUSTERS`: can be specified while running `configure.sh` or directly in `simopt.h`.

Synopsis (in addition to the usual options)

```
cook -f[stride] sysname [simname] [more options]
```

**-f** = **-f1** means that every configuration (frame) is analyzed, otherwise given stride applies.

Negative stride means simname.1, simname.2,... read instead of simname.plb

**-v2** selects verbose mode: prints some technical info and a list of clusters for every configuration.

At the same time the diffusion/conductivity and structure factor can be calculated.

### 14.8.3 Input data

Cluster-specific variables in sysname.def or simname.get are:

**init** First frame (configuration) read from simname.plb (**init=2** means that frame 1 is skipped)

no The last frame (inclusive). To to get the number of frames, run `plbinfo simname.plb`.

; More data in `sysname.def` or `simname.get` after the first ; define the input data for cluster analysis. Both these data sets in `sysname.def` and in `simname.get` are merged; should any of them be empty, at least a line with `end` must be present. A commented example follows:

```
init=1 no=10; ! normal data end here
! more data for cluster analysis start here
off          ! turns off any cluster processing

bonds        ! section defining bonds
AL CL 2.1 ! bond created between AL and CL in different
AL S 3      ! molecules for dist<2.1.  AL CL S are ATOM names

maxcluster=15 ! larger clusters not topologically distinguished
              ! (would be too slow)

! topology of named (registered) clusters is in the che-format
! (see blend for details):
! blank line required after the molecule

cluster AlCl3 ! named cluster for convergence profile
Cl      Cl
  \    /
   Al
   |
  Cl

cluster AlSCN ! another named cluster, Al CL SCN are MOLECULES
Al--SCN

$i extra.cl ! include file (cannot nest)

clusters ! mol- and plb-files will be created for clusters, showing
          ! the connectivity of molecules which are represented by
          ! site[0] only. Use 'show' to view the cluster structure
          ! (OK for monoatomic molecules)

configurations ! mol- and plb-files will be created for whole
               ! configurations (frames), full molecules are shown,
               ! but intermolecular bonds shown between site[0] only

colors ! data used to make a gol-file when "clusters" specified
Al MAGENTA 1.1
Cl GREEN 1.5
```

I YELLOW 1.8

```
end ! reading data stops here
. ! the same as end
```

Comments:

- A bond is created between different molecules if there exists a pair of sites less than the limit (keyword **bond**) apart.
- Of course, molecules may consist of one atom only. Molecule named “Al” may contain only atom “AL” (note case sensitivity!). “Al” refers to “species Al” in a sysname.ble and thus to files Al.mol etc. for blend.
- **maxcluster** is the maximum cluster size (number of molecules) for which the topological analysis is performed. Larger clusters of the same stoichiometry but different structure are not distinguished (even if they are named).
- **cluster NAME** defines a named cluster in simplified CHE-format. There must not be any blank line after the **cluster NAME** line.
- If **colors** is given, gol-files will be created for each cluster shown.
- No gol-file is created for **configurations**. If you wish to have proper colors, use molcfg and link or copy simname.gol to simname.#.gol
- Equal sign = is optional between a keyword and its value, e.g., **\$i=FILE** (bug: FILE must not contain ‘=’). For better compatibility, also **\$iFILE** (without space) is allowed

## 14.8.4 Results

simname.sfd Final statistics of clusters. The clusters are sorted by size and stoichiometry (we call this “standard order”). If the cluster is named, the name is printed. Then, a list of bonds is printed: atoms are numbered consecutively in the order of the stoichiometry formula. E.g.,  $\text{AlCl}_4$  has bonds 0-1 0-2 0-3 0-4 because Al has number 0 and four Cl are numbered 1,2,3,4.

simname.cl.cpa Convergence profile of counts of named clusters. Columns correspond to the named clusters in standard order (not in the order in which the clusters appear in simname.def after **;**). Each line corresponds to one configuration analyzed.

simname.#.mol, simname.#.plb **#=1,2,...** denotes frame number. This applies only if keyword **configurations** has been specified in sysname.def. To view the configuration with molecule-molecule connectivity, use e.g.:

```
show simname.1
```

**\*.mol, \*.plb** To view the clusters. Applies only if keyword **clusters** has been specified. For instance, **A13.C19b.mol** or **A13.C19b.plb**. The last configuration (position of atoms) encountered is used to show the cluster, each molecule is represented by the 1st atom.

**simname.rm.sh** Shell script which will remove all files created with keyword **clusters**: run it as

```
sh simname.rm.sh
```

### 14.8.5 Bugs and caveats

- The topology analyzer is inefficient—it requires  $n1! \ n2! \dots nN!$  operations for a cluster with stoichiometry **MOL1<sub>n1</sub> MOL2<sub>n2</sub>...MOLN<sub>nN</sub>**. Hence, the practical maximum **maxcluster** is around 15.
- Molecule-molecule bonds in **\*.mol** are always created between **site[0]** irrespective of the actual site that passed the distance condition (no problem with one-atom molecules).

### 14.8.6 Bond kinetics

Files **simname.created.cpa** and **simname.broken.cpa** are created:

**column 1** time

**column 2** **created[0]** or **broken[0]** =  $2 * (\# \text{ of broken/created bonds since the previous frame})$

**column 3** **created[1]** or **broken[1]**, as above but the bond is not counted if the same bond has been or will be created/broken immediately before/after

**column 4** **created[2]** or **broken[2]**, as above, one other change allowed between creation/breakage of the same bond

**more** etc., according to compile-time switch **#define BONDHIST**

#### ALGORITHM:

- For bond definition see the ‘cluster analysis’ above
- The algorithm works on the basis of a list of bonded molecules attached to each molecule. The current and previous lists are compared and the differences determine the broken and created bonds (column 2).
- In addition, the list of created and broken bonds (at each molecule) is searched for time-ordered sequences created-broken, broken-created, created-broken-created, etc.; sequences with even number of changes (=the bond returns to the same

state) are omitted, odd sequences are included as one creation or breakage (for time of the middle change, e.g.. for created-broken-created for the time of the middle breakage). These sequences for created[1] or broken[1] must not be interrupted by creation or breakage of another bond (CreateBondTo77-CreateBondTo13-BreakBondTo77 is not recognized as bond to molecule 77 created and immediately broken).

- For created[2] or broken[2], one change in between is allowed (the above example is recognized and does not count for created/broken bonds to molecule 77)
- Etc., up to created[BONDHIST] or broken[BONDHIST]: BONDHIST-1 changes in between are allowed
- It may happen that created[i] or broken[i],  $i > 0$ , are odd since the ‘does not count’ condition may occur for one molecule from the bonded pair, e.g. (A-B C)  $\rightarrow$  (A B-C)  $\rightarrow$  (A-B C). This looks strange but is not a bug—is a feature.

## 14.9 Dihedral angle distribution

If `dih.grid` is selected, the program measures the dihedral angle distributions (see also `#define DIHHIST` in `simopt.h` for options). The results are stored in binary file with extension `.dih`; at the same time, ascii dump `.dia` is printed.

# Chapter 15

## Special versions

I should write more about boundary conditions in `cook`. Now, see Sect. 8.3 for an overview. Also comments in file `simopt.h` may be useful. Only a few particular versions and special terms are listed here.

### 15.1 Fixing positions of selected atoms

Positions of selected atoms can be fixed via harmonic springs. See also see Sect. 15.9. The potential is

$$U_{\text{fix}} = \sum_i \frac{K}{2} (r_i - r_i^{\text{fix}})^2$$

File `simname.fix` defining sites to be kept fixed is needed:

```
# this is comment
! this as well
! the following line specifies that site # 2 will be fixed:
2
! the following line specifies that sites 5 6 7 8 will be fixed:
5-8
! the following line specifies that site 12 will be fixed
! to the position x,y,z given (if no x,y,z are given, the positions
! are taken from the initialization)
12  5.5 6.6 7.7
! then the initial file written while init>=3 is not needed.
! * Site numbers for the first molecule can be found in mol-file or
!   ble-file (table sites), otherwise have to be calculated
! This is incorrect line:
2 3
```

This file has the same format as `molname.keep` or `molname.mark` from `blend` and can be copied. However, atom positions are not present.



- Force constant  $K$  = option `-k`.  $K$  is in units  $\text{K}/\text{\AA}^2$ . Reasonable values are probably in range 100–10000. Too low values do not keep atoms in positions firmly enough, too large values cause bad energy conservation and other artifacts.
- Negative value of option `-k` [default] turns off any fixing and file `simname.fix` is ignored.
- For zero value, `-k0`, see Sect. 15.9.
- If a line containing keyword `fix` is present in `simname.cpi`, energy  $U_{\text{fix}}$  is recorded in the convergence profile

### Initialization (was changed in V2.4j):

- If `simname.fxc` exists, it is read and the positions saved there are used as fix locations.
- If `simname.fxc` does not exist, the positions are taken from the configuration at job start.
- If, in addition, atom positions are present in file `simname.fix`, they replace the positions given above.

In any case, file `simname.fxc` is written to be used next time. In the next start of `cook -k`, `simname.fxc` is used; it may be modified by coordinates specified in `simname.fix`

**Conserved quantities:** Presence of non-symmetric potential causes the momentum and angular momentum (for free b.c.) not to be conserved. For general 3 or more fixed atoms, this is correctly taken into account in calculating the number of degrees of freedom (needed to evaluate `Tkin`). In addition, correcting the momentum and angular momentum for numerical errors is turned off (unless `drift` is specified)

Bugs: for 1 or 2 atoms to be fixed (or linear set of atoms), the number of degrees of freedom calculated and correcting the angular momentum is incorrect and must be set using `drift` and `conserved`.

Tested with FREEBC only, but in principle should work with other b.c. too

## 15.2 Notes on water models

There are THREE versions of TIP3 water:

1. Flexible TIP3 water, as used in `blend`, with small intermolecular H-O and H-H Lennard-Jones terms to prevent H-O charge singularity. Because of flexibility, the H-bond energies are lower (dimer energy = -6.8322 kcal/mol). To use this model also in `cook`, do not use option `-h` in `blend` and run `cook -u9999 -x`.

2. Rigid TIP3 water, with the same intermolecular H-O and H-H Lennard-Jones terms as above. This is used in `cook` when \*.ble file has been created by `blend -h` AND `cook -x` (and no `-uK` option selecting flexible terms is used). Dimer energy = -6.5920 kcal/mol. It is recommended to use this model with the random initializer (`init=3="random"`) to prevent singularities.
3. Rigid ‘original’ TIPS3 water: `blend -h` needed and no `-x` nor `-uK` is used. If possible, optimized code is used (using the optimized code is suppressed by `cook -x0`; the numerical results should be the same). Dimer energy = -6.5390 kcal/mol.

As regards other water models, they are usually defined without H-H and H-O Lennard-Jones terms. There are, however, optimized and unoptimized versions of TIP4P and ST2. (Note: TIP4P dimer energy = -6.2345 kcal/mol).

- `cook` prints a warning when it recognizes water and `blend -h` has not been used.
- `blend` without `-h` and `cook -x -u9999` will use the flexible water also in MD. This is less efficient.
- `cook` need not recognize water (to use optimized code) that has been added to the system in other way than via ‘blend hoh’ and consequently uses the general code (with H-O and H-H terms). The molecule is still rigid (if `blend -h`)

## 15.3 Cut off electrostatic forces (CUTELST)

Instead of the Ewald summation (which can be regarded as more accurate, but is more costly), the simulation in periodic boundary conditions can neglect electrostatic forces beyond certain `cutoff`. This version of `cook` is requested by `#define COULOMB 2` or `#define COULOMB 0` in `simopt.h`.

The  $1/r$  term in the Coulomb energy is replaced by

$$\frac{1}{r} \approx \begin{cases} 1/r - \text{shift}, & \text{for } r < \alpha \text{cutoff} \\ (r - r_1)^3 (A \text{FA}(r) + B \text{FB}(r)), & \text{for } \alpha \text{cutoff} < r < \text{cutoff} \\ 0, & \text{for } \text{cutoff} < r \end{cases}$$

where  $\text{FA}(r) = 1$  and  $\text{FB}(r) = r$  are the base functions.

The electrostatic force is thus neglected beyond the cutoff, shifted at short separations, and smoothly interpolated in between.

The default value of  $\alpha = 0.7$ ; the optimum values are around  $2/3$ . The former default 0.9 is not good [16]. The `cutoff` must be less than one half of the box size.

For `#define COULOMB 2` this cutoff electrostatic is implemented by quadratic splines, for `#define COULOMB 0` directly by formulas. In fact, the `COULOMB=2` version it makes use of the structure of the Ewald summation, where the  $k$ -space part is turned off ( $K=0$ ) and the `erfc` functions (implemented by rational splines, `#define COULOMB -1`) are replaced by quadratic splines that are more suitable. Both the `COULOMB=2`

and COULOMB=0 are approximately of the same speed, but this may depend on the architecture.

The cutoff electrostatic should not be used for systems with free charges. It is acceptable for systems with partial charges but neutral groups, possibly also for charged systems solvated e.g. by water. The minimum `cutoff` is 15–20 Å. Because of the shift, the total energy and pressure may be significantly affected.

The pressure calculation assumes that the virial of electrostatic forces is the same as the electrostatic energy. This holds exactly for the Ewald summation (and therefore the virial pressure calculated by the virtual volume change [see variable `dV`] is the same as the virial calculated directly from pair interactions). However, this is an approximation only for the cut and shifted electrostatic. The virial pressure *is not consistent* with the used potential; this need not mean that it is worse approximation of the full electrostatic system, though.

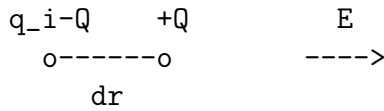
## 15.4 Gravity simulation (STARS)

In this version, requested by `#define STARS`, the sign of the Coulomb interactions is changed to attraction of equal ‘charges’. To be used with FREEBC. For instance, a globular star cluster can be simulated in this way (<http://www.icpf.cas.cz/jiri/movies/stars.htm>).

## 15.5 Polarizable dipoles

For a general theory and formulas, see Appendix 26.

While point induced dipoles are used in `blend`, they are approximated by macroscopic dipoles in `cook`. A polarizable site  $i$  (with an optional charge  $q_i$ ) accepts an additional charge  $-Q$  and there is an auxiliary site  $\delta\vec{r}$  apart with charge  $+Q$ :



It holds

$$\vec{\mu}_i = Q\delta\vec{r}_i = \vec{\alpha}_i \cdot \vec{E}_i = \vec{\alpha}_i \cdot \vec{F}_i / Q \quad (15.1)$$

or

$$\delta\vec{r}_i = \vec{\alpha}_i \cdot \vec{F}_i / Q^2$$

where  $\vec{E}_i$  is the electric field acting on site  $i$ ,  $\vec{F}_i$  is the electric force,  $\vec{\mu}_i$  is the induced dipole, and  $\vec{\alpha}_i$  is the polarizability tensor. (See also Appendix 26.) The larger  $Q$ , the better approximation of point dipoles, but lower numerical accuracy;  $Q = -1000e$  seems to be a good tradeoff.

Particularly, equation (26.6) is approximated by the numerical derivative

$$\vec{f}_{\text{rep},i} = -\kappa_i Q \delta\vec{r}_i \cdot \vec{\nabla}_i \vec{f}_{ij} \approx -\kappa_i Q [\vec{f}_{ij}(\vec{r}_i + \delta\vec{r}_i) - \vec{f}_{ij}(\vec{r}_i)].$$

Equation (15.1) should be valid for all polarizable atoms in the system. Since the field  $\vec{E}$  depends on the induced dipoles on other atoms, we have a set of equations for the so-called self-consistent field (SCF). Let us write it shortly as

$$\mathbf{D} = \text{Iter}(\mathbf{D})$$

where  $\mathbf{D}$  is the set of induced dipoles (or equivalently  $\delta\vec{r}$ 's) and operator  $\text{Iter}()$  denotes one iteration. In this iteration, all electrostatic forces are calculated (using the fixed set of atomic charges and current induced dipoles) and from them the new induced dipoles.

If the iterations diverge, it is called the ‘polarization catastrophe’. It may happen if the polarizability is too large and/or atoms are too close together (e.g., at too high temperature).

The SCF equation can be rewritten as

$$\mathbf{D} = \omega \text{Iter}(\mathbf{D}) + (1 - \omega)\mathbf{D}$$

where  $\omega$  (in the program called `scf.omega`) is the ‘mixing iteration parameter’. For the equations of motion for the induced dipoles, see also Appendix 28.

### 15.5.1 Polarizability models

Several models of polarizability are available. They are selected in the `simopt.h` file by `#define POLAR number`, where `number` is a sum of the following numbers (or via `configure.sh`):

- 1 Repulsive antipolarizability (shell-core model [2]). All polarizable atoms can interact.
- 2 Saturated polarizability (see Sect. 26).
- 4 Makes some optimizations if there are also uncharged atoms. Use this option if there are more than about 1/4 of uncharged atoms in the system.
- 8 Axial polarizability supported. The polarizability is a tensor. The principal axis of this tensor is defined by a pair of atoms, such pairs are listed in table `axials` in the `ble`-file [see the `blend` manual, Section 5: Output format (`ble`-file)]. Bugs: tensor polarizability cannot be saturated. POLAR 8 has been checked with the shell-core model (POLAR 1) and it is not clear whether it can be used without it.
- 16 Special (see `constrdaa.c` and `gear2polg.c`)
- 32 Modifier of 1: repulsive antipolarizability for polar-nonpolar pairs only. (E.g., polarizable anions and nonpolarizable cations).
- 64 The ADIM model [1]: also all intramolecular induced dipole–induced dipole interactions are included

E.g., `#define POLAR 9` selects the shell-core model with a support for axial polarizability tensors; the code will be optimized for no (or a few) uncharged atoms in the system.

If `#define POLAR 0` is specified, only scalar linear polarizability is supported.

If `POLAR` is not `#defined` the polarizability support is turned off.

## 15.5.2 Integration methods

There are three integration methods available.

- Full iterations of the self-consistent field until sufficient accuracy (given by variable `scf.eps`) of the induced dipoles is reached. In most cases, `scf.eps` should be 1e-3 (prog. units) or less. Since prediction of the induced dipoles from previous integration steps is used at the same time, (cf. option `-p`) usually 3–6 iterations are sufficient. The value of the mixing iteration parameter `scf.omega=0.95–0.98` leads for typical systems to faster convergence than direct (unmixed) iterations (`scf.omega=1`).

This method can be used with both Gear integration and Verlet/SHAKE. For Gear, the value of option `-@` should be zero (default with `-m3` or larger). If you wish to use Verlet/SHAKE and full iterations (option `-m2`), set option `-@0`.

This method is most accurate but time demanding because electrostatic forces are calculated several times in one integration step.

- Method using a stable second-order predictor see Sect. 28. It should be used with Verlet/SHAKE. THIS IS OUT-OF-DATE: This method is selected by option `-m2` (the default), option `-@50` (the default for `-m2`), the mixing iteration parameter in range `scf.omega=0.6–1`, and `scf.eps` so large so that there is only one iteration per integration step. The trick is that (with a special predictor given by `-@50`), one iteration (correction) is sufficient and the method is stable (errors in the self-consistent dipoles do not cumulate). The value `scf.omega=0.6` (default) guarantees stability for all (converging) systems though higher values may lead to more accurate trajectories. Typical values are around `scf.omega=0.7–0.8`; values close to 1 will probably cause divergence. It is recommended to try several values of `scf.omega` and to watch the output (columns `polerr` and `polit`) and finally to set `scf.omega` about 0.02–0.05 below the limit of convergence (stability of the predictor-corrector procedure).

Hint: Option `-^` can be used instead of `scf.omega`.

- Car-Parrinello-like approach (also called Lagrangian or extended Lagrangian) is a mechanical model of a polarizable dipole. The Lagrangian is:

$$L = L_0 + \sum_i (M/2) D \dot{\mathbf{r}}_i^2 - \sum_i (K/2) D \mathbf{r}_i^2$$

where  $L_0$  is the unperturbed Lagrangian and  $K = Q^2/\alpha$  (only scalar polarizability is supported). For a fixed configuration at zero temperature, the equilibrium

values of the dipoles are the same as the SCF dipoles, during simulation they will oscillate around the SCF solution.

This method is selected by negative option `-p`.

The ‘mass’ of the additional degree of freedom is set indirectly using its typical correlation time `tau.dip`:

$$M = Q^2 \text{tau.dip}^2 / \alpha$$

(One free ‘mass’  $M$  in a constant field exhibits small harmonic motion with period  $2\pi \text{tau.dip}$ .)

BUG: in FREEBC, the algorithm removing the angular momentum drift is not correct and leads to energy drift (decrease). Use `drift` unless friction thermostat.

Car-Parrinello-like method is new in version 2.0c.

NOTE: name “Car-Parrinello” is rather misleading. It is not the quantum Car-Parrinello method.

## 15.6 Pressure tensor

New in version 2.4a, definition modified in 2.4m. Pressure tensor (also called stress tensor<sup>1</sup>) is for a system of point interacting particles in free space given by

$$\overleftrightarrow{P} = \overleftrightarrow{P}_{\text{kin}} + \overleftrightarrow{P}_{\text{vir}} = \frac{1}{V} \sum_{i=1}^N \left( m_i \dot{\vec{r}}_i \dot{\vec{r}}_i + \vec{f}_i \vec{r}_i \right) \quad (15.2)$$

The scalar pressure is 1/3 of its trace,  $P = \text{tr}(\overleftrightarrow{P})/3$ .<sup>2</sup>

For models with rigid bonds (or rigid models described by sites and rigid bonds, see Sect. 11.3) it is given by

$$\overleftrightarrow{P} = \overleftrightarrow{P}_{\text{kin}} + \overleftrightarrow{P}_{\text{vir}}, = \overleftrightarrow{P}_{\text{kin}} + \overleftrightarrow{P}_{\text{constr}} + \overleftrightarrow{P}_{\text{config}},$$

where

$$\overleftrightarrow{P}_{\text{constr}} = \frac{1}{V} \sum_{a,b} \vec{f}_{ab}^{\text{constr}} \vec{r}_{ab},$$

the sum is over all bonds  $a$ - $b$  and  $\vec{f}_{ab}^{\text{constr}}$  is the constrained force (fictitious force needed to maintain the bond, available readily in the MD code). The configuration al part in periodic b.c. has to be also expressed by pair interactions,

$$\overleftrightarrow{P}_{\text{config}} = \frac{1}{V} \sum_{i < j}^N \vec{f}_{ij} \vec{r}_{ij}. \quad (15.3)$$

For long-range forces the sums must run also over all periodic images. Formulas for the Ewald summation are summarized in excellent paper [19]; for the r-space cutoff

---

<sup>1</sup>Sometimes the stress tensor is defined as minus the pressure tensor.

<sup>2</sup>In versions 2.4a–2.4l the pressure tensor was defined with factor 1/3 so that  $P = \text{tr}(\overleftrightarrow{P})$ .

less than half the minimum box size the the real-space part of  $\vec{P}_{\text{config}}$  is evaluated in the same way as (15.3).

Notes:

- For atomic systems without rigid bonds (constraints) in the NVT ensemble it holds

$$\vec{P}_{\text{kin}} = \frac{NkT}{V} \vec{1} \quad (15.4)$$

The atom-based kinetic part of the pressure tensor for anisotropic systems is no longer isotropic. Full pressure tensor must be therefore used (typical example is surface tension of rigid models of water).

- NEW in V2.6m:

If some integral of motion are preserved (e.g., 3 components of the momentum in the periodic b.c. with the Nosé–Hoover thermostat), then the kinetic part of (15.2) is subject to finite-size errors of the order of  $\propto 1/N$ . More accurate results are obtained if  $\vec{P}_{\text{kin}}$  is multiplied by a factor which restores the equation for the kinetic temperature,

$$\vec{P}_{\text{kin}} = \frac{N_{f0}}{N_f} \frac{1}{V} \sum_i m_i \dot{\vec{r}}_i \dot{\vec{r}}_i \quad (15.5)$$

where  $N_{f0}$  is the number of degrees of freedom not taking into account conserved quantities ( $N_{f0} = 3N - \text{constraints}$ ) but taking into account mechanical constraints (bonds),  $N_f$  is the effective number of degrees of freedom (incl. all conserved quantities);  $N_f$  is used in the formula for the kinetic temperature,  $T_{\text{kin}}$ . For atomic systems without constraints and with canonical thermostats, (15.4) is exactly reproduced; it is not for the Berendsen thermostat where a (smaller) residual finite-size error  $\propto 1/N$  is inevitable. This correction should not be used with the true MTK barostat. See also variable `corr`.

- The diagonal components can be independently calculated by appropriate scaling of the box, see variable `rescale`; the typical application is the surface tension in the slab geometry, see Sect. 15.7. In the case of systems with constrained bonds, scaling based on center-of-mass has to be used. To obtain the full tensor, the corresponding kinetic part calculated from the centers of mass must be added. The result is in the thermodynamic limit (not numerically for every configuration) the same.
- Direct calculation of the center-of-mass-based configurational part of the pressure tensor is not supported (in V2.4a).

All pressure tensor components are given in Pa. Symbols used in statistic analysis are:

- `Pvir` =  $\vec{P}_{\text{vir}} = \vec{P}_{\text{constr}} + \vec{P}_{\text{config}}$
- `Pkin` =  $\vec{P}_{\text{kin}}$  (atom-based)
- `PKin` = molecular (center-of-mass) based kinetic pressure tensor

- $\text{Pvirc} = \text{tr}(\vec{\vec{P}}_{\text{constr}})/3$
- $\text{Pvir pair} = \text{tr}(\vec{\vec{P}}_{\text{vir}})/3$   
(without k-space contribution if Ewald summation is used)
- $\text{Pvir xy-zz} = (P_{\text{vir},xx} + P_{\text{vir},yy} - 2P_{\text{vir},zz})/3$   
(good for surface tension of models without rigid bonds)
- $\text{Pt} = (P_{xx} + P_{yy} - 2P_{zz})/3$   
(good for surface tension of all models, needs PRESSURETENSOR&3)

To record components of the pressure tensor in the .cp files, their codes must be given in the .cpi-file (see Sect. 9.2.3. The four-letter codes are **Pvxx**, **Pvxy**, etc., for the site-based virial parts, **Pkxx** etc. for the site-based kinetic part, and **PKxx** for the center-of-mass based kinetic part, and finally **Ptxx**, **Ptyy**, **Ptzz**, **Ptxy**, **Ptyz**, **Ptzz** for the full pressure tensor components. All these values are without cutoff corrections and the units are Pa.

### 15.6.1 Pressure tensor for Drude oscillators

Let  $\vec{r}'_i$  be positions of the Drude charges and shortly  $r = \{r_i\}_{i=1}^N$ ,  $r' = \{r'_i\}_{i=1}^N$ . Let us denote  $U' = U'(r, r')$  the potential energy as a function of all point charges (normal and Drude). We are interested in the self-consistent potential energy,

$$U(r) = \min_{r'} \left[ U'(r, r') + \sum_i \frac{K_i}{2} (\vec{r}'_i - \vec{r}_i)^2 \right] \quad (15.6)$$

The minimum condition is equivalent to

$$\frac{\partial U'}{\partial r'_i} + K_i(r'_i - r_i) = 0 \quad (15.7)$$

Let us calculate  $U(r + dr)$ :

$$U(r + dr) = \min_{r''} \left[ U'(r + dr, r'') + \sum_i \frac{K_i}{2} (\vec{r}''_i - \vec{r}_i - d\vec{r}_i)^2 \right] \quad (15.8)$$

Here  $r'' = r' + dr'$ , where  $dr'$  is of the same order as  $dr$ . However, the minimum is quadratic in  $r'$  and therefore replacing  $r'' \approx r'$  in (15.8) will lead to a second order error which can be neglected. From the Taylor expansion of  $U(r + dr)$  (or simply taking  $\partial/\partial \vec{r}_i$  of [...] in (15.6)) we get

$$\frac{\partial U}{\partial \vec{r}_i} = \frac{\partial U'}{\partial \vec{r}_i} - K_i(\vec{r}'_i - \vec{r}_i) = \frac{\partial U'}{\partial \vec{r}_i} + \frac{\partial U'}{\partial \vec{r}'_i} \quad (15.9)$$

The computer code (r- and k- space Ewald sums for both real and Drude charges) calculates the following components of the pressure tensor

$$\frac{1}{V} \sum_i \left( \vec{r}_i \vec{f}_i + \vec{r}'_i \vec{f}'_i \right) = -\frac{1}{V} \sum_i \left( \vec{r}_i \frac{\partial U'}{\partial \vec{r}_i} + \vec{r}'_i \frac{\partial U'}{\partial \vec{r}'_i} \right)$$



However, from (15.9) it follows that it should be

$$-\frac{1}{V} \sum_i \vec{r}_i \frac{\partial U}{\partial \vec{r}_i} = -\frac{1}{V} \sum_i \left( \vec{r}_i \frac{\partial U'}{\partial \vec{r}_i} + \vec{r}_i \frac{\partial U'}{\partial \vec{r}_i'} \right)$$

Therefore, the following correction should be added

$$\frac{1}{V} \sum_i (\vec{r}_i' - \vec{r}_i) \frac{\partial U'}{\partial \vec{r}_i} = -\frac{1}{V} \sum_i (\vec{r}_i' - \vec{r}_i) \vec{f}_i'$$

where  $\vec{f}_i'$  is the force acting on the Drude charge. Since in our model the force on the Drude charge is entirely electrostatic, the trace of the above term (contribution to the virial pressure), the above term is  $-(2/V)E_{\text{self}}$ .

## 15.7 Slab geometry and surface tension

To enable the z-density profile and surface tension calculations, SLAB must be `#defined` in the respective `simopt.h`, or selected while running `configure.sh`.

To prepare a slab, define first the box;  $L_x = L_y$  and  $L_z = 3L_x$  is recommended. Example of data in the def-file:

```
x=30
L[0]=x
L[1]=x
L[2]=x*3
```

Note that efficiency of Ewald summation decreases somehow for too an elongated slab even if the linked-cell list method is used; cutoff electrostatics is not affected in this case.

There are several ways how to create a slab.

1. Use `init="slab"` with `tau.T=0.1` or so. Simplest but slowest.
2. Prepare a periodic box first, then enlarge `L[2]` and load again with `load.L[2]=1`; use variable `shift[2]` to shift the configuration
3. Use variables `slab.n`, `slab.Kz`, `slab.z0`, etc. These artificial forces should be turned off for measurement. Example:

```
n=300
N[0]=n
slab.n[0]=n      ! will affect n molecules
slab.Kz[0]=10    ! moderate force
slab.z0[0]=x*0.6 ! slab width=1.2x
```

There are four basic ways of slab simulation::

**Strictly periodic mode** (default) is recommended for liquid with very dilute vapor or two layers of liquids. The system is fully periodic even in the the z-direction. If the the vapor pressure is not negligible, molecules moving in the z-direction cross the boundary and cause that the center of the slab moves slightly in the opposite direction. The z-density profile is then “smoothed out” and the cutoff corrections may be imprecise, too. These errors are acceptable if only a few molecules cross the z-boundary in this way. It is not recommended to use `drift=4+8+16+32` (recenter slab and all drifts after every step) and `el.epsinf=-1` (slab-dipole correction) unless *very few* molecules evaporate.

NEW in 2.6c: the autocenter feature (see `slab.sp`) now shifts the slab automatically before measuring the z-profiles (not physically—the trajectory is not affected).

**Almost periodic mode** is recommended for liquid with moderately dense vapor. In addition, a very small harmonic force keeping the slab in the center is added (see `center.cmn`, `center.cmK[2]`). Particles can still cross the boundary (there is a jump in forces there, but very small) but the slab cannot drift too much. The force constant should be set so that the period of harmonic motion is larger than the evaporation rate; then this method is unsuitable for very low evaporation rates. Incompatible with `drift=4+32`.

**Z-aperiodic mode** is recommended for a slab with dense vapor. It employs two “caps” preventing particles to move in the z-directions. Example:

```
z=x*3
n=300
N[0]=n
slab.n[0]=n      ! will affect n molecules
slab.Kz[0]=300   ! large force
slab.z0[0]=z/2-3 ! 3 A thick repulsive wall
drift=4+8+16+32  ! recenter slab and all drifts after every step
el.epsinf=-1     ! slab-dipole correction (Ewald only)
```

### 15.7.1 Surface tension via virial pressure

The fastest route to the surface tension of liquids with low vapor tension is via the virial of force. Surface tension from slab geometry in case of negligible vapor pressure is:

$$\gamma = -\frac{3}{4}pL_z$$

or more precisely with correction to vapor pressure  $p^s$ :

$$\gamma = -\frac{3}{4}(p + p^s)L_z$$

where  $p^s$  can be estimated from the density (use density profile) and the ideal gas EOS. The value of  $\gamma = \text{gamma}$  printed contains cutoff correction of LJ terms calculated by integration from the current density profile.

The following variables (structures) apply for slab calculations, see Sect. 9.2.5: `center`, `densprof`, `el.epsinf`, `drift`. Do not forget to set `slab.mode=1` and with Ewald `el.epsinf=-1`, check `drift`. Note that with cutoff electrostatics the virial of electrostatic forces is not equal to the electrostatic energy. Consequently, better values of the pressure and surface tension require the virtual volume change method, see below.

### 15.7.2 Surface tension via virtual volume change

This method is equivalent to the above method, but it uses the virtual volume change to calculate the virial pressure. This makes sense with cutoff electrostatics (`COULOMB>=0`), where the directly calculated virial (replacing the virial of electrostatic force by the electrostatic energy) gives usually worse results. However,  $p^s$  is still needed and (within virtual box change methods), the following method is recommended instead. This method is requested by setting `dV` and `rescale`.

### 15.7.3 Surface tension via virtual area change

This method [17] is based on the formula

$$\gamma = \left\langle \left( \frac{\delta U}{\delta A} \right)_V \right\rangle$$

where  $A = 2L_x L_y$  and the derivative is interpreted as scaling of  $x$  and  $y$  coordinates and simultaneous contra-scaling of  $z$  so that the volume is the same (then, no  $p^s$  is needed). For technical reasons, this is implemented as:

$$\begin{aligned} \left( \frac{\delta U}{\delta A} \right)_V &= -\frac{3}{4} L_z (p_{\text{vir}} + p_{\text{cutoff correction}}) \\ p_{\text{vir}} &= -\frac{U(+dV) - U(-dV)}{2dV V} \\ \text{where} \\ U(d) &= U(x \exp(d/3), y \exp(d/3), z \exp(-2d/3)) \end{aligned}$$

This method is turned on by `slab.mode=2`; `dV` must be also set. Example of surface tension measurements:

```
dV=1e-3          ! step for numerical derivative; for water and 40A x 40A
                  ! box, the systematic error in eta is 1e-5 N/m
rescale="CM"      ! rescaling based on molecules (center-of-mass)
slab.mode=3       ! 1=calculate corrections
                  ! 2=use shape scaling V=const
                  ! (default=all coordinates scaled q-times)
```

### 15.7.4 Surface tension via pressure tensor

If cook was configured with `PRESSURETENSOR=3` and `SLAB`, the surface tension is calculated directly from  $P_t = (P_{xx} + P_{yy} - 2P_{zz})/3$  by

$$\gamma = -\frac{3}{4}L_z P_t$$

For other info see above and see Sect. 15.6. The saturated pressure is given (without cutoff correction) by component `Ptzz`. (Note: in versions 2.4a–2.4l by three times `Ptzz`.)

### 15.7.5 Cutoff corrections in the slab geometry

MACSIMUS includes the slab cutoff corrections, calculated by the algorithm taken from [25]. We write the total interatomic potential as

$$u_{ij} = u_{ij,\text{short}} + \Delta u_{ij}. \quad (15.10)$$

In particular, MACSIMUS implements

$$\Delta u(r) = \begin{cases} 0, & \text{for } r \in [0, c_1], \\ u(r) - C(r^2 - c_2^2)^2, & \text{for } r \in [c_1, c_2], \\ u(r), & \text{for } r \in [c_2, \infty], \end{cases} \quad (15.11)$$

where constants  $C$  and  $c_1$  are determined from the cutoff  $c_2$  and the potential so that both  $\Delta u(r)$  and its derivative  $\Delta u'(r)$  are continuous for all positive  $r$ .

Let  $\zeta_i(z)$  denote the  $z$ -density profile of atom  $i$ . The correction in the averaged potential energy  $E$  (residual internal energy) can be approximated by integration over all (assuming uncorrelated) positions of pairs  $ij$  and then by summing over all pairs

$$\Delta E = \sum_{i < j} \frac{\int \zeta_i(z_i) d\vec{r}_i \int \zeta_j(z_j) d\vec{r}_j \Delta u_{ij}(r_{ij})}{\int \zeta_i(z_i) d\vec{r}_i \int \zeta_j(z_j) d\vec{r}_j}. \quad (15.12)$$

This can be partly integrated and rearranged to a 3-dimensional integral,

$$\Delta E = \sum_{i < j} \frac{\int \zeta_i(z_i) dz_i \int \zeta_j(z_j) dz_j I_{ij}(|z_i - z_j|)}{L_x L_y \int \zeta_i(z_i) dz_i \int \zeta_j(z_j) dz_j}, \quad (15.13)$$

where

$$I_{ij}(z) = \int_0^\infty 2\pi r dr \Delta u_{ij}(\sqrt{r^2 + z^2}) = 2\pi \int_0^{1/\max(c_1, z)} \frac{dt}{t^3} \Delta u_{ij}(1/t). \quad (15.14)$$

In practical evaluation we first calculate, for all atom type pairs  $ij$ , tables  $I_{ij}(z)$  by a fourth-order Gaussian integration over  $dt$ . The same grid given by `slab.grid` is used for both  $I_{ij}(z)$  and  $\zeta_i(z)$ . Then the integrals over  $dz_i$  and  $dz_j$  are numerically evaluated by the trapezoidal rule with the same grid as  $\zeta_i(z)$ . For determining the surface tension correction, term  $\Delta u$  in the above formulas should be replaced by

$$\Delta u \longrightarrow \frac{L_z}{4V} \frac{x^2 + y^2 - 2z^2}{r} \Delta u'(r) \quad (15.15)$$

and then

$$I_{ij}(z) = \frac{L_z}{4V} 2\pi \int_0^{1/\max(c_1, z)} \frac{dt}{t^2} [1/t^2 - 3z^2] \Delta u'_{ij}(1/t) \quad (15.16)$$

in (15.13) gives  $\Delta\gamma$  instead of  $\Delta E$ .

### 15.7.6 Interfacial Gibbs energy by the cleaving method

The implemented method is a modification of the method of [24]. It is turned on by specifying `#define SLAB 2` in the `simopt.h` file and recompiling. Also `PRESSURETENSOR=3` is recommended (not necessary in NVT, but NVT is likely not suitable).

Instead of two walls, we use the cleaving potential in the  $z$ -direction

$$\Phi(\sigma) = \sum_i K \sigma b \left( \frac{z_i - z_0}{\sigma} \right) \quad (15.17)$$

where  $b(x)$  is a bell-like function

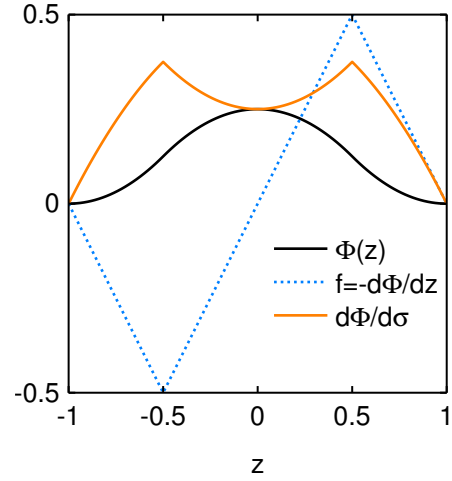
$$b(x) = \begin{cases} 0, & |x| > 1, \\ (1 - |x|)^2/2, & 1/2 < |x| < 1, \\ 1/4 - x^2/2, & |x| < 1/2. \end{cases} \quad b'(x) = \begin{cases} 0, & |x| > 1, \\ x - \text{sign}(x), & 1/2 < |x| < 1, \\ -x, & |x| < 1/2. \end{cases} \quad (15.18)$$

Parameter  $\sigma$  is the half-width of the cleaving potential while  $K/2$  is the maximum steepness. The potential maximum is  $K\sigma/4$ . The figure shows the function and its derivatives for  $\sigma = 1$  and  $K = 1$ .

The force on atom  $i$  in the  $z$ -direction is

$$f_i = -\frac{\partial \Phi}{\partial z_i} = -K b' \left( \frac{z_i - z_0}{\sigma} \right) \quad (15.19)$$

and hence the contribution to the virial of force (and the  $P_{zz}$ -component of the pressure tensor) is  $W_i = (z_i - z_0) f_i$ .



The Gibbs energy as a function of  $\sigma$  is

$$G(\sigma) = -kT \ln \int dV \int \exp(-\beta U - \beta PV - \beta \Phi) dr^N \quad (15.20)$$

and thus

$$\frac{dG}{d\sigma} = \left\langle \frac{\partial \Phi}{\partial \sigma} \right\rangle, \quad (15.21)$$

where

$$\frac{\partial \Phi}{\partial \sigma} = \sum_i K \left[ b \left( \frac{z_i - z_0}{\sigma} \right) - \frac{z_i - z_0}{\sigma} b' \left( \frac{z_i - z_0}{\sigma} \right) \right] = \frac{1}{\sigma} \left( \Phi + \sum_i W_i \right). \quad (15.22)$$

The Gibbs energy is obtained by integration over  $d\sigma$ . The program stores and analyses quantity `Wcleave` normalized by the surface area  $2L_x L_y$ ,

$$W_{\text{cleave}} = \frac{1}{2L_x L_y} \frac{\partial \Phi}{\partial \sigma}. \quad (15.23)$$

In other words,  $W_{\text{cleave}}d\sigma$  is the element of reversible work per surface area needed to increase  $\sigma$  by  $d\sigma$ . The surface cohesion or adhesion energy per surface area is

$$\gamma_{c_1c_2} = \int_0^\infty W_{\text{cleave}}(\sigma)d\sigma, \quad (15.24)$$

where  $c$  stands for a condensed phase ( $s$  or  $l$ ). The interfacial  $l/s$  energy is then

$$\gamma_{l/s} = \gamma_{ls} - \gamma_{ll} - \gamma_{ss} \quad (15.25)$$

It is recommended to plot and integrate the difference of particular averaged forces (reversible work contributions) as a function of  $\sigma$ ,

$$W_{l/s} = W_{ls} - W_{ll} - W_{ss} \quad (15.26)$$

because  $W_{l/s} \rightarrow 0$  for  $\sigma \rightarrow \infty$  faster than the individual contributions.

The following quantities are available:

**cleave.sigma** [default=0] Parameter  $\sigma$ , in Å. Typically should run from zero to several Å in steps by 0.01–0.1 Å.

**cleave.K** [0] Parameter  $K$ , in K/Å. Typical values are 10000–100000.

**cleave.n** [0] Number of cleaving walls, max. 2. Both walls have the same  $K$  and  $\sigma$ , but differ in  $z_0$ .

**cleave.init** [0] If bit  $2^0$  is set (**cleave.init**=1) then the cleaving wall positions given below are used, otherwise the previous versions are loaded. To be used for simulation start. This bit is unset so that next time the wall positions are not changed (similarly to **init** = continue with simulation).

If bit  $2^1$  is set (**cleave.init**=2) then the cleaving walls (for (**cleave.n**=2) are adjusted along with  $L_z$  according to the the  $z$ -components of the pressure tensor using the same **tau.P** and **bulkmodulus**; use **rescale**="ZCM" or similar.

**cleave.z0** [0] The initial position ( $z_0$ ) of the first wall, in the units of  $z$ -box size  $L_z$ .

**cleave.z1** [0] The initial position of the second wall (for **cleave.n**=2).

If two walls are used, **cleave.n**=2, the Andersen (or center-of-mass Andersen) thermostat should be used to avoid mutual “convection” in both compartments.

The barostat should be used in the  $z$ -direction (e.g., **rescale**="ZCM"; thus, **PRESSURETENSOR**=3 configured is required. For **cleave.n**=2 also the wall positions are barostatted to maintain the same pressure in both compartments. Note that the wall positions are stored in the **cfg** file and used unless **cleave.init** is set.

## 15.8 Simulations at walls

The WALL version allows simulations at Lennard-Jones integrated walls (9-3 potential). One wall (**wall.n**=1) at  $z=0$  and two walls (**wall.n**=2) at  $z = 0$  and  $z = L$  are supported.

The GOLD version performs simulations at vicinity of metal (ideal conductor) surface. It is (since version 2.0e) treated as sub-version of WALL. The surface is located at  $z = 0$ , bulk metal at  $z < 0$ , studied configuration at  $z > 0$ . If `wall.n=2`, then another Lennard-Jones (non-conducting) wall is added at  $z = L$ .

Both FREEBC and CUTEELST (see Sect. 15.3) versions are available, the latter is periodic in x and y directions only.

LINKCELL version is available (with CUTEELST). In this case using two walls (`wall.n=2`) is recommended because atom positions  $z \geq L$  are illegal and the program ends with ERROR if a molecule leaves the  $x < L$  slab.

The WIDOM code differs for a slab and provides a z-dependent profile of the chemical potential. WARNING: no cutoff corrections included for WIDOM + SLAB.

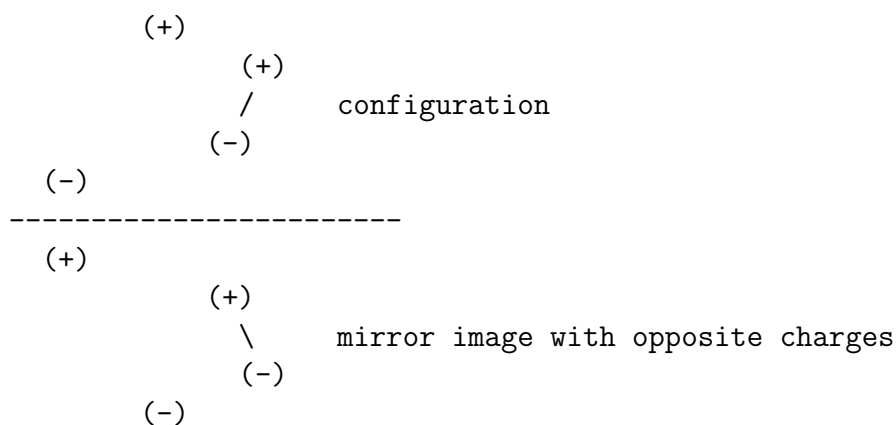
### 15.8.1 Atom-surface force field

The atom-surface force field is averaged (integrated) Lennard-Jones, 11.4, of interaction between a wall atom and a given atom; these parameters are obtained by the usual combining rules from LJ parameters of both atoms, and let rho be the number density of (smoothly distributed) wall atoms. Then the effective wall-atom potential is

$$E = 4\epsilon\rho\sigma^3 \left[ \frac{\pi}{45} \left( \frac{\sigma}{z} \right)^9 - \frac{\pi}{6} \left( \frac{\sigma}{z} \right)^3 \right]$$

### 15.8.2 Atom-metal force field

The electrostatic forces of a charge with a conducting wall (gold) are obtained by the method of “inversion”:



The potential energy:

$$U = \sum_{i < j} q_i q_j \left[ \frac{1}{|r_i - r_j|} - \frac{1}{|r_i^* - r_j|} \right] - \frac{1}{2} \sum_i q_i \frac{1}{|r_i^* - r_i|}$$

where  $\mathbf{r}_i=(x_i,y_i,z_i)$  and the mirror image is  $\mathbf{r}_i^*=(x_i,y_i,-z_i)$ . In the CUTELST version all the  $1/r$  terms are modified (see Sect. 15.3).

(To derive this formula, calculate the electrostatic energy of the ‘double’ system with reflected charged. You will get  $2U$ . Use  $-\mathbf{r}_i^*-\mathbf{r}_j=-\mathbf{r}_i-\mathbf{r}_j^*$ ,  $-\mathbf{r}_i-\mathbf{r}_j=-\mathbf{r}_i^*-\mathbf{r}_j^*$  to simplify.) Note also that  $-\mathbf{r}_i^*-\mathbf{r}_i=2z$ .

Implementation:

Inside all site-site functions, the Coulomb term is replaced by Coulomb+inverted Coulomb. Module `interpot.c` is replaced by `gold.c`, with the same header file `interpot.h`. Boundary conditions macros in `intermac.h` are updated to treat periodic b.c. in x,y only and extra support for inversion is added. The SUM*i* part (along with LJ, see below) is in `force.c`, function `goldforces()`.

CAVEAT: optimized water potentials not (yet) implemented, `gold` must be run with option `-x`.

### 15.8.3 Using WALL and GOLD versions

Programs `wall/gold` derive `rho` from real mass density `wall.rho` in  $\text{kg/m}^3$  which is available in data (default `wall.rho=19320` corresponds to gold). The Lennard-Jones parameters of the wall material must be listed as the LAST atom in table Lennard-Jones in the system `ble-file` and at the same time MUST NOT appear as normal atom in molecules. This can be accomplished by putting it the last position when using `blend` and specifying `N[LASTINDEX]=0` or `blend` option `-n0`, e.g.

```
blend MOLECULE(s) -n0 au
```

where `au.che` is just

```
gold atom
MAU ! for charmm21.par
AU  ! for charmm22.par or GROMOS (to be implemented)
```

WARNING: this requires that the correct data for gold are present in the parameter set (`charmm21.par`)! The current value is

!	alpha	EMIN	Rmin	
!	[A <sup>3</sup> ]	(kcal/mol)	(A)	
MAU	1	-0.039	3.293	! UFF -- JK

based on UFF [J. Am. Chem. Soc. 114,10024 (1992)]. The value for `alpha` (used in the combining rule) is just guess. We need better data from literature, and/or adjust this for given task. This value affects the energy of adsorption.

NOT TESTED: interaction with `blend -n-1`



## 15.8.4 Wall versions and integrals of motion

Minor versions:

FREEBC free b.c. in  $x$ ,  $y$ , and  $z > 0$

NIBC nearest image periodic b.c. in  $x$ ,  $y$ , free in  $z > 0$  (NOT TESTED)

CUTELST cutoff electrostatics: periodic b.c. in  $x$ ,  $y$ , free in  $z > 0$

EWALD Ewald summation not supported

Integrals of motion:

**x,y periodic and FREEBC:** momentum in  $x$  and  $y$  is conserved and is periodically reset to zero,  $z$ -momentum is unchanged; center-of-mass is adjusted to  $(x, y) = (0, 0)$

**FREEBC:** angular momentum in  $z$ -direction is conserved and is periodically reset to zero,  $x$ ,  $y$  are unchanged

Degrees of freedom to subtract for conserving quantities (see `siminit.c`):

- GOLD + FREEBC, no central force (option `-o0`): 3
- GOLD + FREEBC, central force (option `-o#`): 1
- GOLD + periodic: 2
- Fixed atoms (option `-k`) shrinks this to 0 (NOT GENERALLY CORRECT).

## 15.8.5 Initial configuration

Initializer:

- only `init=3` available, both with and without `config` option
- `config` (= option `-n-1` in `blend`): the configuration is shifted in the  $z$ -direction so that the atom with minimum  $z$  coordinate is `wall.minz` from the surface (`wall.minz` is available in the data, default=3). `wall.minz<=0` turns off this function. Suitable for a ‘configuration’ prepared by `blend`.
- standard (`-n-1` not used): when initializing by ‘random shooting’ condition, `min z > wall.minz` is satisfied
- CAVEAT: MC not implemented

## 15.8.6 Input data for the WALL versions

Data (to be placed in def- or get- file):

`wall.g` (0) Gravity towards the  $z=0$  (the golden plate), in  $\text{\AA}/\text{ps}^2$  (=internal prog. units, see `sim/units.h`). Useful especially while initializing to adsorb molecules on a wall. Practical range is up to 10 with intensive thermostatting, otherwise much less.

`wall.minz` (0) minimum  $z$  for GOLD initial configuration, in  $\text{\AA}$

`wall.n` (0) Number of walls. 0=none, 1=bottom, 2=top and bottom (slit pore)

`wall.rho` (19320) mass density of wall atoms, in  $\text{kg}/\text{m}^3$

`wall.scalez` (1) factor by which the randomly shot  $z$ -coordinates are multiplied while GOLD initializing (i.e., when `init=3`)

`wall.rep` (2) `wall.rep=1` causes the  $z=0$  wall repulsive (by using only the  $r^{-9}$  term from the integrated Lennard-Jones potential),  
`wall.rep=2` causes the  $z=L$  wall repulsive, `wall.rep=3` causes both walls repulsive. WARNING: `wall.rep>1` does not imply `wall.n=2`.

## 15.8.7 Wall visualization and more

Option `-\number` was added to `show` to show the surface as `number*number` mesh. Version `-\number\` shows a slit pore, negative argument uses wall made of atoms. Note that `\` should be protected from the shell by doubling. The size is derived from the size of the system (if no box size is specified either in the plb-file or by option `-l`). Example:

```
molcfg MOLNAME1 [MOLNAME2 ...] SIMNAME
show -\\5 -I% SIMNAME
```

Full example (OPTIONS ARE OBSOLETE):

```
blend -o test -p ala -h hoh au
molcfg ala hoh test
goldcut test -y-1 -q-100 -x
```

Bugs/Plans:

- Unify `interpot.c` and `gold.c` into one module???
- POLAR version in some cases unnecessarily repeats calculation of `sqrt()`

WARNING: `LJcutoff<0` means that the effective LJ cutoff is  $-\text{LJcutoff} \times \text{van der Waals diameter}$  (not LJ sigma)

BUG to fix: some extra terms proportional to the shift of the cut-smoothed-and-shifted electrostatic potential are present (in functions `XQQ`, `XQQM`, and `*QQ14`). This affects internal energy and pressure, but not trajectory. Not fixed (yet) for gold!!!

## 15.9 Anchor sites and axes and measure forces

The ANCHOR version allows selected atoms or the center-of-mass to be fixed and at the same time to measure forces and torques acting on them. Run time option `-k0` must be used. Cf. version with harmonic springs, see Sect. 15.1. Input data are in file `simname.fix` in the following format: (warning: the format changed in version 2.0h)

```
site MOLECULE SITE X Y Z # keep site SITE of MOLECULE at (X,Y,Z)
                                # (max 1 site per molecule allowed)
cm MOLECULE X Y Z # keep center-of-mass of MOLECULE at (X,Y,Z)
inert MOLECULE X Y Z # keep principal axis parallel to direction (X,Y,Z)
axis MOLECULE SITE X Y Z # keep vector (center-of-mass,SITE)
                                # parallel to (X,Y,Z)
pair MOLECULE SITE1 SITE2 X Y Z
                                # keep vector SITE1-->SITE2 parallel to (X,Y,Z)
triple MOLECULE SITE1 SITE2 SITE3 X Y Z
                                # keep plane given by (SITE1,SITE2,SITE3)
                                # perpendicular to (X,Y,Z)
group MOLECULE NS SITE_1 ... SITE_NS X Y Z
                                # keep center-of-mass of given group
                                # groups may repeat and even overlap, but no
                                # check of mechanical inconsistency is made
# measurements only (no constraining), of center-of-mass:
r MOLECULE # print position of MOLECULE
v MOLECULE # print velocity of MOLECULE
x MOLECULE # print acceleration of MOLECULE
force MOLECULE # measure force to (center-of-mass of) MOLECULE
moment MOLECULE # measure torque to MOLECULE (center-of-mass based)
```

where MOLECULE is the molecule number (1st molecule has n=0), SITE is the site number. X Y Z is the position or unnormalized direction to be fixed. The keywords can be abbreviated to 1 letter. Line beginning with non-letter (e.g., space) is treated as comment and ignored, i.e., the keywords must start from column 0.

The last two commands, **force** and **moment**, measure the force and the moment of force (torque), respectively, acting to the given molecule. Nothing is constrained. Both quantities are related to the center-of-mass. These measurements are based on atom forces. All other commands constrain some quantity. At the same time, the constraint force is measured. All these forces are (in SI units) printed in file `simname.anc` with a self-explaining header.

Commands **site** and **cm** cause the constraint for position to be satisfied. **cm** is implemented after the Shake iteration because the whole molecule is shifted to satisfy the constraint; at the same time, the constraint force is measured. **site** is implemented within the Shake iteration.

Commands **inert**, **axis**, **pair**, **triple**, cause the molecule to rotated around the center-of-mass to satisfy the constraint; this is exclusive with **site** option, but can be combined with **cm**. It is implemented after Shake because the center-of-mass is

preserved.

If both **force** and **cm** are given, both the force on the center-of-mass and the constraint force (on the center-of-mass) are printed. The constraint equals minus the force.

If both **moment** and either of **inert**, **axis**, **pair**, **triple** are specified for a rigid molecule, the moment with subtracted projection to the given axis equals minus the constraint force with precision proportional to  $h^2$ . If the molecule is not rigid, both quantities differ, but in mechanically well-defined systems (plane defined by atoms far from each other, the principle moments of inertia differing sufficiently) the difference consists of small and fast oscillations.

Example:

```
s 0 1 1 1 1
c 1 5 5 5
t 1 0 1 2 1 1 0
```

Site 1 of molecule 0 is (1,1,1). Center of mass of molecule 1 is (5,5,5). Plane given by sites 0,1,2 of molecule 1 is perpendicular to (1,1,0).

In the input data, **drift**=0 must be specified and **omegac**<1 is recommended. After these coordinates have changed, **drmax** should be set to a small negative value. Suggested value is **drmax**=-0.03, but the final temperature will be too low and equilibration with larger **|drmax|** or none must follow. If **init**=3 and option **-k0** are combined, the initial configuration satisfies these constraints and setting **drmax** is not needed (does not apply for group constraints).

Caveats:

- Some combinations of constraints are illegal even if there is no apparent reason for it.
- Implemented for one version of Shake (SHAKE=1) only – check `simopt.h`
- Constraining axes for flexible molecules is not a well-defined task. Use with caution.
- Note also that rigid molecules generated by **blend** with the dep-files species.dep do not preserve the inertia tensor.
- If **init**=3 is used, the generated configuration satisfies the positional constraints, but not the axis. Equilibrating with **drmax** must follow.
- The above point, or if the constraints are changed (with **drmax**), flexible molecules may adopt fast internal vibrations which (see Sect. 12.1) are only weakly coupled to the rest. Using the Andersen thermostat may help; in case of molecules with rigid bonds, this is incorrect and can only be used temporarily.
- Inconsistent group constraints may lead to integrator crash or infinite SHAKE loop.

## 15.10 Analyze pair energies

Special cook version MARK:

- calculates pair energies between groups of atoms
- one group is the zeroth-group, there are NG (NG<254) other groups
- energy (in vacuum) of n-th group vs. zeroth-group is calculated and recorded in CP-file (note:1st 2 columns have bad heading)

Source of data:

- if option `-r`: files `simname.1`, `simname.2`, ..., `simname.no`
- if not option `-r`: playback file `simname.plb`, frames 1..no

Compiling:

- `#define MARK` needed in `simopt.h`
- link with module `mark` (`mark.h`, `mark.c`)

Bug: inefficient, calculates also pairs that are not needed

Usage:

- prepare file `simname.mkr` (option `-r > 0`) or `simname.mkp` (option `-r ≤ 0`) in the following format (ID is group ID derived from 1st atom ID):

```
NG
0 ID GROUP
1 ID GROUP
2 ID GROUP
...
```

For example (abridged):

```
10
 0 ASN64aC      0
 1 ASN64aCA     0
 2 ASN64aCB     0
...
633 PR010cCG    10
634 PR010cN     10
635 PR010cOC1   10
636 PR010cOC2   10
```

NOTE: in the playback version (no option `-r`), only those atoms actually stored in the playback may be listed.

Option -r summary:

-r0 reads playback simname.plb, uses simname.mkp

-r1 reads configs simname.1 ..., uses simname.mkr

-r-1 reads configs simname.1 ..., uses simname.mkp

NOTES (added 3/2000):

- IDs in the mkp,mkr files are used as info only, they do not select anything nor are checked
- The N[] in the def-file used must conform the original def-file.
- Atom not to be included in any group simply have the group field blank (e.g., "636 PR010c0C2 ")
- simname.mkp and simname.mkr files may be shorter (i.e., not listing all atoms). Not listed atoms are simply ignored. Only if the actual plb file is shorter, a warning is printed.
- Typical usage is for protein in water. For energies without water (not solvated), mkr and mkp files can be created from the protein mol-files (water is ignored). To include water, use a mol-file generated by `molcfg`.

## 15.11 Viscosity by shear stress

Two methods for viscosity determination are supported. The equilibrium method is based on the Einstein formula, See Sect. 14.5. It is generally more accurate.

The NEMD simulation is performed in a rectangular box of sizes  $L_x \times L_y \times L_z$ , where  $L_x = L_y$ . The recommended  $L_z = 3L_x$ . The external acceleration acting on particles in the direction perpendicular to the  $z$ -axis is cos-modulated

$$\ddot{r}_i = C_f \cos \frac{2\pi z_i}{L_z}.$$

We use forces in the direction of vector (1,1,0). Then

$$f_{x,i} = f_{y,i} = K_f m_i \cos \frac{2\pi z_i}{L_z}, \quad \text{where } K_f = C_f / \sqrt{2}$$

We also add a small  $z$ -force to all atoms so that the sum of forces over all atoms is exactly zero—normally it slightly deviates because particles are not distributed ideally uniformly.

The Navier-Stokes equation for a steady Newtonian flow of an incompressible fluid reads as

$$\eta \nabla^2 \vec{v} + \vec{f} = 0,$$

where  $\vec{f}$  is the (averaged) external force per unit volume,  $|\vec{f}| = f = \rho C_f$ , and  $\rho = \sum_i m_i/V$  is the mass density. Its solution is a steady flow proportional to the applied force. Systematic contribution to the (thermal) velocities of individual atoms is then

$$v_i = C_v \cos \frac{2\pi z_i}{L_z}.$$

where

$$\eta = \left(\frac{L_z}{2\pi}\right)^2 \rho \frac{C_f}{C_v}.$$

In order to calculate  $C_v$  and consequently  $\eta$ , we define sums

$$S_f = \sum_i f_{x,i} \cos \frac{2\pi z_i}{L_z}, \quad S_p = \sum_i (v_{x,i} + v_{y,i}) m_i \cos \frac{2\pi z_i}{L_z},$$

The final formula for viscosity then adopts the form

$$\eta = \rho \frac{L_z^2}{2\pi^2} \frac{\langle S_f \rangle}{\langle S_p \rangle}. \quad (15.27)$$

During simulation, also the velocity amplitude

$$C_v = \frac{S_f}{S_p} \frac{C_f}{2}$$

is monitored (name `Cvel`) in the convergence profile to check for the steady state.

**Heating rate** is an important control quantity in the simulation. In the steady state flow it is

$$\dot{E} = \frac{dE}{dt} = \frac{1}{2} \int \eta (\nabla v)^2 dV = \frac{V}{\eta} \left( \frac{C_f \rho L_z}{4\pi} \right)^2.$$

It is useful to express it via expected increase in temperature because it will help us to set up the thermostat used to remove the heat. The crudest estimate of the isochoric heat capacity is that of the ideal gas,  $C_V = N_f k/2$ , where  $N_f$  is the number of degrees of freedom of the simulation box. Then

$$C_f = \sqrt{\frac{8\pi^2 \dot{T} \eta N_f k}{V L_z^2 \rho^2}}$$

and we use product `shear` =  $\dot{T} \eta$  as the parameter of the simulation. If the Berendsen (friction) thermostat is used to remove the friction heat, the following quantity

$$\Delta T = T_{\text{kin}} - T = \dot{T} \tau_T$$

a rough but still useful measure of the deviation of the system from equilibrium. Namely, the Berendsen thermostat also uses the ideal gas approximation to relate the correlation time `tau.T` to the friction term.

For larger systems it takes some time until a steady velocity profile develops. Relaxation to this steady state is exponential with the correlation time given by

$$\tau_v = \frac{\rho}{\eta} \left( \frac{L_z}{2\pi} \right)^2.$$

As soon as an estimate of viscosity is known, the recommended strategy to fast finding the stationary state is to simulate time  $t = \tau_v \ln 2$  with doubled parameter  $\chi$  and half  $\tau_T$ .

The shear stress method is activated by the compile-time option `#define SHEAR` and by data `shear`. The value of `shear` is given in SI units [ $\text{kg m s}^{-2} \text{K}$ ]. Typical values are of the order of  $1\text{e}9$ – $1\text{e}10$ .

The viscosity value is a ratio of two variables denoted as `shear:num` and `shear:den`, see (15.27) (the factors are already embedded in these variables).

To obtain reliable viscosity estimates, a series of measurements with different values of `shear` should be conducted and extrapolated to `shear=0`. The theoretical dependence at the limit of `shear=0` is a linear dependence of  $\eta$  to `shear`.

## 15.12 Widom and scaled insertion particle method

`#define WIDOM` required in `simopt.h` The Widom method gives the residual chemical potential, i.e. the absolute chem. pot. is (for monoatomic gas)

$$\mu = \mu_{\text{res}} + \mu_{\text{id}} = \mu_{\text{res}} + kT \ln (\Lambda^3 N/V)$$

where  $\Lambda$  is the de Broglie wavelength,  $N$  the number of solutes and  $V$  the liquid volume. This should be equal the gas abs. chem. pot. of gas at pressure  $p$  (assumed to be ideal gas,  $N_g$  and  $V_g$  refer to the gas phase)

$$\mu = kT \ln (\Lambda^3 N_g/V_g) = kT \ln (\Lambda^3 p/kT)$$

On comparing, the Henry constant with respect to definition

$$p = c K_c$$

(where concentration of gas in liquid is  $c=n/V$  and  $n$  is in moles), is

$$K_c = RT \exp (\mu_{\text{res}}/RT)$$

and  $\mu_{\text{res}}$  is in macroscopic units [ $\text{J/mol}$ ]. For  $\mu_{\text{res}}$  in program units = [ $\text{K}$ ]:

$$K_c = RT \exp (\mu_{\text{res}}/T).$$

The resulting  $K_c$  is in [ $\text{Pa.m}^3.\text{mol}^{-1}$ ]. OTHER UNITS: For Henry law in form

$$p = x K_x$$

where  $x$  = molar ratio =  $n_{\text{solute}}/n_{\text{solvent}}$ , it holds

$$K_x = \rho_{\text{solvent}}/M_{\text{solvent}}*K_c$$



where `rho_solvent` and `M_solvent` are in matching real units. E.g. (SI), [kg.m<sup>-3</sup>] and [kg.mol<sup>-1</sup>] and then `Kx` is in [Pa]. For Henry law in form

$$mm = p \cdot K_m$$

where `mm` = molality = `n_solute/m_solvent`, it holds

$$K_m = 1/(\rho_{\text{solvent}} \cdot K_c).$$

BUG: the Widom method cannot be used with Ewald (k-sum ignored)

`widom.sp` (default=0) species to (virtually) insert (for the number, see `.ble`)

`widom.n` (0) Number of insertions per cycle (WALL version: and per slab)

`widom.mode` WALL version switch, use a sum of:

- 1 also record symmetrized functions (makes sense for `wall.rep=0` or `wall.rep=3`)
- 2 record  $\langle \exp(-[U_{\text{pair}} + U_{\text{wall}}]/kT) \rangle$ , leading to real  $\mu$
- 4 record  $\langle \exp(-U_{\text{pair}}/kT) \rangle$ ; useful for monoatomic molecules where the wall potential is simply added
- 8 record  $\langle \exp(-U_{\text{wall}}/kT) \rangle$  – the angle averaged Boltzmann factor of the molecule-wall potential

`widom.z0`, `widom.z1` Minimum and maximum `z` for which (in the wall/slab geometry) the insertion will be performed.

`widom.dz` The step.

Note: the data are internally recorded by the statistical (`statics.c`) module in variables of names “Widommode z=z” Output printout is in file `SIMNAME.wid`.

NOTE: The Widom method inserts a virtual molecule into the configuration AFTER the last step in standard versions. However, in the linked-cell list version BEFORE the last step (previous linked-list is re-used). Therefore the results of non-linked-cell and linked-cell are not exactly the same even with option `-z` specified. But if you advance linked-cell by 1 timestep, then the Widom results with `noint=1` are exactly synchronized with non-linked-cell.)

Support for gradual (scaled-particle-like) insertion (Widom-like). It is selected by:

`widom.spreal=<species to change>`

then (instead of virtual insertion of species `widom.sp`), all molecules of species `widom.spreal` are changed into `widom.sp`. Since all the coordinates are unchanged, the molecules should differ in partial charges and Lennard-Jones terms only. Note that `widom.n` does not apply. The recommended strategy to calculate the chemical potential is to place the species into a `ble`-file several times with the extra instances

edited by hand to describe the partial molecules during gradual insertions. If LJ terms are changed, it requires to create copies of atom types.

BUG: cutoff corrections are not calculated if SLAB (slab, wall, ..) is active. Final correction can be added.

WARNING: with slab-centering (`drift&4` set), it is generally illegal to have a molecule close to  $z=L[2]$ . The LINKCELL method may then detect coordinates out of range  $[0,L[2]]$ . Although (in the interactive run) it may be sometimes possible to continue, this is not recommended. A protective potential (see `slab.Kz` etc.) should be used.

NOTES to Widom and gradual insertion with SLAB:

- \* the Widom method inserts a molecule to a place with defined z-coordinate with respect to the reference point as defined in the ble-file (with random rotation added). It is the CM (center-of-mass) by default if blend has been used (with the default option `-y`).
- \* Consequently, in the grow-molecule stages, the partial (scaled) molecule(s) should be placed at CM, too. To do this, `cook*` must be compiled with "anchoring" and run with option `-k0`. File `SIMNAME.fix` must contain the molecule(s) being fixed with keyword `cm`, like

```
cm NUMBER X Y Z
```

- \* Equivalently, if the molecule in the blend stage has a selected site as the reference point (e.g., O for water), use either `cook -k0` with `SIMNAME.fix` containing:

```
site MOL_NUMBER SITE_NUMBER X Y X
```

This fixes the site exactly (as a constraint). For fixing the site by a harmonic spring, use `cook -kK` ( $K$ =force constant) with `SIMNAME.fix` containing:

```
SITE X Y Z
```

CAVEATS: `SIMNAME.fix` has a different format with `-k0` and `-kK`,  $K > 0$  (to be changed in future?) `-k0` requires `-DANCHOR`, `-kK`,  $K > 0$  does not

- \* The slab should be centered. For `-kK`,  $K > 0$ , flag `drift&4` is OK. For `-k0`, use whole-slab harmonic force defined by (in input data)

```
center.cmK[2]=100 ! tau=0.7755*(M/cmK) [M=tot.mass in g/mol]  
drift=0
```

(If appropriate, use `center.cmn` for # of molecules; default=all)

- \* For scaled water, `-x` must be used to suppress true water detection
- \* If the anchored (fixed) atoms are not in place (coordinates in `SIMNAME.fix` have changed or `-k0` is turned on suddenly), `drmax=<negative value>` should be used in input data and then released. Recommended value is

`drmax=-0.1`

If there are problems, use lower (in abs. value) value, e.g.

`drmax=-0.03`

however, too low `drmax` may cause too low temperature and after `drmax=0` is reset, the system must be equilibrated. NEVER USE `drmax` IN PRODUCTIVE RUNS!!!

## 15.13 Metals

??????????

# Chapter 16

## File formats

This section is incomplete and occasionally obsolete...

### 16.1 Configuration

Files `simname.1`, `simname.2`, ..., and `simname.cfg` are binary files of variable record length. They consist of records which, in addition to the variable recorded, contain the length of the record (one integer prepended to the record) and the checksum of all bytes in the record (one integer appended to the record). The end-of-file is marked by a zero integer (because no record may have zero length) and by the system time (long integer, time in s since the standard time/date). The last value is useful to recover files after a crash.

A list of variables follows. It applies to cook version > 2.7a; to read an old version, use `cfgconv` to convert.

`int cfgkey` File version. Valid bits are:

2<sup>0</sup>: 1 = version  $\geq$  2.7a

2<sup>1</sup>: 2 = POLAR version

Other bits are not valid for cook *le* 2.7a

`int optionlist[32]` The values of options, -@, -a, ...

`int nspec` Number of species

`int spec[nspec]` Array [nspec] of species data: `int N; int ns;` (N = number of molecules, ns = number of sites).

`rec[*]` Records of form { `int key; int intval; vector vecval;` } in any order.  
The following keys are recognized:

1 `intval=N` (total number of molecules)

`vecval={t,h,Etot}` (running simulation time [ps], time step [ps], total energy [K] )

- 2    intval=total number of sites  
     vecval=L[3] (box sizes, vector in [Å])
- 3    (reserved for cell shape)
- 4    intval = thermostat  
     vecval[0]=RvdW (for van der Waals radius setup)  
     vecval[1]=tau.T  
     vecval[2]=tau.P
- 5    intval = number of cleaving planes  
     vecval[0] and [1] = positions of cleaving planes
- 17   vecval[0]=tau.dip (POLAR only)

ToInt cfg[GearOrder] The configuration, see ToInt in `simglob.h`. `cfg[0]` contains the size in bytes as integer first, then possible void bytes to satisfy alignment requirements, then the Nose variable `log(s)` as double, vector `lambda` for the barostat, vector `aux` reserved, and then `ns` double vectors corresponding to all sites. `cfg[1]` contains the first derivative of `cfg[0]`, multiplied by `h` (differences for Verlet/SHAKE), and so on as in Taylor expansion.

## 16.2 Convergence profile

File `simname.cp` is normal fixed record length file. One record contains `n` floats (single precision) where `n` depends on compilation options (see the printout of `cook`). There are three types of records:

1. Initial record. IF the first record in the `.cp` file starts with CPmark (number lower than `-1e36`) as the 1st number, then the 2nd number contains `n` either in the 1st or 4th byte (to recognize endianness), the 4th or 1st being zero. The remaining  $(n-2)$  4-byte fields in the record contain max 4 letter info; the 1st two info's are assumed to be "Etot" and "T" (kinetic). IF the first record in the `.cp` file does not start with CPmark, the old format with `n=5` is assumed.
2. Time mark. IF the record (not 1st) starts with CPmark, it contains the system time in the last 4 bytes.
3. Otherwise `n` floats follow. The standard meaning of the 1st 5 is:

Etot Total energy; if, however, `Etot<-1e36`, then this record is invalid containing in its last four bytes the system time.

T Instantaneous (kinetic) temperature

Epot Total potential energy

Eintra Intramolecular non-bonded energy

P or rho Pressure or density

## 16.3 Playback file

The playback file is a file of single precision floating point numbers (4 bytes long). It contains a header of two floats, the 1st one being the number of sites **ns** (in the float format) of the molecule/configuration, the 2nd contains

**L=0** free boundary conditions (old format)

**L>0** periodic b.c. - the box size **L** (fixed, old format)

**L=-3** New format enabling variable vector **L**. One frame consists of float vector **L[\*]** first, then the configuration (**ns** float vectors) follows.

The body of the file consists of configurations recorded at regular time intervals. Each configuration contains **ns** records of vectors of 3 floats (**x,y,z**).

The extension of playback files is **.plb**.

# Chapter 17

## Examples

### 17.1 Example 1: Protein in water

This is a continuation of the example from the blend manual, see Sect. 6.1.

You may use several versions of cook for this example, but let us choose a cheap and fast one. Run `configure.sh` from directory `macsimus/cook` and select defaults (i.e., answer `Enter`) to all questions but:

Select algorithm for ELECTROSTATIC calculations:

0 = Cut-and-shift approximation (MACSIMUS style) calculated directly

File `cookce` should be created (“ce” stands for “cutoff electrostatics”).

In the manual for `blend` we prepared file `crambinw.ble` containing one molecule of protein `crambin` and 999 water molecules. The protein has bond lengths constrained, water molecule is TIP3P treated as rigid body. Now we want to simulate this protein.

First, we must prepare the initial configuration. We will run `cook` interactively so that we need a default (`.def`) file only. We name our project `crambin` so that the default file will be `crambin.def`. A commented example of file `crambinw.def` follows:

```
T=310                ! temperature in K
thermostat="friction" ! Berendsen thermostat
rho=1000             ! final density in kg/m^3 (this is the default)
initrho=900          ! initial density
tau.rho=1            ! typical time for initrho->rho condensing
cutoff=15            ! electrostatic cutoff
;                    ! end of data set
```

Now, let us run `cook*` interactively:

```
cookce -s crambinw
```

You will be prompted for data. Enter:

```

tau.T=0.1          ! thermostat time constant - short for start
init="random"      ! random initial configuration of molecules
;

```

The program will print a lot of information—read it! You may see some warnings “dr in 1 timestep reduced” as a consequence of large initial temperature. They will disappear as soon as the system equilibrates.

Time evolution of temperature (incl. graph), potential energy, total energy, density, pressure, and accuracy of constrained dynamics is printed. After some time (when temperature drops below about 350 K), interrupt the calculations by pressing Ctrl-C and selecting ☐=stop.

To generate a longer trajectory, create file `crambinw.get`:

```

init="start"        ! start simulation (from .cfg)
dt.plb=0.1          ! write playback every 0.1 ps
tau.T=1             ! thermostat time constant
no=2000;            ! number of cycles (by h*noint=0.01 ps)
h*noint             ! check: cycle length
;                  ! end of data set
init="append"       ! restart measurements but append playback and .cp
no=1000             ! another 10 ps
quit=1;             ! end

```

Start simulation by command:

```

cookce crambinw

```

The output protocol is `crambinw.prt`. The simulation can be interrupted (with everything saved) either by signalling -2 (-SIGINT) to the process, or by creating file `crambinw.stp`.

To watch the convergence profiles, run

```

showcp -p crambinw

```

if `plot` is installed properly, you should see the time development of quantities during 30 ps.

To watch the trajectory, check that files `crambinw.mol` and `crambinw.gol` have been created; if not, use command like

```

molcfg crambin TIP3P crambinw

```

To show, use (may be used also while cook is running)

```

show crambinw

```

Useful variants include



```
show crambinw -Ob
show crambinw -o396 -I%@ -1 -x10 -y2 -z2
```

The first form omits all waters, in the second 396 is the number of atoms in crambin, the values of shift -x -y -z should be set according to the actual position.

Basic hot keys are: g G to cycle the viewing mode, 1 to start playback, 0 to stop it, i to start again, and mouse.

## 17.2 Example 2: Melting point of a model of NaCl

First, run the cook configurator `configure.sh` from directory `macsimus/cook` and select defaults (i.e., answer Enter) to all questions but:

```
Select DETAILS of boundary conditions:
  s = Slab (includes: z-forces, density profiles, surface tension)
Are all your molecules smaller than half the box size (y/n)?
y
Select algorithm for PAIR FORCES calculations (affects speed):
  l = Linked-cell list method
Request PRESSURE TENSOR calculations:
  3
```

File `cookewslc` should be created (ew=Ewald summation, s=slab, lc=linked-cell).

### 17.2.1 Force field and molecules

We will use the force field [21, 22] with the geometric combining rules (the combining rules are not mentioned in the original paper, but with other combining rules the density of NaCl is wrong). The parameter file called `sea.par` is included in MACSIMUS.

Molecules are simple, `na.che` and `cl.che` should be:

<pre>ion Na+ parameter_set=sea  NAp1</pre>	<pre>ion Cl- parameter_set=sea  CLm1</pre>
--	--

Prepare also a cluster `na4cl4.che`:

```
Na4Cl4
parameter_set=sea


NAp1 NAp1 NAp1 NAp1

CLn1 CLn1 CLn1 CLn1
```

### 17.2.2 Crystal Na<sub>4</sub>Cl<sub>4</sub>

Use blend:

```
blend -g -y13 na4cl4
```

where -y13 is a sum of 1=show centered, 4=save with positive coordinates, 8=put in a box. Prepare a crystal  $2 \times 2 \times 2$  and place it so that it is parallelly with the axes, then save by .

### 17.2.3 Preparation of data for the simulation

To create the parameter file for simulation, `nacl.ble`, use:

```
blend -o nacl na cl
```

Then, convert `na4cl4.plb` to the cook format:

```
plb2cfg na4cl4.plb nacl.cfg 8
```

### 17.2.4 Crystal Na<sub>108</sub>Cl<sub>108</sub>

Use the following def-file, `nacl.def`:

```
! cookewslc nacl -s
n=108                ! auxiliary
N[0]=n N[1]=n        ! numbers of Na+ and Cl-

rho=2160              ! reference density [kg/m3]

cutoff=8.4            ! elst cutoff [A]
LJcutoff=cutoff       ! LJ cutoff [A]
rdf.grid=20           ! grid for RDF
el.epsk=1 el.epsr=0.2 ! Ewald summation accuracy [K/A]
noint=40 h=0.1/noint  ! number of steps/cycle and timestep [ps]
no=100                ! number of cycles
dt.plb=1              ! how often to write playback [ps]

thermostat="Andersen" ! Maxwell-Boltzmann-like thermostat at random times
T=300                 ! temperature [K]
tau.T=1               ! time constant of the thermostat [ps]

P=101325              ! pressure [Pa]
tau.P=100             ! constant of barostat; less for a liquid

init="start"          ! start from a prepared configuration
;
```

The microcrystal  $\text{Na}_4\text{Cl}_4$  (in `nacl.cfg`) will be repeated (3 times in x,y,z) and then shrunk by

```
cookewslc nacl -s -[333
```

with the following data entered:

```
no=50          ! 50 cycles
tau.rho=1      ! shrink fast to a cube of given density
tau.P=0        ! no barostat now
;
tau.rho=0      ! density control canceled
tau.P=50       ! barostat
;
quit=1;        ! the end
```

Some warnings during initialization can be safely ignored.

To follow convergence, use:

```
showcp -p nacl
show -I% nacl
```

## 17.2.5 Equilibrium simulation of the crystal

Interactively:

```
cookewslc nacl -s
```

with input data:

```
no=100;
quit=1;
```

or in the batch mode

```
cookewslc nacl
```

where the input data should be put into `nacl.get`.

Analyze the results by:

```
showcp -p nacl
show -I% nacl
rdfig nacl pu
```

The last command will show the radial distribution functions.

## 17.2.6 Melt

Make a copy of the configuration:

```
cp nacl.cfg nacl-1.cfg
```

and simulate at a higher temperature:

```
cookewslc nacl nacl-1 -s
```

with the following data

```
tau.T=0.5 T=2000 tau.P=20 no=50;  
;  
;
```

repeat ';' several times until the system is equilibrated. Watch the trajectory:

```
show nacl nacl-1 -I%---i
```

and restart again (do not forget T=2000) and watch the rdf:

```
rdfg nacl-1 pu
```

## 17.2.7 Melt-crystal equilibrium

1. Choose a temperature (the melting temperature is 1340 K; try various temperatures below and above) and repeat the crystal simulation. Suggested control data are `tau.T=0.5 tau.P=20` in the beginning, then followed by `tau.T=1 tau.P=100`.
2. Copy the configuration

```
cp nacl.cfg slab.cfg  
cp nacl.def slab.def
```

and edit `slab.def` as follows:

```
n=108*3          ! 3 times as much  
rho=1800          ! ref. density (estimate)  
L[0]=1 L[1]=1 L[2]=3 ! box sizes ratio  
T=TEMPERATURE    ! your choice  
tau.P=20          ! faster barostat  
rescale="ZCM"    ! box changed in z using pressure tensor component
```

3. Replicate the box and sort by z

```
cookewslc nacl slab -s -[113
```

with the data:

```
sort="z" ! sort (mlecules separately) by z-coordinate
no=0;    ! no simulation - write only
quit=1;
```

#### 4. Melt half of the box:

```
cookewslc nacl slab -s -j108
```

where -j108 means that first 108 molecules ( $\text{Na}^+$ ) are fixed. Use data:

```
T=5000 ~~~~~ ! melting
tau.T=0.2 tau.P=200
no=50;
tau.P=15      ! z-pressure
T=TEMPERATURE; ! equilibrate at T chosen
quit=1;
```

To have a look:

```
show slab "-I%----*****yirrr=" -l -z12
```

#### 5. Final run

```
cookewslc nacl slab -s
```

with data no=200; and more.

# Chapter 18

## Source C files and making cook

The program is written in ANSI C. All source files are placed in three directories. Cook versions are compiled (best via script `configure.sh` from a subdirectory of `cook`, see Sect. 8.3).

`gen` General software

`sim` Simulation software

`cook` Particular `cook` source code

All these directories should be subdirectories of the same directory. Selected modules are commented below:

### 18.1 General software in directory `gen`

`alloc.c alloc.h` Smart allocation routines

`calc.c` The calculator, `#included` by `ground.c`

`linregr.c linregr.h` Linear regression

`conjgrad.c conjgrad.h` Conjugate gradient method for sparse linear eqs.

`cputime.h cputime.c` Time measurements

`draw.c draw.h drawx11.c mydraw.c mydraw.h` Graphics

`gear2.c gear.c` Gear integration (`gear.c` is used for the Hamiltonian formalism (not active anymore).

`ground.c ground.h` I/O extension (get/put data), scrolling, basic macros, ...

`options.h` Some argument parsing macros

`rndgen.c rndgen.h rndgeni.c` Random numbers, 4 tap register-shift

`statics.c statics.h` Statistical analysis of correlated time series  
`varfile.c varfile.h` Files of variable record lengths (a relic from times of unreliable disks)

## 18.2 Simulation software in directory `sim`

`*/sitesite.c */sitesite.h` Various site-site interactions, e.g., Lennard-Jones (`lj`), Buckingham (`buck`), etc.

`anchora.c anchorg.c anchorm.c anchors.c` Support for anchoring, i.e., fixing atoms or some degrees of freedom and measurements of constraint forces

`cluster.c` Clusters calculated from bonds defined by distances

`constrd.c constrd.h` Constraint dynamics (Hamilton, Lagrange, SHAKE)

`cpmark.h` Mark for convergence profile files

`elst.c elst.h` Electrostatics switching module, #includes `erfc.c` (functions needed for the real-space part of the Ewald sums), `cutelst.c`, `cutelstd.c`, `fgelst.c` (versions of short-range electrostatics).

`ewald.c ewald.h ewaldprt.c ewpass1.c ewpass2.c ewpass2m.c`  $k$ -space part of the Ewald sums, parameter setting, estimates and tests of cutoff errors

`forces.c forces.h` Forces calculation (not for the linked cell list method)

`gear2pol.c` Version of `gear2.c` for POLAR

`interpot.c interpot.h internp.c interpol.c` Non-bonded force terms (LJ + real-space part of Coulomb)

`intrapot.c intrapot.h` Intramolecular bonded force terms (angle bending, torsions, ...)

`main.c` The main (options, i/o, run control, ^C handler, ...)

`norm.c norm.h` Normalizing and rescaling a configuration, constraint errors and corrections

`rdf.h` Typedef for radial distribution functions

`rhs.c rhs.h` r.h.s. of the equations of motion (forces+constraint dynamics)

`simcg.c simcg.h` Support for using `conjgrad` for the constraints

`simgear.c simgear.h` Gear predictor-corrector method

`setss.c setss.h` Set site-site interactions, calculate Lennard-Jones cutoff corrections

`simglob.c simglob.h` Typedef's of basic control structures and definitions of widely used global variables

`simmeas.c simmeas.h` Measurements, control of convergence profiles, etc.

`simils.c simils.h` Initialize/Load/Save configurations incl. playback

`siminit.c siminit.h` Initialize site and molecule tables

`simmeas.c simmeas.h` Support for measurements of convergence profile, radial distribution functions, dihedral angle distributions

`sfdx.c sfdx.h` Structure Factor, Diffusivity, cross-section

`units.h` Program's internal units of measurements

`vector.h` 3D vector macros

`xforces.c xforces.h` Extra forces (slab, central, fix atoms)

## 18.3 Particular cook source code

`cooknmea.c cookmeas.c cookm.c cookmm.c` #included by `simpot.c`

`simdef.c simdef.h` Reads `blend`-file and does some other initializations

`simget.c` #included by `main.c`, cook-specific get data

`simpot.c simpot.h` Molecule-molecule forces

`lcforces.c lcforces.h` Replaces `peoforce` #ifdef `LINKCELL`

`linklist.c linklist.h` Linked-cell list method support

`parinter.c` Support for parallelization of non-bonded forces

`simopt.h` Compilation options (a set of #defines). May be placed in your project directory.

## 18.4 Making cook

The following #defines are global and must apply (e.g. as options `-D` in macro `OPT` in `metamake` or the generated `makefile`, or in Compile/Define menu in Turbo C) to ALL modules compiled.

`DOS` Must be #defined for 16-bit version and Turbo C graphics

`SCR` Optional — to enable scrolling screens of output



Simulation-specific `#defines` are listed in file `simopt.h`. They do not affect the general modules. File `simopt.h` is so well documented that it is not necessary to repeat it here. A sample `simopt.h` is in directory `cook`. It can be edited/copied/moved to your project directory; if your project directory differs from `cook`, a version of `simopt.h` in it has higher precedence.

Remember only that you must set `#define FREEBC` if you want to make `cookfree` (version for free boundary conditions) and `#define LINKCELL` if you want to make `cooklink` (version with Ewald summation and linked cell list method). Both `#defines` must be disabled for `cook` (version with all-pairs method for calculating pair interactions and Ewald summation).

Compiling UNIX and GCC/EMX(DOS or OS/2) versions:

1. Compile `makemake.c`
2. Edit file `metamake`: write the names of directories of your project, general, simulation, and `cook` modules to statement `!dir` and the fully qualified name of their parent directory to statement `!home`. Your project directory (if differs from `cook`) must be first to have the highest priority when searching for `simopt.h`.
3. Run `makemake u` for the UNIX version, `makemake e` for the EMX version (the difference is only in `.exe` extension for EMX).
4. Run `make`.

Compiling Turbo-C DOS version (obsolete):

1. Compile `makemake.c`
2. Edit file `metamake` as above; remember, that the compiler is `tc.exe` or `tcc.exe` irrespective of the value of `CC`. Write the path of TC-directory as `!tcdire=path` statement.
3. Run `makemake l` (for using `make/tc`), or `makemake P` (for using Turbo C 2.0 IDE).
4. Edit `turboc.cfg` or set IDE options (Directories, Large memory model, `#define DOS`, etc.)
5. Run Turbo `make -fmakefile.dos` or load `tc.exe` and make the chosen project (file `cook.prj`).

## 18.5 Library interface

This Section is obsolete!

Installing:

1. in **metamake**, add an 'auxiliary' executable identical to the chosen version (e.g., **cookfree**) with module **main** replaced by **mainlib** (more replacements in future to optimize the functions of interest). Then, add a library making rule, using the respective variable **OBJ#**, where **#** is the consecutive number of the 'auxiliary' executable to be made. Example (**cookfree4lib** is the 2nd executable):

```
-----
# will generate OBJ2, to be used for cookfree.a
!cookfree4lib : alloc conjgrad ground rndgen0 statics varfile \
  bitfile cputime \
  peocg peogear interpot setss intrapot sitesite peonorm \
  peoforce constrd peorhs peostruc pakcp \
  simils siminit mainlib simpot simdef water sinmeas sfdx linregr

# this is for GNU ar, may differ on other systems
cookfree.a : $(OBJ2)
        ar rcs cookfree.a $(OBJ2)

# and this is for compiling a test 'testlib.c'
# not complete..
testlib.o : testlib.c $(D2)/ground.h $(D2)/func.h $(D3)/peostruc.h
        $(CC) -o testlib.o -I$(D1) -I$(D2) -I$(D3) -I$(D4) \
        -I$(D5) $(OPT) -c $(D1)/testlib.c

testlib : testlib.o cookfree.a
        $(CC) -o testlib testlib.o cookfree.a $(LIBOPT)
```

2. make a blend file listing molecules of interest, e.g.

```
blend -_6 -o myblefile peptide -h hoh
```

3. in your program, use the following header files

```
#include "simglob.h"
#include "mainlib.h"
```

4. assign options, if necessary, e.g.

```
option('u')=100000;
```

to cause all bonds be soft (harmonic)

```
option('u')=0;
```

to turn off bonds (this is the default)

5. call the initializer

```
initcook(BLEND_FILE,CUTOFF,LJCUTOFF);
```

where CUTOFF=electrostatic cutoff (for free b.c. is sharp), LJCUTOFF=Lennard-Jones cutoff (will be smooth); negative LJCUTOFF=in multiples of LJ van der Waals radius

6. now you may use the following functions (see also `mainlib.c` and `mainlib.h`):

- energy of species `sp` (numbered as in the blend file, starting from 0) with coordinates given by `r[0]..r[#_OF_SITES-1]`:

```
double energy1(int sp,vector *r);
```

- pair energy of species `sp1..sp2` (=sum of all pair interactions). Does not contain internal terms of `sp1` and `sp2`:

```
double energy2(int sp1,int sp2,vector *r1,vector *r2);
```

Implementation:

A ‘configuration’ is build up containing just one molecule of each species in the ble-file. Functions calculating energy just copy the coordinates to the ‘configuration’ and then call the function according to the table of functions (`charmm1`, `charmm2`, `TIP3P`,...). Forces are thrown away.

Caveats (to be cleaned later...):

1. The library just contains everything, including many modules that are not needed.
2. There are many global variables with ugly short names (`t`, `T`, `a`,...). Because of 1,, many of them are not needed at all.
3. The implementation is inefficient:
  - unnecessarily calculates also forces which are thrown away
  - uses extra copying

## Part III

## Utilities

This manual page describes various utilities and supporting software.

Note that a brief but up-to-date help on most utilities can be obtained by running the utility without any parameter. (A help for `ray` is obtained by `ray -h`. Often, more info can be found in the source.

Utilities marked by \* in front of the \*Source field need a make file (see Sect. [19.1](#)), otherwise they can be compiled by a command written in the first line of the source file.

Most utilities also print names of related utilities as `See also:.`

# Chapter 19

## General utilities

### 19.1 makemake: Makefile and project interface

Projects are in MACSIMUS written in files called `metamake`. Utility `makemake` then finds dependencies and converts these files to standard makefiles (or Turbo C projects).

Installing MACSIMUS software thus typically requires the following steps:

1. Edit the project `metamake`. The sample `metamake` files contain enough comments...
2. Running `makemake` may be as simple as

```
makemake linux
```

or in case of project- and system-dependent switches something like

```
makemake unix digital polarlj
```

3. If there are no errors, run `make` on the requested target (s), e.g.

```
make blend pdb
```

See comments in the source file for details!

Source: `c/makemake.c`

### 19.2 plot: Plot a graph (with formulas and mouse-rescaling)

Plots graphs of tabular ASCII data. This utility allows to plot formulas as well as use a mouse to rescale — it did so several years before `gnuplot` supported these features. It has developed from the viewer of the NSK project and I consider it as the most useful general program I have written.

`plot` accepts input files in similar format as `gnuplot`: of white (SPACE, tab) separated numbers. Comment lines are marked by `#` in the 1st column. When plotting by lines, the comments as well as blank lines break the plotted curves.

Simple examples:

```
plot sim.g          # plots file sim.g: column 2 (y-axis) vs. col. 1 (x)
plot sim.g:2:3      # plots column 3 vs. column 2
plot sim.g:0:3      # plots column 3 vs. consecutive numbers 0,1,2,...
plot sim.g:0:3:p     # as above, using points
plot sim.g:0:3:p:4   # as above, using points w. error bars in col. 4
plot sim.g:A:B^2     # plots col. 2 squared vs. col. 1
plot :2:3 *.g        # plot all *.g files using cols. 2 and 3
plot "[0:10]" "[1000]:A:sin(A)" # plot function y=sin(x)
```

Usage in detail:

```
plot [RANGE] FILE-ARG [FILE-ARG] ..
```

where

```
FILE-ARG = { [FILE]:X:Y  [FILE]:Y  [FILE]:X:Y:STYLE[:DY] | @RESPONSE-FILE }
RANGE = [FROMX:TOX, FROMY:TOY]
```

In the RANGE, the brackets `[ ]` are part of the argument. If RANGE is missing (or some data in it are missing), the missing ranges are data determined from FILES (maxima and minima).

```
FILE = { FN | -FN | [POINTS] | [POINTS:FROMX:TOX] | - | @ }
```

FN file to draw

-FN file not to draw (but this argument is considered when advancing colors or parsing the arguments after `:`). [Note: this looks strange and may be removed in future]

[POINTS] ( `[ ]` are parts of the argument): the same as file of POINTS+1 points in interval [FROMX, TOX]. Use `A` in expressions for `x` or `@` for numbers 0..POINTS. (`x n` can be still used, too, but are deprecated.)

- take data from stdin (this argument may repeat because `plot` creates and uses a temporary file of input stream)

dummy argument (useful to set X,Y,STYLE for a consequent set of files)

X

Y The x- and y-coordinate to draw.

DX error bar, also asymmetric [FILE]:X:Y:STYLE:DYFROM:DYTO Allowed primitives are:

1, 2, ... column

#1, #2, ... the same as above (but usable in expressions)

A, B, ... the same as above (but usable in expressions)

x, y, z the same as A, B, C

01, 02, ... the same as 1, 2, ... but hotkeys 0, 1, ... are disabled

+1 or (1) etc. constant 1 (not column 1)

0 or or #0 or n counter: is 0 for the 1st line, 1 for the 2nd, etc. It is reset to zero if a blank line or comment is encountered in the file.

In addition, expressions using numbers, parentheses, variables mentioned above, and simple functions can be used.

**STYLE** It is a string of chars:

- lines (this is the default, but when - is given explicitly, it cannot be changed by hot keys)
- = thick lines
- d dotted lines
- . pixel-size dots
- p dots or very small circles
- P small circles
- o circles
- O larger circles

Colors are normally advanced for consecutive files in order: White Yellow Cyan Magenta Green Red Blue Brown Darkcyan Darkmagenta Gray Darkgray Darkblue. This can be changed by modifiers:

- c set White
- cc set Yellow
- ccc set Cyan, etc. (cumbersome — to be changed/extended)
- d hotkey SPACE enabled even for files with explicitly determined type of line/point
- C toggle: turns off and on advancing color (since first used, the color is the same for subsequent files until C is used again)

Type of line/point must be explicitly used with c C. Missing arguments FILE,X,Y,STYLE retain their values from previous arguments. The initial default is @:1:2:-.

Environment:

**PLOTGEOMETRY** X11 only: The initial plot window geometry, e.g. 300x200-1+1. The default depends on the display size



**PLOTNAME** X11 only: The name of the window and icon. Useful when **plot** is called from scripts or other programs

**LAZYX11** X11 only: The value must be an integer. Makes X11 slower but suppresses unnecessary redrawing on slow networks.

**PLOTINIT** X11 only: The sequence of hot keys to execute. E.g., to write a PostScript file **plot.ps** with the graph, set **PLOTINIT** to **#Q**.

**PRTSCR** DOS only: Output file/device for printscreen (hotkey @). Default=LPT1.

Mouse:

**drag left button** rectangle to rescale (zoom in)

**click left button** place file information (delete it by hotkey Ctrl-D or Del)

**double click left button** Print cursor position in current x,y coordinates. If environment variable **TOCLIP** is set, put the data to the X11 clipboard.

**click middle button** quit

**click right button** redraw

Hot keys:

**h**

**? Help**

**q**

**e**

**ESC Quit**

**K Kill** all currently launched programs **plot**. Requires script (utility) **Kill**, accepting one argument — the name of the program to kill. My script looks like:

```
ps u | fgrep $1 | fgrep -v Kill | cut -c1-15 | \
  lemon jiri "-kill "$2 > /tmp/KillL
chmod +x /tmp/KillL
rm /tmp/KillL
```

**+** Zoom in (enlarge details)

**-** Zoom out

**u** Undo last zoom or scaling

**X** Zoom in in x direction

- x Zoom out in x direction
- Y Zoom in in y direction
- y Zoom out in y direction
- i Reset the initial (min-max or specified) scaling
- r Redraw
- R Round coordinates up so that the ends of the coordinate axes are ‘round numbers’.  
E.g., if the range of the x-axis is [0.1278:0.333], it will become [0.12:0.34].
- R Round coordinates (to the nearest ‘round’ number) so that the ends of the  
coordinate axes are round numbers. E.g., if the range of the x-axis is  
[0.1278:0.333], it will become [0.13:0.33].

**cursors** Move viewpoint

**Ctrl-W**

**PgUp** Place file or argument info to the top

**Ctrl-V**

**PgDn** Place file or argument info to the bottom

**Ctrl-D**

**Del** Remove file or argument info

**= \ |** Standard thin line, points off

**:** ; Dotted line, points off

**l**

**L** Toggle line style or change thickness (points unchanged)

**p**

**P** Toggle points (circles) off or change diameter

**g**

**b** Toggle grid and labeling off or thickness (not for PostScript)

**SPACE** Toggle style lines/circles/circles+lines. Useful!

**!** Dump file **fig.dat**, using **fig.def**. This is for an obsolete graphical system in Pascal (can be sent on demand).

Printscreen prefix. Then use hot key:

- 1 DOS only: printscreen on LaserJet printer

- e DOS only: printscreen on Epson 9 pin printer (in 72 dpi)
  - p DOS only: printscreen on Epson 24 pin printer (in 180 dpi)
  - o Dump screen in PostScript, black background (see `#` for better PostScript output!)
  - O Dump screen in PostScript, white background (see `#` for better PostScript output!)
  - m Dump screen to a PPM file, black background
  - M Dump screen to a PPM file, reversed (white background)
- 1
- 2
- 3 ... : Plot given column as Y. Suppressed if the column argument Y was written as 01 etc.
- [ ] Change the column by -1 or +1.
- 0 Prefix: e.g., `0 1 2` will plot column 12. `0 0 4` is the same as `4`. Max column is 26.
- # Make PostScript file (`plot.ps` or `plot.eps`), use control data in `ps.def`. This file may contain the following commands. The command key letter must start from column 1. All sizes are in pt (only for command `w`, negative values denote cm).
- ! anything
  - # anything comment
  - s SIZE [NAME ] Set font size (default=14) and name (default=Helvetica)
  - m l Set landscape mode (default)
  - m p Set portrait mode
  - m e Set eps mode (encapsulated postscript)
  - x LEFTMARGIN RIGHTMARGIN TEXT [TEXT ] Set margins (of the graph without axes labelling) of axis x and its label.
  - y BOTTOMMARGIN TOPMARGIN TEXT [TEXT ] Set margins (of the graph without axes labelling) of axis y and its label.
  - w XSIZE YSIZE Set window size. Negative values denote sizes are in cm. Note that command `m p` used after `w` swaps `XSIZE` and `YSIZE`. Default is `w 576 432`, i.e., 8x6 inches.
  - t LINETHICKNESS FRAMETHICKNESS Set the default line thickness and the thickness of the frame and tics. The default is `t 1 0.5`.
  - f NOXTICKS NOYTICKS Sets the approximate number of ticks and corresponding labelling on axes. The default is `f 5 5`. The program determines the tick distance to be a round number of a power of 10 time 1, 2, or 5, and thus the actual number of ticks may differ.

# R G B[,DASH,LINWIDTH] Set the color (in RGB) of line number # (1st file=0, 2nd file=1, if not changed by hot keys `[c]`, `[C]`). DASH is a space-separated even number of numbers denoting DASH PAUSE DASH PAUSE... Example: 1 1 0 0,5 3 1 3,0.5

r ANGLE Rotation angle for subsequent l and y, in degrees.

l X Y STRING

l X +[DY STRING] Print string at given position. The second form advances from the previous position down by DY (default DY=line feed).

STRING for commands l, x, y may contain the following charcters:

\ Next character is symbol (Greek letter): \a is alpha, etc.

\_ Next character is subscript

^ Next character is superscript

Examples: t\_c\_o\_r\_r/ps, r^\a

\*Source: show/plot.c

## 19.3 tabproc: Command-prompt spreadsheet

Some people like excel, I like command prompt and pipes... Input data are processed using given formulas and formats and.

```
tabproc EXPR1[:FMT1] [EXPR2[:FMT2]] ... < INPUT > OUTPUT
```

EXPR Expression made of letters A B C... for columns 1,2,3..., and usual mathematical operators and functions. Instead of A B C..., #1 #2 #3... can be used. Consecutive numbers are coded by @ or n and are reset to zero on a blank line. Examples of expressions: A/2+sin(B), #1/2+sin(#2), @\*pi^2.

FMT Format in the C-style; the initial % may be omitted. Examples: 7.4f, %7.4f, err=%.2e. The default format is that of the previous argument or %g if no format is given.

Lines beginning by '#' or '!' are copied as comments. Max 26 columns are allowed. Environment variable NOLF suppresses line feeds (may be useful in scripts, see Sect. 19.6).

\*Source: c/tabproc.c

## 19.4 mergetab: Merge columns of data

This utility merges columns (white-separated) of data. Lines beginning with **#** are ignored.

Call by:

```
mergetab [FILE1]:[-OMIT:] [/STRIDE:] COL:COL...  
        [FILE2]:[-OMIT:] [/[-]STRIDE:] {COL|COL=COL1}:{COL|COL=COL1}...  
        ...
```

**FILEi** File name, or - for stdin (max once allowed). Missing **FILEi** repeats previous, missing **FILE1**=stdin.

**OMIT** # of noncomment lines omitted from the top of file

**STRIDE** Consider only every **STRIDE**-th noncomment line

**-STRIDE** As above, starting with (**STRIDE**-1)-th line **OMIT** and **-STRIDE** may be combined, e.g., **:-2:/-10** is the same as **:-11:/10**

**COL** Column to extract (print)

**COL=COL1** Synchronizes with column **COL1** of **FILE1** but does not print. The data in columns to synchronize should be in increasing order.

Example:

```
mergetab st2.g:1:2 tip4p.g:1=1:2 | plot -:A:C-B
```

will pipe columns 1 and 2 of **st2.g** and column 2 of **tip4p.g** shifted so that it matches column 1. If **tip4p.g** has finer grid, extra data are omitted, if **st2** finer grid, missing data in **tip4p.g** are marked as n.a. The piped stream has thus three columns. Finally, the difference of **tip4p** and **st2** is plotted.

Source: `c/mergetab.c`

## 19.5 tab: Column table of consecutive numbers

Call by:

```
tab FROM TO [BY [FORMAT]]
```

**TO** is included. If no **BY** is given, 1 is the default. If no **FORMAT** is given, "%g" is the default.

Example (Fourier transform of a saw-like function, [21.11](#))

```

tab 0 100 0.01 | \
  tabproc "A-int(A)" | \
  spectrum 10000 | \
  plot -:0:1

```

Source: c/tab.c

## 19.6 ev: Calculator

To run a line-oriented calculator, suitable for Unix with fast pasting of numbers, run **ev** **--c** and type **?** for help. Also supported: plot graph of function (utility plot required, see Sect. 19.2), sums, products, numerical integrals and derivatives, and numerical root finding. Can read a list of constants (file **.evdata**).

To calculate an expression once, give it as an argument, e.g.:

```
ev "pi^3*exp(2)"
```

Source: c/ev.c

## 19.7 endian: Change endian (order of bytes) in binary files

The playback files (extensions **.plb**, **.p00**, **.p01**) and the convergence profiles (extension **.cp**) contain float (single precision) numbers in binary format. This format is the same for most computer architectures with one exception — order of bytes (the endian or sex). Utility **endian** changes endian by four-byte words. Call by:

```
endian source_file { dest_file | drive: }
```

**drive:** means drive letter and applies for DOS only. Directory name cannot be used instead of the output file name.

**endian** cannot be used for other binary files. Some utilities can read binary files (even other than **.cp** and **.plb**) with opposite sex so that **endian** is not needed.

Source: c/endian.c

## 19.8 start: Start application according to file extension

Motivation:

This is a command-prompt analogue of the Windows (or MacIntosh, Gnome, KDE, OS/2 ...) mechanism of starting application according to the type of the associated

file. On some versions of Windows there is command 'start' of the same function (unfortunately very buggy). Such a style of work may be considered strange by Windows folk. But my brain is not able to find a file of interest among more than ten icons in a graphical folder (or Norton Commander list) and I consider typing the file name much faster (especially with file completion and wildcards).

Simple examples:

```
start pig.jpg          --> xv pig.jpg
start archive.zip      --> unzip -v archive.zip | less
```

Extended examples:

```
start -expand 2.5 pig.jpg  --> xv -expand 2.5 pig.jpg
start pig.jpg -expand 2.5   ERROR unknown ext
start pig.jpg "-expand 2.5" --> xv pig.jpg -expand 2.5
start mol cfg.plb -I%       --> show mol cfg -I%
```

Details:

1. Scans arguments given and determines the extension of the last file argument. (File argument is an argument not starting with - nor + and with an extension and not containing a space. The extension is the suffix after the last . in the file name provided that the name before this . is not empty.)
2. Looks up the associated application
3. Starts this application with arguments, using system()

File registration:

Edit the array of structures `reg[]` in `start.c` and recompile. The last line must be `NULL, NULL`.

Source: `c/start.c`

## 19.9 sortcite: Sort LaTeX citations

This utility sorts all `\bibitem{ }` statements in the `thebibliography` environment so that they appear in the order in which they are referenced using statements `\cite{ }`, `\Cite{ }`, and `\Hide{ }` (see `cite.sty` for the latter two).

Usage:

```
sortcite FILE [-]
```

`FILE.tex` is input, `FILE.ren` output. Optional `-` means that not referenced bibitems will be removed from output.

Source: `c/sortcite.c`

# Chapter 20

## Program ‘pdb’ version 1.4a

`pdb` reads protein in the PDB format and converts it to mol-format understood by program `blend`. See also `pdb2pdb`, see Sect. 24.1 (rearranges the lines of `pdb`-files to a ‘more standard’ order).

### 20.1 Running

#### 20.1.1 Environment

Environment variable `BLENDPATH` can be set to point to the path that contains the needed residue-files (\*.rsd by default). If it is empty, the files are looked for in the working directory. A subdirectory of this directory, i.e., the used set of the residue files, may be specified by option `-r`.

Example for UNIX (csh, tcsh):

```
setenv BLENDPATH /home/jiri/macsimus/blend/data
```

Example for UNIX (sh, bash):

```
export BLENDPATH=/home/jiri/macsimus/blend/data
```

Example for DOS:

```
set BLENDPATH=D:\MACSIMUS\BLEND\DATA
```

#### 20.1.2 Synopsis

The command line to run `pdb` is:

```
pdb [options] pdbname [molname]
```

If molname is missing then it is assumed that molname=pdbname.

Run `pdb` without options to get a brief help.



### 20.1.3 Files

pdbname.pdb Source PDB-file

molname.pdb Output PDB-file (only if option -p)

molname.####.pdb Output PDB-file (only if option -p with range)

molname.rep

pdbname.rep Optional pattern replacement file. If molname.rep is not found, pdbname.rep is tried, if this file does not exist either, no pattern replacement occurs. Example (to fix wrong OT type for C-O-C oxygens in sugar-sugar bond; `pdb` has been run with `-d2` option).

```
CH1E OT-0.65 CH1E      CH1E OE-0.5 CH1E
CH2E OT-0.65 CH1E      CH2E OE-0.5 CH1E
CH1E OE H0              CH1E OE DEL
CH2E OE H0              CH2E OE DEL
```

It means that all structures CH1E-OT-CH1E with partial charge on OT -0.65 will be replaced by CH1E-OE-CH1E with partial charge on OT -0.5, etc. DEL means deletion of the atom. The file is executed by lines so that all changes caused by one line are available when executing the following line. Currently only replacement in groups by 3 atoms defined by atom types is available. No wildcards available for atom types. If necessary, will be extended in future.

molname.sel Residue selection file. Example:

```
option -n0 ! should match with runtime option -n
! neutral versions of all negatively charged residue
!RSD # chain replacement
ASP *    *    asph
GLU 11   A    gluh
```

The format is free, - in the chain field means no chain, \* in the chain field means any chain, \* or -1 as # matches any residue irrespective of its # in the PDB file. RSD MUST be uppercase, replacement refers to .rsd files and for UNIX is case sensitive, however, lowercase is recommended.

molname.mol Output mol-file (see `blend`) Default output from `pdb`.

molname.3db Molecular configuration in binary format `float[] [3]`, deprecated (use .plb), see option -b.

molname.plb Molecular configuration in the playback format, the default since V1.4a (see the manuals for `cook` and `blend` and option -b).

molname.3dt Molecular configuration in text 3 column (x y z) format.

molname.plt Molecular configuration in text 3 column format with playback-like header line (one line of 2 numbers, 1st=number of sites, 2nd=0).

\*.rsd Extension of the residue files, see also BLENDPATH and option -r.

## 20.1.4 Options

-a Some atoms in the pdb-file may be given alternative locations, i.e., different coordinates. They are numbered by letters A,B,...

-a0 Use location A (default)

-a1 Use average of all location weighted by occupancies

-a2 Use location with maximum occupancy number.

-b Write (if not -p) or read (if -p) binary file with the molecular configuration.

-b0 -b- Don't read/write any binary coordinate file.

-b -b1 Binary coordinate file is molname.3db.

-b2 Use playback file molname.plb as the molecular coordinate file (default).

-b3 Both -b1 and -b2 (not for input).

-b-1 As -b1 with reversed endian.

-b-2 As -b2 with reversed endian.

-b-3 Both -b-1 and -b-2 (not for input).

-cnumber Calculate S-S (CYS-CYS) bonds from the configuration. number is the distance limit in Å, missing number means distance limit 3Å (equilibrium bond length is 2.04Å). Use if SSBOND and/or CONECT commands are missing.

-dkey Delete (do not include) the atoms which are in the residue files but are not present in the pdb-file. Note that typically hydrogens are NOT present in the pdb-file and should be included but sometimes a missing heavy atom should be removed because a chemical bond is present (hopefully described by CONECT statement). E.g., sugars are given with all -OH groups in the residue files but if they are chemically bonded then two -OH groups create one -O- bond and one O and two H should be removed. (See also molname.rep). The values of key are as follows:

key>0 Deletions apply only to MOLECULES (1st keyword in the residue file must be **molecule**)

key<0 Deletions apply to all residues

1,-1 Deletions apply to hydrogens only

2,-2 Deletions apply to heavy atoms only; free hydrogens after removing these heavy atoms are also removed.

3,-3 Deletions apply to any atoms

- eRSD Use RSD.rsd as end (C-terminal patch) for ending peptide chains. The default is cter.rsd ( $\text{COO}^-$ ). Extension .rsd can be changed by option -f.
- fPARSET PARSET.par will be the used force field (parameter\_set). Information on the parameter set is written to the mol-file created so that **blend** can make use of it. In addition, file PARSET.par contains line rsddir rsddir with the name of the directory in which the residue files (extension \*.rsd) are looked for. If no -f is specified, PARSET=RSDDIR is assumed (see option -r).
- g Gap in residue numbering terminates chain. If the difference of numbers of two consecutive residues is not 1 than the protein chain is terminated (normally by CTER) and started again (normally by NTER), as if there was a TER statement between them.
- hRSD Use RSD.rsd as head (N-terminal patch) for starting peptide chains. The default is nter.rsd ( $\text{NH}_3^+$ ). Extension .rsd can be changed by option -f.
- key Ignore atoms which are present in the pdb-file but are not supported by the residue-files. Typical case is when all hydrogens are in the pdb-file (incl. aliphatic H) but the aliphatic H are not supported in the united atom representation. Typical key is 1 (ignore H in molecules) or -1 (ignore all H, in molecules and aminoacid residues). See option -d for details of key.
- m- -m0 Don't write .mol file
- n Use neutral residues:
  - n0 Charged residue are used for acids (ASP and GLU) and bases (ARG, LYS, and HIS). This is the default. In V1.3c and older, HIS was neutral by default, now HIS is charged (protonated) by default and -n (see below) specifies its uncharged state.
  - n Use neutral residues instead of 'standard' charged ones:
    - ARG -> ARGN
    - ASP -> ASPH
    - GLU -> GLUH
    - LYS -> LYSN
    - HIS -> HISN
    - NTER -> NTERN
    - CTERH -> CTER
- (In versions V1.3h and older did not work for the termini.)
- n-1 Use charged residues and compensate charges by counterions  $\text{Na}^+$  or  $\text{Cl}^-$ . Also charges of NTER and CTER are compensated. NOTE: Positions of ions are calculated from known atoms of the side-chain (for ARG, ASP, GLU, LYS, HIS) or backbone (for terminals) so that the ion is placed in the direction of the chain off the molecule. E.g., for N-ter the formula is 2\*r[N]-1\*r[CA]. When counterions are used, **blend** should be called for the first time with option -k-3 to allow energy minimization with the molecule skeleton fixed and free ions and hydrogens.

- n-% As above where the distance of the counterion from the ion being compensated is -% percent of the previous bond., e.g.:  $(q/100+1)*r[N]-q/100*r[CA]$
- oRSD Omit residue RSD. Typical use is such as -oH0H (or -oWAT—check the PDB file!) to remove all water molecules.
- pnumber[:FROM:TO] Paste a configuration from input file (typically the playback file molname.plb to a PDB-file. Both pdbrname and molname must be specified on the command line (and must differ). File pdbrname.pdb is the input PDB-file and molname.pdb the output. The source and format of input configuration is given by options -b and -t (only one of them may be active; since -b is default, it must be turned off if -t is to be used). -pnumber with a positive argument selects number-th configuration on the playback file (supported only for binary format), -p is the same as -p1 (the 1st configuration), -p-1 means the last configuration in the playback file, -p-2 the 2nd last, etc. The version with range processes a range of frames. Option -b2 meaning playback file molname.plb as input is forced (also -t2?). The output files are numbered molname.####.pdb, where #### is the frame number. In this case, molname=pdbrname is allowed. Cf. options -r4:FROM:TO:BY -w10 | -w20 of **blend**.
- qnumber Charge multiplication factor in %. Default is 100%.
- rRSDDIR The directory of residue files, relative to BLENDPATH (see Sect. 20.1.1). If also option -fPARSET is given, option -rRSDDIR overrides the value of rsddir stored in file PARSET.par. Slash or backslash after directory name is not allowed.
- snumber Scrolling enabled with number kB buffer, see **blend** for details.
- t Write (if not -p) or read (if -p) text file with the molecular configuration.
  - t -t1 Coordinate file is molname.3dt.
  - t2 Coordinate file is molname.plt with playback-like header line.
  - t3 Both -t1 and -t2.
- u Enable more relaxed atom matching algorithm when assigning PDB atoms to atom names in the residue files; for instance, with -u, HE3 in PDB-file matches HE1 in the residue file if there is only one (Unique) HE atom in the residue file. Without -u, they do not match. If the PDB-file contains aliphatic hydrogens (but no charged or hydrogen bonded hydrogen), the default -u0 must be used. Some PDB formats, however, may use different numbering and then option -u may help.
- v Verbosity level.
  - v -v1 Verbose (default).
  - v- -v0 Most of the info messages and warnings are suppressed.
  - v-1 Partly verbose. Only connection and re-connection reports are suppressed.

- x Wildcard A in atom names in the PDB file is enabled. This occurs when a group like -CNH<sub>2</sub>O in GLN may rotate by 180 degrees so that it is not possible to distinguish both heavy atoms.
- z (new in V1.3n)
  - z0 Patched residue keeps name of the residue even for atoms added (i.e., the added atoms are treated as part of the residue).
  - z2 The patched part has always the name of the patch (i.e., the added atoms are treated as new ‘residue’ under the patch name). If the patch is specified by -h/-e options, first three letters of the patch name (in uppercase) are used.
  - z1 (default) Option -z0 applies for patches specified in the PDB file while -z2 for patches specified by -h and -e options.

## 20.2 Residues

The topology (chemical structure) along with partial charges are defined in the residue files. The name of a residue file is usually the same as the residue name (but is lowercase) and the extension is defined by the parameter set used (see option -f).

### 20.2.1 Format of residues

The format of residue files is almost the same as .mol format described in the manual for **blend**.

Differences are:

1. First line must start by one of the following keywords:

**aminoacid** Aminoacid residue, to be connected by a peptide bond

**patch** A part of molecule that modifies another molecule, for C/N-termini deprecated

**nter** Chain head (N-terminus). A peptide bond to the next residue is added. This residue keeps its name in the PDB file.

**nterp** Patch for chain head (N-terminus), merged with the following residue. A peptide bond to the next residue is added even if not specified in the patch.

**cter** Chain end (C-terminus). A peptide bond to the previous residue is added. This residue keeps its name in the PDB file.

**cterp** Patch for chain end (C-terminus), merged with the following residue. A peptide bond to the previous residue is added.

**water** Water model

**molecule** Any other molecule

2. If the type is **patch**, **nter**, or **cter**, atomid field may be divided by a colon into 2 parts. The second part is atomid to be replaced in the residue and the first part is the new name. Example:

```
patch NH3+ (N-terminal patch for all aminoacids but GLY and PRO)
```

```
number_of_atoms = 5
! charge=1
```

```
atoms
! i  id      type  charge  chir  nb  bonded_atoms
  0  N:N      NT   -0.30   0   4   1 2 3 4
  1  NH1:H    HC    0.35   0   1   0
  2  NH2      HC    0.35   0   1   0
  3  NH3      HC    0.35   0   1   0
  4  CA:CA    CH1E  0.25   1   1   0
```

Since *Nothing is compatible*, the same atom can be coded in different PDB files by different id's. These aliases can be listed in a residue-file as a comma-separated list.

```
4  C,CY      C      0.51   0   0   2 0 5
```

Only the first one is printed in info/error messages.

## 20.2.2 Termini

The residue files for aminoacids contain residues as they enter the peptide chain (-N-C<sub>α</sub>-CO-).

The first aminoacid (N-terminus) must be modified as follows:

- +charged (=protonated): there is NH<sub>3</sub><sup>+</sup> instead of N, patch=NTER
- neutral (**pdb -n**) : there is NH<sub>2</sub> instead of N, patch=NTERN
- counterion (**pdb -n-1**) : there is NH<sub>3</sub><sup>+</sup>Cl<sup>-</sup> instead of N, patch=NTERCL

This is accomplished by applying given patch (a patch is represented by a residue file with keyword **patch**, **nter**, or **cter** in the 1st line, e.g., file **nter.rsd** is the patch NTER for charmm21).

Residues PRO and GLY are exceptional and different patches must be used:

- PRO:
- +charged (=protonated): there is NH<sub>2</sub><sup>+</sup> instead of N, patch=PROP
  - neutral (**pdb -n**): there is NH instead of N, patch=PROPN
  - counterion (**pdb -n-1**): there is NH<sub>2</sub><sup>+</sup>Cl<sup>-</sup> instead of N, patch=PROPCL
- GLY:
- +charged (=protonated): there is NH<sub>3</sub><sup>+</sup> instead of N, patch=GLYP

- neutral (`pdb -n`) : there is  $\text{NH}_2$  instead of N, `patch=GLYPN`
- counterion (`pdb -n-1`) : there is  $\text{NH}_3^+\text{Cl}^-$  instead of N, `patch=GLYPCL`  
(The reason for a special treatment of GLY is because of CH2E instead of CH1E as  $\text{C}_\alpha$ )

`pdb` since V1.3i selects the appropriate patch automatically according to the value of option `-n`. However, if the patch is specified by `-hpatch`, no check is made and the requested patch is used.

WARNING: names of hydrogens vary in different versions of `pdb` files and they need not be recognized. In `PROPN` the hydrogen is called `HNC` while in `PROP` the two hydrogens are called `NH1` and `NH2`. This is not logical, but corresponds to the `pdb` files I have seen. Edit the files if you meet another names. Unrecognized hydrogens may be usually safely ignored—they are calculated in the blend stage.

### 20.2.3 List of residues

The most common residues follow. The corresponding file name is in lowercase and has the residue extension appended. One-letter code for aminoacids is in parentheses.

`nter ACE` acetylated N-terminus,  $\text{CH}_3\text{-CO-}$

`aminoacid ALA` alanine (A)

`cter AMI` amidated C-terminus,  $\text{-CO-NH}_2$ , also called `CT2`

`aminoacid ARG` arginine (R)

`aminoacid ASN` asparagine (N)

`aminoacid ASP` aspartic acid (D), anion

`aminoacid ASPH` aspartic acid, protonated

`cter CT1` methylated C-terminus  $\text{-CO-O-CH}_3$

`cter CT2` amidated C-terminus,  $\text{-CO-NH}_2$ , also called `AMI`

`cter CT3` n-methylamide C-terminus  $\text{-CO-NH-CH}_3$ , also called `CMAM`

`cter CTER` C-terminal patch for aminoacids, anion  $\text{-COO}^-$

`cter CTERH` C-terminal patch for aminoacids, protonated  $\text{-COOH}$

`aminoacid CYS` cysteine (C)

`aminoacid CYSS` cysteine (to bind to another CYS)

`aminoacid GLN` glutamine (Q)

`aminoacid GLU` glutamic acid (E), anion

aminoacid GLUH glutamic acid, protonated

aminoacid GLY glycine (G)

nter GLYP terminal patch for GLY instead of NTER

molecule HEM heme BUGS: in some force fields, planarity is accomplished by different force fields terms than original. It may affect vibrational frequencies and similar. Binding of the heme with the protein can accomplished by files `.sel` and `.rep` files which are only partly available. Since there are 6 bonds of Fe, `blend` must be compiled with `#define MAXVAL 4` in `blend/blendpar.h`

aminoacid HIS histidine (H), cation (protonated)

aminoacid HISN histidine (neutral with proton on ND1)

aminoacid HISNE histidine (neutral isomer with proton on NE2)

water HOH TIP3P water model

aminoacid ILE isoleucine (I)

aminoacid LEU leucine (L)

aminoacid LYS lysine (K), positively charged

aminoacid LYSN lysine (K), neutral (not in all force fields)

aminoacid MET methionine (K)

nter NTER N-terminal patch,  $-\text{NH}_3^+$ , for all aminoacids but PRO and GLY

nter NTERN neutral N-terminal patch for all aminoacids but PRO and GLY

aminoacid PHE phenylalanine (F)

aminoacid PRO proline (P)

nter PROP N-terminal patch for PRO instead of NTER. `prop.che` is the whole residue

aminoacid SER serine (S)

aminoacid THR threonine (T)

aminoacid TRP tryptophan (W) BUG: in some force fields, planarity is accomplished by different force fields terms than original. It may affect vibrational frequencies and similar.

aminoacid TYR tyrosine (Y)

aminoacid VAL valine (V)

molecule ZN  $\text{Zn}^{++}$



## 20.3 Bugs and caveats

1. In spite of its name, MACSIMUS is now being developed towards simulations of fluids, ionic systems, systems with polarizability and not biochemistry applications. MACSIMUS' basic support of protein modeling and simulation will not be extended in future.
2. Only aminoacids are supported and a few other compounds.
3. There are several variants of the PDB format and therefore there is no guarantee that `pdb` will always work. See comments in the program (`pdb.c`) for details.
4. Information in PDB commands `SSBOND` and `CONNECT` is duplicated. `pdb` writes a notice 'reconnected' if the same bond is connected again. The same holds for option `-c`.
5. Cannot use different N- and C-terminal patches for different chains.
6. Some residues (esp. in alternate charge states) may be missing for some force fields.

# Chapter 21

## Data analysis

### 21.1 showcp: Show and analyze convergence profiles

`cook` (and in special cases also `blend`) records time-development of selected quantities (see the manual for `cook`, file extension `.cpi`). `showcp` enables viewing and analyzing these ‘convergence profiles’. `showcp` recognizes the endian so that analyzing the cp-files on a different computer is transparent.

IMPORTANT NOTE: Always the *time development of quantities* is shown and never a *cumulative average* (sometimes also called ‘convergence profile’) from start of simulation because this quantity hides important information like sudden jumps, oscillations, etc! (To show a cumulative average, use `runsum`).

#### Usage:

```
showcp [OPTIONS] {SIMNAME | SIMNAME.cp | SIMNAME.cpz} [OPTIONS] [NAME ..]
```

#### Files:

`SIMNAME.cp` convergence profile file

`SIMNAME.cpz` packed convergence profile file (see Sect. [21.9](#))

`SIMNAME` try `SIMNAME.cp`, if it does not exist then `SIMNAME.cpz`

#### Options:

**Select columns.** The default (if no such option is given) is all columns selected.

`-#` Select column number `#` (integer from 1). This option can repeat and can be combined with `NAME`.

`NAME` Select column of given name.

**-o#[,#...]** To be used with option **-p**: merge given columns to One plot. Several **-o**'s can be repeated. The default (if no option **-o** is given) is **-o2,6,7**, i.e., T, Tin, and Ttr are merged in one plot. If at least one **-o** is given, this default is overridden, further **-o**'s define variables for another plot. The first column must be also selected (by **-#** or **NAME**), the remaining are added. Columns in the cp-file (cpa-file) not listed are plotted separately. Example: to plot 512.cp, blocked by 10, with columns 4 and 5 merged, 2, 6, 7 merged, and all other columns separate, use:

```
showcp -b10 -p 512.cp -o4,5 -o2,6,7
```

### Range/lag options:

- f#** (1) Analyze or show from record #
- t#** (0) Analyze or show to record # (incl.). The default is **-t0** (= the last record)
- b#** (0) Block (sub-average) size for showing and additional analysis. The default is **-b0**, which will adjust the block size to a power of 10 (for plotting) or according to screen width.
- h#** Set the timestep (**noint\*h** from cook run), applies to **-x**, **-c1**, **-c2**.
- l#** (29) Set lag for statistics (up to which autocorrelation analysis is performed). **-l0** = no statistics
- n#** (12) Number of consecutive blockings by 2 (not for **-c1,2**). Blocked data enable better error estimate.

### Actions:

- a** Write ASCII file **SIMNAME.cpa** in format compatible with **gnuplot** or **plot** (see Sect. 19.2)
- a-1** As above and add column **Etot-Etot0**
- Output is **stdout** instead of a cpa-file and any other information but the resulting ASCII convergence profile is prepended by **#**. To be used with option **-a** to pipe the results to other programs. Example:

```
showcp - -a test.cp | plot -:0:4 :5
```

will show a merged graph of columns 4 and 5.

- c#** (0) Sum of:
  - 1 write files **SIMNAME.NAME.tcf** with time correlation functions
  - 2 write files **SIMNAME.NAME.cov** with covariances

- 4 detailed autocorrelation analysis (not just summary)
- 8 autocorrelation analysis of data blocked by option -b
- 16 autocorrelation analysis with removed linear T-dependence

Option -l must be set, too. The default is -c0 which means that only a brief summary of errors is printed.

- g print pseudoGraphs [default if columns selected and no other action]
- m print Merged y-x pseudograph (1 line = 1 block, variables are shown by letters)
- p Plot blocked selected columns with not-constant data (forces -a)
- x write blocked x-y files SIMNAME.NAME.xy
- e statistical analysis and regression of the first two selected columns

### Options:

- k# pseudograph info line (blow the graph) contains:
  - 1 column info, min, max, range, blocking
  - 2 column info, first, last, difference, blocking
  - 3 both 1+2
- d# (1) Set delay # s between plots, to be used with option -p. Increase when showcp is run on a distant computer and the connection is slow
- r Reverse endian (obsolete)
- s# Scroll (buffer #kB, -s=-s30) (obsolete)
- u Character set for pseudograph (option -g) is
  - u Braille UTF-8
  - u- ASCII

The default is guessed from LOCALE.

### Example:

```
showcp mysimul -p Etot Tkin -5
```

Will show three graphs, total energy (column 1), temperatures (white=Tkin=total, yellow=translational, cyan=rotational and intramolecular), and column 5=pressure or density. The data will be averaged in appropriately selected blocks

\*Source: util/showcp.c

## 21.2 cp2cp: Manipulate convergence profile files

Usage:

```
cp2cp [OPTIONS] INPUT.cp OUTPUT.cp [OPTIONS]
```

Files:

INPUT.cp input convergence profile file

OUTPUT.cp output convergence profile file

Options:

- b# input data are blocked (sub-averages are made) by # lines of data
- s# stride by #: every #-th value of input is taken, the first one considered is the #-th
- s# -b## strided data are blocked
- rKEY Rearrange columns. KEY=string of {1,2,...,9,A,...,0}, where 1=1st column, A=10th column, 0=empty column (filled by 0)
- hNAME... headers of empty columns (added by using key 0 in -k), every 4 chars correspond to one 0
- f# read the input file from record # (records are numbered from 0)
- n# max. # of records read (must appear after -f)
- t# read the input to record # (record # is not included; equivalent to the sum of -f -n)

BUGS: does not accept old 5-column format, opposite endian, packed (.cpz) files

Examples:

Omit first 1000 records, delete columns 6+7 (of 12), and block by 10:

```
cp2cp in.cp out.cp -f1000 -r1234589ABC -b10
```

Replace column 4 by 0 (header ZERO) and swap columns 6 and 7:

```
cp2cp in.cp out.cp -r1230576 -hZERO
```

Split a.cp (100000 records long) into chunks by 10000 records

```
tab 10000 100000 10000 "cp2cp a.cp b0.cp -f0 -n10000" | sh
```

Source: util/cp2cp.c

## 21.3 rdfg: Analyze and show radial distribution functions

**rdfg** prints and/or plots the radial distribution functions from binary data stored in the rdf-file. The running coordination numbers are provided, too (as column 4 of the g-files).

Usage:

```
rdfg SIMNAME[.rdf] [KEY [FMT]]
```

KEY is a string of:

- u** Unix version only: Create separate files **SIMNAME.SITE1.SITE2.g** for each site-site pair in the rdf-file
- d** DOS style: Create separate files **SITE1SITE2.g** for each site-site pair in the rdf-file
- none of d u** Create one merged file **SIMNAME.g** of all site-site RDFs
- p** Start plot of the of generated g-file(s). See Sect. [19.2](#).
- r** Reverse endian on input. To be used for rdf-files obtained on a computer with the opposite sex (endian)

FMT is the C-format for printing g(r), default=" %8.4f" (four dec. digits).

Example (generate files with DOS-names, accuracy=5 dec.digits, plot them):

```
rdfg mysim dp %9.5f
```

\*Source: util/rdfg.c

## 21.4 smoothg: Smooth histogram-based RDF

Smoothing of g(r) [obtained by histogram]. Call by:

```
smoothg WINDOW ORDER DR DATA[:COLR:COLG] FROMR[:TOR[:DR]] [FORMAT]
```

WINDOW RECT[+TRIANG] rectangular width+triangular half-width

-WIDTH Gaussian half-width:  $\exp[-(x/WIDTH)^2/2]$

ORDER order of the polynomial (1=linear)

DR grid of R in DATAFILE (must match file!)

DATA file of r,g data

COLR:COLG columns of r,g (default=1,2), COLR=0 means 0,1,..

FROMR:TOR:DR output range and grid of R

FORMAT output format for double x,y (default="%.12g %.12g")

\*Source: util/smoothg.c

## 21.5 staprt: Print a sta-file

Prints the contents of a sta-file generated e.g. by cook. Usage:

```
staprt FILE.sta [mode]
```

where MODE is

- + More decimal digits of the output
- Print merged time correlation to stdout

anything else Print time correlation files to files FILE.NAME.tcf, where NAME is derived from the variable name (problematic characters are edited)

Warning: some changes in DOS version

\*Source: util/staprt.c

## 21.6 sfourier: Structure factor from RDF

Using the fast Fourier transform, RDF (from a g-file) is converted to a structure factor (SF). Cf. structure factors obtained directly from configurations (see cook, option -f).

To get a partial SF from site-site RDF:

```
sfourier GRID CUTOFF NS <FILE.X.X.g > FILE.gsf
```

where FILE.X.X.g is one of g-files obtained by rdfg FILE u

The total SF for a mixture from a set of RDFs

```
sfourier GRID CUTOFF NS SITE1:b1 SITE2:b2 ... < FILE.g > FILE.gsf
```

where FILE.g is obtained by rdfg FILE (or cat FILE.\*.\*.g) Note that sfourier reads also information printed in the header of the g-files.

GRID Number of grid points/Å, must match the grid of the RDF files

CUTOFF Should be `rdf.cutoff` (see cook input data). If `CUTOFF < rdf.cutoff`, then then the data are truncated; if `CUTOFF > rdf.cutoff` then the data are padded by  $g(r) = 1$ .

NS total number of sites (atoms, not types of sites)

SITE1, SITE2 name of site of given number. Sites may be listed in any order

b# scattering length of site #

\*Source: `util/sfourier.c`

## 21.7 coordn: Coordination number

Calculates the coordination number from a (`r g(r)`) file. Normally not needed because `rdffg` prints the coordination number as well. Call by:

```
coordn COLUMN_OF_G DR [RHO GO CONST] < INFILE > OUTFILE
```

Calculates  $CONST + RHO * \int_0^R [g(r) - G0] dV$ . Default  $RHO=1$ ,  $CONST=0$ . The integral is over  $dV = 4 \pi r^2 dr$  and is replaced by the sum over  $dV = 4 \pi / 3 * [(r+DR/2)^3 - (r-DR/2)^3]$

Source: `util/coordn.c`

## 21.8 hbonds: H-bonds for liquid water

Hydrogen bonds are generated for water based on intermolecular O-H distance. The default version is for TIP4P, this can be changed in the source file.

Call by:

```
hbonds PLBFILE [O-H distance]
```

where PLBFILE is the input playback file and the second parameter is the threshold O-H distance to define an H-bond; the default is 2.44. A series of files `hb####.mol`, `hb####.gol`, `hb####.plb` is generated where `####` are consecutive frames.

Source: `util/hbonds.c`

## 21.9 cppak: Loss (de)compression of convergence profile files

This compressor samples the min-max interval of variables using integers of given number of bits and stores the differences. Typical compression rates are to 25–50%, depending on the number of bits and possible columns of zeros only.

Call by:



`cppak FILE.EXT [NBITS]`

where the action is determined by the extension .EXT:

`.cp` compresses FILE.cp to FILE.cpz

`.cpz` decompresses FILE.plz to FILE.cp

NBIT is the number of bits to store the min-max interval; default=12 (accuracy 3-4 dec. digits).

Packed cp-files are transparently recorded by `cook` (see variable `CPnbit`) and shown or analyzed by `showcp` (see Sect. 21.1).

Source: util/cppak

## 21.10 autocorr: Statistical analysis using autocorrelation function

Prints autocorrelation and error analysis of time-dependent data. Call by:

`autocorr {FILE|-} [OPTIONS]`

Options:

- `-l#` The lag (# of autocorrelation coefficients) for statistics [default=29]
- `-b#` The lag for calculations with data blocked by 2, 4, 8, ... items [default=2]
- `-n#` Number of blocked calculations: the maximum block size is  $2^{\#}$  [default=8]
- `-h` Higher precision (more digits) on output [default=off]
- `-m-1` Input data are angles in radians (mod  $2\pi$ ) and they are reconstructed to be continuous. To be used for the time correlation function of, e.g., dihedral angles.
- `-m#` As above, modulo # (must be positive integer). E.g., for angles in degrees use `-m360` (as for output of `ramachan`)
- `-c#` Column of data [default=1]
- `-t` Write time correlation functions (column 1 = 0,1,2,...; column 2 = autocorrelation coefficients) in separate files `NAME#.tcf` Number # is used only if given by option `-c#`. [default=off]
- `-` Instead of FILE means stdin

The output contains first the name, number of data and their range. In the second line there is the Mean and Variance. Then, the autocorrelation coefficients and other statistics are printed:

```

c[t] = Cov(c[0],c[t])/Var(c)
tau = SUM_t'=1^t c[t'] (partial sum)
StDev^2 = <(1+2tau) Var(c)>/n

```

where  $\text{Cov}(X,Y)=\langle(X-\langle X\rangle).(Y-\langle Y\rangle)\rangle$  is the covariance and  $\text{Var}(X)=\text{Cov}(X,X)$  is the variance; both are estimated from the data. tau is the correlation length and StDev is estimated standard deviation of the mean (arithmetic average).

\*-lines contain the same information calculated from blocks of averages (sub-averages) by 2, 4, 8... data. Example (realistic):

```

> autocorr logfile.anc -c3 -n12
logfile.anc.3          No=320688  range=[-1.40727e-09,2.12488e-09]=3.53215e-09
Mean = 1.30584217e-10  Var = 5.16207293e-20
_t__c[t]__1+2tau__StDev_  _t__c[t]__1+2tau__StDev_  _t__c[t]__1+2tau__StDev_
 0 1.0000  1.000 4.0e-13   1 0.9960  2.992 6.9e-13   2 0.9844  4.961 8.9e-13
 3 0.9667  6.894 1.1e-12   4 0.9450  8.784 1.2e-12   5 0.9214 10.627 1.3e-12
 6 0.8978 12.423 1.4e-12   7 0.8754 14.173 1.5e-12   8 0.8552 15.884 1.6e-12
 9 0.8374 17.559 1.7e-12  10 0.8220 19.202 1.8e-12  11 0.8084 20.819 1.8e-12
12 0.7958 22.411 1.9e-12  13 0.7834 23.978 2.0e-12  14 0.7702 25.518 2.0e-12
15 0.7556 27.029 2.1e-12  16 0.7389 28.507 2.1e-12  17 0.7201 29.947 2.2e-12
18 0.6994 31.346 2.2e-12  19 0.6771 32.700 2.3e-12  20 0.6539 34.008 2.3e-12
21 0.6304 35.269 2.4e-12  22 0.6074 36.484 2.4e-12  23 0.5853 37.654 2.5e-12
24 0.5645 38.783 2.5e-12  25 0.5452 39.874 2.5e-12  26 0.5274 40.928 2.6e-12
27 0.5110 41.950 2.6e-12  28 0.4957 42.942 2.6e-12  29 0.4812 43.904 2.7e-12
* 0 1.0000  1.000 5.7e-13   1 0.9848  2.970 9.8e-13   2 0.9465  4.863 1.2e-12
* 0 1.0000  1.000 8.0e-13   1 0.9516  2.903 1.4e-12   2 0.8663  4.636 1.7e-12
* 0 1.0000  1.000 1.1e-12   1 0.8936  2.787 1.9e-12   2 0.7559  4.299 2.3e-12
* 0 1.0000  1.000 1.5e-12   1 0.7907  2.581 2.5e-12   2 0.4804  3.542 2.9e-12
* 0 1.0000  1.000 2.1e-12   1 0.5645  2.129 3.0e-12   2 0.1880  2.505 3.2e-12
* 0 1.0000  1.000 2.6e-12   1 0.3299  1.660 3.3e-12   2 0.0953  1.850 3.5e-12
* 0 1.0000  1.000 2.9e-12   1 0.2409  1.482 3.6e-12   2 0.0838  1.649 3.8e-12
* 0 1.0000  1.000 3.3e-12   1 0.1694  1.339 3.8e-12   2 0.0917  1.522 4.1e-12
* 0 1.0000  1.000 3.6e-12   1 0.1880  1.376 4.2e-12   2 0.1144  1.605 4.5e-12
* 0 1.0000  1.000 3.9e-12   1 0.2105  1.421 4.7e-12   2 0.0248  1.471 4.8e-12
* 0 1.0000  1.000 4.2e-12   1 0.1515  1.303 4.8e-12   2-0.0060  1.291 4.7e-12
* 0 1.0000  1.000 4.4e-12   1 0.0376  1.075 4.5e-12   2 0.0331  1.141 4.7e-12
# 1.30584217191e-10 4.76e-12 9.53e-12 320688 (av std 2*std no)

```

Analysis: These are highly correlated data. The average is 1.3058e-10. The autocorrelation coefficient with lag of 29 timesteps is 0.4812 so that the error estimate of 2.7e-12 is severely underestimated. In lines beginning with \* are error estimated with blocking 2,4,8,... The higher blocking, the less influence of correlations, but fewer data so that less reliable estimate. The best results are from the t=1 column, i.e., in addition to blocking the first autocorrelation coefficient is taken into account. Watch the c[1] column and find the line with c[1] small enough, let us say,  $c[1] < 0.2$ , or  $c[2] < 0.1$ . It gives stdev=4.8e-12. The last #-line contains the maximum which is usually reasonable error estimate obtained automatically. Therefore the result is 1.306e-10 +- 4.8e-12 (or perhaps more optimistic 1.306e-10 +- 3.6e-12; it is difficult to say which value is more reliable. This is general problem in simulations with slowly decaying correlations.), i.e.,

with probability about 68% the value is within  $1.306\text{e-}10 \pm 4.8\text{e-}12$ , with probability 95% within twice this margin.

If there are enough data and the error estimates are reliable, then

Note: the autocorrelation analysis is embedded also in `cook` and `showcp`

Bug: some non-alphanumeric characters are removed from `FILE`.

\*Source: `util/autocorr.c`

## 21.11 `spectrum`: Spectrum (Fourier transform)

Calculates a spectrum (Fourier transform) of data. Call by:

```
spectrum [-]NDATA [KDATA [DT]] < FILE
```

`FILE` should contain `NDATA` numbers (SPACE, tab, or EOL separated) and `KDATA` is the number of output frequencies (if not given, `KDATA=NDATA/2`). If time resolution `DT` is given, the output has two columns (1st column = frequency, 2nd column = squared amplitude) and peaks are also calculated. If `DT` is omitted, the output has only one column (squared amplitude). With `-NDATA`, the input data will be multiplied by bell-like window  $\sin^2(\pi * i / NDATA)$ . The peaks are then ‘more peaky’ (delta-function like), but the resolution actually somehow worsens.

May be used for determining vibrational frequencies from MD data. For instance, let `sim.cp` be a 1000-records-long convergence profile obtained by running `cookfree` at low temperature for one molecule with `h=0.001` and `noint=1`. Then the following ‘pipe’:

```
showcp - -a sim | \  
mergetab -:2 | \  
spectrum 1000 2000 0.002 | \  
plot -
```

will plot a graph with peaks corresponding to vibrational frequencies. These should match the results of `blend -N sim`. Note that parameter `DT` has been set to TWICE the sampling rate `h*noint` because the kinetic temperature is a quadratic quantity and gives doubled frequencies.

Note: the algorithm uses FFT optimized for factors 2 and 3. The calculation is slowest for `NDATA=prime number`.

\*Source: `util/spectrum.c`

# Chapter 22

## Working with playback files

The calculated configurations or trajectories are usually stored in files with extensions `.plb` (whole configuration) or `.p00` (one molecule). In some cases they can be analyzed again by `blend` or `cook` that have generated them. Here are utilities to manipulate with the `plb` files.

### 22.1 `plbinfo`: Get information on `plb`-files

Get information on binary playback files. Call by:

```
plbinfo [-]FILE [[-]FILE ...]
```

The file(s) must be with extensions (`.plb`, `.p00...`) - in front of file names reverses endian on input.

Returns # of errors encountered (0 on success)

Source: `util/plbinfo.c`

### 22.2 `plbcheck`: Some checks on binary playback files.

Similar to `plbinfo` with some checks added. Call without arguments to get help.

Source: `util/plbcheck.c`

### 22.3 `plbconv`: Converts old and new `plb` formats

The format of `plb` files changed from a version supporting only fixed box size to a version with variable 3D box. `plbconv` converts these two formats.

Call without arguments to get help.

Source: `util/plbconv.c`

## 22.4 plb2plb: Extract selected sites

Call without arguments to get help.

Source: `util/plb2plb.c`

## 22.5 plb2asc: Conversion of plb-files to ASCII

Call without arguments to get help.

See also see Sect. [24.9](#).

Source: `util/plb2asc.c`

## 22.6 asc2plb: plb-files from ASCII

Call without arguments to get help.

Source: `util/asc2plb.c`

## 22.7 frame: Extract one frame from a plb-file

Obsolete, old plb format only. Use `plbcut` instead.

Call by:

```
frame {FILE|FILE.plb|FILE.p00|FILE.ppp} [FRAME [NS [{a|t|b}]]]
```

**argument 1** FILE without extension means `FILE.plb`, if not found then `FILE.p00`  
[[`??`, if not found then `FILE.ppp` ]]

**argument 2**=FRAME FRAME is the frame # to extract, default=1=1st frame, -1=last frame

**argument 3**=NS optional number of sites to truncate; default=0=use all sites\n\

**argument 4** a=t output is ASCII file `FILE.3dt`. The default is b = binary file `FILE.3db`

Source: `util/frame.c`

## 22.8 cutplb: Edit plb-files

Obsolete, old plb format only. Use `plbcut` instead.

To extract parts from a plb file:

```
cutplb INPUT_FILE OUTPUT_FILE BY [FROM]
```

where

INPUT\_FILE The input binary playback file

OUTPUT\_FILE The output binary playback file

BY The stride: extracts every BY-th configuration

FROM The first configuration extracted, default=1

To check coordinates of SITE, remove repeating frames, and generate `cutplb.chk` with removal info:

```
cutplb INPUT_FILE OUTPUT_FILE -BY [SITE]
```

where SITE is the site number. This version is useful when a long simulation crashed and was restarted so that some frames have been included twice into a plb-file. Not all versions of `cook` work in a strictly deterministic manner. The POLAR version depends on the start of the self-consistent field and in parallel versions the result may depend on the order of calculations. Even tiny numerical errors multiply in long simulation times! It is therefore better in case of any crash to truncate the plb file before restart – information on this restart is printed in the prt-file.

Source: `util/cutplb.c`

## 22.9 plbcut: extracts parts of a playback file

Call without arguments to get help.

Source: `util/plbcut.c`

## 22.10 plbbox: Change box size of a plb-file

Call without arguments to get help.

Source: `util/plbbox.c`

## 22.11 densprof: Selected density profile angular correlations.

Call without arguments to get help.

Source: `util/densprof.c`

## 22.12 plb2cryst: Sort sites to files according to crystal-like structure.

Call without arguments to get help.

Source: util/plb2cryst.c

## 22.13 plb2nbr: Sort sites to files according to the number of neighbors.

Call without arguments to get help.

Source: util/plb2nbr.c

## 22.14 plbmerge: Merge several plb files

Merge several plb files into another plb file, frames synchronously. Call without arguments to get help.

Source: util/plbmerge.c

## 22.15 atomdist: Atom-atom distances

Prints atom-atom distances from playback files Call by:

```
atomdist FILE INDEX1 INDEX2 [NS]
```

FILE The playback file (with extension .plb, p00, ...) or a 3db-file

INDEX1 INDEX2 Atom indices (see the mol-file)

NS Number of sites (iff FILE a 3db file)

Source: util/atomdist.c

## 22.16 smoothpl: Smooth the playback file

A simple method to filter out the thermal motion is smoothing the coordinates. It is, for each atom vector  $\mathbf{r}$ , given by

$$\mathbf{r}'[t] = \text{SUM } w[i] \mathbf{r}[t+i]$$

where **w** is the weight function.

Call by:

```
smoothpl IN OUT WINDOW [GAUSS [SITES]]
```

**IN** input playback file

**OUT** output playback file. It is shorted by the window width which is **WINDOW+GAUSS**

**WINDOW** rectangular window  $[1/\text{WINDOW}, \dots, 1/\text{WINDOW}]$ . Negative **WINDOW** means to reverse endian on input

**GAUSS** # of  $[0.5, 0.5]$  windows to convolute. This creates binomial function or approximately Gaussian function

**SITES** optional # of sites. If **SITES** is given, no header is assumed, otherwise 2 floats

Example: let **sim.plb** is a file of a 1ns long run, sampled by 1ps (there are 1000 frames in it):

```
smoothpl sim.plb smoothed.plb 20 20
show sim smoothed
```

will calculate and show the smoothed motion. Try different **WINDOW** and **GAUSS**.

Source: `util/smoothpl.c`

## 22.17 plbmsd: Mean square displacement of atoms

Calculates the (mean) square displacement of selected sites. Call by:

```
plbmsd [-]FILE[:FROM] L SITE [SITE ...]
```

where

**FILE.p00** or **FILE.plb** input playback file

**FROM** start for linear regression, default=2=first datum (1=zero)

**FILE.msd** output file

**-FILE** reverse endian of input playback file

**L** box size ( $L=0$ : take from **FILE.EXT**,  $L < 0$ : force **FREE** boundary conditions)

**SITE** site # to process (can enter more sites)

\*Source: `util/plbmsd.c`



## 22.18 density: Calculate local density

Call by:

```
density FILE.plb[:FROMFRAME] X Y Z R
```

Calculates # of atoms in a sphere of center (X,Y,Z) and radius R. Reads the whole file starting with frame FROMFRAME (default=1).

Source: util/density.c

## 22.19 mergeplb: Merge several plb files into one

Merge several plb files into another plb file, frames synchronously. Stops when any plb-file reaches EOF.

```
mergeplb PLB-FILE [PLB-FILE ...] > MERGED-PLB-FILE
```

The box size L is set to the maximum L Bug: cannot for DOS because of binary redirection Hint: use molcfg to prepare mol (and gol) files for showing.

Source: util/mergeplb.c

## 22.20 filtplb: Convert a plb-file for a subset of atoms

Obsolete (old format only), use plbfilt instead.

Call by:

```
filtplb RICH.mol [-]RICH.plb POOR.mol POOR.plb
```

Reads playback file RICH.plb which corresponds to molecule RICH.mol, selects only sites which are contained in molecule POOR.mol and creates playback file POOR.plb. POOR.mol must be a subset of RICH.mol, i.e., it POOR.mol must contain atoms of IDs present also in RICH.mol. Sign - in front of RICH.plb means reversed endian.

Typically used for essential dynamics.

Source: util/filtplb.c

## 22.21 plbpak: Loss (de)compression of playback files

This compressor rounds the coordinates to the nearest grid point and stores differences in coordinates. Typical compression rates are to 25–30% for resolution 10/Å and 30–40% for resolution 100/Å. Best lossless compressors (bzip2, rar) give about 90%.

Call by:

```
plbpak FILE.EXT [GRID]
```

where the action is determined by the extension .EXT:

.plb compresses FILE.plb to FILE.plz

.p00 compresses FILE.p00 to FILE.plz

.plz decompresses FILE.plz to FILE.plb

GRID is the resolution of points per 1Å. The default is GRID=10, i.e., the distance of grid points is 0.1. This is sufficient for most tasks like visualization or diffusion or conductivity calculations. BUG: available as a stand-alone utility only and not included into cook nor other programs.

\*Source: util/plbpak.c

## 22.22 plb2diff: Diffusion, conductivity, and viscosity calculations

This utility helps to calculate the diffusion, conductivity (both partial and bulk) from plb- and cp-files. If cook\* has been called with -n option (and therefore .vlb files have been stored), viscosity is calculated, too.

plb2diff calls cook\* several times for different blocks of configurations and makes averages.

See comments in the source file for details!

\*Source: util/plb2diff.c

## 22.23 shownear: re-color atoms according to their distance

```
shownear FILE.plb[:FRAME] FILE.gol \  
FROM TO [-]DIST COLORNEAR [COLORMARK] > OUTFILE.gol
```

Creates a gol-file that, when used with show, will mark atoms closer than —DIST— to atoms [FROM..TO) by COLORNEAR and optionally atoms [FROM..TO) by COLORMARK. -DIST means that atoms [FROM..TO) are not included in tests. Configuration (frame) FRAME is analyzed from file FILE.plb. TO<=0 means TO=number of sites.

Source: util/shownear.c

## 22.24 tomoil: Conversion to MOIL

Converts mol- and plb-files to the MOIL format. If you do not know what MOIL is, you probably do not need this utility. Run `tomoil` to get help.

\*Source: `show/tomoil`

# Chapter 23

## Molecule visualization

### 23.1 molcfg: Create configuration mol- and gol-files

This is a supporting utility to be used with `show` to show a configuration recorded by `cook*`. It generates a configuration mol-file `SIMNAME.mol` and optionally a gol-file `SIMNAME.gol` from mol-files `FILE1.mol`, `FILE2.mol` ... and optionally gol-files `FILE1.gol`, `FILE2.gol` ...

Usage 0:

```
blend -o SYSNAME MOL1 MOL2
...
cook* SYSNAME SIMNAME
```

In the above “standard” way to call `cook` with a force field generated by `blend`, `molcfg` is called transparently from `cook*`. Problems may occur with “optimized water models” recognized by `cook`, which may change the name. Then see below

Usage 1:

```
molcfg FILE1 FILE2 ... SIMNAME
```

Repeats `FILE1.mol` and `FILE1.gol` (if exists) `N[0]` times, `FILE1.mol` `N[1]` times, etc., where `N[0]`, `N[1]`, etc., are read from `SIMNAME.def` used in the simulation. Note: only variable `init`, `no`, `noint` are accepted as temporary variables in `SIMNAME.def` if you wish to use formulas to calculate `N[]`: `init=20 N[0]=init^3` is correct, `eps=20 N[0]=eps^3` is incorrect.

Example 1:

```
molcfg Li Al Cl I salt
show salt
```

Usage 2:

```
molcfg -COUNT1[PREFIX1:SUFFIX1:FROM1] FILE1 \
[-COUNT2][PREFIX2:SUFFIX2:FROM2] FILE2 ... SIMNAME
```

Repeats `FILE1.mol` `COUNT1` times, etc.; `SIMNAME.def` is ignored. `PREFIX` and `SUFFIX` are added to atom ID and may contain format (as `%d`) to hold file number, starting from `FROM` (default=1)

Example 2:

```
molcfg -1 cyto -10w hoh -3:p%d proton -1 etoh config
show config -Yp1 -Yp2 -Yp3
```

\*Source: `util/molcfg.c`

## 23.2 show V 1.0c: Viewing playback (trajectory) files

`show` shows playback files (trajectories) generated typically by `cook` (with option `-y` and variable `dt.plb` set). These files have typical extensions `.plb`. In addition, description of molecules (mol-file and optional gol-file) is needed; details are explained in the manual of `blend`. See also `molcfg` (see Sect. 23.1) and other utilities working with the playback files (see Sect. 22).

Run it without parameters to get list of options, type `[h]` to get a list of hot keys. More information can be found in the source file `show/show.c`. Some options may be turned on/off at compile time and need not be available.

Usage:

```
show [OPTIONS] MOLNAME[.mol|.gol|.plb] [ PLBNAME.EXT ]
```

Input files:

**MOLNAME.mol** The mol-file, generated by `pdb`, `blend`, `molcfg` (see Sect. 23.1), or `cook` (which actually calls `molcfg`)

**MOLNAME.gol** Optional file with colors and atom diameters (based on van der Waals radii) generated by `blend`, `molcfg` (see Sect. 23.1), or `cook`. If missing, colors and diameters are derived from atom types which is less precise (some atom names need not be recognized) but in most cases acceptable.

**PLBNAME.plb** Playback file generated by `cook` `[-y]` or `blend` `[-p]`; the corresponding mol- and gol-files can be obtained by `molcfg` (see Sect. 23.1)

**PLBNAME.3db** Obsolete: `blend` configuration file (1 frame, cf. option `-k0`)

**FORMATFN** Playback file name with int format (in C convention), e.g., `nm%04d.plb`. Cf. option `-h`.

Missing PLBNAME means PLBNAME=MOLNAME

Output files are numbered PLBNAME.0000.EXT, PLBNAME.0001.EXT, etc. The extensions .EXT are:

.ps PostScript: see hotkeys W, C

.nff ray scene file (in Neutral File Format), see Sect. 23.3: N

.pov PovRay scene file: V

.plb Output playback: L or Ctrl-S

.zb Z-buffer (for stereo, see Sect. 23.7): B

.ppm Portable Pixel Map (P6: raw, depth=255): P

NOTE: the P6 PPM file has the following format:

```
P6
# optional comment (some interpreters allow max 1 line of comment)
XSIZE YSIZE
DEPTH
RGBRGB...
```

where XSIZE YSIZE are ASCII-coded integers, DEPTH=255, and RGB are binary 1-byte values.

Data up to DEPTH are standard Unix (LF-separated) lines. After the last LF, a series of XSIZE\*YSIZE\*3 bytes follows, in the range [0,DEPTH]; usually DEPTH=255. These data are interpreted as YSIZE lines of XSIZE triples RGB. It is therefore very easy to manipulate the PPM files. There are publicly available libraries for manipulating the ppm files, and I have also written a couple of them (cutting and cropping, rescaling, making palette, lettering, merging (side-by-side or over)...). For simplicity, I include into MACSIMUS only 23.4 and 23.5.

.edt blend edit file, see option -

Lowercase options:

-a% [100; DOS full screen VGA:120] Pixel aspect ratio for bar and ball modes

-b# [0] -b1 Show the backbone only provided that the mol-file has been originally obtained from a PDB source

-b-1 As above and also show hetero molecules

-b2 Show the backbone obtained by a general backbone algorithm. This may be imprecise for proteins

-bg XCOLOR X11: set background color, e.g., -bg white or -bg #b000ff

- cXYZ [0] Center molecule, XYZ are digits 0 or 1 (e.g., -c010 centers in y-direction, -c111 centers in all coordinates).
- d# [50] Initial delay between frames. The units used to be ms, but now they are arbitrary. Cf. hot keys s, S.
- e# [1] Extent of group. Size of group for options -n and -p. -e1: group (defined by bonded neighbors), -e0: group=individual atom.
- f# [-2] Free (not bonded) atoms are shown (in the bond showing modes) as a cross (X-shaped) of size # pixels. If #<0, too small atoms are shown as one pixel also in the ball modes.
- g XGEOMETRY X11: Set window size and position, e.g., -g 300x300+100+100
- g# [5] DOS: set initial showing mode (1..7)
- h# [0] Set # for the 1st file for FORMATFN
- i# [DOS:ns\*1.1+10; X11:ns\*2+10] Deprecated: Set the initial number of bonds to show. No longer needed because the field grows on demand.
- j# Show water in a shell (of width # Å) around a protein (or any non-water molecule). Water molecule is recognized by HOH or w in the atom ID; use the following form of molcfg to do it (example):
 

```
molcfg -1 pept -999:w%d hoh mysystem
```

See also hotkeys j, J to change the shell thickness.
- k# [2: plb; 0:3db] Skip header of # floats (plb files have normally a header of 2 floats but 3db-files don't)
- l# Use periodic boundary conditions with box size L=#
- l As above but read L from the plb-file
- mPLBFILE:# A playback file (incl. extension) to be matched with PLBNAME.plb. # is the frame to be matched. Matching means rotating and translating the molecule so that the sum of squared distances of corresponding atoms is minimum. See hot keys ' and " (used to be m and M in older versions)..
- m-PLBFILE:# As above, with reversed endian of PLBFILE
- n%@ Show negative: re-color atom by color @ (@ is one of WYRGBCM — see option -W\$\$\$) if it is a member of a group of charge smaller than % e (in per cent). See also option -e. Example: -n-80Y will show all groups more negative than -0.8e in yellow.

- o# In bar or bar-and-ball modes, show always the first # atoms as (big) balls. E.g., for showing a protein as balls and water around as bars, find the number of atoms (in the mol-file) and use -o#, then switch to the bar mode (hot key !). Number # can be also changed by typing o or O.
- p%@ Show positive: re-color atom by color @ (@ is one of WYRGBCM — see option -W\$\$\$) if it is a member of a group of charge bigger than % e (in per cent). See also option -e. Example: -n-80Y will show all groups more negative than -0.8e in yellow.
- q% [50] Cue depth: in ball modes, atoms that are farther from the viewpoint are darker, creating thus an illusion of space. -q0 turns the cue depth off (all atoms are of the same luminance), -q100 is maximum (the farthest balls all so dark that almost invisible). May require readjusting on displays with nonstandard gamma.
- r Reverse endian of input playback (and .3db) files. See Sect. 19.7.
- r%g%b% [-r30g59b11] Coefficients for RGB → gray conversion for gray PostScript output (hotkey O)
- s% [100] Center molecule + set the initial scale factor (in % of certain default value derived from the molecule actual size). Cf. hotkeys +, -.
- s-# Do not center and set the initial scaling factor absolutely (in units of 0.001A/display width). Useful for setting reproducible viewpoint and angle of two different molecules.
- t# [-1] Floor (or ‘table’) will be (-t) or will not be (-t0) included in the NFF or POV scene dumped by hot keys N or V, respectively. The default (-t-1) is context sensitive: when dumping one frame -t applies, when dumping a series (which is supposed to serve for creating a movie) -t0 applies.
- u0 No limit for sphere/bar size, i.e., hotkey R can create arbitrarily large balls or bars. The default is that these sizes cannot reach certain limit because showing extremely large objects is slow.
- u Thick bars (in the ball-and-bar modes) of the size of balls are shown instead of dumbbells (balls connected by thin bars for bonds). This looks ugly because of cheap method of showing bars, but raytraced pictures show nicely the molecule skeleton. The same as hot key D.
- w% [0] Add extra white to colors for PostScript or NFF output. To adjust output for some printers where colors are too dark.
- x#
- y#
- z# Add (x,y,z) in units of L (=box size) to the respective coordinates. Add (Cf. option -l.



Uppercase and special character options:

**-I\$\$\$** X11 only: initial keystroke (sequence of hot keys). Quit is **Q**. Useful in scripts.  
NOTE: because of a delayed action of some keys, you should include toggle **k** (once) before any output (see hotkey **[k]**) Example: **-s70 "-I\*\*xk%NoQ"** will rescale the image to 70% (**-s70**), increases the rotation angle 4x (**\*\***), rotates by x-axis (**x**), sets the 'every key' mode (**k**), dumps one NFF-file (**No**), and quits (**Q**).

**-W\$\$\$ -Y\$\$\$ -R\$\$\$ -G\$\$\$ -B\$\$\$ -C\$\$\$ -M\$\$\$ -O\$\$\$** Atoms and bonds with string \$\$\$ in ID will be colored (instead of their native colors derived from the mol or gol file) as follows: W=white, Y=yellow, R=red, G=green, B=blue, C=cyan, M=magenta, O=omit. Several these options can be combined, in this case they are scanned from right (i.e., the more distinct should be left). Example:

```
show cytox.mol -YLYS7 -MLYS
```

will show lysine 7 (and also lysine 70 :-( ) in yellow and other lysines in magenta while `show cytox.mol -MLYS -YLYS7` would show all lysines in magenta.

**-K** Carbons are black, not cyan.

**-H[#] [@]** Show hydrogen bonds (with distance acceptor—donor less than #) by color @. The defaults (with **-H**) are: # = 2.24, @ = G.

**-Aacceptor**

**-Ddonor** Set the acceptor and donor atom types for **-H**. The defaults are **-AO**, **-DH**.

**-\#** For the GOLD version of **cook**: show the surface as square grid (lattice) of # bars in each direction. Negative #: show the surface using 'gold atoms' Note: \ must be protected from the shell, e.g., **-\6** Default: show box Turn off by **-\0**

**-\#rRAD** As above, with RAD as the GOLD atom van der Waals radius (will be multiplied by 70% to get sphere radius and shift of the plane in the grid mode)

**-\#rRAD**

**-\#rRAD\** Show slit pore, see above for details. Example: **-\-1r2.3\** (show slit pore by atoms of diameter 2.3\*0.7)

**-\** Show the simulation box (cube) around

**- # [0]** Record clicked atoms in a **blend** edit file (extension **.edt**). Such a file can be then used by **blend** for editing the molecule.

- 1 Use command **ra** (=remove atom), append the edit file (also only - )
- 2 Use command **ra** (=remove atom), rewrite the edit file.
- -1 Use command **rm** (=remove molecule), append the edit file.
- -2 Use command **rm** (=remove molecule), rewrite the edit file.

Mouse:

**left button click** Show atom info and show the distance and angle to atoms clicked previously. If option **-e**, write atom name to file **PLBN####.edt**. BUG: works reliably in bond modes only

**left button drag** Rotate

**middle button click** Quit — need to be clicked twice.

**middle button drag** Move in x and y directions

**right button click** Redraw screen

**right button drag** Horizontal: rotate around z-axis, vertical: rescale screen

Hot keys:

**SPACE** Redraw screen

**ESC** Quit or break: must be pressed once more. (To end dumping a series or merged file, use **e**)

**0** Pause the playback

**1** Show every frame (timestep=1 frame). This is the default

**2** Show every second frame (timestep=2 frames), etc.

**9** Show every 9-th frame (timestep=9 frames)

**A** DOS only: Dump a VGA animation file. Can be played by hot key **a** or separately by **showvga**

**a** DOS only: Play the VGA animation file

**b** Toggle forward/backward playback

**B** Ball mode only: Dump the z-buffer. To be used by **stereo** (see Sect. 23.7) to create a stereogram. Should be followed by one of the following keys:

**o** dump one frame

**s** start dumping a series: any subsequent showing a frame will dump a file too, with the number increased by 1. You should use hot key **e** to end dumping the series

**m** start dumping a merged file: any subsequent showing a frame will append the information to the file. You should use hot key **e** to end dumping and close the file

**c** Recenter molecule

- C Bar and ball modes only: Dump color postscript image as a copy of the screen. See B for consequent keys
- d Re-set the default ball and bar sizes
- D In bar-and-ball (dumbbell) modes, make balls of the same size as bars. The same as option `-u` — see there for details.
- e End dumping a series or merged file
- f Freeze (stop-go) toggle, the same as 0 and number etc.
- g
- G Cycle between 7 showing modes (see ! @ # \$ % ^ &)
- h Print help (on hot keys and mouse)
- i Start playback from the beginning of file
- J With option `-j`: increase water shell width
- j With option `-j`: decrease water shell width
- k Toggle between mode ‘every key processed’ and ‘cumulate keys’. The default is ‘cumulate keys’: if e.g. several hot keys x are pressed and rendering is slow (i.e., extra x are pressed while showing), only the final rotated molecule is shown. It is faster. In the ‘every key processed’ mode, all intermediate orientations are shown, too. This is slower but looks like animation.
- K Toggle using of alternate colors in dumping scenes. To be used with keys N and V. Can be used for certain special effects like moves of molecules depicted by different colors
- l Toggle color of cutting plane (see \_)
- L Ball mode only: dump new playback. See B for consequent keys
- ’ Perform match of the current frame with the configuration specified by option `-mPLBFILE:#` (old versions: m)
- " As above, more accurately (lower error in the minimization) (old versions: M)
- m Toggle rotate/move the whole configuration/marked atoms.
- M Unmark all atoms
- N Bar and ball modes only: Dump a NFF file (`ray` scene file, see Sect. 23.3). See B for consequent keys
- o

- O Decrease/increase # of atoms shown always in the ball mode. See option `-o`.
- P Dump a PPM file. E.g., to create a ‘movie’ cheaply, you may use the following sequence of hot keys: `% P s i e`. The resulting series of files can be (via an appropriate script) converted to gif-files (e.g., use utility called `pnmto gif`), then merge these gifs to one animated gif (e.g., use `gifmerge`) that can be shown easily by a web browser.
- q Quit or break: must be pressed once more. (To end dumping a series or merged file, use `e`)
- Q Quit (enough to be pressed once)
- r Smaller balls or thicker bars
- R Larger balls or thinner bars
- s Slows down the playback speed (includes longer delays)
- S Speeds up the playback (includes shorter delays)
- Ctrl-S** Save the current frame (as `PLBNAME.####.plb`)
- t Toggle the trace mode. In this mode, the old screen is not erased before showing the next frame. In the bond modes, the old frame(s) are blue, on other modes just all is drawn over (with proper visibility)
- T As above, and erases (by random black pixels) gradually the previous picture. Repeating T causes faster erasing. Turned off by pressing t.
- u In bar-and-ball (dumbbell) modes, decrease the relative size of balls to bars
- U In bar-and-ball (dumbbell) modes, increase the relative size of balls to bars
- Ctrl-U** Unmark all atoms (also `M`)
- V Bar and ball modes: Dump a POV file (scene for PovRay raytracer). See `B` for consequent keys. You will need include file `show/show.inc`. See comments in `show/show.c` on how to raytrace with PovRay
- w ‘swing’: when the end of playback file is reached, plays backwards, etc. Useful e.g. for showing normal mode vibrations (see `blend`, option `-N`).
- W Dump black and white postscript image. In the bar and bond modes, this is a copy of the screen, in the bond mode (`^`), a black-and-white postscript drawing is dumped. See `B` for consequent keys. Color postscript is `C`. (In old versions, this was `O`)
- x
- y

- z Rotate around axes (anticlockwise). The x-axis is horizontal, y is vertical, and z points up towards the viewpoint
- X
- Y
- Z Rotate around axes (clockwise)
- ' Repeat command prefix (similar to `ESC` in emacs). E.g., `' 1 2 ]` will go by 12 frames forward. In the 'every key processed' mode (see `k`), all frames will be shown, otherwise only the final frame.
- ? Print help (on hot keys and mouse)
- ! Show molecule using bars (sticks) of the same color  
Show molecule using bars (sticks) colored in halves by the color of the closest atom
- # As mode `!` and equally sized balls for atoms (dumbbell mode)
- \$ As mode `@` and equally sized balls for atoms (dumbbell mode with halved bars)
- % Show molecule using balls (spheres) of sizes proportional to van der Waals diameters. BUG: the balls in central projection are displayed only approximately. The closer the viewpoint and larger sphere, the larger error. E.g., when viewing hard spheres, the spheres may look like they overlap a bit though in reality they don't. Use the parallel projection `=` to see the real overlaps.
- ~ Show molecule using bonds of the same color (if not changed by e.g. option `-W$$$`). Similar but much faster than mode `!`.
- & Show molecule using halved bonds of colors derived from the closest atom. Similar but much faster than mode `@`.
- \* Double the rotation angle for keys `x` `X` etc. Maximum angle is 90 degrees.
- / Halve the rotation angle for keys `x` `X` etc. The minimum angle is 0.044 degrees.
- cursors** Move the molecule in x and y directions
- =** Toggles central and parallel projection modes
- End** Central projection only: Move the viewpoint (eye) out of the screen
- Home** Central projection only: Move the viewpoint (eye) closer to the screen
- +** Enlarge molecule (rescale screen). The position of the viewpoint rescales in the same ratio

- Shrink molecule (rescale screen). The position of the viewpoint rescales in the same ratio
- . Go to the beginning of the playback (show first frame)
- , Go to the end of the playback (show last frame)
- [ Go by 1 frame backwards or forwards; if the playback file name contains format (see **FORMATFN**), load the previous or next file
- { Go by 10% of the file length backwards or forwards
- In ball mode only: toggle cutting the molecule by a plane perpendicular to the z-axis. See also hotkey 1
- \
- | Toggle WALL (gold) showing modes
- <
- > Move the cutting plane in the z-direction (see \_)
- (
- ) Move the cutting plane in the z-direction (see \_) – fine move

**Ctrl-C** Kill

\*Source: **show/show.c**

## 23.3 ray: The raytracer

A Reasonably Intelligent Raytracer by Mark VandeWettering, modified by J. Kolafa. The recommended raytracer for MACSIMUS. I found it more suitable for rendering molecules than PovRay (unless you want to see wooden atoms in fog..) because (1) it is simple, (2) can better handle the ambient light, and (3) uses the normal right-handed coordinate system. Nevertheless, if you prefer PovRay, you will find info on how to use it in **show/show.c**.

The scene file for this raytracer has extension **.nff** (for Neutral File Format) and can be generated by **show**. The output file is PPM (see Sect. 23.2).

To get help on options, run

```
ray -h
```

Simple example (render **sim-0000.nff** dumped by **show**):

```
ray -n sim-0000
```

Another example (as above, 1/2 size, no antialiasing (faster), watch progress, view the final picture by 'xv')

```
ray -n sim-0000 -S.5 -j1 -t -vxv
```

Usage details:

```
ray { -Option Argument | -OptionArgument } ...
```

-h Help

-i FILE Input scene file (Neutral File Format), recommended extension **.nff**

-o FILE Output file (P6 Portable Pixel Map), recommended extension **.ppm**

-n FILE Input=FILE.nff, output=FILE.ppm

-t Show progress indicator in %

-u Cheap and fast antialiasing by 2x2 blur (formerly -f)

-v VIEW Show picture using viewer VIEW. Usual viewers (Unix) are **xv**, **display**; for Windoze, e.g. IrfanView understands the ppm format. In case of problems, try [23.6](#).

-j # Antialiasing and jittering. The default is -j-9

#=0 No antialiasing nor jittering (fastest)

#>0, #!=N<sup>2</sup> Random jittering with # samples/pixel

#>0, #=N<sup>2</sup> Antialiasing by N\*N supersamples/pixel in a square

#<0, #=-N<sup>2</sup> As above but supersample only if contrast > c.

#<0, #=-N<sup>2</sup> -d# With option -d#: if error > c, enlarge supersample N  
→2N+1 (N even) or N →3N (N odd)

-d # Diffuse light. # is the light size for Gaussian jittering (for each light source).  
Use -j with a large argument (at least 100). Will create soft shadows.

-c # Threshold for smart supersampling (-j-#), default=0.02

-x # x size in pixels, default=command 'resolution' in NFF file

-y # y size in pixels, default=command 'resolution' in NFF file

-r # The same as -x# -y#

-s # Scale view angle (zoom) by # [default=1]. Use # < 1 if you do not see the whole molecule, # > 1 if you wish to look closer

- S # Scale x and y size by # [default=1]. Will create smaller image but does not change proportions.
- a # Pixel aspect ratio (y/x) [default=1]
- f # Fog from z-coordinate (no fog in front of this z) [0]
- F # Fog thickness for exponential attenuation to 1/e (to background color), 0=off [0]  
Note: the fog algorithm is simple and works OK only with uniform background, best light cyan or blue
- b FILE Use given background image (must be P6 PPM file), instead of color (command b in the NFF file)
- X # Scale background image # times horizontally [1]
- Y # Scale background image # times vertically [1]
- U # Move background image up by #\*height [0]
- R # Move background image right by #\*width [0]
- B # How the background image is treated [default=+2]
  - #>0: tile background image
  - #<0: chessboard mirror tile (make nonperiodic images continuous)
  - +1 -1 Use the color of command b in the NFF file front background (this is not directly visible, but may reflect in the rendered spheres)
  - +2 -2 Calculate background color for front as background image average [default]
  - 1 # Light scaling factor [1] (effective brightness adjusted to -I -N). Use # > 1 if the image is too dark and vice versa.
- I # Ambient isotropic light [0.1]. This is the light shining on the scene from all sides isotropically and reflecting isotropically.
- N # Ambient light proportional to cos angle(normal,ray) [0.2]. This is the light shining on the scene from all sides isotropically but reflecting more to the normal. Both -I -N improve the appearance of spheres.

NFF: Detailed explanation is in `ray/NFF.desc`. Note that `ray` uses the normal right-handed coordinate system.

# Comment

v 1st command to start scene description

from X Y Z The viewpoint (eye)

at X Y Z Look at point (center of the 'screen' or 'paper')



up X Y Z Direction up vector on the ‘screen’ or ‘paper’

angle ANGLE Viewing angle of the width of the  
‘screen’ or ‘paper’, in degrees

hither 1 ?

resolution 533 400 Size of the image in pixels

l X Y Z Position of a point light source. There may be several light sources. All have  
the same intensity

b R G B Background, in RGB (Red Green Blue in interval [0,1]).

f R G B diffuse specular Phong transmittance index Set the material. **diffuse**  
is the amount in [0,1] of the light reflected diffusely, **specular** is the amount in  
[0,1] of the light reflected in the same angle (this makes the ‘mirror’ effect), Phong  
is the power determining the size and brightness of the highlights. For molecules,  
the **transmittance** and the **index** of refraction will never be used.

s X Y Z RADIUS A sphere

c X1 Y1 Z1 RADIUS1 X2 Y2 Z2 RADIUS2 A cone of the axis defined by (X1,Y1,Z1)-  
(X2,Y2,Z2). If both radii are the same, this is a cylinder.

p # Polygon of # vertices. # lines of X Y Z should follow.

pp # Polygonal patch primitive (one-sided). # lines of X Y Z Xnormal Ynormal  
Znormal should follow. Not used by **show**

\*Source: ray/main.c

## 23.4 ppm2ps: PPM, PBM, PGM to PostScript conversion

Both **ray** and **show** generate pictures in the PPM format. Since I was not satisfied with the quality of printing these files using available utilities, I wrote my own. All six versions (P1–P6) are supported.

Call by:

```
ppm2ps [OPTIONS] [INFILE [OUTFILE]] [OPTIONS]
```

Options:

**-r#** Resolution in DPI (default = –75).  
# < 0: minimum resolution (=small pictures small, large fit to page),  
# = 0: smart autoselect good for 600dpi printers.

- a# y pixel aspect ratio (VGA 320x200 has 1.2) (default=1)
- x# x-position in cm: 0=center (default), # > 0: left margin, # < 0: right margin.  
Ignored for -e
- y# y-position in cm: 0=center (default), # > 0: top margin, # < 0: bottom margin.  
Ignored for -e
- X# x size of the picture in cm, overrides -r
- Y# y size in cm, overrides -r; both -X and -Y override both -r and -a
- p Portrait orientation (default)
- l Landscape orientation
- i Invert colors or gray scale (processed after -d,-w)
- e Output is Encapsulated PostScript level 1 (-x and -y are ignored)
- c Compress image (using run length encoding). Useful if the image contains large one-color areas. Usually not good for dithered B/W images (-l -d#).
- 0 Autoselect output format [default]
- 1[#] Output B/W image [default for P1,P4]. #=dither square [default=threshold (change by -d)]. (THIS OPTION HAS CHANGED RECENTLY)
- 2 Output gray scale image (converted from color if necessary, default for P2,P5) (THIS OPTION HAS CHANGED RECENTLY)
- 3 Output RGB image (default for P3,P6)
- 4 Output CMYK image (some printers like this)
- d# Change input depth for gray (default=depth in file)
- d#,#,# Change depths for RGB (default=depth in file)
- R# Add pseudo-random number (in 5\*5 square mod 8, to fix dithering): try -R32 for DeskJet
- R#,#,# As above, for each color separate amplitude
- w The same as -d248,252,248 (16 bit TrueColor adjustment of white)
- g The same as -d248,248,248 (16 bit gray adjustment of white)
- L Letter paper (default=A4)
- G# Gamma correction [default=1.0=no correction]
- # Do not copy PPM/PGM/PBM comment to PS/EPS (here # stands for itself, not a number)

Missing `OUTFILE = stdout`, missing both `INFILE` and `OUTFILE = filter` (cannot for DOS).

Source: `ray/ppm2ps.c`

## 23.5 `ppminfo`: Get information on ppm,pbm,pgm-files

Call by (verbose info):

```
ppminfo FILE
```

Call by (brief info):

```
ppminfo FILE FILE [FILE ...]
```

Source: `ray/ppminfo.c`

## 23.6 `showppm`: View a (sequence of) ppm-file(s)

Very primitive PPM (P6) X11/DOS viewer. To show one ppm-file:

```
showppm PPM-FILE
```

To show a sequence (e.g., generated by `show` or a raytracer):

```
showppm MOVIE%INTFMT.ppm [FROM]
```

Where `%INTFMT` is a suitable integer format. For instance, a series of ppm files dumped by `show mysim@` can be viewed by

```
showppm mysi%04d.ppm
```

There are some hot keys that can be obtained (in X11 only?) by pressing h

\*Source: `show/showppm.c`

## 23.7 `stereo`: Stereogram

This program calculates stereograms normally with the eye cross point behind the paper, i.e., your eyes should watch an imaginary point lying behind the paper in approximately the same distance as the eye-paper distance (this is for the recommended default of the x-pattern size one half of the eye's distance). Different people may prefer different distances or even stereograms with the cross point above the paper.

It has several version, in the newest one (`#define COLOR 2`), it works with PPM files. Two input files are needed:

1. The ZB-file with a z-buffer data. It can be generated by **show**, or it may be even an ASCII file.
2. The pattern file. Its x-size should be about one half of the eye distance with the resolution you would like to use. Certain random patterns may be generated in **stereo** directly (use **pattern=5** and other variables).

To run interactively (asks for data and file names):

```
stereo
```

To process ZB-file(s) using the def-file **stereo.def** and stop:

```
stereo FILE.zb [...]
```

For more information, see file **util/stereo.def** and the source.

\*Source: **util/stereo.c**

# Chapter 24

## Miscellaneous utilities

### 24.1 pdb2pdb: Rearrange pdb-files

Rearranges a pdb-file so that the order of the backbone atoms is N-CA-C-O[-sidechain]. This is ‘standard’, though most programs (incl. `pdb`) do not care. Call by:

```
pdb2pdb [MAXLINES] < INPUTPDB > OUTPUTPDB
```

Source: `blend/pdb2pdb.c`

### 24.2 ramachan: Ramachandran plot from blend and playback files

Call by:

```
ramachan [OPTIONS] SYSNAME [SIMNAME.plb|SIMNAME.p00] [OPTIONS]
```

Options:

- a** Angles in interval [0,360) [default=[-180,180]]
- r** Write results to separate files `SIMNAME.r#`, where `#`=frame number. [default=all frames concatenated to one file `SYSNAME.ram`]
- s** Write summary phi,psi file (1line=1frame; for columns see `SYSNAME.mar`) This file can be used, e.g, by `autocorr` to obtain time correlation functions. Example:

```
autocorr -t -c2 -m360 test.sum
```

where see `test.sum` which dihedral is column 2 (of `-c2`) In addition, this file can be used as `SIMNAME.ddf` for the `DIHHIST=-1` version of `cook` as selection of phi,psi angles.

- f# From frame (first frame to process) [default=1]
- t# To frame (last frame processed) [default=-1=until eof]
- b# By frame (stride) [default=1=every frame]
- p# The calculated Ramachandran plot is plotted (using `plot`, see Sect. 19.2)).
- n# Number of molecules. It should be the same as `N[0]` specified in the simulation def-file. [default=1] BUG: Only simulations of identical molecules are supported (`N[1]=N[2]=..=0` in the simulation def-file). WARNING: if no option `-n#` is specified, only the 1st molecule from the plb-file is processed. Similarly if `-n#` is less than the actual number of molecules (`N[0]`).
- PPARSET Parameter set (prepends environment variable `BLENDPATH`) The default parameter set is that defined in file `SYSNAME.ble`.

Files:

`SYSNAME.ble` Input ble-file

`SIMNAME.plb,SIMNAME.p00` Input playback file

`SYSNAME.mar` Output dihedral angle info extracted from `SYSNAME.ble`

`SYSNAME.ram` Output Ramachandran plot(s) omega,phi,psi,omega: The first table from `SYSNAME.ble` with site info, tables calculated from playback files follow (if not option `-r`).

`SYSNAME.sum` Output Ramachandran summary of phi,psi by columns, 1 line= 1 frame of plb-file

`SIMNAME.r1,...` Tables calculated from playback files (if `-r`)

`${BLENDPATH}/PARSET.par` Parameter file (if given by option `-p`). Only table 'backbone' is used from this file.

Undefined angles/sites are denoted 999/-1

Source: `blend/ramachan.c`

## 24.3 makepept: Makes a peptide in che-format

Makes a peptide of given residues in a che-format. The environment is the same as for `pdb` and `blend`.

Print one letter aminoacid codes:

```
makepept ANYPARM
```

Make a peptide:

```
makepept RSDDIR {rsd | RRR} [rsd | RRR ...]
```

RSDDIR Subdirectory of BLENDPATH with residues in che-format.

rsd Residue or terminus name in lowercase

RRR Chain of one-letter aminoacid codes (uppercase)

Example (make peptide Acetyl-ALA-PRO-THR-HIS(neutral)-Methyl)

```
makepept charmm22 ace APT hisn ct1 > pept.che
```

BUG: for termini, the output file has to be edited (ct1 above)

Recommended blend commands:

```
blend -e40 pept.che      # no pept.mol,pept.3db
blend -e40 -r2 pept.che # to try again (overwrite pept.mol,pept.3db)
```

Source: blend/makepept.c

## 24.4 blefilt: Blend-file filter.

Extracts some information ‘hidden’ in ble-files. Call by:

```
blefilt TABLE [COL [COL2 ...]] [TABLE2 ... ] < INPUT.ble > OUTPUT
```

where TABLE is one of { sites bonds angles dihedrals impropers aromatics } and optional list of columns follows. The default column is the column with the value of bond length or angle, or x y z for sites. The output is in the order of the blend-file. Examples:

```
blefilt angles dihedrals < cyto.ble > cyto.int
blefilt sites < cyto.ble > cyto.xyz
blefilt bonds 2 4 7 < cyto.ble > cyto.int
```

Source: blend/blefilt.c

## 24.5 bonds: Make (show-able) mol-file from coordinates

Call by:

```
bonds { FILE.3dt | FILE.atm | FILE.pdb } [DIST]
```

**FILE.3dt** Data are in 3 columns x,y,z. DIST is the bond threshold [default=1.5]

**FILE.atm** Data are in 4 columns, ATOM,x,y,z, where ATOMS is atom symbol. The bond threshold =  $r_0 \cdot \text{DIST}$ , where  $r_0$  is maximum bond length for given atom-atom bond. Only atoms H C O N S P are explicitly considered, other atoms use certain general values that may be OK in many cases. [default DIST=1.2, values less than 1 do not have too much sense]

**FILE.pdb** As above, PDB format.

**FILE.mol** output mol-file

**FILE.plb** output plb-file

The output mol and plb files can be used by **show**, but are not suitable for blend.

Hint: combination of **bonds FILE.pdb**; **show FILE** can serve as a simple PDB viewer.

Source: `blend/bonds.c`

## 24.6 cutprt: Shorten a prt-file

Shortens prt-files (generated by **cook**) leaving the first header and the final statistics. Call by:

```
cutprt [-]FILE.prt [[-]FILE.prt ...]
```

**FILE.prt** rewrites file **FILE.prt** by the shortened one, the old one is renamed to **FILE.prt**

**-FILE.prt** output to stdout

Source: `util/cutprt.c`

## 24.7 lattice: Make a cubic lattice

Call by

```
lattice N L LATTICE
```

**N** Number of vertices

**L** Box size

**LATTICE=1** Simple cubic lattice

**LATTICE=2** Body centered lattice=1

**LATTICE=3** Face centered lattice

Source: `c/lattice.c`



## 24.8 showpro: Show sorted pro-files

pro-files are generated by `blend` and contain energies of the ‘probe’ (atom or water molecule) in a grid around a protein. Call by:

```
showpro PRO-FILE [LE]
```

where `LE > 1` means that # of data in low-energy histogram will be divided by `LE`; default=1.

Source: `blend/showpro.c`

## 24.9 view3db: View 3db files or convert to 3dt

Makes ASCII dump of 3db files. Call by:

```
view3db 3db-FILE  
view3db 3db-FILE > 3dt-FILE
```

See also see Sect. [22.5](#).

Source: `blend/view3db.c`

# Part IV

## Appendixes

# Chapter 25

## Ewald summation

To simplify notation, let us first define function

$$e(y) = \frac{\operatorname{erfc}(y)}{y} = \frac{1}{y} \frac{2}{\sqrt{\pi}} \int_y^\infty \exp(-t^2) dt \quad (25.1)$$

Basic formula for the Ewald electrostatic energy in periodic boundary conditions surrounded by a dielectric continuum (dielectric constant  $\epsilon'_r$ ) in infinity is in SI

$$4\pi\epsilon_0 U = \sum_{j<l, r_{lj}<r_c} q_j q_l \alpha e(\alpha r_{lj}) \quad (25.2)$$

$$+ \sum_{\vec{k} \neq \vec{0}, |\vec{k}/\vec{L}| < \kappa} \frac{\exp[-(\pi/\alpha)^2 (\vec{k}/\vec{L})^2]}{2\pi V (\vec{k}/\vec{L})^2} Q(\vec{k})^2 \quad (25.3)$$

$$+ \frac{2\pi}{2\epsilon'_r + 1} \frac{M^2}{V} - \frac{\alpha}{\sqrt{\pi}} \sum_j q_j^2 \quad (25.4)$$

where

$$Q(\vec{k}) = \sum_j q_j \exp[2\pi i (\vec{k}/\vec{L}) \cdot \vec{r}_j] \quad (25.5)$$

$$\vec{M} = \sum_j q_j \vec{r}_j \quad (25.6)$$

and the symbol for “division of vectors” is interpreted as vector

$$(\vec{k}/\vec{L}) = (k_x/L_x, k_y/L_y, k_z/L_z) \quad (25.7)$$

The first sum in (25.4) (the r-space or pair sum) assumes that the cutoff  $r_c$  is less than half the minimum box size.  $\vec{M}$  is the dipole moment of the simulation cell. It is well-defined only for configurations of neutral molecules which are never split by a periodic cell boundary.

The implementation of  $e(y)$  in MACSIMUS is based on the following two functions,

$$\operatorname{eru}(x) = \beta \alpha e(\alpha \sqrt{x}) \quad (25.8)$$

$$\operatorname{erd}(x) = -\beta \alpha^3 \frac{e'(\alpha \sqrt{x})}{\alpha \sqrt{x}} \quad (25.9)$$

where  $e'(x) = de(x)/dx$  and  $\beta = 1$ . Note that the term appearing in the r-space sum is  $\alpha e(\alpha r_{lj}) = \text{eru}(r_{lj}^2)$ , where  $r_{lj}^2$  is directly available. These functions are implemented by splines, See Sect. 11.6.

The forces are

$$\begin{aligned}
4\pi\epsilon_0 \vec{f}_j &= -4\pi\epsilon_0 \frac{\partial U}{\partial \vec{r}_j} \\
&= \sum_{l, r_{lj} < r_c} q_j q_l \alpha^3 \frac{e'(\alpha r_{lj})}{\alpha r_{lj}} \vec{r}_{lj} \quad \text{NB: } \alpha^3 \frac{e'(\alpha r_{lj})}{\alpha r_{lj}} = \text{erd}(r_{lj}^2) \\
&+ \sum_{\vec{k} \neq 0, |\vec{k}/\vec{L}| < \kappa} \frac{\exp[-(\pi/\alpha)^2 (\vec{k}/\vec{L})^2]}{V(\vec{k}/\vec{L})^2} 2(\vec{k}/\vec{L}) \text{Im} \left\{ Q(k)^* \exp[2\pi i (\vec{k}/\vec{L}) \cdot \vec{r}_j] \right\} q_j \\
&+ \frac{4\pi}{2\epsilon'_r + 1} q_j \frac{\vec{M}}{V}
\end{aligned}$$

The electrostatic potential is

$$\begin{aligned}
4\pi\epsilon_0 \Phi(r_j) &= 4\pi\epsilon_0 \frac{\partial U}{\partial q_j} \\
&= \sum_{l, r_{lj} < r_c} q_l e(\alpha r_{lj}) \\
&+ \sum_{\vec{k} \neq 0, |\vec{k}/\vec{L}| < \kappa} \frac{\exp[-(\pi/\alpha)^2 (\vec{k}/\vec{L})^2]}{2\pi V(\vec{k}/\vec{L})^2} 2 \text{Re} \left\{ Q(k) \exp[2\pi i (\vec{k}/\vec{L}) \cdot \vec{r}_j] \right\} \\
&+ \frac{4\pi}{2\epsilon'_r + 1} \frac{\vec{r}_j \cdot \vec{M}}{V} - \frac{2\alpha}{\sqrt{\pi}} q_j
\end{aligned}$$

Note that contributions of  $\vec{k}$  and  $-\vec{k}$  are the same in the k-space sums, which is (along with treating null components of  $\vec{k}$  separately) used in the code.

For ready-to-code formulas including the pressure tensor, see [19]. Differences from our notation are:

- $h = 2\pi(\vec{k}/\vec{L})$  (vector)
- the r-space parameter is called  $\kappa$  (here:  $\alpha$ , while  $\kappa$  means k-space)
- dielectric response  $\epsilon'_r$

# Chapter 26

## MD of Polarizable Force Fields

Jiří Kolafa, 1999

### 26.1 Notation

Vector from  $i$  to  $j$ :  $\vec{r}_{ij} = \vec{r}_j - \vec{r}_i$ ,  $r_{ij} = |\vec{r}_{ij}|$

Gradient:  $\vec{\nabla}_i = \partial/\partial\vec{r}_i$ ,  $\vec{\nabla}_i r_{ij} = -\vec{r}_{ij}/r_{ij}$ ,  $\vec{\nabla}_j r_{ij} = \vec{r}_{ij}/r_{ij}$

Direct (tensor) product:  $\overleftrightarrow{A} = \overrightarrow{a} \overrightarrow{b}$

Dot (scalar) product of two vectors:  $A = \overrightarrow{a} \cdot \overrightarrow{b}$

Dot (matrix) product of two tensors:  $\overleftrightarrow{C} = \overleftrightarrow{A} \cdot \overleftrightarrow{B}$

Inverse tensor:  $\overleftrightarrow{A} \cdot \overleftrightarrow{A}^{-1} = \overleftrightarrow{A}^{-1} \cdot \overleftrightarrow{A} = 1$

### 26.2 Polarizability

Induced dipole on particle  $i$  is generally a function of field  $\vec{E}_i$  at position  $\vec{r}_i$ :

$$\vec{\mu}_i = \vec{\mu}_i(\vec{E}_i) \quad (26.1)$$

where we assume that  $\vec{\mu}_i = 0$  for  $\vec{E}_i = 0$ . As we shall see later, field  $\vec{E}_i$  is rather an “effective” field which may contain certain terms behaving as electrostatic field. We shall consider two cases, the *linear polarizability*

$$\vec{\mu}_i(\vec{E}_i) = \overleftrightarrow{\alpha}_i \cdot \vec{E}_i$$

where  $\overleftrightarrow{\alpha}_i$  is the polarizability tensor, and a simple model of *hyperpolarizability* exhibiting saturation given implicitly by

$$\vec{\mu}_i(\vec{E}_i) = \overleftrightarrow{\alpha}_i \cdot \frac{\vec{E}_i}{1 + \vec{E}_i \cdot \vec{\mu}_i / E_{\text{sat}}}$$

which can be solved for  $\vec{E}_i$ :

$$\vec{E}_i = \frac{\vec{\alpha}_i^{-1} \cdot \vec{\mu}_i}{1 - \vec{\mu}_i \cdot \vec{\alpha}_i^{-1} \cdot \vec{\mu}_i / E_{\text{sat}}}$$

The “saturation energy”  $E_{\text{sat}}$  has a physical meaning of the energy of significant deviation from the linear law. For the denominators in the above formulas it holds

$$(1 + \vec{E}_i \cdot \vec{\mu}_i / E_{\text{sat}})(1 - \vec{\mu}_i \cdot \vec{\alpha}_i^{-1} \cdot \vec{\mu}_i / E_{\text{sat}}) = 1$$

Finally, for  $\vec{\alpha}_i = \alpha_i$  (scalar):

$$\vec{\mu}_i(\vec{E}_i) = \frac{\alpha_i \vec{E}_i}{\frac{1}{2} + \sqrt{\frac{1}{4} + \alpha_i E_i^2 / E_{\text{sat}}}}$$

## 26.3 Pair operators of electrostatic interaction

$$T_{ij} = \frac{1}{r_{ij}}$$

$$\vec{T}_{ij} = \vec{\nabla}_i T_{ij} = \frac{\vec{r}_{ij}}{r_{ij}^3} = -\vec{\nabla}_j T_{ij} = -\vec{T}_{ji}$$

$$\vec{\nabla}_i \vec{\nabla}_j T_{ij} = \vec{\nabla}_j \vec{T}_{ij} = -\vec{\nabla}_i \vec{T}_{ij} = \frac{1}{r_{ij}^3} - \frac{3\vec{r}_{ij}\vec{r}_{ij}}{r_{ij}^5}$$

$$\vec{\nabla}_i \vec{\nabla}_j \vec{T}_{ij} = -\vec{\nabla}_j \vec{\nabla}_i \vec{T}_{ij} = 3\frac{\vec{1}\vec{r}_{ij}}{r_{ij}^5} + 3\frac{(\vec{r}_{ij})}{r_{ij}^5} + 3\frac{\vec{r}_{ij}\vec{1}}{r_{ij}^5} - 15\frac{\vec{r}_{ij}\vec{r}_{ij}\vec{r}_{ij}}{r_{ij}^7}$$

where  $\vec{1}$  is a unit isotropic tensor,  $\vec{1}_{\alpha\beta} = \delta_{\alpha\beta}$ , and  $(\vec{r})_{\alpha\beta\gamma} = \delta_{\alpha\gamma}\vec{r}_\beta$  so that  $\vec{a} \cdot (\vec{r}) \cdot \vec{b} = (\vec{a} \cdot \vec{b})\vec{r}$ .

## 26.4 Electrostatic energies

Charge-charge:  $U_{\text{CC}}(i, j) = q_i T_{ij} q_j$

Dipole-charge:  $U_{\text{DC}}(i, j) = \vec{\mu}_i \cdot \vec{T}_{ij} q_j$

Dipole-dipole:  $U_{\text{DD}}(i, j) = \vec{\mu}_i \cdot \vec{\nabla}_i \vec{T}_{ij} \cdot \vec{\mu}_j$

Polarization self-energy of dipole  $\vec{\mu}_i$  is

$$U_{\text{self}}(\vec{\mu}_i) = \int d\vec{\mu}_i \cdot \vec{E}_i(\vec{\mu}_i)$$

where the integration is along any path from zero induced dipole to  $\vec{\mu}_i$  and  $\vec{E}_i$  is the total effective electrostatic field acting on the dipole (what “effective” means becomes clear later). For the linear and saturated models, respectively:

$$U_{\text{lin}}(\vec{\mu}_i) = \frac{1}{2} \vec{\mu}_i \cdot \vec{\alpha}_i^{-1} \cdot \vec{\mu}_i = \frac{1}{2} \vec{E}_i \cdot \vec{\mu}_i$$

$$U_{\text{sat}}(\vec{\mu}_i) = -\frac{E_{\text{sat}}}{2} \ln \left( 1 - \frac{\vec{\mu}_i \cdot \overset{\leftrightarrow}{\alpha}_i^{-1} \cdot \vec{\mu}_i}{E_{\text{sat}}} \right) = \frac{E_{\text{sat}}}{2} \ln \left( 1 + \frac{\vec{E}_i \cdot \vec{\mu}_i}{E_{\text{sat}}} \right)$$

Note that  $\lim_{E_{\text{sat}} \rightarrow \infty} U_{\text{sat}}(\vec{\mu}_i) = U_{\text{lin}}(\vec{\mu}_i)$ .

*Repulsive antipolarization* (shell-core model, deformable dipole model—how shall we call it?) is based on an idea that close contact of two atoms pushes the electron shell around atom (normally anion) out of center giving rise to a dipole in the opposite direction that is the polarized dipole:

$$U_{\text{rep}} = -\kappa_i \vec{\mu}_i \cdot \vec{f}_{ij} \quad (26.2)$$

where  $i$  is a polarizable atom,  $j$  any other atom, and

$$\vec{f}_{ij} = -\vec{\nabla}_i u_{ij}(r_{ij}) = \frac{\vec{r}_{ij}}{r_{ij}} u'_{ij}(r_{ij}) = -\vec{f}_{ji}$$

is the repulsive pair force acting on atom  $i$  due to atom  $j$  and  $u_{ij}$  is the repulsive pair potential. This potential does not include electrostatic interaction<sup>1</sup>. In the typical case  $i$  is an anion,  $j$  a cation, and then  $\kappa_i$  is positive.

## 26.5 Electrostatic field

Field at  $\vec{r}_i$  caused by charge  $q_j$  at  $\vec{r}_j$ :  $\vec{E}_i = -q_j \vec{T}_{ij}$

Field at  $\vec{r}_i$  caused by dipole  $\vec{\mu}_j$  at  $\vec{r}_j$ :  $\vec{E}_i = -\vec{\mu}_j \cdot \vec{T}_{ij}$

Fictitious field caused by repulsive antipolarization:  $\vec{E}_i = \kappa_i \vec{f}_{ij}$

This is not real electrostatic field but as regards its interaction with dipoles, (26.2) and (26.1), can be treated in the same way.

## 26.6 Total energy

The total electrostatic energy of a set of  $n$  polarizable atoms is

$$U = \sum_{i,j,i < j} q_i T_{ij} q_j + \sum_{i,j,i \neq j} \vec{\mu}_i \cdot \vec{T}_{ij} q_j + \sum_{i,j,i < j} \vec{\mu}_i \cdot \vec{T}_{ij} \cdot \vec{\mu}_j - \sum_{i,j,i \neq j} \kappa_i \vec{\mu}_i \cdot \vec{f}_{ij} + \sum_i U_{\text{self}}(\vec{\mu}_i) \quad (26.3)$$

This  $U(\{\vec{\mu}_i\})$  as a functional of  $\{\vec{\mu}_i\}$  reaches a minimum for  $\{\vec{\mu}_i\}$  satisfying (26.1) for all  $i$  where

$$\vec{E}_i \equiv \frac{dU(\vec{\mu}_i)}{d\vec{\mu}_i} = \sum_{j,j \neq i} \left( -q_j \vec{T}_{ij} - \vec{\mu}_j \cdot \vec{T}_{ij} + \kappa_i \vec{f}_{ij} \right) \quad (26.4)$$

is the total (real electrostatic and fictitious) field at  $\vec{r}_i$ .

---

<sup>1</sup>There is a question whether it should generally include dispersion forces; the Tosi et al. potential does not include cation-anion dispersion forces at all.

For  $\kappa_i = 0$  (term (26.2) is missing), we can get a somehow simpler expression not containing the dipole-dipole energy by inserting (26.1) and (26.4) into (26.3):

$$\begin{aligned}
U = & \sum_{i,j,i < j} q_i T_{ij} q_j + \frac{1}{2} \sum_{i,j,i \neq j} \vec{\mu}_i \cdot \vec{T}_{ij} q_j \\
& + \begin{cases} 0 & \text{for linear polarizability} \\ \sum_i \frac{1}{2} \left[ E_{\text{sat}} \ln(1 + \vec{E}_i \cdot \vec{\mu}_i / E_{\text{sat}}) - \vec{E}_i \cdot \vec{\mu}_i \right] & \text{for saturated polarizability} \end{cases}
\end{aligned} \tag{26.5}$$

which is especially simple if the polarizability is linear.

## 26.7 Forces

Let us take a minus gradient of (26.3). Since  $\vec{\mu}_j$  depend on  $\vec{r}_i$ , several terms containing  $\vec{\nabla}_i \vec{\mu}_j$  appear, however, they all cancel out as a consequence of (26.1) with (26.4)<sup>2</sup>. The resulting force ready for implementing is

$$\begin{aligned}
\vec{f}_i = -\vec{\nabla}_i U = & \sum_{j,j \neq i} \left( -q_i \vec{T}_{ij} q_j + \vec{\mu}_i \cdot \vec{T}_{ij} q_j - q_i \vec{T}_{ij} \cdot \vec{\mu}_j - \vec{\mu}_i \cdot \vec{T}_{ij} \cdot \vec{\mu}_j - \kappa_i \vec{\mu}_i \cdot \vec{\nabla}_i \vec{f}_{ij} \right) \\
& - \begin{cases} \vec{\mu}_i \cdot \vec{\nabla}_i \vec{\alpha}_i^{-1} \cdot \vec{\mu}_i & \text{for linear polarizability} \\ \vec{\mu}_i \cdot \vec{\nabla}_i \vec{\alpha}_i^{-1} \cdot \vec{\mu}_i (1 + \vec{E}_i \cdot \vec{\mu}_i / E_{\text{sat}}) & \text{for saturated polarizability} \end{cases}
\end{aligned}$$

### 26.7.1 Gradient of the repulsive antipolarization

The force caused by the repulsive antipolarization term is

$$\vec{f}_{\text{rep},i} = -\kappa_i \vec{\mu}_i \cdot \vec{\nabla}_i \vec{f}_{ij} = \kappa_i \left[ \frac{u'_{ij}}{r_{ij}} \vec{\mu}_i + \frac{\vec{\mu}_i \cdot \vec{r}_{ij}}{r_{ij}} \left( \frac{u'_{ij}}{r_{ij}} \right)' \vec{r}_{ij} \right], \tag{26.6}$$

to be completed by  $\vec{f}_{\text{rep},j} = -\vec{f}_{\text{rep},i}$ .

### 26.7.2 Gradient of the polarizability tensor

For isotropic and constant (=not depending on configuration) polarizability,  $\vec{\alpha}_i = \vec{1} \alpha_i$ ,  $\vec{\nabla}_j \alpha_i = 0$ , the last term in (26.6) is zero. Let us consider the simplest case of axial polarizability of atom  $i$  where the polarizability in the direction of a chemical bond (towards certain atom with position  $\vec{r}_t$ ) is  $\alpha_{zz}$  and in the perpendicular directions it is  $\alpha_{xx} = \alpha_{yy}$ . The polarization tensor is

$$\vec{\alpha}_i = \alpha_{xx} \vec{1} + (\alpha_{zz} - \alpha_{xx}) \frac{\vec{r}_{it} \vec{r}_{it}}{r^2} \tag{26.7}$$

---

<sup>2</sup>That is why it is more convenient to take a gradient of (26.3) and not of the simpler form (26.5) which would in fact lead to more complicated calculations to eliminate terms containing  $\vec{\nabla}_j \vec{\mu}_i$



The inverse tensor is

$$\overset{\leftrightarrow}{\alpha}_i^{-1} = \alpha_{xx}^{-1} \overset{\leftrightarrow}{1} - \frac{\alpha_{zz} - \alpha_{xx}}{\alpha_{zz}\alpha_{xx}} \frac{\vec{r}_{it} \vec{r}_{it}}{r^2} \quad (26.8)$$

by taking the gradient we obtain the corresponding ‘axial polarization’ force term

$$\vec{f}_{\text{ax},i} = -\vec{\mu}_i \cdot \vec{\nabla}_i \overset{\leftrightarrow}{\alpha}_i^{-1} \cdot \vec{\mu}_i = \frac{\alpha_{zz} - \alpha_{xx}}{\alpha_{zz}\alpha_{xx}} \left[ \frac{\vec{\mu}_i \cdot \vec{r}_{it}}{r_{it}^2} \vec{\mu}_i - \frac{(\vec{\mu}_i \cdot \vec{r}_{it})^2}{r_{it}^4} \vec{r}_{it} \right]$$

and finally  $\vec{f}_{\text{ax},t} = -\vec{f}_{\text{ax},i}$ .

### 26.7.3 Fluctuating charge

Currently, MACSIMUS implements the fluctuating charge model of water with four charges. The method comes from the FQ models [26].

Let us consider a neutral molecule with four interaction sites numbered  $i \in \{0, 1, 2, 3\}$  which do not lie in a plane. Each site bears a permanent charge  $q_{i0}$ ,  $i \in \{0, 1, 2, 3\}$ ,  $\sum_i q_{i0} = 0$ . In electric field with potential  $\Phi_i$  at site  $i$  the charges change into  $q_i = q_{i0} + \delta q_i$ . We write the electrostatic energy as

$$U = U_{\text{self}} + \sum_i (q_{i0} + \delta q_i) \Phi_i, \quad (26.9)$$

where the self energy is [26]

$$U_{\text{self}} = \sum_i \sum_j \frac{A_{ij}}{2} \delta q_i \delta q_j.$$

At fixed field  $\Phi_i$  the configuration of charges minimizes energy  $U$  under constraint  $\sum_i \delta q_i = 0$ . By adding term  $\lambda \sum_i \delta q_i$  to (26.9) ( $\lambda$  is the Lagrange multiplier) and taking derivatives over  $\delta q_i$ , we arrive at four equations

$$\sum_j A_{ij} \delta q_j + \Phi_i + \lambda = 0. \quad (26.10)$$

Given  $A_{ij}$  and  $\Phi_i$ , these linear equations (along with  $\sum_i \delta q_i = 0$ ) yield the unknown charges.

The polarizability tensor  $\alpha$  can be calculated by applying homogeneous fields in three directions and calculating the response dipole moment. However, in constructing the force field the task is opposite: to determine ten independent components of symmetric matrix  $A_{ij}$  given six independent components of  $\alpha$ . The task is overdetermined, but all solutions are equivalent because the set of shifted constants

$$A'_{ij} = A_{ij} + a_i \quad (26.11)$$

gives the same charges and thus the same energy (the Lagrange multiplier  $\lambda$  need not be the same, though). The dimensionality of this space is four.

For the FQ4 model of water there are a number of symmetries. We have five independent components of matrix  $A$

$$A = \begin{pmatrix} A_{HH} & A_{H'H} & A_{HL} & A_{HL} \\ A_{H'H} & A_{HH} & A_{HL} & A_{HL} \\ A_{HL} & A_{HL} & A_{LL} & A_{L'L} \\ A_{HL} & A_{HL} & A_{L'L} & A_{LL} \end{pmatrix} \quad (26.12)$$

where HH stands for the diagonal (quadratic) terms,  $H_i H_i$ , while  $H'H$  denotes the off-diagonal terms,  $H_i H_j$ ,  $i \neq j$ . There are only two independent conditions (26.11). We therefore we will impose  $A_{H'H} = A_{L'L} = 0$ . The solution is

$$\delta q_{H1} = \frac{(4A_{HL} - 2A_{HH} - A_{LL})\Phi_{H1} + (A_{LL} - 4A_{HL})\Phi_{H2} + A_{HH}(\Phi_{L1} + \Phi_{L2})}{2A_{HH}(A_{HH} + A_{LL} - 4A_{HL})} \quad (26.13)$$

and the remaining  $q_i$  can be obtained by  $H1 \leftrightarrow H2$  and  $H \leftrightarrow L$  symmetries.

For the FQ4 model of water in the standard orientation the polarizability tensor is diagonal and thus the remaining three values,  $A_{HL}$ ,  $A_{H'H}$ ,  $A_{L'L}$ , can be determined from the diagonal components of the polarizability tensor.

Let the positions of charge sites are (with respect to Oxygen):

$$\vec{r}_{H1} = (H_x, H_y, 0) \quad \vec{r}_{H2} = (H_x, -H_y, 0) \quad (26.14)$$

$$\vec{r}_{L1} = (L_x, 0, L_z) \quad \vec{r}_{L2} = (L_x, 0, -L_z) \quad (26.15)$$

The the polarizability tensor is

$$\alpha = \text{diag} \left( \frac{2(H_x - L_x)^2}{A_{HH} + A_{LL} - 4A_{HL}}, \frac{2H_y^2}{A_{HH}}, \frac{2L_z^2}{A_{LL}} \right) \quad (26.16)$$

from which the unknown constants  $A_{HH}$ ,  $A_{LL}$ ,  $A_{HL}$  can be readily calculated.

#### 26.7.4 Implementation

Fluctuating charges are stored in the 2nd vector (rpol). ASPC is directly applicable.

# Chapter 27

## Time-reversible predictors for Verlet+SHAKE with a velocity-dependent rhs

### 27.1 The task

Our task is to integrate numerically by the Verlet method (with optional SHAKE) the following set of equations

$$\ddot{x} = a(x, \dot{x})$$

where  $x$  stands for the set of  $\vec{r}_i$  and the Nose variable  $\xi = \ln s$ . The Verlet algorithm reads as

$$x(t+h) = 2x(t) - x(t-h) + h^2 a(t)$$

where  $\dot{x}(t)$  (needed to calculate  $a(t)$ ) is not known. It may be expressed by

$$\dot{x}(t) = \frac{x(t+h) - x(t-h)}{2h} + \mathcal{O}(h^2)$$

but the resulting equations have to be calculated by iterations. If SHAKE is involved, the SHAKE part must be repeated in this iteration. Another possibility is the velocity Verlet with RATTLE.

### 27.2 MACSIMUS solution

MACSIMUS implements a set of predictors with a good time-reversibility. The velocity predictor can be written in the form

$$\dot{x}^p(t) = \frac{1}{h} \sum_{i=0}^{k+1} A_i x(t - ih),$$

The  $k+2$   $A_i, i = 0, \dots, k+1$  can be determined from the Taylor expansion of the right-hand side

$$\sum_{i=0}^{k+1} A_i x(t - ih) = \sum_{i=0}^{\infty} X_i x^{(i)} h^i$$

It must hold ( $k + 2$  equations)

$$\begin{aligned}
X_0 &= \sum_{i=0}^{k+1} A_i = 0 \\
X_1 &= -\sum_{i=0}^{k+1} i A_i = -1 \\
X_2 &= \sum_{i=0}^{k+1} i^2 A_i = 0 \\
X_4 &= \sum_{i=0}^{k+1} i^4 A_i = 0 \\
&\vdots \\
X_{2k} &= \sum_{i=0}^{k+1} i^{2k} A_i = 0
\end{aligned}$$

The odd terms are not nullified because they are time-reversible.

The solution is

$$\begin{aligned}
A_0 &= \frac{2k+1}{k+1} \\
A_1 &= -2(2k+1) \frac{1}{k+2} \\
A_2 &= +2(2k+1) \frac{k}{(k+2)(k+3)} \\
A_3 &= -2(2k+1) \frac{k(k-1)}{(k+2)(k+3)(k+4)} \\
&\vdots
\end{aligned}$$

MACSIMUS code uses expansion in the first differences

$$\dot{x}^p(t) = \sum_{i=0}^k B_i \frac{x(t-ih) - x(t-(i+1)h)}{h}$$

with  $B_0 = A_0$  and

$$B_j = (-1)^j (2k+1) \frac{k(k-1) \cdots (k+1-j)}{(k+1)(k+2) \cdots (k+1+j)}$$

or recursively

$$\begin{aligned}
B_0 &= \frac{2k+1}{k+1} \\
B_j &= -B_{j-1} \cdot \frac{k+1-j}{k+1+j}, \quad j > 0
\end{aligned}$$

which is directly coded in MACSIMUS. The default is  $k = 2$ .

## 27.3 Algorithm

One step of the combined Verlet+SHAKE method is

- calculate forces (accelerations)  $a(t)$  from known  $x(t)$
- predict velocities  $\dot{x}$  from known  $x(t) - x(t - h)$ ,  $x(t - h) - x(t - 2h)$ ,  $\dots$ ,  $x(t - kh) - x(t - (k + 1)h)$
- perform one step of the Verlet method (12.1) to get  $r(t + h)$
- run the SHAKE algorithm;  $x(t + h)$  is modified and  $x(t + h) - x(t)$  recalculated
- calculate the kinetic temperature (MACSIMUS supports several formulas, see the switch VERLET; VERLET=9 uses the predicted value)
- perform one step of the Verlet method (12.1) to get  $\dot{\xi}(t + h)$  and  $\xi(t + h)$
- advance time,  $t := t + h$ , and evaluate the Hamiltonian

# Chapter 28

## Always Stable Predictor-Corrector (ASPC) instant

### 28.1 Task

To integrate numerically the Newton equations of motion

$$\ddot{\mathbf{r}}_i = \frac{1}{m_i} \mathbf{f}_i(\mathbf{r}_1, \dots, \mathbf{r}_N; \boldsymbol{\mu}_1, \dots, \boldsymbol{\mu}_m)$$

where  $\mathbf{r}_i$  are positions of atoms (nuclei) and  $\boldsymbol{\mu}_i$  are any variables which are given by an implicit equation of a self-consistent field type:

$$\boldsymbol{\mu}_i = \mathbf{M}_i(\mathbf{r}_1, \dots, \mathbf{r}_N; \boldsymbol{\mu}_1, \dots, \boldsymbol{\mu}_m), \quad i = 1, \dots, m$$

This equation must converge for all  $\mathbf{r}_1, \dots, \mathbf{r}_N$  from the trajectory and (a linearization of)  $M_{ij} = \mathbf{M}_i(\boldsymbol{\mu}_j)$  must be symmetric.

Example of  $\boldsymbol{\mu}$ : induced dipoles,  $\boldsymbol{\mu}_i = \alpha_i \mathbf{E}_i(\mathbf{r}_1, \dots, \mathbf{r}_N; \boldsymbol{\mu}_1, \dots, \boldsymbol{\mu}_m)$ , where  $\mathbf{E}_i$  is the electric field

### 28.2 ASPC

The recommended version with  $\mathcal{O}(h^7)$  time reversibility follows:

**Predictor:**

$$\mu^p(t) = 2.8\mu(t-h) - 2.8\mu(t-2h) + 1.2\mu(t-3h) - 0.2\mu(t-4h)$$

where  $\mu \equiv \{\boldsymbol{\mu}_1, \dots, \boldsymbol{\mu}_m\}$ , and similarly below.

**Corrector:**

$$\mu(t) = \omega M(r(t); \mu^p(t)) + (1 - \omega)\mu^p(t)$$

where  $\omega = \frac{4}{7}$  guarantees stability (for any converging SCF equation), but a certain  $\omega > \frac{4}{7}$  may give more accurate results.

In the above equation,  $M(r(t); \mu^p(t))$  is calculated at time  $t$  and current  $r(t)$ . Normally the forces  $f(t)$  (along with  $E$ ) are calculated at the same step. Then one step of Verlet gives  $r(t+h)$  (which becomes  $r(t)$  in the next step) and the corrector gives  $\mu(t)$  (which becomes  $\mu(t-h)$  in the next step, ready for the predictor).

The method may work also in certain cases of asymmetric  $M_{ij}$ . It will probably work also for complex  $\mu$ .

# Chapter 29

## Specific heat $C_V$ in the molecular dynamics microcanonical ensemble

In the *canonical* ensemble we have for the kinetic energy  $E_k$  and potential (configurational) energy  $E_p$ :

$$\text{Var}E_x = kT^2 \frac{\partial E_x}{\partial T} \equiv kT^2 E'_x$$

where  $x \in \{k, p\}$  and

$$\text{Cov}(E_p, E_k) = 0$$

$$E'_k = fk/2$$

$$E'_p + E'_k = C_V$$

where  $f$  is the number of degrees of freedom and  $C_V$  is the heat capacity (at constant volume) of the whole system.

Thus the (unnormalized) probability distribution for given  $T$  and a state with given  $E_k$  and  $E_p$  is:

$$w(T, E_p, E_k) \sim \exp \left[ -\frac{(\delta E_p - E'_p \delta T)^2}{2 \text{Var}E_p} - \frac{(\delta E_k - E'_k \delta T)^2}{2 \text{Var}E_k} \right]$$

where we linearize at certain point  $(T, E_p, E_k) = (T_0, E_{p,0}, E_{k,0})$ ,  $\delta X \equiv X - X_0$ . In the MD NVE ensemble there is  $E_p + E_k = \text{const.}$  We write  $E_p = -E_k \equiv E$  and express  $E'_x$  via  $\text{Var}E_x$ :

$$w(T, E) \sim \exp \left[ -\left( \frac{1}{2kT^2 E'_p} + \frac{1}{2kT^2 E'_k} \right) \delta E^2 - \left( \frac{E'_p}{2kT^2} + \frac{E'_k}{2kT^2} \right) \delta T^2 \right]$$

Note that there is no  $\delta E \delta T$  cross term! Thus, from the term at  $\delta T^2$ ,

$$\text{Var}T = \frac{kT^2}{E'_p + E'_k} = \frac{kT^2}{C_V}$$

which is known formula for the fluctuation of temperature in the microcanonical ensemble — useless for MD because we do not know  $T$  (the “kinetic temperature” derived from  $E_k$  is *not*  $T$ ).



From the term at  $\delta E^2$  we have

$$\text{Var}E = \text{Var}E_p = \text{Var}E_k = \frac{kT^2}{1/E'_p + 1/E'_k} = \frac{kT^2}{1/(C_V - E'_k) + 1/E'_k}$$

and because we know  $E'_k$  finally

$$C_V = \frac{fk}{2} \left[ \left( \frac{2T^2}{f \text{Var}T_k} - 1 \right)^{-1} + 1 \right]$$

where  $T_k \equiv E_k/(fk/2)$  is the “kinetic temperature”.

# Chapter 30

## Dielectric constant in SI

The permittivity  $\epsilon$  is defined via the electric field intensity  $\vec{E}$  and the induction (displacement field)  $\vec{D}$ . In SI it holds

$$\vec{D} = \epsilon \vec{E} = \epsilon_0 \epsilon_r \vec{E}$$

where  $\epsilon_0 = 8.85418782 \times 10^{-12}$  F/m is the vacuum permittivity and  $\epsilon_r$  is the relative permittivity which equals the dielectric constant defined in the CGS units. The permittivity of an isotropic medium is a scalar.

In CGS simply  $D = \epsilon_r \vec{E}$  (usually subscript  $r$  is not used, but we will keep it for clarity). Note that MACSIMUS uses internally CGS-like (Gaussian electrostatic) units, however, it transparently transforms the units mostly into SI; one notable exceptions is the dipole moment where the internal units are used because it is not clear whether to prefer Debye or Cm.

The electric induction (displacement field) is

$$\begin{aligned} \vec{D} &= \epsilon_0 \vec{E} + \vec{P} & \text{SI} \\ \vec{D} &= \vec{E} + 4\pi \vec{P} & \text{CGS} \end{aligned}$$

where

$$\vec{P} = \frac{\vec{M}}{V}$$

is the electric polarization which equals the volume density of dipole moment  $\vec{M}$ ; this equation in SI and CGS is formally identical, only the units differ.

### 30.1 Dielectric constant from external field

In simulations we apply an external (microscopic) field  $\vec{E}^{\text{ext}}$  (entered as `Eelst[]` in input data in the SI units of V/m). The total intensity  $\vec{E}$  is a sum of the the external field and the field caused by charges in the system. It depends on the boundary conditions. In the Ewald periodic boundary conditions we have a continuum (of relative permittivity<sup>1</sup>  $\epsilon'_r = \text{el.epsinf}$ ) surrounding the infinite array of periodic boxes

---

<sup>1</sup>For the short-range (cutoff) electrostatics `el.epsinf` = 1 (?)

spherically in infinity; to be able to write formulas, we will consider the radius  $R \gg L$ . The polarization  $\vec{P}$  of the sphere is equivalent to a point dipole  $4\pi R^3 P/3$  placed in the center. It induces charges at the surface of the  $R$ -cavity and hence a field which damps the external field. It holds [27] (in SI):

$$\vec{E} = \vec{E}^{\text{ext}} - \frac{1}{\epsilon_0} \frac{\vec{P}}{2\epsilon'_r + 1} \quad (30.1)$$

Let the external field acts in the direction of  $z$  (`Eelst`[2]). During simulation we determine quantity

$$\chi_z = \frac{\langle M_z \rangle}{\epsilon_0 E_z^{\text{ext}} V} \quad (30.2)$$

or in the units used by MACSIMUS

$$\chi_z = \frac{4.4266218\text{e}+09 \times (\text{Mz}/\text{prog.unit})}{[E_z^{\text{ext}}/(\text{V/m})] \times (V/\text{\AA}^3)}$$

The dielectric constant is then

$$\epsilon_r = \frac{\epsilon_0 E_z + P_z}{\epsilon_0 E_z} = 1 + \frac{2\epsilon'_r + 1}{2\epsilon'_r + 1 - \chi_z} \chi_z = 1 + \frac{1}{\frac{1}{\chi_z} - \frac{1}{2\epsilon'_r + 1}} \quad (30.3)$$

where we inserted  $E_z$  from (30.1). Some like the equivalent implicit formula,

$$\chi_z = \frac{(\epsilon_r - 1)(2\epsilon'_r + 1)}{(2\epsilon'_r + \epsilon_r)} \quad (30.4)$$

For the tin-foil boundary conditions ( $\epsilon'_r = \text{el.epsinf} = \infty$ ) it holds  $\epsilon_r = 1 + \chi_z$ .

It is inefficient to use  $\epsilon'_r = 1$  for models with large  $\epsilon_r$ , because then  $\chi_z$  is close to 3 and we have  $3 - \chi_z$  in the denominator of (30.2). On the other hand, large  $\epsilon'_r$  increases correlation length and may also lead to dipole moments close to saturation and systematic errors. Anyway, stronger external field is needed to get the same response for small  $\epsilon'_r$ .

Formula (30.3) is exact for both polarizable and nonpolarizable systems provided that the field is small enough. It is recommended to run several simulations with decreasing field (and increasing length to compensate for decreased accuracy).

## 30.2 Fluctuation formulas

For the dipole moment of the simulation cell (one configuration of charges) under influence of the external electric field in the  $z$ -direction it holds

$$\vec{M}(\vec{E}^{\text{ext}}) = \vec{M}(0) + \frac{\partial \vec{M}}{\partial \vec{E}^{\text{ext}}} \vec{E}^{\text{ext}}$$

where  $\partial \vec{M} / \partial \vec{E}^{\text{ext}}$  stands for the induced dipole moment per unit electric field at constant positions of atoms. (Generally, it is a tensor, however, the off-diagonal terms

are small and their average in isotropic systems is zero.) By repeating the derivation of the fluctuation formula [27] we get a response caused by a small external field. We write it in the  $z$ -direction

$$\langle M_z(E_z^{\text{ext}}) \rangle = \left( \left\langle \frac{\partial M}{\partial E_z^{\text{ext}}} \right\rangle + \frac{\text{Var}M_z}{kT} \right) E_z^{\text{ext}}$$

In isotropic systems  $\text{Var}M = 3 \text{Var}M_z$ . We will express the direct response by a dimensionless number  $\chi^{\text{hf}}$  (a sort of high-frequency susceptibility including boundary conditions<sup>2</sup>)

$$\begin{aligned} \chi^{\text{hf}} &= \frac{1}{\epsilon_0 V} \left\langle \frac{\partial M_z}{\partial E_z^{\text{ext}}} \right\rangle \quad \text{SI} \\ \chi^{\text{hf}} &= \frac{4\pi}{V} \left\langle \frac{\partial M_z}{\partial E_z^{\text{ext}}} \right\rangle \quad \text{CGS} \end{aligned}$$

and the total response

$$\begin{aligned} \chi &= \chi^{\text{hf}} + \frac{\text{Var}M}{3\epsilon_0 V kT} \quad \text{SI} \\ \chi &= \chi^{\text{hf}} + \frac{4\pi \text{Var}M}{3V kT} \quad \text{CGS} \end{aligned}$$

Similarly as (30.3) and (30.4) one can derive

$$\epsilon_r = 1 + \frac{1}{\frac{1}{\chi} - \frac{1}{2\epsilon'_r + 1}} \quad (30.5)$$

or implicitly

$$\chi = \frac{(\epsilon_r - 1)(2\epsilon'_r + 1)}{(2\epsilon'_r + \epsilon_r)}$$

Note that MACSIMUS calculates  $\text{Var}M$  as  $\langle M^2 \rangle$  unless an external field is applied; in the latter case  $\langle M_z \rangle^2$  is subtracted.

For polarizable models it remains to determine  $\langle \chi^{\text{hf}} \rangle$ . MACSIMUS offers a direct numerical method based on applying an external field (`scf.E`) and iterations until the dipoles are accurate enough (`scf.epsx`, `scf.omegax`); the original and new dipole moment are subtracted and divided by the field.

By the same arguments which lead from (30.1) to (30.3) one can relate the “susceptibility”  $\chi^{\text{hf}}$  to the high-frequency (instantaneous, optical) dielectric constant  $\epsilon_r^{\text{hf}}$ ,

$$\chi^{\text{hf}} = (\epsilon_r^{\text{hf}} - 1) \frac{1 + 2\epsilon'_r}{2\epsilon'_r + \epsilon_r^{\text{hf}}}$$

It can be approximated by the the Clausius-Mossotti formula from the total polarizability (assumed to be linear), see eq. (2.6) in [28] (with  $\epsilon_r^{\text{hf}} = \epsilon^\infty$ )

$$\frac{\epsilon_r^{\text{hf}} - 1}{\epsilon_r^{\text{hf}} + 2} = \frac{1}{3\epsilon_0 V} \sum_i \alpha_i^{\text{SI}} = \frac{4\pi}{3V} \sum_i \alpha_i \quad (30.6)$$

---

<sup>2</sup>susceptibility in CGS is normally without the  $4\pi$  factor

In the formula above,  $\alpha_i^{\text{SI}}$  is the SI polarizability of particle  $i$  and  $\alpha_i = \alpha_i^{\text{SI}}/4\pi\epsilon_0$  is its polarizability volume (identical to “polarizability” in CGS and usually referred to as just “polarizability”; in MACSIMUS measured in  $\text{\AA}^3$ ). The induced dipole in SI is  $\vec{\mu} = \alpha_{\text{SI}}\vec{E} = 4\pi\epsilon_0\alpha\vec{E}$ .

### 30.2.1 Notes

Eq. (30.6) is an approximation based on the assumption that a polarizable dipole is in a spherical cavity surrounded by a continuum of the optical dielectric constant  $\epsilon_r^{\text{hf}}$ . An error in determination of  $\epsilon_r^{\text{hf}}$  propagates to the final result more for  $\epsilon'_r = 1$  and less for  $\epsilon'_r = \infty$ . E.g., for the DC97 water model, the error  $\epsilon_r^{\text{hf}} \pm 0.01$  causes the error  $\epsilon_r \pm 7$  with  $\epsilon'_r = 1$  while only negligible  $\epsilon_r \pm 0.01$  with  $\epsilon'_r = \infty$ .

Eq. (30.5) for  $\epsilon'_r = \infty$  (tin-foil b.c.) reduces to

$$\epsilon_r = \epsilon_r^{\text{hf}} + \chi$$

which is identical to the formula in [29].

In cook V2.6e and older, formula

$$\chi^{\text{hf}} = \frac{4\pi}{V} \sum_i \alpha_i \quad (30.7)$$

was incorrectly used. This formula neglects mutual interactions of induced dipoles. It formula is identical to (30.6) for  $\epsilon'_r = 1$  and also for small  $\epsilon_r^{\text{hf}}$  (a second-order term in  $(\epsilon_r^{\text{hf}} - 1) \approx \frac{4\pi}{V} \sum_i \alpha_i^2$  is neglected). E.g., for water at 25 °C there is  $\frac{4\pi}{V} \sum_i \alpha_i = 0.616$  and  $\epsilon_r^{\text{hf}} = 1.775$ . The error caused by using equation (30.7) is 0.16, to be compared to the bulk water value  $\epsilon_r = 78$ .

# Chapter 31

## Fourier transform

### 31.1 Basic formulas

The basic formula in 1D is

$$\tilde{f}(k) = \int_{-\infty}^{\infty} f(x) \exp(-ikx) dx$$

and the inverse transform is

$$f(x) = \frac{1}{2\pi} \int_{-\infty}^{\infty} \tilde{f}(k) \exp(ikx) dk.$$

In 3D

$$\tilde{f}(\vec{k}) = \int f(\vec{r}) \exp(-i\vec{k} \cdot \vec{r}) d\vec{r}, \quad f(\vec{r}) = \frac{1}{(2\pi)^3} \int \tilde{f}(\vec{k}) \exp(i\vec{k} \cdot \vec{r}) d\vec{k}. \quad (31.1)$$

If  $f(\vec{r})$  is spherically symmetric, then (31.1) becomes

$$\tilde{f}(k) = \int_0^{\infty} r^2 dr \int_{-1}^1 d\cos\theta \int_0^{2\pi} f(r) \exp(-i\cos\theta kr) d\phi.$$

After performing the integration over angles

$$k\tilde{f}(k) = 4\pi \int_0^{\infty} \sin(kr) r f(r) dr$$

and back in the same way

$$r f(r) = \frac{1}{2\pi^2} \int_0^{\infty} \sin(kr) k \tilde{f}(k) dk.$$

### 31.2 Implementation

The Fourier transforms in MACSIMUS (see `gen/fourier.c`, `gen/fft.c` and `gen/fft.h`), are implemented as generalized fast Fourier transform. The number of points  $n$  is factored and the algorithm proceeds recursively by taking the prime factors. The cases 2 and 3 are optimized. The efficiency deteriorates for large factors.

In addition to the basic complex Fourier transform, transforms of real, even and odd functions as well as the 3D Fourier transform are available.

## 31.3 Structure factor

To derive the formula (14.14) for  $S_{IJ}$ , let us start from the definitions of  $g_{IJ}$ :

$$g_{IJ}(r) = V \langle \delta(r_{ij} - r) \rangle \times \begin{cases} 1 - 1/N_I & \text{for } I = J \\ 1 & \text{for } I \neq J \end{cases}, \quad (31.2)$$

where  $i(j)$  is any atom of type  $I(J)$ , respectively, and  $\delta$  is the Dirac delta-function. By taking the Fourier transform of (31.2) (for  $I \neq J$ ):

$$\int (g_{IJ}(r) \exp(-i\vec{k} \cdot \vec{r}) d\vec{r} = V \langle \exp(i\vec{k} \cdot \vec{r}) \exp(-i\vec{k} \cdot \vec{r}) \rangle = V \frac{\langle Q_I^*(\vec{k}) Q_J(\vec{k}) \rangle}{N_I N_J}$$

and using (31.3), one gets the partial structure factors

$$S_{IJ}(k) = 1 + N \left[ \frac{\langle Q_I^*(\vec{k}) Q_J(\vec{k}) \rangle}{N_I N_J} - 1 \right]$$

and finally (14.14).

### 31.3.1 Mixtures

Formula (14.14) comes from the following ‘combining rule’ for the ‘partial structure factors’  $S_{IJ}$ , where uppercase letters  $I, J$  index species:

$$S = \sum_I \sum_J w_{IJ} S_{IJ}, \quad w_{IJ} = \frac{N_I b_I N_J b_J}{(\sum_I N_I b_I)^2}$$

and  $N_I$  is the number of atoms of species  $I$ . Note that  $\sum_I N_I b_I = \sum_j b_j$ .

For the ‘partial structure factors’  $S_{IJ}$  it holds

$$S_{IJ} = 1 + \frac{N}{V} \int (g_{IJ} - 1) \exp[-2\pi i \vec{k} \cdot \vec{r}] d\vec{r} \quad (31.3)$$

# References

- [1] J. Applequist, J. R. Carl, K.-K. Fung: *J. Am. Chem. Soc.* **94**, 2952 (1972).
- [2] Z. Akdeniz, G. Pastore, M.P. Tosi: *Phys. Chem. Liq.* **32**, 191 (1996); Z. Akdeniz, G. Pastore, M.P. Tosi: *Nuovo Cimento*, **20**, 595 (1998).
- [3] J. Kolafa: Numerical integration of equations of motion with a self-consistent field given by an implicit equation, *Mol. Simulat.* **18**, 193-212 (1996)
- [4] J. Genzer, J. Kolafa: Molecular dynamics of potential models with polarizability: comparison of methods, *J. Mol. Liq.* **109** 63–72 (2004)
- [5] J. Kolafa: Time-Reversible Always Stable Predictor-Corrector Method for Molecular Dynamics of Polarizable Molecules, *J. Comput. Chem.* **25** 335–342 (2004)
- [6] J. Kolafa: Gear formalism of the always stable predictor-corrector method for molecular dynamics of polarizable molecules, *J. Chem. Phys.* **122** 164105 (2005)
- [7] S. Nosé: *Mol. Phys.* **52**, 255 (1984).
- [8] W.G. Hoover: *Phys. Rev. A* **31**, 1695 (1985).
- [9] 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. Comput. Chem.* **4**, 187, (1983).
- [10] M. L. Bjerking: Grafisk brugerinterface til et Molekyle-dynamik (MD) program, *MSc. thesis*, Odense University (1993).
- [11] *QUANTA Parameter Handbook*, Polygen Corporation (1990).
- [12] W. H. Press, S. A. Teukolsky, W. T. Vetterling, B. P. Flannery: Numerical Recipes in FORTRAN, Cambridge University Press (1992), Chap. 10.6.
- [13] A.D. MacKerell et al.: All-atom empirical potential for molecular modeling and dynamics studies of proteins, *J. Phys. Chem B* **102**, 3586–3616 (1998).
- [14] Jorgensen et al.: *J. Chem. Phys.* **79**, 926 (1983); Jorgensen et al.: *Molec. Phys.* **56**, 1381 (1985).
- [15] M Levitt., M Hirshberg., R Sharon., V Daggett.: *Comput. Phys. Commun.* **91**, 215 (1995).



- [16] J. Kolafa, F. Moučka, I. Nezbeda: Handling Electrostatic Interactions in Molecular Simulations: A Systematic Study *Collect. Czech. Chem. Commun.* **73**, 481–506 (2008).
- [17] G.J. Gloor, G. Jackson, F.J. Blas, E. de Miguel: Test-area simulation method for the direct determination of the interfacial tension of systems with continuous or discontinuous potentials, *J. Chem. Phys.* **123**, 134703 (2005).
- [18] C. Yeh, M.L. Berkowitz, *J. Chem. Phys.* **111**, 3155 (1999).
- [19] A. Aguado, P.A. Madden: *J. Chem. Phys.* **119**, 7471 (2003).
- [20] J. Kolafa, J.W. Perram: Cutoff errors in the Ewald summation formulae for point charge systems, *Mol. Simulat.* **9** 351–368 (1992).
- [21] D.E. Smith, L.X. Dang: *J. Chem. Phys.* **100**, 3757 (1994).
- [22] J.P. Brodholt: *Chem. Geol.* **151**, 11 (1998).
- [23] M.P. Allen: Back to Basics, in *Computer Simulation in Chemical Physics*, ed. M.P. Allen and D.J. Tildesley, Proceedings of the NATO Advanced Study Institute on New Perspectives in Computer Simulation in Chemical Physics, Alghero, Sardinia, Italy, September 14–24 (1992).
- [24] R. L. Davidchack, B. B. Laird: Direct calculation of the crystal–melt interfacial free energies for continuous potentials: Application to the Lennard-Jones system, *J. Chem. Phys.* **118**, 7651 (2003).
- [25] J. Picalek, B. Minofar, J. Kolafa, P. Jungwirth: *Phys. Chem. Chem. Phys.* **10**, 5765–5775 (2008).
- [26] S. W. Rick, S. J. Stuart, B. J. Berne: *J. Chem. Phys.* **101**, 6141 (1994).
- [27] S.W. de Leeuw, J.W. Perram, E.R. Smith: Simulation of electrostatic systems in periodic boundary conditions. I. Lattice sums and dielectric constants. *Proc. R. Soc. Lond. A* **373**, 27–56 (1980).
- [28] A. D. Buckingham: A Theory of the Dielectric Polarization of Polar Substances, *Proc. R. Soc. London A* 238, 235–244 (1956).
- [29] G. Lamoureux, A.D. MacKerell, Jr., B. Roux: *J. Chem. Phys.* **119**, 5185 (2003).
- [30] B. Dünweg, K. Kremer: *J. Chem. Phys.* **99**, 6093 (1994).
- [31] J. Malohlava (University of Ostrava) and J. Kolafa (2010), unpublished results.
- [32] G.J. Martyna, D.J. Tobias, M.L. Klein: Constant pressure molecular dynamics algorithms, *J. Chem. Phys.* **101**, 4177–4189 (1994).
- [33] G.J. Martyna, M.E. Tuckerman, D.J. Tobias, M.L. Klein: Explicit reversible integrators for extended systems dynamics, *Mol. Phys.* **87**, 1117–1157 (1995).

# Index

.1, 103  
.3db, 36, 233  
.3dt, 36, 233  
.S1, 107  
.acc, 98  
.anc, 103  
.ang, 30  
.asc, 98, 103  
.bin, 39  
.ble, 39, 71  
.cfg, 103, 204  
.che, 36, 68  
.cp, 103, 205  
.cpa, 103  
.cpi, 103  
.cpz, 105  
.dcp, 106  
.ddh, 106  
.def, 105  
.dep, 36  
.dia, 106  
.dih, 106  
.dpr, 106  
.edt, 36  
.fix, 106  
.for, 98, 106  
.g, 106  
.geo, 38  
.get, 106  
.gol, 38  
.gra, 98  
.jet, 25, 38  
.keep, 36  
.loc, 99, 106  
.mark, 36  
.mkr, 197  
.mol, 22, 38, 67, 233

- .msd, 31
- .nff, 262, 270
- .p00, 106
- .par, 20, 39, 48, 235
- .pch, 36
- .pdb, 233
- .plb, 38, 206, 233
- .pol, 101, 107
- .pov, 262
- .ppm, 262, 273
- .pro, 37
- .prt, 107
- .ps, 273
- .rdf, 107
- .rea, 37
- .rep, 233
- .s-s, 107
- .sel, 233
- .sfd, 107
- .sta, 107
- .stp, 107
- .sym, 38
- .vel, 98
- .wid, 107
- .z, 107
- .zb, 262
  
- accuracy, 135
  - of constraints, 117, 136
  - of Ewald, 116
  - of induced dipoles, 126, 137
- alpha helix, 28
- alternate location, 234
- aminoacid, 239
- anchor, 195
- aromatic dihedral, 64
- asc2plb, 253
- atomdist, 255
- autocorr, 249
  
- barostat, 128, 151
- bin, 20
- blefilt, 279
- blend-file, 71
- BLENDPATH, 232
- bonds, 279
  
- cache, 109

- calculator, [230](#)
- Car-Parrinello, [181](#)
- center forces, [127](#)
- center molecule, [30](#)
- center of mass, [30](#)
- che, [22](#)
- chemical file, [68](#)
- chemical potential, [200](#)
- chirality, [22](#), [63](#)
- cleaving, [189](#)
- cluster, [26](#)
  - analysis, [171](#)
  - Na<sub>4</sub>Cl<sub>4</sub>, [83](#)
- combining rule, [53](#)
- conductivity, [166](#)
- conjugate gradients, [46](#)
- constrained
  - dihedral, [26](#)
- constraint, [38](#)
- constraint dynamics, [140](#)
- constraint force, [195](#)
- convergence profile, [97](#), [103](#), [158](#), [205](#)
- coordn, [248](#)
- cp2cp, [245](#)
- cppak, [248](#)
- crambin, [81](#)
- cross section, [130](#)
- crystal, [156](#)
- cutoff, [29](#), [112](#), [119](#), [144](#)
- cutoff electrostatics, [178](#)
- cutplb, [253](#)
- cutprt, [280](#)
- cystein bond, [234](#)
  
- density, [125](#), [257](#)
- density profile, [112](#)
- densprof, [254](#)
- dependant, [78](#), [143](#)
- dielectric constant, [115](#), [116](#), [298](#)
- diffusivity, [160](#), [164](#)
- dihedral
  - angle distribution, [112](#)
  - constrained, [26](#)
  - potential, [62](#)
- dipole moment, [45](#)
- DOS, [17](#)
- DOS graphics, [18](#)

- drift, [113](#)
- Drude
  - pressure tensor, [184](#)
- Einstein relation, [164](#)
- endian, [230](#)
- energy conservation, [115](#), [137](#)
- energy minimization, [46](#)
- essential dynamics, [31](#)
- ev, [230](#)
- Ewald parameters setting, [138](#)
- Ewald test, [117](#)
- filtplb, [257](#)
- fix atom, [176](#)
- force field, [16](#)
  - parameter file, [48](#)
- Fourier transform, [302](#)
- frame, [253](#)
- fundamental mode, [33](#)
- get data, [39](#), [108](#)
- GPL, [1](#)
- graphics, [42](#)
- hbonds, [248](#)
- Henry constant, [200](#)
- improper torsion, [62](#)
- inertia matrix, [32](#)
- initial configuration, [118](#), [155](#)
- insertion of particle, [200](#)
- integration method, [98](#)
- interface
  - Gibbs energy, [189](#)
- internal coordinate, [30](#)
- interrupt, [41](#), [97](#), [132](#)
- keep atom, [176](#)
- keep atoms, [26](#)
- kinetic quantity, [160](#)
- lattice, [280](#)
- Lennard-Jones potential, [144](#)
- library, [217](#)
- linked-cell list, [122](#), [133](#)
- lock file, [99](#), [106](#)
- makemake, [222](#)
- makepept, [278](#)

- mar, 278
- mark, 197
- matching rule, 66
- mean square displacement, 31
- mergeplb, 257
- mergetab, 229
- metal wall, 192
- minimize energy, 26
- mirror inversion, 122
- missing coordinate, 47
- mixing iteration parameter, 126
- mixing rule, 53
- molcfg, 260
- molecular dynamics
  - equilibrium, 160
- molecule description, 67
- moment of inertia, 32
- MSD, 31
  
- neutral file format, 262, 270
- normal mode, 33
- normalize configuration, 113
- NPT, 128, 151
- NVT, 147
  
- optimization, 46
- optimize, 26
  
- pair energy, 29, 197
- parallelization, 103, 133
- parameter\_set, 68
- partial charge, 36
- PDB
  - write, 29
- pdb, 232
- pdb2pdb, 277
- periodicity, 65
- permittivity, 115, 298
- playback, 46
  - write, 27
- playback file, 206
- plb2asc, 253
- plb2cryst, 255
- plb2diff, 258
- plb2nbr, 255
- plb2plb, 253
- plbbox, 254
- plbcheck, 252

plbconv, [252](#)  
plbcut, [254](#)  
plbinfo, [252](#)  
plbmerge, [255](#)  
plbmsd, [256](#)  
plbpak, [257](#)  
plot, [222](#)  
polarizability, [179](#)  
portable pixel map, [273](#)  
PostScript, [273](#)  
PovRay, [262](#)  
ppm2ps, [273](#)  
ppminfo, [275](#)  
precision, [135](#)  
pressure tensor, [144](#), [182](#)  
probe, [37](#)  
protein in water, [81](#), [207](#)  
  
quadrupole moment, [45](#)  
  
radial distribution function, [169](#), [246](#)  
radial distribution functions, [125](#)  
ram, [278](#)  
ramachan, [277](#)  
Ramachandran plot, [277](#)  
ray, [270](#)  
raytracing, [270](#)  
RDF, [125](#), [169](#), [246](#)  
rdfg, [246](#)  
reaction, [37](#)  
relaxation parameter for SHAKE, [123](#)  
rescaling, [124](#)  
residue, [239](#)  
  
scroll, [40](#)  
scrolling, [19](#), [100](#), [236](#)  
sfourier, [247](#)  
SHAKE, [123](#), [140](#)  
shear viscosity, [163](#)  
shift, [126](#)  
show, [261](#)  
show molecule, [42](#)  
showcp, [242](#)  
shownear, [258](#)  
showppm, [275](#)  
showpro, [281](#)  
slab, [112](#), [185](#)  
slit pore, [190](#)

smoothg, [246](#)  
smoothpl, [255](#)  
sortcite, [231](#)  
spectrum, [251](#)  
spline, [145](#)  
staprt, [247](#)  
statistical error, [159](#)  
statistics, [120](#)  
steepest descent, [46](#)  
stereo, [275](#)  
stress tensor, [144](#), [182](#)  
structure factor, [167](#), [303](#)  
sum, [278](#)  
surface tension, [185](#)

tab, [229](#)  
tabproc, [228](#)  
temperature, [128](#)  
thermostat, [129](#), [147](#)  
time measurement, [100](#)  
timestep, [118](#), [144](#)  
tomoil, [259](#)  
torsion, [61](#)

units, [127](#)

Verlet, [140](#)  
view3db, [281](#)  
virtual area change, [187](#)  
virtual volume change, [114](#), [187](#)  
viscosity, [163](#), [198](#)

wall, [190](#)  
Widom method, [200](#)

X11, [17](#)

Z-buffer, [262](#), [266](#)  
z-profile, [112](#)