

EMC: Monte Carlo Simulations

A General Simulation Package
EMC version 9.4.4, 1 August 2021

Pieter J. in 't Veld

This manual is for EMC (version 9.4.4, 1 August 2021), a simulation package that can build and simulate both atomistic and coarse-grained systems using Monte Carlo techniques.

Copyright © 2007-2021 Pieter J. in 't Veld.

Permission is granted to copy, distribute and/or modify this document under the terms of the GNU Free Documentation License, Version 1.1 or any later version published by the Free Software Foundation; with no Invariant Sections, with the Front-Cover texts being "A GNU Manual," and with the Back-Cover Texts as in (a) below. A copy of the license is included in the section entitled "GNU Free Documentation License."

(a) The FSF's Back-Cover Text is: "You have freedom to copy and modify this GNU Manual, like GNU software. Copies published by the Free Software Foundation raise funds for GNU development."

Table of Contents

1	Introduction	1
1.1	General Introduction	1
1.2	Distribution Content	1
2	Methodology	3
2.1	Lists	3
2.2	Force Fields	3
3	Program Structure	6
3.1	Molecular Representation	6
3.1.1	Sites	6
3.1.2	Groups	6
3.1.3	Clusters	6
3.2	Molecular Interactions	7
3.2.1	Types	7
3.2.2	Forces	7
3.3	Systems	7
3.4	Configurational Moves	7
3.5	Measurements	7
4	Simulation Setup	8
4.1	General	8
4.2	Setup Usage	9
4.3	Extensions	10
4.4	Environment Options	12
4.5	Chemistry Options	15
4.6	File Formats	32
4.6.1	Environment File	32
4.6.2	Chemistry File	35
4.6.2.1	General	35
4.6.2.2	Shorthand	37
4.6.2.3	Groups	37
4.6.2.4	Clusters	38
4.6.2.5	Polymers	42
4.6.2.6	DPD Additions	42
4.6.3	Field File	45
4.6.3.1	General	45
4.6.3.2	Define	46
4.6.4	References File	49
4.6.5	Parameters File	50
4.7	Examples	51
4.7.1	References	51

4.7.2	Chemistry Mode.....	51
4.7.2.1	Bulk Mixture.....	52
4.7.2.2	Force Fields.....	55
4.7.2.3	Record.....	56
4.7.2.4	Polymers.....	58
4.7.2.5	Multiphase Systems.....	61
4.7.2.6	Surfaces.....	63
4.7.3	Environment Mode.....	65
4.7.3.1	User-Defined Force Fields.....	66
4.7.3.2	Shear.....	69
4.8	Help Output.....	74
5	Scripting Commands.....	81
5.1	Build.....	82
5.1.1	Syntax.....	82
5.1.2	Usage.....	84
5.1.3	Default.....	84
5.1.4	Examples.....	84
5.2	Cancel.....	86
5.2.1	Syntax.....	86
5.2.2	Usage.....	86
5.2.3	Default.....	86
5.3	Carve.....	87
5.3.1	Syntax.....	87
5.3.2	Usage.....	87
5.3.3	Default.....	87
5.3.4	Examples.....	88
5.4	Clusters.....	89
5.4.1	Syntax.....	89
5.4.2	Usage.....	90
5.4.3	Default.....	90
5.4.4	Examples.....	90
5.5	Crystal.....	93
5.5.1	Syntax.....	93
5.5.2	Usage.....	93
5.5.3	Default.....	93
5.5.4	Examples.....	93
5.6	Cut.....	95
5.6.1	Syntax.....	95
5.6.2	Usage.....	95
5.6.3	Default.....	95
5.6.4	Examples.....	95
5.7	Deform.....	96
5.7.1	Syntax.....	96
5.7.2	Usage.....	96
5.7.3	Default.....	96
5.7.4	Examples.....	96
5.8	Duplicate.....	97

5.8.1	Syntax	97
5.8.2	Usage	97
5.9	Export	98
5.9.1	Syntax	98
5.9.2	Usage	98
5.9.3	Default	98
5.9.4	Examples	98
5.9.5	Data Interpretation	98
5.10	Field	101
5.10.1	Syntax	101
5.10.2	Usage	102
5.10.3	Formats	102
5.10.4	Default	105
5.11	Flag	106
5.11.1	Syntax	106
5.11.2	Usage	106
5.11.3	Default	107
5.12	Focus (Command)	108
5.12.1	Syntax	108
5.12.2	Usage	108
5.12.3	Default	108
5.13	Force	109
5.13.1	Syntax	109
5.13.2	Usage	109
5.13.3	Default	109
5.14	Former	110
5.14.1	Syntax	110
5.14.2	Usage	110
5.14.3	Default	110
5.15	Get	111
5.15.1	Syntax	111
5.15.2	Usage	111
5.15.3	Default	111
5.16	Groups	112
5.16.1	Syntax	112
5.16.2	Usage	113
5.16.3	Default	113
5.16.4	Examples	114
5.17	Insight	116
5.17.1	Syntax	116
5.17.2	Usage	117
5.17.3	Default	117
5.17.4	References	117
5.18	Lammps	118
5.18.1	Syntax	118
5.18.2	Usage	119
5.18.3	Default	119
5.18.4	Examples	119

5.18.5	References	120
5.19	Memory	121
5.19.1	Syntax	121
5.19.2	Usage	121
5.19.3	Default	121
5.19.4	Examples	121
5.20	Moves	122
5.20.1	Syntax	122
5.20.2	Usage	122
5.20.3	Default	122
5.21	PDB	123
5.21.1	Syntax	123
5.21.2	Usage	124
5.21.3	Default	125
5.21.4	References	125
5.22	Put	126
5.22.1	Syntax	126
5.22.2	Usage	126
5.22.3	Default	126
5.23	Rename	127
5.23.1	Syntax	127
5.23.2	Usage	127
5.23.3	Default	127
5.23.4	Examples	127
5.24	Remove	128
5.24.1	Syntax	128
5.24.2	Usage	128
5.24.3	Default	128
5.24.4	Examples	128
5.25	Reset	130
5.25.1	Syntax	130
5.25.2	Usage	130
5.25.3	Default	130
5.26	Restart	131
5.26.1	Syntax	131
5.26.2	Usage	131
5.26.3	Default	131
5.27	Retype	132
5.27.1	Syntax	132
5.27.2	Usage	133
5.27.3	Default	133
5.27.4	Examples	133
5.28	Run	134
5.28.1	Syntax	134
5.28.2	Usage	134
5.28.3	Default	134
5.29	Sample	135
5.29.1	Syntax	135

5.29.2	Usage	135
5.29.3	Default	135
5.29.4	References	135
5.30	Shell	136
5.30.1	Syntax	136
5.30.2	Usage	136
5.30.3	Default	136
5.31	Sites	137
5.31.1	Syntax	137
5.31.2	Usage	137
5.31.3	Default	137
5.31.4	Examples	137
5.32	Simulation	138
5.32.1	Syntax	138
5.32.2	Usage	138
5.32.3	Examples	138
5.33	Split	140
5.33.1	Syntax	140
5.33.2	Usage	140
5.33.3	Default	140
5.34	Terminate	141
5.34.1	Syntax	141
5.34.2	Usage	141
5.34.3	Default	141
5.35	Timing	142
5.35.1	Syntax	142
5.35.2	Usage	142
5.35.3	Default	142
5.36	Traject	143
5.36.1	Syntax	143
5.36.2	Usage	143
5.36.3	Default	143
5.36.4	Examples	144
5.37	Translate	145
5.37.1	Syntax	145
5.37.2	Usage	145
5.37.3	Default	145
5.37.4	Examples	145
5.38	Types	146
5.38.1	Syntax	146
5.38.2	Usage	147
5.38.3	Default	147
5.39	Variables	148
5.39.1	Syntax	148
5.39.2	Usage	148
5.39.3	Examples	148
5.40	XYZ	149
5.40.1	Syntax	149

5.40.2	Usage	149
5.40.3	Default	150
5.40.4	References	150
6	Sampling Tools	151
6.1	Bond	152
6.1.1	Syntax	152
6.1.2	Usage	152
6.1.3	Default	152
6.1.4	Example	153
6.2	Cavity	155
6.2.1	Syntax	155
6.2.2	Usage	156
6.2.3	Default	156
6.2.4	Example	157
6.2.5	References	159
6.3	Gr	160
6.3.1	Syntax	160
6.3.2	Usage	160
6.3.3	Default	160
6.3.4	Pair Correlation Functions	161
6.3.5	Definition	161
6.3.6	Relations Involving $g(r)$	162
6.3.6.1	Structure Factor	162
6.3.6.2	Compressibility Equation	163
6.3.6.3	Potential of Mean Force	163
6.3.6.4	Energy Equation	164
6.3.6.5	Pressure Equation of State	164
6.3.6.6	Thermodynamic Properties	164
6.3.7	References	164
6.4	Gyration	165
6.4.1	Syntax	165
6.4.2	Usage	165
6.4.3	Default	165
6.4.4	Theory	166
6.4.5	References	166
6.5	Interaction	167
6.5.1	Syntax	167
6.5.2	Usage	168
6.5.3	Default	168
6.6	Profiles	170
6.7	Examples	171

7	Variable Descriptions	172
7.1	Constants	174
7.1.1	Syntax	174
7.1.2	Usage	174
7.1.3	Default	174
7.2	System Flags	175
7.2.1	Syntax	175
7.2.2	Default	175
7.3	Focus	176
7.3.1	Syntax	176
7.3.2	Usage	176
7.3.3	Default	176
7.4	Moves	177
7.4.1	Syntax	177
7.4.2	Usage	177
7.4.3	Default	177
7.4.4	References	178
7.4.5	Displace	179
7.4.5.1	Syntax	179
7.4.5.2	Usage	179
7.4.5.3	Default	179
7.5	Port	180
7.5.1	Syntax	180
7.5.2	Default	180
7.6	Profiles	181
7.6.1	Syntax	181
7.6.2	Usage	181
7.6.3	Default	181
7.7	Region	182
7.7.1	Syntax	182
7.7.2	Usage	182
7.7.3	Default	182
7.8	SMILES	183
7.8.1	Syntax	183
7.8.2	Usage	183
7.8.3	Examples	184
7.9	Splines	185
7.9.1	Introduction ¹	185
7.9.2	Linear Spline	185
7.9.3	Cubic Spline	185
7.9.4	References	186
7.10	System Flags	187
7.10.1	Syntax	187
7.10.2	Default	187
7.11	Systems	188
7.11.1	Syntax	188
7.11.2	Usage	188
7.11.3	Default	189

7.12	Types.....	190
7.12.1	Syntax.....	190
7.12.2	Usage.....	192
7.12.3	Default.....	192
7.12.4	Boltzmann.....	193
7.12.4.1	Syntax.....	193
7.12.4.2	Example.....	193
7.12.4.3	References.....	194
7.12.4.4	Pair.....	195
7.12.4.5	Bond.....	196
7.12.4.6	Angle.....	197
7.12.5	Born.....	198
7.12.5.1	Syntax.....	198
7.12.5.2	Example.....	198
7.12.5.3	References.....	199
7.12.5.4	Pair.....	200
7.12.5.5	Bond.....	202
7.12.5.6	Angle.....	203
7.12.5.7	Torsion.....	204
7.12.5.8	Improper.....	205
7.12.6	CFF.....	206
7.12.6.1	Syntax.....	206
7.12.6.2	Examples.....	207
7.12.6.3	References.....	208
7.12.6.4	Pair.....	209
7.12.6.5	Bond.....	211
7.12.6.6	Angle.....	212
7.12.6.7	Bond-Bond.....	213
7.12.6.8	Bond-Angle.....	214
7.12.6.9	Torsion.....	215
7.12.6.10	End-Bond-Torsion.....	216
7.12.6.11	Middle-Bond-Torsion.....	217
7.12.6.12	Bond-Bond-13.....	218
7.12.6.13	Angle-Torsion.....	219
7.12.6.14	Angle-Angle-Torsion.....	220
7.12.6.15	Improper.....	221
7.12.7	CHARMM.....	222
7.12.7.1	Syntax.....	222
7.12.7.2	Examples.....	222
7.12.7.3	References.....	223
7.12.7.4	Pair.....	225
7.12.7.5	Bond.....	227
7.12.7.6	Angle.....	228
7.12.7.7	Urey.....	229
7.12.7.8	Torsion.....	230
7.12.7.9	Pair14.....	231
7.12.7.10	Improper.....	232
7.12.8	Coarse.....	233

7.12.8.1	Syntax	233
7.12.8.2	Examples	233
7.12.8.3	References	234
7.12.8.4	LJ	235
7.12.8.5	Repulsive	237
7.12.8.6	Sphere	239
7.12.8.7	Colloid	241
7.12.8.8	DPD	243
7.12.8.9	Charge	245
7.12.8.10	FENE	247
7.12.8.11	Angle	248
7.12.9	Colloid	249
7.12.9.1	Syntax	249
7.12.9.2	Example	249
7.12.9.3	References	250
7.12.9.4	Pair	251
7.12.9.5	Charge	253
7.12.9.6	Bond	255
7.12.9.7	Angle	256
7.12.10	Coulomb	257
7.12.10.1	Syntax	257
7.12.10.2	Examples	257
7.12.10.3	References	258
7.12.10.4	Pair	259
7.12.10.5	Charge	260
7.12.11	DPD	261
7.12.11.1	Syntax	261
7.12.11.2	Example	261
7.12.11.3	References	262
7.12.11.4	Pair	263
7.12.11.5	Bond	265
7.12.11.6	Angle	266
7.12.11.7	Torsion	267
7.12.11.8	Improper	269
7.12.12	GROMACS	270
7.12.12.1	Syntax	270
7.12.12.2	Example	270
7.12.12.3	References	271
7.12.12.4	Pair	272
7.12.12.5	Bond	274
7.12.12.6	Angle	275
7.12.12.7	Torsion	276
7.12.12.8	Pair14	277
7.12.12.9	Improper	278
7.12.13	MARTINI	279
7.12.13.1	Syntax	279
7.12.13.2	Example	279
7.12.13.3	References	280

7.12.13.4	Pair	281
7.12.13.5	Bond	283
7.12.13.6	Angle	284
7.12.13.7	Torsion	285
7.12.13.8	Improper	286
7.12.14	OPLS	287
7.12.14.1	Syntax	287
7.12.14.2	Example	287
7.12.14.3	References	288
7.12.14.4	Pair	289
7.12.14.5	Bond	291
7.12.14.6	Angle	292
7.12.14.7	Torsion	293
7.12.14.8	Pair14	294
7.12.14.9	Improper	295
7.12.15	SDK	296
7.12.15.1	Syntax	296
7.12.15.2	Example	296
7.12.15.3	References	297
7.12.15.4	Pair	298
7.12.15.5	Bond	300
7.12.15.6	Angle	301
7.12.15.7	Torsion	302
7.12.15.8	Pair14	303
7.12.15.9	Improper	304
7.12.16	Spline	305
7.12.16.1	Syntax	305
7.12.16.2	Bond	305
7.12.16.3	Angle	306
7.12.16.4	Torsion	306
7.12.16.5	Pair	307
7.12.16.6	Example	308
7.12.16.7	References	308
7.12.17	Standard	309
7.12.17.1	Syntax	309
7.12.17.2	Example	309
7.12.17.3	References	310
7.12.17.4	Pair	311
7.12.17.5	Bond	313
7.12.17.6	Angle	314
7.12.17.7	Torsion	315
7.12.17.8	Improper	317
7.12.18	Table	318
7.12.18.1	Syntax	318
7.12.18.2	Table Syntax	318
7.12.18.3	Example	319
7.12.18.4	Pair	320
7.12.18.5	Bond	321

7.12.18.6	Angle	322
7.12.18.7	Torsion	323
7.12.19	TraPPE	324
7.12.19.1	Syntax	324
7.12.19.2	Example	324
7.12.19.3	References	325
7.12.19.4	Pair	326
7.12.19.5	Bond	328
7.12.19.6	Angle	329
7.12.19.7	Torsion	330
7.12.19.8	Improper	331
7.13	Units	332
7.13.1	Syntax	332
7.13.2	Usage	332
7.13.3	Default	333
7.14	Vector	334
7.14.1	Syntax	334
7.14.2	Usage	334
7.14.3	Default	334
7.15	Voigt	335
7.15.1	Syntax	335
7.15.2	Usage	335
7.15.3	Default	335
Index	336	

1 Introduction

1.1 General Introduction

Monte Carlo simulation techniques are wide spread through branches of scientific research and financial industry. It finds its application in statistical sampling of complex functional descriptions, giving numerical solutions to theoretically unsolvable equations. Typically codes using Monte Carlo techniques are written with a specific purpose in mind. These approaches include scientific implementations where versatility and not expandability is the main design goal. The latter finds an example in Towhee, a code which offers a wide array of force fields and sampling techniques, but encounters challenges when the need arises for implementation of new techniques. The code presented here does not try to offer the same breath, but rather follows a strategy of expandability. Strict modular design isolates key features in separate objects, yet a general parser allows for easy integration of these objects in a coherent design, thus allowing for a more flexible and expandable design. The purpose of this paper is to present a design philosophy for simulation engines in general, while applying this philosophy to the field of scientific Monte Carlo simulations. This is achieved by the identification of key features, such as input/output features, force field calculation, and configurational alterations, features which are highlighted in later sections.

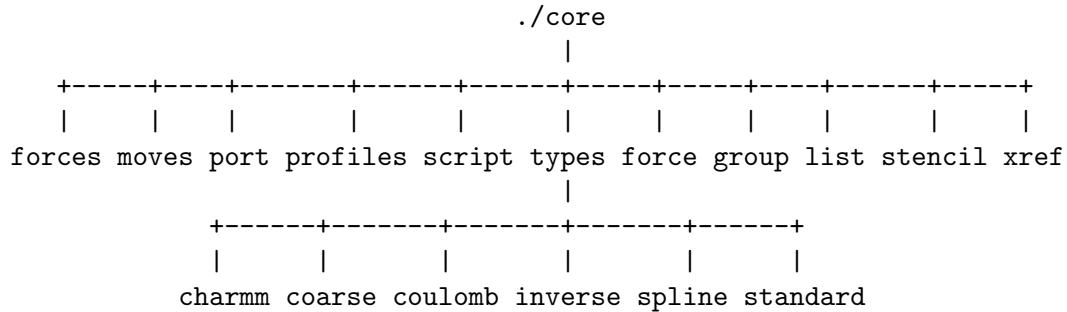
Each simulation code hinges around computational efficiency. Gain in speed can be obtained by an increase in processor speed, but more importantly, by judicious design of the program framework. This requires the identification of key factors of importance. Specifically, computational efficiency for this type of simulations is hampered by its serial nature: atoms or sites are moved one after another and not in a parallel fashion as is the case with Molecular Dynamics. Efficiency of energy calculations becomes the major concern when designing Monte Carlo codes for scientific applications. These applications hinge around the concept of an acceptance rule, in which typically is formed by a combination of system temperature with the difference in energy as the result of a change in configuration. This configuration change or move normally entails the displacement of one or a few sites. The serial nature of these moves necessitates the calculation of configurational energy both before and after the move. Optimization of energy calculations therefore plays an integral role in improving computational efficiency, for which this paper offers suggestions.

Alternatively, the speed with which a scientific field is allowed to advance rests on the foundations on which new developments can be build. Normally, code optimization results in an increase in code complexity, which in turn creates barriers for non-professional programmers to implement alterations necessary to test new scientific ideas. Strict separation of modules, yet flexible integration of concepts, creates a library-type of environment in which non-professional programmers – with the help of simple development tools – can use already existing modules as templates for their new ideas, thus creating a canvas on which only the minimal necessary changes need to be made.

1.2 Distribution Content

This distribution consist of a set of hierarchical directories, each of which represents a part of the code or documentation. The main directories are `./core`, `./emc`, `./examples`, `./scripts`, `./modules`, `./lib`, and `./texinfo`. A general make file `./Makefile` governs

compilation of separate pieces of this distribution. A ‘make’ in the root directory displays all compilation options. General routines are grouped under `./core`, which has the following structure



Each subdirectory represents a substructure of main structures. Any module or object files placed in any of the directories are automatically compiled into each executable, provided that they conform to a required format associated with the substructure. Definition of macros based on file headers facilitates automatic compilation. This particular feature aids programming by example, which means an already existing module can be used as a template, rather than having to completely write a new model. A copy script to facilitate this process is provided by `./scripts/cpmod.pl`.

Currently, the main program, which includes a scripting front-end, is located in `./emc`. This directory also contains example building and running scripts in `./examples`. Upon compilation through execution of ‘make emc’ generates a new directory `./emc/objects`, which contains all `.o` files. Examples in `./examples` make use of predefined crystal structures in `./lib`.

All documentation is organized in `./texinfo`. This PDF document is generated by a ‘make pdf’ in this directory.

2 Methodology

2.1 Lists

The presented code employs a combination of cell lists and verlet lists, where a separate object represents each list. Spatial discretization forms the cell list basis and is determined by either the largest or the smallest force field cut off, combined with a user supplied skin. The choice of cut off depends on invoking a conventional cell list, in which case the largest cut off is selected, or a multi-neighbor cell list, in which case the smallest cut off is chosen (*Intveld2008*). Both lists store their members in a linked list, which is accomplished by the application of a stencil or template. This template describes which cells neighbor the central cell. A distance criterion – based on the cut off and skin – then decides which sites are eventually added to a verlet list. Besides an integer vector to neighboring cells, a multi-neighbor template also contains the minimum distance between the cells containing the each site contributing to the pair interaction, which enables a primary selection of sites based on their cut off and therefore negating the initial need of a distance calculation. Inclusion in a verlet list only then requires a distance calculation when the neighboring particle falls within the stored minimum distance. Application of the multi-neighbor principle is optional, although initial tests do not indicate a significant drop in performance compared conventional cell list implementations.

Verlet lists describe which neighboring sites fall within a certain distance criterion of a site of interest. These lists are described by linked lists. Cross-linking between verlet lists has been added to allow for easy removal of a site when performing a Monte Carlo move. Practically, this means that each verlet list member of one site knows which verlet list member of another site it is connected to, thus creating members that contain pointers to child and parent in one particle’s list and a pointer to a member of another particle’s list. Additionally, the squared distance between the two sites has been added to negate the need for distance calculations during non-bonded force field evaluation.

Typically, the application of verlet lists in Molecular Dynamics makes use of the principle of a skin, which is a distance added to the force field cut off, and serves the purpose of reducing the number of verlet list rebuilds. However, an increase in distance calculations, as a result of inclusion of more sites in the verlet list, forms the downside of skin application. Hence, a balance needs to be found between performance gain as a result from less verlet list builds and performance loss as a result of an increase in distance calculations. Typical numbers used for the skin are 0.3σ in Lennard-Jones (LJ) units or 1 \AA in atomistic units. An added advantage of the usage of skins is that the decision to rebuild a verlet list can be coupled to the distance with which a site has moved from when the verlet list was built last for that site. A safe criterion is to rebuild the list when the site has moved by more than half the skin. Application of this criterion enables a dynamic update of verlet lists based on particle positions only and not by an external constraint such as a preset number of passed time steps or cycles, as is commonly used in packages such as LAMMPS (*Plimpton1995*).

2.2 Force Fields

Implemented force fields include standard representations of the OPLS force field as described by Jorgensen et al. (*Jorgensen*), the CHARMM force field as described by MacKerell

et al. (*MacKerell1998*), and a Class 2 force field representation used by PCFF or COMPASS force fields as described by Sun (*Sun1998*). A separate module describes coulombic interaction and allows for the use of either a cut off or Ewald summations to account for the long-range character (*Frenkel*). Coarse-grained force fields are provided by a colloidal force field as described by Everaers and Etjehadi (*Everaers2003*), which includes additional cross-terms for interaction between Lennard-Jones and colloidal particles (*Grest2007, Veld2008*), and the FENE bond potential as described by Kremer and Grest (*Kremer*). Pressures are calculation through standard virial expressions, which were derived when needed. All interactions can be compounded without significant effects on performance.

Generalization of virial calculations requires a solution to the gradient of a potential with respect to the cartesian coordinate system, which entails splitting partial derivatives into a product of linear parts and cartesian parts. For a nonbonded potential dependent on radius r this results in

$$\frac{\partial U(r)}{\partial \vec{x}} = \frac{\partial U(r)}{\partial r} \frac{\partial r}{\partial \vec{x}} = \frac{\partial U(r)}{\partial r} \frac{\vec{x}}{r},$$

where \vec{x} represents the constituent vector of a cartesian axis frame. Consequently, the virial tensor W_r resulting from this expression becomes

$$W_r(r_{ij}) = -\frac{\partial U(r)}{\partial r} r_{ij} \Delta \vec{r}_{ij} \Delta \vec{r}_{ij}.$$

where r_{ij} denotes the distance between site i and site j and $\Delta \vec{r}_{ij} \Delta \vec{r}_{ij}$ denotes a vector product resulting in a tensor. Bonded potentials potentials depend on bond length l_{ij} , angle θ_{ijk} , torsion ϕ_{ijkl} , and improper ψ_{ijkl} , which are defined as

$$\begin{aligned} \Delta \vec{r}_{ij} &= \frac{\vec{r}_j - \vec{r}_i}{|\vec{r}_j - \vec{r}_i|} \\ \vec{n}_{ijk} &= \frac{\Delta \vec{r}_{jk} \times \Delta \vec{r}_{ij}}{|\Delta \vec{r}_{jk} \times \Delta \vec{r}_{ij}|}, \\ r_{ij} &= l_{ij} = |\vec{r}_j - \vec{r}_i|, \\ \theta_{ijk} &= \Delta \vec{r}_{ij} \cdot \Delta \vec{r}_{jk}, \\ \phi_{ijkl} &= \vec{n}_{ijk} \cdot \vec{n}_{jkl}, \\ \psi_{ijkl} &= \vec{n}_{ijk} \cdot \vec{n}_{ijl}. \end{aligned}$$

Note that j represents the center site of impropers. The bond virial tensor W_l for potentials depending on l_{ij} is described by

$$W_l(l_{ij}) = -\frac{\partial U(l_{ij})}{\partial l_{ij}} l_{ij} \Delta \vec{r}_{ij} \Delta \vec{r}_{ij}.$$

The angular virial tensor W_θ for potentials depending on θ_{ijk} is described by

$$W_\theta(\theta_{ijk}) = -\frac{\partial U(\theta_{ijk})}{\partial \theta_{ijk}} \frac{1}{\sin(\theta_{ijk})} [\Delta \vec{r}_{ij} (\Delta \vec{r}_{jk} + \cos(\theta_{ijk}) \Delta \vec{r}_{ij}) + \Delta \vec{r}_{jk} (\Delta \vec{r}_{ij} + \cos(\theta_{ijk}) \Delta \vec{r}_{jk})].$$

Note that vector product $\Delta\vec{r}_{ij}\Delta\vec{r}_{jk}$ is noncommutative, i.e. $\Delta\vec{r}_{ij}\Delta\vec{r}_{jk} \neq \Delta\vec{r}_{jk}\Delta\vec{r}_{ij}$. The torsional virial tensor W_ϕ for potentials depending on ϕ_{ijkl} is described by

$$W_\phi(\phi_{ijkl}) = -\frac{\partial U(\phi_{ijkl})}{\partial \phi_{ijkl}} \frac{1}{\sin(\phi_{ijkl})} [$$

$$(\cos(\phi_{ijkl})\Delta\vec{r}_{jk} \times \vec{n}_{ijk} + \Delta\vec{r}_{jk} \times \vec{n}_{jkl}) \Delta\vec{r}_{ij}$$

$$+ (\cos(\phi_{ijkl}) (\Delta\vec{r}_{kl} \times \vec{n}_{jkl} - \Delta\vec{r}_{ij} \times \vec{n}_{ijk}) + \Delta\vec{r}_{kl} \times \vec{n}_{ijk} - \Delta\vec{r}_{ij} \times \vec{n}_{jkl}) \Delta\vec{r}_{jk}$$

$$- (\cos(\phi_{ijkl})\Delta\vec{r}_{jk} \times \vec{n}_{jkl} + \Delta\vec{r}_{jk} \times \vec{n}_{ijk}) \Delta\vec{r}_{kl}].$$

The improper virial tensor W_ψ for potentials depending on ψ_{ijkl} is described by

$$W_\psi(\psi_{ijkl}) = -\frac{\partial U(\psi_{ijkl})}{\partial \psi_{ijkl}} \frac{1}{\sin(\psi_{ijkl})} [$$

$$(\cos(\psi_{ijkl})\Delta\vec{r}_{jk} \times \vec{n}_{ijk} + \Delta\vec{r}_{jk} \times \vec{n}_{ijl}) \Delta\vec{r}_{ij}$$

$$+ \dots$$

$$+ (\cos(\psi_{ijkl})\Delta\vec{r}_{jk} \times \vec{n}_{ijl} + \Delta\vec{r}_{jk} \times \vec{n}_{ijk}) \Delta\vec{r}_{kl}].$$

3 Program Structure

This particular Monte Carlo simulation code is modular in approach. While it uses C as its programming language foundation, it incorporates distinct object oriented features. Communication by all modules to the outside, it being storage or inter-node communication, happens through a generalized parsing module, which can be used for both textual and binary output. Binary output can be either to memory or file. Variables are stored in a tree structure of pointers with cross-links between branches when so required. The textual input/output file bears an identical but simplified structure compared to the tree structure in memory: communication only conveys essential information and cross-links are implied. The structure 'simulation' presents the top structure, consisting of pointers to the structures 'io', 'identity', 'clocks', 'output', 'units', 'channels', 'types', 'systems', 'moves', 'statistics', 'forces', 'profiles', 'clusters', 'groups', and 'sites'. These structures are grouped in the following categories:

- Molecular representation by 'clusters', 'groups', and 'sites'
- Molecular interaction by 'types' and 'forces'
- Simulation parameter definition by 'systems'
- Configuration changes by 'moves'
- Measurements by 'statistics' and 'profiles'
- Communication by 'io', 'output', and 'units', and 'channels'
- Simulation timing by 'clocks'
- Simulator identification by 'identity'

3.1 Molecular Representation

A need for generality motivates the chosen terminology of 'clusters', 'groups', and 'sites'. Clusters can represent molecules and generally define closed networks of connected sites. Groups are of the order of repeat unit or protein residue and generally define subsets of molecules, creating the possibility of group operators when applying configuration changes. Sites can resemble atoms, but could also represent a more coarse-grained model of a group of atoms lumped into one site. Sites are the backbone of all interactions. This section describes the hierarchy from the bottom up, starting with 'sites', followed by 'groups' and 'clusters'.

3.1.1 Sites

Description of sites.

3.1.2 Groups

Description of groups.

3.1.3 Clusters

Description of clusters.

3.2 Molecular Interactions

3.2.1 Types

Description of types.

3.2.2 Forces

Description of forces.

3.3 Systems

Description of systems.

3.4 Configurational Moves

Description of moves.

3.5 Measurements

Description of distributions.

4 Simulation Setup

4.1 General

EMC provides a setup script `emc_setup.pl`, which allows for creation of EMC build and LAMMPS input scripts. The EMC setup script serves as a wrapper around EMC itself, creating a context for abstraction of EMC scripting commands. In its simplest appearance, setup scripts only contain a few lines describing simulated chemistry and their quantities, where subsequent options are defined through the command line. More complex scripts contain several paragraphs for describing simulation conditions and chemistry. This mode will be referred to as the chemistry mode (see Section 4.6.2 [Chemistry File], page 35). The most complex mode creates an environment in which several simulations are defined in one script, allowing for looping over multiple simulation conditions. This mode will be referred to as the environment mode (see Section 4.6.1 [Environment File], page 32). EMC environment setup scripts function as a wrapper around chemistry scripts. An environment setup script can hold multiple chemistry scripts. Paragraphs use keyword `ITEM` as demarcation, where identifiers follow this keyword at the beginning of each paragraphs. Their ending is marked by `ITEM END`.

Normal parametrization uses force fields as provided by EMC (see Section 5.10 [Field], page 101) for typing. These force fields can be found in `${EMC_ROOT}/field/`. Optional `references.csv` and `parameters.csv` files can be provided for DPD force fields. The latter is called from the chemistry file. Alternatively, a more general way is provided by using the field paragraph (`ITEM FIELD`) in an environment script. Both produce a force field file with extension `.prm`. Force fields are governed over `types` definitions (see Section 7.12 [Types], page 190). The EMC setup script can create multiphase initial configuration, but does not address the grafting capabilities of the EMC build capabilities. The EMC setup script displays all available options when called without arguments. It is, in general, called by

```
emc_setup.pl [-option[=#]] project [phase 1 clusters [+ ...]]
```

It can also, however, be used as an interpreter, where EMC setup scripts can function as executables when using EMC in a UNIX-type environment. An interpreter allows for including all settings and options into an EMC script, thus folding the description of one simulation or a set of simulations into one comprehensive script.

4.2 Setup Usage

Examples can be found in `${EMC_ROOT}/examples/setup/`. Creating EMC and LAMMPS input with these examples occurs in steps. The first step is to convert the example setup script by means of `emc_setup.pl` through

```
emc_setup.pl example.esh
```

which either creates an input script for EMC, called `build.emc` by default, and an input script for LAMMPS, called `example.in` by default, or sets up an execution environment by creating subdirectories `build`, `run`, `analysis`, and `chemistry`. The first three directories contain `bash` scripts with the purpose of respectively building input structures, running molecular dynamics simulations, and analysis of end results. The `chemistry` directory can contain force fields, variation in chemistry, polymer make up, etc. Default names can be altered when desired (see [\[Environment Options\]](#), page [\(undefined\)](#)).

The second step involves a call of EMC itself, by for instance for LINUX,

```
emc_linux build.emc 2>&1 | tee build.out
```

The latter part of the command allows for writing output to screen and `build.out` simultaneously. Five extra files appear upon completion of this command, i.e. `example.pdb`, `example.psf`, and `example.vmd` for visualization with VMD, and `example.data` and `example.params` for use with LAMMPS. Visualization supports checking the validity of the final structure. The call to VMD, when in the user's path, is facilitated by,

```
vmd -e example.vmd
```

The `example.vmd` file represents a small Tcl script, which contains information about particle diameters (as defined by the chosen force field) and the type of VMD presentation to use for visualization. A molecular dynamics simulation with LAMMPS occurs with for example

```
lmp_linux <example.in 2>&1 | tee example.out
```

The `example.in` LAMMPS script contains all the necessary information for executing the run. Please note, that the presented examples in the `${EMC_ROOT}/examples/setup/` directory are primarily meant as building examples. Strategies for managing sets of simulations are provided by the environment mode of `emc_setup.pl` (see Section 4.6.1 [\[Environment File\]](#), page 32), for which an example can be found in `${EMC_ROOT}/examples/setup/shear/`.

4.3 Extensions

A comprehensive mode for EMC Setup is when using environment scripts, which creates a modeling environment consisting of build, run, and analysis scripts, complemented with a hierarchical structure for dealing with produced simulation data. The simpler EMC chemistry script creates a subset of the more comprehensive environment script, but is useful for prototyping simulations. Extensions and explanations for scripts and data files produced by both modes following from EMC Setup, EMC, LAMMPS or postprocessing analysis can be found in the following table,

Extension	Description
<code>.data</code>	EMC output containing LAMMPS data file defining coordinates, types, and topology
<code>.csv</code>	Postprocessed data derived from EMC or LAMMPS output resulting from execution of EMC analysis scripts; normally used for further processing by user generated scripts (using e.g. Python, Perl, Matlab, or Mathematica)
<code>.density</code>	Density profile produced by LAMMPS
<code>.density3d</code>	3D density profile produced by LAMMPS
<code>.emc</code>	EMC script as interpreted by EMC itself when produced by <code>emc_setup</code> ; used to create input structures by means of energetic considerations; EMC produces a comprehensive EMC format with the same extension as output containing all topological and morphological information
<code>.energy</code>	Time-averaged energetic information produced by LAMMPS
<code>.esh</code>	EMC Setup script as interpreted by wrapper <code>emc_setup.pl</code> ; can be either a chemistry or an environment script
<code>.in</code>	LAMMPS input script according to specifications as made in a EMC chemistry file
<code>.params</code>	EMC output following from a successful build; input needed by LAMMPS, containing all parameters defining a full simulation
<code>.pdb</code>	PDB format containing initial coordinates
<code>.pressure</code>	Time-averaged pressure information produced by LAMMPS
<code>.prm</code>	EMC parameter file as produced by either EMC chemistry or environment script; used by EMC to set force field energetics

<code>.psf</code>	PSF format containing types and (initial) topology
<code>.vmd</code>	VMD Tcl script used to read in both <code>.pdb</code> and <code>.psf</code> , additionally defining bead sizes following from the used force field
<code>.volume</code>	Time-averaged volume information produced by LAMMPS

4.4 Enviroment Options

Option	Default	Description
<code>analyze_archive</code>	<code>true</code>	Toggle archiving of file names associated with analyzed data; file name lists can be used directly or at a later date for transferring data between computational clusters and local machines.
<code>analyze_data</code>	<code>true</code>	Toggle creation of tar archive from exchange file list; this data is used in combination with <code>./scripts/run_host.sh -exchange</code> to transfer data from computational clusters to local machines; archives should not be created when analysis scripts subsequently submit analysis to queue.
<code>analyze_last</code>	<code>false</code>	Toggle inclusion of the last trajectory frame during analysis (deprecated); analysis is transfered to the analysis paragraph (see Section 4.6.1 [Environment File], page 32).
<code>analyze_replace</code>	<code>true</code>	Toggle replacement of already exisiting analysis results.
<code>analyze_skip</code>	<code>0</code>	Set the number of initial frames to skip during analysis.
<code>analyze_user</code>	<code>-</code>	Set directory for user analysis scripts, which can be accessed through an analysis paragraph (see Section 4.6.1 [Environment File], page 32).
<code>analyze_window</code>	<code>1</code>	Set the number of frames over which to apply a window averaging during analysis.
<code>name_analyze</code>		Set the name of job analysis scripts; analysis scripts are stored in <code>./analyze/</code> ; takes the base name of the chemistry script when omitted; ignored when <code>"-"</code> is chosen as name.
<code>name_build</code>		Set the name of job build scripts; build scripts are stored in <code>./build/</code> ; takes the base name of the chemistry script when omitted; ignored when <code>"-"</code> is chosen as name.

<code>name_run</code>		Set the name of job run scripts; run scripts are stored in <code>./run/</code> ; takes the base name of the chemistry script when omitted; ignored when <code>"-"</code> is chosen as name.
<code>name_scripts</code>	-	Simultaneously set analysis, build and run script names.
<code>name_testdir</code>	-	Set the name of a test directory in which to test the chemistry paragraph in an environment script; a setup script is created in <code>./test/</code> ; ignored as default.
<code>nchains</code>	-	Set the number of chains to be used for executing LAMMPS jobs; allows for having the next job wait on the previous (currently only with LSF).
<code>ncores</code>	-	Set the number of cores for execution of LAMMPS jobs.
<code>ncorespernode</code>	<code>default</code>	Set number of cores per node for execution of packed jobs; equivalent to <code>queue_ppn</code> .
<code>project</code>	-	Set the project name; derived from setup script name when not given.
<code>queue</code>	<code>account=none, analyze=default, build=default, memory=default, ncores=-, ppn=default, run=default, user=none</code>	Directly set queueing system settings, where option <code>account</code> sets billing account, <code>analyze</code> sets the analysis queue, <code>build</code> sets the build queue, <code>memory</code> sets the amount of memory needed, <code>ncores</code> sets the number of cores for running, <code>ppn</code> sets the number of processors per node, <code>run</code> sets the run queue, and <code>user</code> sets user options; the latter is dependent on the installed queueing system.
<code>queue_account</code>	<code>none</code>	Set queue account for billing.
<code>queue_analyze</code>	<code>default</code>	Set the queue used in analysis scripts; option <code>default</code> indicates the use of queue, which is set as system default; option <code>local</code> will sequentially execute all jobs on local machine.
<code>queue_build</code>	<code>default</code>	Set the queue used in build scripts; option <code>default</code> indicates the use of queue, which is set as system default; option <code>local</code> will sequentially execute all jobs on local machine.

<code>queue_memory</code>	<code>default</code>	Set memory per core in gb for executed jobs.
<code>queue_ppn</code>	<code>default</code>	Set cores per node for executed jobs.
<code>queue_run</code>	<code>default</code>	Set the queue used in run scripts; option <code>default</code> indicates the use of queue, which is set as system default; option <code>local</code> will sequentially execute all jobs on local machine.
<code>queue_user</code>	<code>none</code>	Options to be passed directly to queuing system; allowed option are dependent on the installed queueing system.
<code>replace</code>	<code>false</code>	Replace or overwrite all written script files.
<code>time_analyze</code>	<code>00:30:00</code>	Set the desired wall time for analysis scripts.
<code>time_build</code>	<code>00:10:00</code>	Set the desired wall time for build scripts.
<code>time_run</code>	<code>24:00:00</code>	Set the desired wall time for run scripts.
<code>workdir</code>	<code>-</code>	Set the work directory, which is used as a base for creating the EMC work environment.

4.5 Chemistry Options

Option	Default	Description
angle	5,180	This option can be used to set angle constants in three ways: 1. set DPD angle constants (k , theta0), 2. provide additional angle table entries through type1 , type2 , type3 , k , theta0 , 3. set force field handling for angle typing; valid options are ignore , complete , warn , empty , and error .
auto	false	Include wildcard mass entry in generated DPD .prm force field file; options are true or false .
binsize	0.01	Set the general bin size to be used in LAMMPS profiles; note, that profiles are recorded in reduced units, i.e. coordinates run from 0 to 1.
bond	25,1	Set DPD bond constants (k , 10); can also be used to provide additional bond table entries through -bond=type1,type2,k,10 .
build	build	Set build script name.
build_center	false	Place first site at the origin (see also option build_origin); options are true or false .
build_dir	../build	Set location of build directory for LAMMPS script.
build_order	false	Build clusters in the order as defined by option phases ; options are random or sequence .
build_origin	x=0, y=0, z=0	Set alternate origin at which to place the first site.
build_replace	false	Toggle replacement of already existing build scripts and results, which will skip build execution when a LAMMPS data file exists.
build_theta	false	Set the minimal insertion angle
charge	true	Indicator for occurrence of charges in chemistry.csv .
charge_cut	9.5	Set pairwise charge interaction cut off.

<code>chemistry</code>	<code>chemistry</code>	Set chemistry input file name; extension <code>.csv</code> is implied.
<code>chunk</code>	<code>true</code>	Use chunk approach for computing profiles and samples in LAMMPS script; options are <code>true</code> or <code>false</code> .
<code>communicate</code>	<code>false</code>	Use <code>communicate</code> keyword instead of <code>comm_modify</code> in LAMMPS input script; used with DPD simulations for backwards compatibility with older versions of LAMMPS; options are <code>true</code> or <code>false</code> .
<code>core</code>	<code>-1</code>	Set particle core diameter; used with Born potentials.
<code>cross</code>	<code>false</code>	Include nonbond cross terms in LAMMPS params file; default depends on chosen force field; options are <code>true</code> or <code>false</code> .
<code>crystal</code>	<code>false</code>	Let EMC derive a crystal structure based on the offered import; options are <code>true</code> or <code>false</code> .
<code>cut</code>	<code>9.5</code>	Set pairwise interaction cutoff; options are either definite positive numbers or <code>repulsive</code> ; the latter transfers the the parameters for a purely repulsive cut and shift potential to LAMMPS
<code>cutoff</code>	<code>center=-1, charge=9.5, ghost=-1, inner=-1, outer=-1, pair=9.5, repulsive=0</code>	Set various cutoffs; options are either definite positive numbers; negative numbers refer to internal defaults; the different cutoffs can also be set with single options <code>charge_cut</code> , <code>ghost_cut</code> , <code>inner_cut</code> , <code>outer_cut</code> , and <code>cut</code> .
<code>debug</code>	<code>-</code>	Turn debugging information on; options are <code>true</code> or <code>false</code> .
<code>deform</code>	<code>nblocks=1, ncycles=100, type=relative, xx=1, yy=1, zz=1, zy=0, zx=0, yx=0</code>	Invoke affine deformation using Monte Carlo relaxation when <code>ncycles</code> is larger than zero; deformation <code>type</code> can either be <code>absolute</code> or <code>relative</code> .

density	1[,...]	Set simulation density; density of separate phases can be entered through separating values for each phase by commas; units and default depend on chosen force field type .
depth	auto	Set the depth with which rings - as defined in groups - are recursively assigned; options are either auto or values larger than 2.
dielectric	0.2 or 1	Set dielectric constant of medium; default depends on chosen force field.
direction	x	Set direction in which phases in setup scripts with multiple phases are build; valid options are x , y , or z .
dt dump	100000	Frequency with which LAMMPS adds to trajectory files.
dt restart	100000	Frequency with which LAMMPS writes restart files.
dt thermo	1000	Frequency with which LAMMPS generates thermodynamical output.
emc	true	Create EMC build script (.emc file); options are true or false .
emc_exclude	build=false	Exclude specified sections from the resulting EMC script; excluding build omits morphology generation, which can be desired when exporting SMILES.
emc_execute	-	Execute the EMC build script as created by the setup script; options false or - do not execute; option true allows EMC setup decide which EMC version to use, based on operating system, host option, and path; any other option value is interpreted as the location and name of the EMC version to use.
emc_export	smiles=false	Add export of specified format; smiles controls export of SMILES for all built clusters in both group-based and site-based representations; option false deselects export; other valid options are csv , json , and math for comma separated values, JSON, and Mathematica formats respectively.

<code>emc_moves</code>	<code>displace=1</code>	Control frequency of selected Monte Carlo move for positive values; zero or negative values unselect.
<code>emc_output</code>	<code>debug=false,</code> <code>exit=true,</code> <code>info=true,</code> <code>warning=true</code>	Control output as generated by EMC scripts; options for each separate keyword are <code>true</code> or <code>false</code> .
<code>emc_progress</code>	<code>build=true,</code> <code>clusters=false</code>	Control progress indicators resulting from executing EMC script; keyword <code>build</code> controls progress output during build; keyword <code>cluster</code> controls progress output during sequencing and cluster construction; options for each separate keyword are <code>true</code> or <code>false</code> .
<code>emc_run</code>	<code>nblocks=100,</code> <code>ncycles=0,</code> <code>nequil=0,</code> <code>clusters=all,</code> <code>groups=all,</code> <code>sites=all</code>	Execute a Monte Carlo simulation after building for <code>ncycles</code> ; optionally, an equilibration phase is selected when <code>nequil</code> is larger than zero; <code>nblocks</code> selects the frequency with which output is generated; specific selections can be set when defining either <code>clusters</code> , <code>groups</code> , or <code>sites</code> separately or in combination; <code>all</code> indicates that all members of a selection are considered.
<code>emc_test</code>	<code>false</code>	Test the validity of an EMC environment script; checks up to execution of EMC build files, as generated by EMC environment; options are <code>true</code> or <code>false</code> .
<code>emc_traject</code>	<code>append=true,</code> <code>frequency=0</code>	Add generation of trajectory file during Monte Carlo run when <code>frequency</code> is larger than zero.
<code>environment</code>	<code>false</code>	Interpret EMC shell script (<code>.esh</code>) as an environment script; options are <code>true</code> and <code>false</code> .
<code>ewald</code>	<code>true</code>	Switch the electrostatics long-range treatment through Ewald summations on or off; default depends on the chosen force field; options are <code>true</code> or <code>false</code> .
<code>expert</code>	<code>false</code>	Invokes expert mode, which allows for overrides of certain consistency checks (e.g. <code>nrepeat</code> for polymers); options are <code>true</code> or <code>false</code> .
<code>extension</code>	<code>.esh</code>	Set the extension of environment scripts; extensions are derived from given environment scripts, when not defined.

<code>extra</code>	-	Create copies of existing nonbond interactions; format is <code>destination:source:offset</code> , where <code>offset</code> defines the DPD offset of the new interaction constant compared to the original (deprecated, replaced by <code>replica</code>)
<code>field</code>	<code>opls</code>	Set force field type and name based on root location; more general access to EMC provided force fields uses the <code>type</code> option below.
<code>field_angle</code>	-	Set error handling for typing of angles; valid options are <code>complete</code> , <code>empty</code> , <code>error</code> , <code>ignore</code> , and <code>warn</code> .
<code>field_bond</code>	-	Set error handling for typing of bonds; valid options are <code>complete</code> , <code>empty</code> , <code>error</code> , <code>ignore</code> , and <code>warn</code> .
<code>field_dpd</code>	<code>auto=false</code> , <code>bond=false</code>	Set options influencing the generation of a DPD force field; possible options are <code>auto</code> and <code>bond</code> with values of either <code>true</code> or <code>false</code> ; <code>auto</code> controls the addition of wildcard additions to mass and nonbonded force field paragraphs; <code>bond</code> allows for verbatim transcription of pair to bond interactions.
<code>field_debug</code>	<code>false</code>	Set debug option, showing individual typing steps; valid options are <code>full</code> , <code>reduced</code> , and <code>false</code> ; <code>full</code> outputs verbose debugging information, <code>reduced</code> generates an abridged selection, and <code>false</code> turns debugging information off.
<code>field_error</code>	<code>true</code>	Exit upon error (set to <code>false</code> when debugging a force field).
<code>field_format</code>	<code>%15.10e</code>	Sets the format of parameters in the produced field; should be format as is standard for floating point values when used in combination with <code>printf()</code> .
<code>field_group</code>	-	Set field group option for applying force fields to groups as defined by their <code>TEMPLATE</code> section.
<code>field_id</code>	-	Set force field id.
<code>field_improper</code>	-	Set error handling for typing of impropers; valid options are <code>complete</code> , <code>empty</code> , <code>error</code> , <code>ignore</code> , and <code>warn</code> .

<code>field_increment</code>	-	Set error handling for typing of bond increments; valid options are <code>complete</code> , <code>empty</code> , <code>error</code> , <code>ignore</code> , and <code>warn</code> .
<code>field_location</code>	-	Provide force field location; intended for expert use; use <code>field_type</code> for EMC provided force fields.
<code>field_name</code>	-	Provide force field name; intended for expert use= use <code>field_type</code> for EMC provided force fields.
<code>field_nbonded</code>	0	Define the number of sites, that are bonded, but should be excluded from nonbond interactions, e.g. a value of 1 excludes 1-2 interactions, a value of 2 excludes 1-2 and 1-3 interactions, etc.; allowed values are larger or equal than 0; this option only applies to DPD force fields.
<code>field_torsion</code>	-	Set error handling for typing of torsions; valid options are <code>complete</code> , <code>empty</code> , <code>error</code> , <code>ignore</code> , and <code>warn</code> .
<code>field_type</code>	<code>opls</code>	Provide force field type; invoking this option will automatically set <code>field</code> , <code>field_location</code> , and <code>field_name</code> options.
<code>field_write</code>	<code>true</code>	Interpret and write the force field as defined in the ITEM FIELD paragraph (see Section 4.6.3 [Field File], page 45)
<code>focus</code>	-	List of clusters to focus on.
<code>grace</code>	<code>0.9999, 0.9999, 0</code>	Deprecated command, use <code>weight</code> instead; user provided grace for building procedure; given in order of nonbonded, bonded, and focussed interactions.
<code>ghost_cut</code>	-	Set molecular dynamics ghost region size for property communication.
<code>help</code>	-	Display script help.
<code>hexadecimal</code>	<code>false</code>	Set hexadecimal output in PDB files.
<code>host</code>	-	Set host on which to run EMC and LAMMPS.

<code>improper</code>	-	Set force field handling for improper typing; valid options are <code>ignore</code> , <code>complete</code> , <code>warn</code> , <code>empty</code> , and <code>error</code> .
<code>increment</code>	-	Set force field handling for increment typing; valid options are <code>ignore</code> , <code>complete</code> , <code>warn</code> , <code>empty</code> , and <code>error</code> .
<code>info</code>	-	Turn on information during setup script execution.
<code>inner</code>	-	Set inner cutoff; use and default depend of chosen force field; note that <code>inner</code> and <code>outer</code> cutoffs are interpreted as fractions for colloidal force fields.
<code>insight</code>	<code>true</code>	Create InsightII CAR and MDF output files.
<code>insight_ compress</code>	<code>true</code>	Control compression of produced CAR and MDF files; options are <code>true</code> or <code>false</code> .
<code>insight_pbc</code>	<code>true</code>	Toggle application of periodic boundary conditions; options are <code>true</code> or <code>false</code> .
<code>insight_unwrap</code>	<code>clusters</code>	Set CAR unwrapping mode; options are <code>none</code> , <code>clusters</code> , or <code>sites</code> .
<code>kappa</code>	1 or 4	Set long range electrostatics Ewald summation kappa; default depends on chosen force field.
<code>lammmps</code>	<code>true</code>	Create LAMMPS input script (.in file) or set lammmps version using year, e.g. <code>-lammmps=2014</code> (new versions start at 2015) or keywords <code>old</code> or <code>new</code> ; alters the settings for <code>-communicate</code> and <code>-chunk</code> ; default is <code>new</code> .
<code>lammmps_cutoff</code>	<code>false</code>	Toggle output of pairwise cut off in parameter file; valid options are <code>true</code> or <code>false</code> .
<code>lammmps_dlimit</code>	0.1	Set LAMMPS nve/limit distance used during equilibration.
<code>lammmps_pdamp</code>	1000	Set LAMMPS barostat damping constant for NPT simulations.

<code>lammops_tdamp</code>	<code>100</code>	Set LAMMPS thermostat damping constant for both NVT and NPT simulations.
<code>location</code>	<code>analyze=., field=., include=.</code>	Prepend path for locations of analysis scripts, force field files, and include files; always included are for 1) <code>analyze:</code> directories <code>.</code> and <code>\${EMC_ROOT}/scripts/analyze</code> , 2) <code>field:</code> directories <code>.</code> and <code>\${EMC_ROOT}/field</code> , and 3) <code>include:</code> directories <code>.</code> and <code>\${WORKDIR}/chemistry/include</code> when existing; multiple path entries are separated with a colon.
<code>mass</code>	<code>false</code>	Assume mass fractions in <code>chemistry.csv</code> input file.
<code>memorypercore</code>	<code>default</code>	Set memory per core in gigabyte for executed jobs; equivalent to <code>queue_memory</code> .
<code>modules</code>	<code>-</code>	Swap out modules; format is defined by ' <code>[old=new]</code> ' separated by commas.
<code>mol</code>	<code>true</code>	Assume mol fractions in <code>chemistry.csv</code> input file.
<code>momentum</code>	<code>100,1,1,1, angular</code>	Control zeroing of linear and angular momentum during LAMMPS execution; order of entries (separated by commas only!), i.e. <code>N,xflag,yflag,zflag,flag</code> (see LAMMPS) manual; <code>flag</code> is omitted in the LAMMPS input script when the keyword <code>none</code> is used.
<code>moves_cluster</code>	<code>active=false, cut=0.05, frequency=1, limit=auto:auto, max=0:0, min=auto:auto</code>	Define cluster move settings used to optimize builds; multiple options are available, where option <code>active</code> switches the move on or off, <code>cut</code> sets a percentage of the maximum allowed displacement and rotation at which to turn the move off, <code>frequency</code> sets the frequency of the move, <code>limit</code> sets the maximum allowed displacement (in force field units) and rotation (in rad), <code>max</code> sets estimation for 50% acceptance of displacement and rotation (0:0 will trigger internal estimation), and <code>min</code> sets explicit rather than relative conditions to turn the move off.
<code>msd</code>	<code>false</code>	Add mean square displacement analysis to LAMMPS output; options are <code>true</code> or <code>false</code> .
<code>namd</code>	<code>false</code>	Create NAMD input script and parameter file.

<code>namd_dtcoulomb</code>	<code>1</code>	Set electrostatic interaction update frequency.
<code>namd_dtdcd</code>	<code>10000</code>	Set frequency with which snapshots are written to a DCD file.
<code>namd_dtnonbond</code>	<code>1</code>	Set nonbond interaction update frequency.
<code>namd_dtrestart</code>	<code>100000</code>	Set output frequency of restart files.
<code>namd_dtthermo</code>	<code>1000</code>	Set output frequency of thermodynamic quantities.
<code>namd_dttiming</code>	<code>10000</code>	Set timing frequency.
<code>namd_dtupdate</code>	<code>20</code>	Set update frequency.
<code>namd_pres_decay</code>	<code>50</code>	Set pressure ensemble decay.
<code>namd_pres_period</code>	<code>100</code>	Set pressure ensemble period.
<code>namd_temp_damp</code>	<code>3</code>	Set temperature ensemble damping.
<code>namd_tminimize</code>	<code>50000</code>	Set number of initial minimization steps.
<code>namd_trun</code>	<code>10000000</code>	Set total number of timesteps for execution run.
<code>niterations</code>	<code>1000</code>	Controls the number of iterations used for inserting an atom during the build process.
<code>norestart</code>	<code>false</code>	Control the capability for using run scripts – as resulting from environment scripts – for restarting; options are <code>true</code> or <code>false</code> ; the latter means, that restarting with the same run script is possible.
<code>nparallel</code>	<code>auto</code>	Set the number of repeat units in the direction parallel to a surface defined in the chemistry file; values larger than 1 override the <code>auto</code> setting.
<code>nrelax</code>	<code>100</code>	Alter the number of relaxation cycles used during the EMC building process.
<code>nsample</code>	<code>1000</code>	Number of configurations used for averaging during execution of LAMMPS analysis routines.
<code>ntotal</code>	<code>10000</code>	Total number of created sites or atoms.

<code>number</code>	<code>false</code>	Interpret column 3 of <code>SHORTHAND</code> or <code>CLUSTERS</code> as the number of desired clusters; this number is an integer (see Section 4.6.2 [Chemistry File], page 35).
<code>omit</code>	<code>false</code>	Omit fractions from chemistry file; options are <code>true</code> or <code>false</code> .
<code>outer</code>	<code>-</code>	Set outer cutoff; note that <code>inner</code> and <code>outer</code> cutoffs are interpreted as fractions for colloidal force fields.
<code>pair</code>	<code>a=25, gamma=4.5, r=1</code>	Set DPD pair constants; only definite positive values are allowed.
<code>parameters</code>	<code>parameters</code>	Alternative parameter file name; the extension <code>.csv</code> is implied.
<code>params</code>	<code>true</code>	Create force field parameter file (<code>.prm</code>); currently only for DPD; options are <code>true</code> or <code>false</code> .
<code>pdb</code>	<code>true</code>	Create PDB and PSF output files.
<code>pdb_atom</code>	<code>index</code>	Set atom name behavior in PDB and PSF output; options are <code>detect</code> , <code>index</code> , and <code>series</code> .
<code>pdb_compress</code>	<code>true</code>	Control compression of produced PDB and PSF files; options are <code>true</code> or <code>false</code> .
<code>pdb_connect</code>	<code>false</code>	Add connectivity to PDB; options are <code>true</code> or <code>false</code> .
<code>pdb_cut</code>	<code>false</code>	Cut bonds in PSF output, which span the simulation box; options are <code>true</code> or <code>false</code> .
<code>pdb_extend</code>	<code>false</code>	Use extended format in PSF output; options are <code>true</code> or <code>false</code> .
<code>pdb_parameters</code>	<code>false</code>	Toggle output of NAMD parameter file; options are <code>true</code> or <code>false</code> .
<code>pdb_pbc</code>	<code>true</code>	Toggle application of periodic boundary conditions; options are <code>true</code> or <code>false</code> .
<code>pdb_fixed</code>	<code>true</code>	exclude flagged fixed sites when unwrapping clusters; options are <code>true</code> or <code>false</code> .

<code>pdb_rigid</code>	<code>true</code>	exclude flagged rigid sites when unwrapping clusters; options are <code>true</code> or <code>false</code> .
<code>pdb_residue</code>	<code>index</code>	Set residue name behavior in PDB and PSF output; options are <code>detect</code> , <code>index</code> , and <code>series</code> .
<code>pdb_segment</code>	<code>index</code>	Set segment name behavior in PDB and PSF output; options are <code>detect</code> , <code>index</code> , and <code>series</code> .
<code>pdb_unwrap</code>	<code>clusters</code>	Set PDB unwrapping mode; options are <code>none</code> , <code>clusters</code> , or <code>sites</code> .
<code>pdb_vdw</code>	<code>true</code>	Add Van der Waals representation to the VDW script created by EMC; options are <code>true</code> and <code>false</code> .
<code>percolate</code>	<code>false</code>	Treat imported structures as percolating InsightII structures; uses boundary crossing definitions as provided in the <code>.mdf</code> file; behavior of <code>crystal</code> option might be preferred; options are <code>true</code> or <code>false</code> .
<code>phases</code>	<code>all</code>	Sets which clusters to assign to each phase; each phase is separated by a <code>+</code> -sign; default assigns all clusters to phase 1; build order is defined by option <code>build_order</code>
<code>polymer</code>	<code>bias=none, fraction=number, niterations=-1, order=list</code>	Set group polymer distribution global defaults; <code>bias</code> helps in limiting construction time by controlling exclusion of unsuccessful subpolymers during polymer construction; valid <code>bias</code> options are <code>none</code> for no bias, <code>binary</code> for exclusion of subpolymers on a binary basis, and <code>accumulative</code> for exclusion on an accumulative basis; valid <code>fraction</code> options are <code>number</code> for using number of molecules and <code>mass</code> molecule mass as distribution entries; valid <code>order</code> options are <code>list</code> for sequentially and <code>random</code> of randomly interpreting the provided polymer distribution; the <code>niterations</code> keyword sets the maximum allowed number of iterations used during polymer sequence determination
<code>precision</code>	<code>0.001</code>	Provide LAMMPS <code>kspace</code> long range Ewald summation precision.
<code>prefix</code>	<code>false</code>	Set project name as prefix to LAMMPS output files.

pressure	false, direction= x+y+z, couple	Set the system pressure and invoke an NPT ensemble; optionally specify barostat direction and/or directional coupling; valid directions are x , y , and z ; valid coupling options are couple , uncouple , true , and false , where true corresponds to coupled directions; definition of the coupling direction follows the couple keyword through separation by a colon; a + sign separates multiple directions, e.g. atomistic 1,couple=y+z sets a pressure of 1 atm in all directions with coupling in y and z directions; omission of coupling directions assumes an isotropic barostat.
profile	density=false, density3d=false, pressure=false	Generate density and/or pressure profiles of all clusters while executing LAMMPS; options are density , density3d , or pressure , followed by either true or false , separated by a colon; true is assumed when omitted; options are separated by a comma.
project	-	Set the project name; derived from setup script name when not given.
quiet	-	Suppress all output generated by this setup script.
radius	5	Alter the radius of the spherical volume in which already built sites are relaxed; used during the EMC building process.
record	cut=false, frequency=1, inactive=true, name="", pbcb=true, unwrap=true	Defines the record entry in the build paragraph, which records the build process by outputting a set of PDBs and PSFs; valid options are cut for cutting bonds in the last recorded frame, frequency for setting the frequency with which the relaxation process is written in each separate PDB, inactive for whether to add inactive entries to PDBs (denoted by positions (0,0,0)), name for the base name of the written PDBs and PSFs, pbcb for mapping clusters back into the periodic box by their center of mass, and unwrap to toggle unwrapping of clusters.
references	references	Provide an alternative name for the references file; the extension .csv is implied.
region_epsilon	0.1	Set epsilon for excluded regions when importing structures; the given value is expected to be positive; other are ignored.

<code>region_sigma</code>	<code>1</code>	Set sigma for excluded regions when importing structures; the given value is expected to be positive; other are ignored.
<code>replace</code>	<code>false</code>	Replace or overwrite all written script files.
<code>replica</code>	<code>-</code>	Create copies of existing nonbond interactions; format is <code>destination:source:offset</code> , where offset defines the DPD offset of the new interaction constant compared to the original.
<code>restart</code>	<code>false,..</code>	Create LAMMPS restart scripts in chemistry file mode; this option allows setting an alternative data directory using in job run scripts, from which the most recent restart file will be used as a starting point; job run scripts will always try to restart – also when this option is set to <code>false</code> – from the specified data directory, creating a new serial directory by adding 1 to the current highest serial; serial numbers start at 00 and currently have a maximum of 99.
<code>rlength</code>	<code>-</code>	Provide a reference length; used when selecting non-dimensional force fields.
<code>rmass</code>	<code>-</code>	Provide a reference mass; used when selecting non-dimensional force fields.
<code>rtype</code>	<code>-</code>	Provide a reference type; used when selecting non-dimensional force fields.
<code>sample</code>	<code>energy=false, msd=false, pressure=true, volume=false, green-kubo=false</code>	Set sampling sections in LAMMPS input script for averaging of energy for itemized energetic contributions, msd for mean square displacements, pressure for the pressure tensor, volume for the volume tensor, or green-kubo for Green-Kubo output concerning correlations between the off-diagonal pressure tensor contributions as used in viscosity calculations; output files will have their respective identifier as extension; options are either <code>true</code> or <code>false</code> , expect for keyword msd which has average as additional option for time averaged mean square displacement.

seed	-1	Provide initial random seed; a -1 seed will invoke the use of the number of seconds since January 1, 1970 as an initial seed.
shake	-	Either switch off use of shake for select force fields or set masses, types, bonds, and/or angles for which to apply the SHAKE algorithm to; allowed keywords are active , mass , type , bond , angle ; contributing types are separated by a colon ':', e.g. to freeze the angle of TraPPE water in LAMMPS use bond=hw:ow , angle=hw:ow:hw ; see 'fix shake' in LAMMPS manual for shake interpretation.
shake_ iterations	20	Set the maximum number of iterations used during SHAKE.
shake_output	never	Set the output frequency with which SHAKE statistics are written to the LAMMPS log file; options are integer numbers larger than 0 and never , which is equivalent to numbers smaller than 1.
shake_tolerance	0.0001	Set the SHAKE tolerance, which defines a successful SHAKE.
shape	1	Provide the desired shape of the simulation box; the shape factor refers to the relative fraction of lx/ly .
shear	false	Add shear paragraph to LAMMPS input script; options are true or false .

<code>split</code>	<code>phase=1, fraction=0.5, thickness=1, type=relative, mode=random, sites=all, groups=all, clusters=all</code>	Defines the selection criteria for applying a fractional split of clusters at the edges of the box; <code>phase</code> sets the phase to which to apply the split; <code>fraction</code> specifies the fractional split; <code>thickness</code> sets the thickness of the region taken into consideration; <code>type</code> sets the units of the thickness (valid options are <code>absolute</code> and <code>relative</code>); <code>mode</code> specifies the selections algorithm (valid options are <code>distance</code> and <code>random</code>); <code>sites</code> sets the sites to include in the selection (valid options are <code>all</code> for all available sites or a selected list of site types separated by colons); <code>groups</code> sets the groups to include in the selection (valid options are <code>all</code> for all available sites or a selected list of group ids separated by colons); <code>sites</code> sets the sites to include in the selection (valid options are <code>all</code> for all available clusters build in the set <code>phase</code> or a selected list of cluster ids separated by colons).
<code>suffix</code>	<code>_\$HOSTNAME</code>	Set EMC and LAMMPS suffix.
<code>system</code>	<code>charge=true, geometry=true, map=true, pbc=true, id=main</code>	Perform various system checks or set system id; <code>charge</code> checks if the total system charge equals to zero; <code>geometry</code> checks if already existing clusters span newly created volume upon adding of a new phase; <code>map</code> allows for mapping box geometry to its minimum image; <code>pbc</code> applies periodic boundary conditions to existing molecules when changing box shape; valid options are <code>true</code> and <code>false</code> ; <code>id</code> sets the id used for the targetted system.
<code>system_charge</code>	<code>true</code>	Check if the total system charge equals to zero; valid options are <code>true</code> and <code>false</code> .
<code>system_id</code>	<code>main</code>	Set the EMC system ID; a string is expected.
<code>system_geometry</code>	<code>true</code>	Checks if already existing clusters span newly created volume upon adding of a new phase; valid options are <code>true</code> and <code>false</code> .
<code>system_map</code>	<code>true</code>	Map the active system box to its minimum shape; valid options are <code>true</code> and <code>false</code> .
<code>system_pbc</code>	<code>true</code>	Apply periodic boundary conditions after building; valid options are <code>true</code> and <code>false</code> .

<code>temperature</code>	300 or 1	Provide the simulation temperature; atomistic and dimensionless force fields use 300 K and 1 respectively as default.
<code>tequil</code>	1000	Set the LAMMPS equilibration time
<code>tfreq</code>	10	Number of time steps skipped before adding a configuration to an average as sampled in LAMMPS input scripts.
<code>tighten</code>	false	Set margin for tightening the simulation box when importing structures; options are false or a value to represent the desired margin, e.g. 3 Angstroms for atomistic systems; note, that tightening only occurs in the chosen build direction (see <code>direction</code> option).
<code>timestep</code>		Provides the LAMMPS MD time step.
<code>torsion</code>	-	Set force field handling for torsion typing; valid options are ignore , complete , warn , empty , and error .
<code>triclinic</code>	false	Set LAMMPS triclinic mode; needed for simulation of triclinic boxes; valid options are true and false .
<code>trun</code>	10000000	Set LAMMPS run time; setting <code>trun</code> to - avoids its addition to job run scripts, thus not overriding different settings in subsequent chemistry files.
<code>units</code>	-	Set type of units; valid options are reduced , real , and si ; alternatively, the outdated option <code>lj</code> can be used for reduced .
<code>units_energy</code>	-	Set units for energetic scale.
<code>units</code>	-	Set units for length scale.
<code>volume</code>	false	Assume volume fractions in <code>chemistry.csv</code> file; options are true or false .
<code>warn</code>	true	Control output of warning information; options are true or false .

<code>weight</code>	<code>bond=0.0001,</code> <code>focus=1,</code> <code>nonbond=0.0001</code>	User provided energetic weight for building procedure controlling nonbonded, bonded, and focussed interactions; the latter applies to imported structures.
<code>width</code>	<code>false</code>	Sets double width for generated output scripts; options are <code>true</code> for 160 characters or <code>false</code> for 80 characters.
<code>workdir</code>	-	Set the work directory, which is used as a base for creating the EMC work environment.

4.6 File Formats

All entries within files are pasted into the resulting EMC scripts and force field files. Use of a chemistry file is mandatory. Use of references and parameters files is optional. All entries are assumed to be comma separated unless otherwise stated. All **ITEM** identifiers can be followed by an optional **comment** keyword, which can be set to either **true** or **false**.

```
ITEM  COMMAND [comment=[true|false]]
```

The full **ITEM** paragraph will be ignored until the next **ITEM END**, when keyword **comment** is set to **true**.

4.6.1 Environment File

The environment file functions as a wrap-around to chemistry files. It allows for setting up multiple simulations within one file, including looping over variables, thus enabling setting up series of simulations scanning parameter space. The environment format allows for defining parameter scans and simulation definition in one succinct file, thus furthering compact overviews. The keyword **ITEM** is required to precede the identifiers listed in the following table. The identifiers are listed in the preferred order in which they are to appear in the environment script.

Identifier	Description
ANALYSIS	Controls the type of analysis performed during execution of analysis scripts as generated in directory analyze ; this paragraph allows to include pre-defined or user-defined analysis scripts into a project's analysis; sample scripts can be found in <code>\${EMC_ROOT}/scripts/analyze</code> ; current supported types are cavity , density , energy , green-kubo , last , pressure , and volume ; option active controls inclusion of the analysis type; valid values are true or false ; options archive , dir , skip , and window are automatically passed to the analysis script, bearing internal settings; these options can be overridden, but is not advised; options and values are separated by an equal sign (=); different options are separated by a comma, tab or space; location of scripts is looked for in <code>\${EMC_ROOT}/scripts/analyze/</code> , <code>chemistry/analyze</code> , or the path associated with the indicated script
CLUSTERS	Sets general cluster definitions which replace the <code>@{CLUSTERS}</code> reference in the chemistry template in the same STAGE and TRIAL section; CLUSTERS contents is stored in <code>chemistry/clusters/\$stage/trial.dat</code>
COMMENTS	Sets the start of a comments section, which can only be terminated by a matching END
END	Marks the end of a paragraph

ENVIRONMENT	Sets optional variables related to the environment (formerly <code>OPTIONS</code>); environment options are defined previously (see [Environment Options] , page [undefined])
FIELD	Allows for a direct definition of force field parameters within the environment concept; the resulting <code>.prm</code> parameter file and <code>.top</code> topology file are stored in <code>chemistry/field/\$stage/project.{prm top}</code> and should be referred to by using this location (see Section 4.6.3 [Field File] , page 45)
GROUPS	Sets general group definitions which replace the <code>@{GROUPS}</code> reference in the chemistry template in the same <code>STAGE</code> and <code>TRIAL</code> section; <code>GROUPS</code> contents is stored in <code>chemistry/groups/\$stage/trial.dat</code>
INCLUDE	Specifies a files to be included containing a subset of commands; the file name follows <code>INCLUDE</code> directly; <code>INCLUDE</code> is a single line item and is not closed with an <code>END</code>
LOOPS	Lists variables over which to loop; variables are expected to be lower case; variables <code>stage</code> , <code>trial</code> and <code>copy</code> are reserved; variables can be paired or coupled by adding <code>:p</code> behind the variable, which couples the current variable to the previous; add <code>:h</code> in order to hide or exclude the variable from the data directory structure; an <code>:h</code> automatically implies the variable is paired with its predecessor; add <code>:d</code> to the predecessor when double occurrences are intended (useful in case of pairing); later reference to any and all variables recorded in <code>LOOPS</code> are to be preceded with <code>@</code> , enclosed within curly brackets <code>{}</code> , and written in all-caps; these references can also be used in subsequent loop variables
PARAMETERS	Allows for direct inclusion of DPD parameters; the resulting parameter file is stored under <code>chemistry/field/\$stage/parameters.csv</code> (see Section 4.6.5 [Parameters File] , page 50)
POLYMERS	Sets general polymer definitions which replace the <code>@{POLYMERS}</code> reference in the chemistry template in the same <code>STAGE</code> and <code>TRIAL</code> section; <code>POLYMERS</code> contents is stored in <code>chemistry/polymers/\$stage/trial.dat</code>
REFERENCES	Allows for direct inclusion of DPD references; the resulting reference file is stored under <code>chemistry/field/\$stage/references.csv</code> (see Section 4.6.4 [References File] , page 49)
SHORTHAND	Sets general shorthand definitions which replace the <code>@{SHORTHAND}</code> reference in the chemistry template in the same <code>STAGE</code> and <code>TRIAL</code> section; <code>CLUSTERS</code> contents is stored in <code>chemistry/shorthand/\$stage/trial.dat</code>

STRUCTURES	Defines a list of structures, relating directly to <code>trial</code> loop entries; structures are referred to by <code>@{STRUCTURE}</code>
STAGE	Indicates the start of a section relating to loop variable <code>stage</code> ; <code>STAGE</code> is followed by a stage indicator, which should appear in the enumerated loop variable <code>stage</code> ; all following paragraphs relate to the indicated stage; <code>STAGE</code> is a single line item and is not closed with an <code>END</code> ; all identifiers following <code>STAGE</code> refer to loop variable <code>stage</code>
TEMPLATE	Defines a chemistry template; templates are stored in <code>chemistry/stages/stage.esh</code> ; templates are used to create complete chemistry files upon execution of the associated run script; references to loop variables start with <code>@{}</code> and are written in all caps; references are treated as environment variables
TRIAL	Indicates the start of a section relating to loop variable <code>trial</code> ; <code>TRIAL</code> is a single line item and is not closed with an <code>END</code> ; all identifiers following <code>TRIAL</code> refer to loop variable <code>trial</code>
VARIABLES	Lists extra environment variables; environment variables are referred to with an at symbol <code>@</code> and are enclosed by curly brackets <code>{}</code> ; reserved variables are <code>@{EMCROOT}</code> for the root location of EMC and <code>@{WORKDIR}</code> for the directory in which the collection of simulations takes place
WRITE	Specify a message line to be written to the output; the written text directly follows <code>WRITE</code> ; <code>WRITE</code> is a single line item and is not closed with an <code>END</code>

EMC setup only assumes environment modus, when the `environment` variable is set to `true` in the first occurring `OPTIONS` paragraph. Environment mode allows for optionally setting names associated with build, analyze, and run scripts, as well as defining which queues to be used. Dashes imply undefining parameters. The `ncores` parameter is mandatory. For certain problems, it can be desired to simulate multiple independent structures in order to obtain a correct statistical sample. To this end, a loop variable `copy` is available, which creates multiple copies with the same conditions, but executing building with different random seeds, thus creating independent initial structures. Note, however, that use of this features quickly creates many simulations, which harbors the danger of overcrowding high-performance computing queueing systems and over-requesting available computational resources. An example for environment mode application can be found in `${EMC_ROOT}/examples/setup/environment/shear`.

4.6.2 Chemistry File

The chemistry file – with default file name `chemistry.esh` – supports environment, new-style, and legacy formats. Columns can only be separated by commas or tabs. Next line extension occurs by using an ampersand `&` at the end of to be extended line.

4.6.2.1 General

The new-style format allows for the definition of polymers, which uses keyword `ITEM` followed by an identifier to distinguish between different contributing paragraphs, as described by

Identifier	Description
CLUSTERS	Sets cluster definitions, using previously defined cluster groups
COMMENTS	Sets the start of a comments section, which can only be terminated by a matching <code>END</code>
EMC	Defines a section to be copied verbatim into the EMC build script; optionally, the keyword can be followed by either one or two numeric values, identifying which phase and where the block verbatim text is to be added to; alternatively, identifiers <code>phase</code> and <code>spot</code> can be used, followed by an equal sign (<code>=</code>); valid values are the respective phase for identifier <code>phase</code> and a number between 0 and 2 for <code>spot</code> to place the defined section at specific spots in the selected phase section in the EMC build script; defaults for <code>phase</code> and <code>spot</code> are 1 and 0 respectively when value and identifiers are omitted;
END	Marks the end of a paragraph
FIELD	Allows for a direct definition of force field parameters; the resulting <code>.prm</code> parameter file and <code>.top</code> topology file are stored in the current directory (see Section 4.6.3 [Field File], page 45)
GROUPS	Sets separate group definitions, including polymeric repeat units
INCLUDE	Specifies a files to be included containing a subset of commands; the file name follows <code>INCLUDE</code> directly; <code>INCLUDE</code> is a single line item and is not closed with an <code>END</code>

LAMMPS	Defines a section to be copied verbatim into the LAMMPS execution script, appearing atop the simulation section; alternatively, identifiers stage and spot can be used, followed by an equal sign (=); allowed options for stage are – in order of appearance in the LAMMPS execution script – header , variables , interaction , equilibration , simulation , integration , sampling , intermediate , run ; allowed options for spot are head and tail ; stage options correspond to the comments in the resulting LAMMPS execution script; defined additional variables as described above can be used when captured between <code>\${..}</code> .
OPTIONS	Allows for setting all command line options of <code>emc_setup.pl</code> inside chemistry files, thus creating the option of consolidating a simulation setup within one file only; see the previous paragraph; syntax is <code>option,value[...]</code> , where <i>option</i> refers to any of the options as defined previously (see (undefined) [Chemistry Options], page (undefined))
POLYMERS	Defines polymers, using previously defined polymeric groups; this paragraph can be used for defining polymers either through the GROUPS or CLUSTERS paragraph
PARAMETERS	Allows for direct inclusion of DPD parameters; parameters are not stored but used internally (see Section 4.6.5 [Parameters File], page 50)
PROFILES	Defines additional mass profiles, where each line generates an additional profile; syntax is <code>name, mode[:type[:binsize]], contributor[,...]</code> ; <i>mode</i> can be either cluster or type ; <i>type</i> sets the type of profile, which can either be density , density3d , or pressure ; density is assumed when omitted; <i>binsize</i> sets the individual bin size; the general binsize is assumed when omitted; <i>contributor</i> refers to either a cluster name or type respectively; note, that profile coordinates are in reduced units and run from 0 to 1
REFERENCES	Allows for direct inclusion of DPD references; references are not stored but used internally (see Section 4.6.4 [References File], page 49)
SHORTHAND	Legacy format, allowing for group and cluster definitions in shorthand notation; polymeric definitions are not possible within this format
VARIABLES	Defines additional variables, which are either used by read EMC structures from the EMC structure library (<code>\$EMC_ROOT/lib</code>), serve as pass-through to subsequent simulation packages (e.g. LAMMPS), or are used in the generated EMC build script; the section identifier VARIABLES can be followed by keywords head for appearance at the top or tail for appearance at the bottom of the EMC script variable block
WRITE	Specify a message line to be written to the output; the written text directly follows WRITE ; WRITE is a single line item and is not closed with an END

4.6.2.2 Shorthand

The **SHORTHAND** format can be used without a line starting with keyword **ITEM**, provided its lines are at the start of the chemistry file. Conversely, the shorthand format also provides quick definition of non-polymeric molecules without the need of defining groups and clusters. Its format is given by the following table.

Entry	Description
1	Cluster and Sets the group name, which optionally can be followed by <code>f[ield]=id[,...]</code> , <code>m[ass]=#</code> , or <code>t[erm]</code> – separated by colons – where characters between <code>[]</code> are optional and can be omitted; multiple options can be set in one line; <code>f[ield]=id</code> sets a specific field <code>id</code> when using multiple fields for typing, e.g for CHARMM (see Section 5.10 [Field], page 101), where <code>id</code> can be part of the full identifier; <code>m[ass]=#</code> overrides automated setting of group mass, where <code>#</code> represents the desired mass; <code>t[erm]</code> marks group as a terminator (see Section 5.16 [Groups], page 112)
2	Chemical representation in the form of a SMILES string (see Section 7.8 [SMILES], page 183)
3	Fraction or count; can be either molecular, mass, or volume fraction, or the number of clusters based provided option (see <code><undefined></code> [Chemistry Options], page <code><undefined></code>), for options <code>mass</code> , <code>mol</code> , or <code>number</code>)
4	Molecular mass in g/mol
5	Molecular volume in cc/mol

Column 4 and 5 are optional.

4.6.2.3 Groups

The **GROUPS** paragraph defines multiple groups, including polymeric groups. Its format is given by the following table.

Entry	Description
-------	-------------

1	Sets the group name, which optionally can be followed by <code>f[ield]=id[,...]</code> , <code>m[ass]=#</code> , or <code>t[erm]</code> – separated by colons – where characters between <code>[]</code> are optional and can be omitted; multiple options can be set in one line; <code>f[ield]=id</code> sets a specific field <code>id</code> when using multiple fields for typing, e.g for CHARMM (see Section 5.10 [Field], page 101), where <code>id</code> can be part of the full identifier; <code>m[ass]=#</code> overrides automated setting of group mass, where <code>#</code> represents the desired mass; <code>t[erm]</code> marks group as a terminator (see Section 5.16 [Groups], page 112); note, that groups with only one connection are automatically terminators
---	---

- 2 Sets the group chemistry using SMILES strings (see Section 7.8 [SMILES], page 183) or polymer type to turn on polymer mode; in SMILES polymeric connectivity points are marked with *****; valid polymer types are **alternate**, **block**, and **random**; column 3 and onwards are omitted when using groups in polymer mode; the polymer distribution is defined through the **POLYMERS** paragraph
- 3 Defines connection index for polymeric groups of current group; indices refer to ***** occurrences in the previously set SMILES string
- 4 Defines the connecting group **name** and corresponding **index** in the format **name:index**
- 5- Repeats of column 3-4 for subsequent connections

4.6.2.4 Clusters

The **CLUSTERS** paragraph defines multiple clusters. Options following keyword **import** in entry 2 can be used in any order when superceeded by their respective identifier and separated by an = sign. Column 4 and onward are optional and have different meaning depending on the mode in which the cluster option is called. In case of polymeric definitions, column 4 through 5 are used. In case of imported structures, all columns 4 and onward can be used. Identifiers instead of columns can be used when importing structures. Each identifier is described in the paragraph and table hereafter. The use of commas or tab as option separators between entries is advised when defining an import. Its format is given by the following table, where the identifier column refers to the next table.

Entry	Identifier	Description
1	-	Sets the cluster name
2	-	Sets the associated group, the type of polymer – indicated by alternate , block , or random – or whether to expect a imported structure from file as incicated by keyword import ; consequently, group names are not allowed to have reserved names alternate , block , random , or import
3	- or ncells	Sets a fraction or count, which can be either a mol, mass, or volume fraction, or number of clusters, based provided command line option (see [Chemistry Options] , page [Chemistry Options] , for options mass , mol , or number); in case of keyword import in entry 2, this entry contains the number of repeat units (id ncells)
4	- or name	Molecular mass in g/mol or file name (id name) in case of keyword import in entry 2

5	<code>- or mode</code>	Molecular volume in cc/mol or import mode (<code>id mode</code>) when selecting keyword <code>import</code> in entry 2
6	<code>type</code>	Optional entry for import type
7	<code>flag</code>	Optional entry for setting mobility flags of imported structures
8	<code>density</code>	Optional entry for setting the density treatment of imported structures
9	<code>focus</code>	Optional entry for marking the imported structure as focus
10	<code>tighten</code>	Optional entry to trim or tighten imported structures
11	<code>ntrial</code>	Optional entry for setting the number of trials
12	<code>periodic</code>	Optional entry for setting periodicity; currently not implemented
13	<code>field</code>	Optional entry for enforcing a specific force field for force field typing of the imported structure;
14	<code>exclude</code>	Optional entry for setting an exclusion region
15	<code>depth</code>	Optional entry for setting ring recognition recursive depth
16	<code>percolate</code>	Optional entry for identifying a percolating crystalline structure
17	<code>unwrap</code>	Optional entry for unwrapping imported structures
18	<code>guess</code>	Optional entry for guessing unwrap status based on input
19	<code>charges</code>	Optional entry for using charges from imported morphologies
20	<code>formal</code>	Optional entry for using formal charges from imported morphologies when available

The following table describes import identifiers, which can be used without having to abide by strict column sequences. Identifiers and their values are separated by equal signs (=), omitting any spaces. Examples are mentioned after this table.

Identifier	Default	Description
<code>charges</code>	<code>false</code>	Use charges from imported structures; allowed values are <code>true</code> or <code>false</code>

density	mass	Sets density treatment of imported structures, which is needed for calculating correct box dimensions when adding material; allowed values are mass or number
depth	auto	Sets the maximum size of rings to be recognized; the provided input should be a definite positive integer or keyword auto
exclude	true	Toggles whether to add an repulsive exclusion region around the imported structure; allowed values are true or false
field	-	Enforces a specific force field for force field typing of the imported structure
flag	rigid	Sets mobility flag for imported structures; allowed flags are fixed , rigid , or mobile
focus	true	Toggles whether to focus on the imported structure, which excludes its volume for additionally built molecules; allowed values are true or false
formal	true	Include formal charges from imported morphologies when provided in input structure; allowed values are true or false
guess	auto	Sets optional guessing of unwrapping based on imported structures; only functions when unwrap is set to auto ; allowed values are auto , true , or false
mode	emc	Optionally enforce import mode when not using name extensions in identifier name ; allowed modes are emc for EMC files, pdb for PDB files – providing both .pdb and .psf files, or insight for Insight II files – providing both .car and .mdf files
name	-	Sets file name; allowed file name extensions are .emc for EMC files, .car or .mdf for InsightII files, and .pdb or .psf for PDB files; default is .emc ; extensions override mode defaults
ncells	1:auto:auto	Sets the number of repeated unit cells along the direction as set by option direction in the options paragraph (see [Chemistry Options] , page (undefined)); repeats in multiple directions can be set when dividing input by colons in order of <i>x</i> , <i>y</i> , and <i>z</i> direction respectively; default is 1 in the main direction and auto in the remaining directions, e.g. auto:1:auto for main direction <i>y</i>

ntrials	10000	Sets the number of trial iterations used for determining the volume of an import of type= <i>line</i> or structure (see column 6), which employs a Monte Carlo algorithm; option is a number larger than zero
percolate	<i>auto</i>	Sets whether imported structure reflects a percolating crystal; allowed values are <i>auto</i> , <i>true</i> , or <i>false</i>
periodic	-	Sets periodicity; currently not implemented
tighten	-	Set the tightening distance, which effectively shrink-wraps the box around what is imported; allowed option is a distance; default is a distance of 1 or 3 for coarse grained or atomistic force fields respectively
type	<i>surface</i>	Sets the imported structure type; allowed types are crystal when adding no additional sites, surface when adding sites in one direction as set by option direction in the options paragraph (see [Chemistry Options] , page (undefined)), tube when adding in two directions, or structure when adding in three directions
unwrap	<i>false</i>	Sets whether to unwrap the imported structure; allowed values are <i>auto</i> , <i>true</i> , or <i>false</i>

Column 4 is optional when defining molecules, but mandatory when importing structures. Importing options defined in columns 3 through 14 can be preceded by their respective identifiers, i.e. **ncells**, **name**, **mode**, **type**, **flag**, **density**, **focus**, **tighten**, and **ntrials**, separated by an equal sign. The order as given in the table above is assumed when omitting this identifier. The specified default is taken, when the an option is omitted. A formal example of an import line is given by

```
surface      import, name=$root+"lib/fcc", ncells=8, mode=emc, &
              type=surface
```

which creates a surface with cluster name **surface**, using a nonbonded fcc lattice from the EMC library, consisting of 8 repeat units in the x-direction. Note, that any given order can be used, when identifiers are specified. A shorter, but equivalent definition is given by

```
surface      import 8      $root+"lib/fcc" emc
```

where now the option order matters, therefore creating the need for strictly following the column order as given in the above table. Alternatively, the import mode can be determined by EMC setup through adding the file extension, i.e.

```
surface      import name=$root+"lib/fcc.emc" ncells=8 type=surface
```

or

```
surface      import 8      $root+"lib/fcc.emc"
```

Note, that a space or comma can also be used as separator. However, this option is not advised when using a space or comma in the file name. Currently, only one structure at a time can be imported by EMC.

4.6.2.5 Polymers

In the case of a polymer, the cluster name has to appear in the **POLYMERS** paragraph, when defining the type of polymer in column 2. Otherwise, a group name is assumed. The **POLYMERS** paragraph describes the composition of one polymer at a time. Each polymer definition starts with the polymer's cluster name on a separate line. The polymer name optionally can be followed by keywords **fraction**, **niterations**, or **order**, separated from their value by an = sign. Defining any of these keywords overrides global definitions for the indicated polymer only (see keyword **polymer** under **OPTIONS** for globals (see (undefined) [Chemistry Options], page (undefined))). Valid **fraction** options are **number** for using number of molecules and **mass** molecule mass as distribution entries. Valid **order** options are **list** for sequentially and **random** of randomly interpreting the provided polymer distribution. The **niterations** keyword sets the maximum allowed number of iterations used during polymer sequence determination. Lines following the polymer name define the polymer's distribution of contributing groups. Multiple polymer definitions can be provided within this paragraph. The format of contributing groups is given by the following table.

Entry Description

- | | |
|----|---|
| 1 | Mol or mass fraction of the polymer cluster defined by this line; mass fractions are only allowed when defining polymers through groups |
| 2 | Group name; multiple group names can be given through separation with a ':'; weights – with which groups are randomly picked – follow an '=' and are separated by a ':', e.g. A:B=1:2 ; equal weights are assumed when omitted |
| 3 | Number of repeat units for the above group |
| 4- | Repeats of columns 2-3 for subsequent groups |

Capping or end groups should be included in the above list for mass calculation purposes, although they are neglected by EMC. EMC selects end groups internally.

4.6.2.6 DPD Additions

The following identifiers are valid only, when selecting DPD as force field:

Identifier	Description
ANGLES	Defines a paragraph for additional specific angle entries, which should be provided in the order of <code>type1</code> , <code>type2</code> , <code>type3</code> , <code>k</code> , <code>theta</code>
BONDS	Defines a paragraph for additional specific bond entries, which should be provided in the order of <code>type1</code> , <code>type2</code> , <code>k</code> , <code>l</code>
MASSES	Defines or – in case of replicas – redefines masses, which should be provided in the order of <code>type</code> , <code>mass</code> ; <code>mass</code> is a definite positive number and has reduced units when no reference type has been chosen; otherwise, <code>mass</code> is in units of the reference type (see Section 4.4 [Options], page 12)
NONBONDS	Defines a paragraph for nonbond entries, which should be provided in the order of <code>type1</code> , <code>type2</code> , <code>a</code> [, <code>cutoff</code> [, <code>gamma</code>]]; these entries overwrite already existing entries
REPLICAS	Creates duplicates of existing nonbond entries, which should be provided in the order of <code>target[:factor]</code> , <code>source[:fraction[:flag]]</code> [, <code>source[:fraction[:flag]]</code> [, ...]], <code>offset</code> ; interaction constants, mass, cutoff, and gamma are averaged when multiple sources are provided; <code>offset</code> defines the offset of the new DPD interaction constant a_{ij} with respect to the original; <code>factor</code> defines a multiplication factor with respect to the <code>offset</code> ; <code>factor</code> is set to 1 when omitted; <code>fraction</code> defines which fraction to use of each contributing component; equal fractions are assumed when omitted; normalization of <code>fraction</code> is controlled by <code>flag</code> , which can either be <code>true</code> or <code>false</code> ; by default <code>fraction</code> is normalized; <code>fraction</code> is not normalized when any one <code>flag</code> is set to <code>false</code>
TORSIONS	Defines a paragraph for additional specific torsion entries, which should be provided in the order of <code>type1</code> , <code>type2</code> , <code>type3</code> , <code>type4</code> , <code>k</code> , <code>n</code> , <code>delta</code> , and optionally following triplets of <code>k</code> , <code>n</code> , and <code>delta</code>

Both keyword and identifier must be in all caps and separated by a comma or tab. Currently, alterations or additions to force field files only function for the DPD force field. Types can be replicated in order to either create duplicates or compounded types, where

$$a_{ij}^* = f(a_{ij}^* - 25) + \Delta$$

expresses the new interaction parameter a_{ij}^* . Assume, that a new type AIR is to be created based on an alkane representation C4, where multiplication factor $f = 1.5$ and offset $\Delta = 25$. This would result in the following REPLICAS entry

```
AIR:1.5      C4      25
```


For each of the parameter additions (i.e. nonbonds, bonds, angles, or torsions), entries containing a wildcard `*` will appear in the `AUTO` paragraph of the resulting force field file.

4.6.3 Field File

The field file – using the extension `.define` – allows for defining force field parameters and rules within one file. Translation of this file into an EMC `.prm` parameter file and `.top` topology file occurs by applying `field.pl`, which can be found in the EMC script directory `${EMC_ROOT}/scripts`. This format can also be included in environment and chemistry files by adding the field definition between identifiers `FIELD` and `END`, both preceded by keyword `ITEM`.

4.6.3.1 General

The field format itself also uses keyword `ITEM` to identify subsequent main functionality paragraphs, as described by

Identifier	Description
DEFINE	Defines different force field settings as described below; this paragraph is mandatory
MASSES	Defines the types and their associated masses and subsequent definitions; the syntax is <i>type, mass, element, number of connections, formal charge, comment</i> ; this paragraph is mandatory
COMMENTS	Allows for inclusion of comments specific to the force field's derivation; this paragraph is optional
REFERENCES	Reports the literature references used to construct the force field; the syntax is <i>year, volume, page, journal</i> ; this paragraph is optional
PRECEDENCE	Defines the precedence table as described below; this paragraph is optional
EQUIVALENCE	Describes the equivalences of the force field; equivalences state which type can be used instead of the official type for abstracting the associated force field parameters; the syntax is <i>type, pair, bond, angle, torsion, improper</i> ; this paragraph is optional
NONBOND	Defines the nonbond parameters of the force field; the syntax starts with <i>type1, type2</i> , followed by the actual parameters (e.g. <i>epsilon, sigma</i> for Lennard-Jones force fields, and <i>a, cutoff, gamme</i> for a DPD force field); the expected number of parameters is dependent on the chosen force field mode; this paragraph is mandatory
BOND	Defines the parameters for bond length interactions; the syntax is <i>type1, type2, k, l0</i> ; the interaction function depends on the the force field mode, but is in general harmonic in nature; this paragraph is optional

ANGLE	Defines the parameters for bond angle interactions; the syntax is <i>type1, type2, type3 k, theta0</i> ; the interaction function depends on the the force field mode, but is in general harmonic in nature; this paragraph is optional
TORSION	Defines the parameters for dihedral interactions; the syntax is <i>type1, type2, type3, type4, k, n, delta[, k, n, delta ...]</i> ; the allowed number of additional parameter entries depends on the chosen force field mode; the interaction function depends on the the force field mode, but is in general a Fourier series of cosines (e.g. see Section 7.12.17 [Standard], page 309, Torsion paragraph); this paragraph is optional
IMPROPER	Defines improper interactions; the syntax is <i>type1, type2, type3, type4, k, psi0</i> ; the interaction function depends on the the force field mode, but is in general harmonic in nature; this paragraph is optional
RULES	Defines the rules associated with the types as defined by the MASSES paragraph; the syntax is <i>type, partial charge, rule[, rule ...]</i> ; one type can have multiple rules; a rule describes a unique chemical environment, which defines the type at hand; this paragraph is optional
TEMPLATES	Defines templates, which can be used as aliases when defining group chemistry; this paragraph is optional

Each entry can be separated by either a **TAB**, **COMMA**, or **SPACE**. A **TAB** represents the preferred separator. Wildcards as represented by ***** can be used for types in parameter definitions. Partial wildcards are also allowed. The partial wild cards can be understood by the following example: assume **c4** generally describes an SP3 carbon with no hydrogen attached, **c4h** with one hydrogen attached, and **c4h2** with two hydrogens attached, then wildcard **c4*** would describe all three of these types.

4.6.3.2 Define

Specific alternative settings can be defined for force field by using the **DEFINE** paragraph, as decribed by the following

Identifier	Description
ANGLE	Sets the error handling for typing of angle interactions; options are ignore , complete , warn , empty , and error
CREATED	Sets the creation date of the force field
CUTOFF	Sets the cut off for all nonbonded interactions

DENSITY	Sets the unit of length as used by the parameters in the force field; options are <code>g/cc</code> , <code>kg/m^3</code> , and <code>reduced</code> ; both <code>LENGTH</code> and <code>ENERGY</code> are set to <code>reduced</code> when the latter option is chosen, thus resulting in a force field in reduced units
ENERGY	Sets the unit of length as used by the parameters in the force field; options are <code>j/mol</code> , <code>kJ/mol</code> , <code>cal/mol</code> , <code>kcal/mol</code> , <code>kelvin</code> , and <code>reduced</code> ; both <code>LENGTH</code> and <code>DENSITY</code> are set to <code>reduced</code> when the latter option is chosen, thus resulting in a force field in reduced units
FFDEPTH	Sets the maximum recursive with which to trace the chemical surrounding of a site during typing; this depth should equal the maximum depth of the provided rules
FFMODE	Sets the name or mode of the force field; options are <code>born</code> , <code>charmm</code> , <code>dpd</code> , <code>martini</code> , <code>mie</code> , <code>opls</code> , <code>sdk</code> , <code>standard</code> , and <code>trappe</code>
FFTYPE	Sets the force field type; options are <code>atomistic</code> , <code>united</code> , and <code>coarse</code>
IMPROP	Sets the error handling for typing of improper interactions; options are <code>ignore</code> , <code>complete</code> , <code>warn</code> , <code>empty</code> , and <code>error</code>
INNER	Sets the inner cut off when using a force field with a switching function for nonbonded interactions (e.g. as in the CHARMM force field (see Section 7.12.7 [CHARMM], page 222))
LENGTH	Sets the unit of length as used by the parameters in the force field; options are <code>angstrom</code> , <code>nanometer</code> , <code>micrometer</code> , <code>meter</code> , and <code>reduced</code> ; both <code>DENSITY</code> and <code>ENERGY</code> are set to <code>reduced</code> when the latter option is chosen, thus resulting in a force field in reduced units
MIX	Sets the mixing rule; options are <code>none</code> , <code>bethelot</code> , <code>arithmetic</code> , <code>geometric</code> , and <code>sixth_power</code>
NBONDED	Sets the number of bonded atoms to exclude from nonbonded interactions; the number varies by force field
PAIR14	Sets the error handling for typing of 1-4 pair interactions; options are <code>off</code> , <code>false</code> , <code>exclude</code> , <code>on</code> , <code>true</code> , and <code>include</code>
TORSION	Sets the error handling for typing of dihedral interactions; options are <code>ignore</code> , <code>complete</code> , <code>warn</code> , <code>empty</code> , and <code>error</code>
VERSION	Sets the force field version

Error handling, as mentioned in the latter options of the above table, can be dealt with in a number of ways. Any errors can be either fully ignored without any output by EMC during typing by choosing option **ignore**. A warning will be generated, when choosing option **warn**. An empty parameter entry will be generated without any output to screen when choosing option **empty**. EMC will continue after all of these options. However, EMC will generate warnings for all missing parameters and will cease execution when choosing options **error**. Examples of **.define** files can be found for several force fields as provided in the `${EMC_ROOT}/field/` force field directory (e.g. the CHARMM, OPLS, and TraPPE force fields).

4.6.4 References File

Reference files are optional and are used to create comprehensive EMC force field files (using `.prm` extensions). Currently only DPD force fields are used when interpreting the reference file. Its default file name is `references.csv`.

Column	Description
1	Short ID as referred to in <code>parameters.csv</code>
2	ID as represented in produced <code>.prm</code> force field files
3	Bead mass in g/mol
4	Bead volume in nm ³
5	Number of connections that this bead can have
6	Effective charge of bead
7	Number of repeat units represented in this bead (e.g. 3 when 3 ethyleneoxide monomers are captured by one bead)
8	Comment describing the origin of the bead

Future `emc_setup.pl` versions are intended to span a wider choice of force fields when using reference files.

4.6.5 Parameters File

Parameter files are optional and are used to provide parameters. Currently they are only used in case of the DPD force field (see Section 7.12.11 [DPD], page 261). Its default file name is `parameters.csv`.

Column	Line	Description
1	1	Mandatory empty field
	2-	Alphanumeric type name of bead A (short)
2	1	Mandatory empty field
	2-	Alphanumeric type name of bead B (short)
3	1	Temperature at which the interaction parameters in this column were determined.
	2-	Numerical value referring to the interaction of bead A and B
...		Identical to column 3.

Column 4 and subsequent columns can hold parameters determined at different temperatures. Their syntax is the same as for entries in column 3. Future `emc_setup.pl` versions are intended to include a wider choice of force field when using parameter files.

4.7 Examples

The directory `${EMC_ROOT}/examples/setup/` holds a number of examples with different complexity, all using `emc_setup.pl` as a base for creating EMC build and LAMMPS input scripts. The examples are subdivided in chemistry and environment related examples. This section describes ways to build bulk systems with a mixture of chemicals, typing of various force fields, construction of various types of polymers, building multiphase simulations, and ways to build material between two surfaces. A second set of examples in `${EMC_ROOT}/examples/build/` illustrate adaptation of EMC scripts directly for specific solutions. These scripts are not discussed here.

4.7.1 References

1. VMD - Visual Molecular Dynamics, '<http://www.ks.uiuc.edu/Research/vmd/>'
2. Humphrey, W., Dalke, A. and Schulten, K., "VMD - Visual Molecular Dynamics", *J. Molec. Graphics* **1996**, *14*, 33-38
3. LAMMPS - Molecular Dynamics Simulator, '<http://lammps.sandia.gov/>'
4. Plimpton, S., "Fast Parallel Algorithms for Short-Range Molecular Dynamics", *J. Comput. Phys.* **1995**, *117*, 1-19

4.7.2 Chemistry Mode

Chemistry mode examples show how to set up single system simulations using a number of different common problems, which include setting up bulk systems, systems with different force fields, polymer systems, systems with multiple phases, and systems with surfaces. Chemistry examples can be found in `${EMC_ROOT}/examples/setup/chemistry/`.

4.7.2.1 Bulk Mixture

Assume the desire exists to simulate a system consisting of 45% water, 40% alcohol, 5% salt, and 10% sugar by mass, where the force field of choice is PCFF and consists of about 2000 particles once built. The setup utility offers multiple ways for setting up a simulation. The quickest way is to use `emc_setup.pl`'s legacy format. For this format, one first creates a `chemistry.esh` file with the following content:

```
water      0,45
alcohol    CC0,40
salt       [Na+] . [Cl-] ,5
sugar      OCC1OC(O)C(O)C(O)C10,10
```

Note, that the fraction indications – 45, 40, 5, and 10 – in the last column are normalized internally, and can therefore be any number. Here, percentages are taken for illustrating purposes. The next step is to invoke `emc_setup.pl`,

```
emc_setup.pl -field=pcff -ntotal=2000 -mass -build_dir=. bulk
```

where `-field` selects the force field, `-ntotal` sets the number of particles, `-mass` assumes mass fractions in the `chemistry.esh` file, and `-build_dir` sets the origin of the created files to the current directory. An alternative chemistry file with the name `bulk.esh` uses shorthand notation in combination with options. Here all command line options are folded into the chemistry file, which reads as follows,

```
#!/usr/bin/env emc_setup.pl
```

```
# Options
```

```
ITEM      OPTIONS
```

```
replace   true
mass       true
ntotal     2000
field      pcff
density    1
build_dir  .
```

```
ITEM      END
```

```
# Shorthand
```

```
ITEM      SHORTHAND
```

```
water      0,45
alcohol    CC0,40
salt       [Na+] . [Cl-] ,5
```

```
sugar      OCC10C(O)C(O)C(O)C1O,10

ITEM      END
```

A more general way of describing the same simulations is by the use of **GROUPS** and **CLUSTERS**. EMC internally uses sites, groups, and clusters to represent atoms, repeat units, and molecules respectively. Using groups in case of small molecules makes less sense than using groups for polymers. However, as a general notation, this problem can also be defined by the following

```
#!/usr/bin/env emc_setup.pl

# Options

ITEM      OPTIONS

replace   true
mass      true
ntotal    2000
field     pcff
density   1
build_dir .

ITEM      END

# Groups

ITEM      GROUPS

water     0
alcohol   CCO
salt      [Na+] . [Cl-]
glucose   OCC10C(O)C(O)C(O)C1O

ITEM      END

# Clusters

ITEM      CLUSTERS

water     water,45
alcohol   alcohol,40
salt      salt,5
sugar     glucose,10

ITEM      END
```

Execution can be performed in one of two ways,

```
emc_setup.pl bulk
```

or, in the case of a Unix-like operating system,

```
./bulk.esh
```

provided `bulk.esh` is executable and `emc_setup.pl` can be found in the shell's predefined path. This solution allows for a comprehensive representation of a simulation, without having a number of separate files describing it. Once executed, `emc_setup.pl` creates the EMC build script `build.emc` and the LAMMPS input script `bulk.in`. To build an input structure, the EMC build script needs to be executed using EMC. Assume, that the LINUX version `emc_linux` is used, which location is in your path variable `$PATH`. Then,

```
emc_linux build.emc
```

will create input structures in PDB format – `bulk.pdb` and `bulk.psf` – and LAMMPS format – `bulk.in`. The resulting structure can be inspected by means of VMD (Visual Molecular Dynamics)^{1,2} through

```
vmd -e bulk.vmd
```

assuming, that `vmd` is in your path. The advantage of using `bulk.vmd`, is that this file contains the correct Van der Waals radii for the simulated beads. A molecular dynamics simulation with LAMMPS (Large-scale Atomic/Molecular Massively Parallel Simulator)^{3,4} can be started with

```
lmp_linux < bulk.in
```

assuming, that `lmp_linux` is in your path. A similar example can be found in the `${EMC_ROOT}/examples/setup/chemistry/bulk` directory.

4.7.2.2 Force Fields

EMC allows for the use of various force fields. To illustrate this, five examples are included for commonly used force fields, which are separated into atomistic (CHARMM, OPLS-AA, PCFF), united atom (OPLS-UA, TrAPPE), and coarse grained (DPD) examples. Force field choice is indicated by the `-field` flag of the `emc_setup.pl` script, which will try to find a corresponding force field in the `${EMC_ROOT}/field` directory. For the latter to work, it is assumed, that `emc_setup.pl` is located in the `${EMC_ROOT}/scripts` directory. With exception of the coarse-grained example, all examples use a molar ratio of 1:4 for a ethanol:water mixture. The corresponding chemistry file `chemistry.esh` for non coarse-grained examples contains the following lines

```
ethanol,CCO,20
water,O,80
```

Fraction indications do not have to reflect percentages: 1 instead of 20, combined with 4 instead of 80 is equally valid, due to internal normalization. For a system with 1000 atoms typed with the CHARMM force field and project name `solution`, the call to `emc_setup.pl` is given by

```
emc_setup.pl -ntotal=1000 -field=charmm -replace solution
```

where `-ntotal` indicates the number of atoms, `-field` specifies the desired force field, and `-replace` selects overwriting of possibly existing build and input scripts. An alternative to `chemistry.esh` is presented by `solution.esh`, in which case `emc_setup.pl` command line options are integrated into the script by means of an options section. Execution creates an EMC `build.emc` and a LAMMPS^{3,4} `solution.in` script. A change of force field and its corresponding typing occurs by altering the `-field` flag to e.g. `-field=trappe`, when desiring TraPPE typing. Note, that each force field only includes chemistries, for which it was parameterized. This means, that not all chemistries can be typed by all force fields. Also be aware, that typing is not guaranteed to be correct. It is kindly requested to report bugs, so that improvements to force fields can be included in next versions. Next steps for executing EMC and LAMMPS are identical to the steps as mentioned in the previous section for the bulk mixture example.

4.7.2.3 Record

EMC offers the possibility to follow the building process by creating a set of PDB's and accompanying PSF's during building, which is illustrated by the example found in `${EMC_ROOT}/examples/setup/chemistry/record/`. The build of a 60 carbon fullerene has been chosen to illustrate this option. EMC will produce a set of PDB's and PSF's, which are combined by `cat.sh`. The latter produces one PDB and PSF file, and one VMD script, after which it deletes the PDB's and PSF's produced by EMC. A `exec.sh` script illustrates the work flow. It uses `fullerene.esh` as the describing chemistry file, which set the following options,

ITEM	OPTIONS
<code>seed</code>	<code>-1</code>
<code>ntotal</code>	<code>60</code>
<code>density</code>	<code>0.1</code>
<code>temperature</code>	<code>300</code>
<code>grace</code>	<code>0.999</code>
<code>nrelax</code>	<code>200</code>
<code>radius</code>	<code>9.5</code>
<code>center</code>	<code>true</code>
<code>field</code>	<code>opls-ua</code>
<code>record</code>	<code>"record",10,true</code>
<code>replace</code>	<code>true</code>
<code>depth</code>	<code>6</code>
ITEM	END

A `seed` of -1 indicates for EMC to take the system clock as a seed. The total number of sites is set by `ntotal`, while the `density` is chosen to be 0.1 g/cc. The simulation `temperature` is set at 300 K. The keyword `grace` is used to allow for overlap during initial insertion, which is corrected during relaxation. The number of relaxation moves after insertion, `nrelax`, is set to 200, with a site inclusion `radius` of 0.95 nm. The first inserted site is set at the center of the box by setting keyword `center` to `true`. The chosen force field – as set by defining `field` – is OPLS-UA. Recording is switched on by defining `record`, where PDB's and PSF's are written out for the inserting of each site. The written out files have root file name `"record"`, where each file contains a trajectory resulting from relaxation. Here, snapshots are written every 10 cycles. Inactive sites with positions at (0,0,0) are included in PDB's. All files are opted to be replaced by setting the `replace` keyword to `true`. Ring recognition normally is automatic. However, this would take unnecessarily long for fullerenes due to the recursive nature of the underlying algorithm. EMC allows for reducing the recursion depth of this algorithm. Here, we chose to set the recursion `depth` to 6, since none of the rings in a fullerene have more than 6 members.

Fullerene chemistry is defined in the shorthand paragraph by

ITEM	SHORTHAND
------	-----------

```
fullerene      c12c3c4c5c1c6c7c8c2c9c1c3c2c3c4c4c%10c5c5c6c6c7c7c%11
                c8c9c8c9c1c2c1c2c3c4c3c4c%10c5c5c6c6c7c7c%11c8c8c9c1
                c1c2c3c2c4c5c6c3c7c8c1c23,1

ITEM           END
```

Intricate connectivity is illustrated by connectivity numbering (see Section 7.8 [SMILES], page 183). The SMILES for this example as well as for many chemistries can be found in Wikipedia.¹

Execution of this example is best performed by workflow script

```
./exec.sh
```

which calls `emc_setup.pl`, followed by EMC. It also concatenates building trajectories as produced by EMC into one set of files, which can be viewed with VMD by

```
vmd -e record.vmd
```

assuming VMD is in the user's path.^{2,3}

References

1. N.N., "Buckminsterfullerene", Wikipedia, 1/1/2016.
2. VMD - Visual Molecular Dynamics, '<http://www.ks.uiuc.edu/Research/vmd/>'
3. Humphrey, W., Dalke, A. and Schulten, K., "VMD - Visual Molecular Dynamics", *J. Molec. Graphics* **1996**, 14, 33-38

4.7.2.4 Polymers

Polymers have a wide field of application in chemical industry. EMC offers the possibility of creating a variety polymer classes. Examples can be found in `#{EMC_ROOT}/examples/setup/chemistry/polymer`. The focus in this section lays on building random, block, and alternate copolymers.

Let us assume, that we would like to study a copolymer consisting of four A beads, four B beads, and two D terminators. We would like to study a coarse-grained model with 1000 beads, using a general DPD force field with a number density of 3.

The `OPTIONS` section is defined by (see (undefined) [Chemistry Options], page (undefined)),

```

ITEM          OPTIONS

project       dpd
field         dpd/general
auto          true
ntotal        1000
density       3
replace       true
build_dir     .

ITEM          END

```

The first line sets the project name with keyword `project`, while the second selects the general DPD force field with keyword `field`. The option `auto` toggles the automatic addition of masses to the force field mass paragraph, in case masses are not provided. The total number of beads is set with `ntotal` and the system density with `density`. The keyword `replace` allows for existing files to be overwritten. The current directory is chosen as the build directory by keyword `build_dir`.

The generalized DPD form does not require a `parameters.csv` or `reference.csv` file, when not specifically defining pair-wise interactions. With this information, `emc_setup.pl` will create `dpd.prm` in the current directory, which is a DPD force field parameter file. This force field file is needed by EMC, in order to be able to type the different beads.

A polymer chemistry file cannot use shorthand notation as discussed in the previous paragraphs. Instead, group, cluster, and polymer specific sections need to be used (see Section 4.6.2 [Chemistry File], page 35).

First, three contributing groups are defined to describe beads A, B, and D,

```

ITEM          GROUPS

a              *A*,1,a:2,2,b:1
b              *B*,1,b:2
d              *D,1,a:1,1,a:2,1,b:1,1,b:2

ITEM          END

```

As with the bulk case, SMILES syntax is used to describe group chemistry. Additionally, connectivity between the different monomers needs to be defined. This connectivity is indicated by an asterisk *. By definition, a monomer has two or more connections and a terminator only one. With the connection points defined per monomer, we have to still define where these connection points connect to. Let us consider group **a** as an example. We would like to connect group **a** to both itself and group **b**,

```
*A*,1,a:2,2,b:1
```

A indicates the SMILES representation, followed by connection pairs. The first pair **1,a:2** means, that the first occurring * of ***A*** connects to the second occurring * of group **a**. The first pair **2,b:1** means, that the second occurring * of ***A*** connects to the first occurring * of group **b**. The same logic applies for subsequent definitions of groups **b** and **d**. The setup script adds all redundant connections, i.e. when connecting group **a** to group **b** with **2,b:1**, a connection on group **b** reading **1,a:2** is implied and does not have to be defined explicitly. One could define group **b** as

```
*B*,1,b:2,2,a:1
```

which would work, but it is not necessary. The definition of group **d** through

```
*D,1,a:1,1,a:2,1,b:1,1,b:2
```

defines all capping groups to the correct connectors. With the groups defined, we can now define the type of polymer we would like to build,

```
ITEM      CLUSTERS

poly      random,1

ITEM      END
```

The first keyword defines the polymer's name, **poly** here. Its name is followed by the type of polymer. The three available options are **random**, **block**, and **alternate**. Examples are available for each of these options in `#{EMC_ROOT}/examples/polymer`. Here we chose to create a random copolymer. The last entry denotes the polymer's fraction, here chosen to be 1. Different clusters and polymers can be added to this paragraph to form any mixture desired.

Finally, we define the polymer's morphology. We wanted 4 monomers **A**, 4 monomers **B**, and two terminators **D**,

```
ITEM      POLYMERS

poly
1         a,4,b,4,d,2
```



```
ITEM      END
```

Each polymer definition starts with its name as a header on a separate line. This name refers to the name chosen in the clusters section, here `poly`. The polymer's name is followed by all morphologies associated with this name. Here we only have one morphology, but we could also define a distribution of polymer morphologies. Each line contains the mol fraction of each contribution, followed by its morphological definition,

```
1          a,4,b,4,d,2
```

A subsequent polymer would start with its name on a separate line, followed by its definition(s), e.g. this section with two polymers would look like

```
ITEM      POLYMERS

poly
1          a,4,b,4,d,2

another
1          a,2,b,2,d,2
2          a,3,b,3,d,2
1          a,4,b,4,d,2

ITEM      END
```

Here, a polymer called `another` was added and assumed to have three contributing subpolymers, thus creating a distribution of polymer lengths within this polymer. The distribution uses mol fractions, which means, that the above example has a distribution of 1:2:1, or 25% of the first entry, 50% of the second, and 25% of the third. The `emc_setup.pl` script normalizes fractions in the resulting EMC build script.

4.7.2.5 Multiphase Systems

EMC offers the possibility of building multiphase systems by building one phase after another. This allows for setting up systems, for which it is known a priori, that the equilibrated state phase separates. This then allows for studying mass distribution and interfacial tension behavior. Let us investigate this possibility by means of an example, in which we study the behavior of sodium dodecyl sulfate (SDS) in a system of water and oil. This example can be found in `${EMC_ROOT}/examples/setup/chemistry/multiphase`, where the describing file `multiphase.esh` resides.

We assume the density of the water, SDS, and oil phase to be 1, 0.8, and 0.8 g/cc respectively. This system is at a temperature of 300K. We would like to describe the interactions with the CHARMM force field. The final simulation should have about 5000 particles. These prerequisites are captures in the `OPTIONS` paragraph (see [\[Chemistry Options\]](#), page [\(undefined\)](#)),

ITEM	OPTIONS
<code>ntotal</code>	5000
<code>temperature</code>	300
<code>density</code>	1,0.8,0.8
<code>phases</code>	water + sds + oil
<code>field</code>	charmm/c32b1/all_27_prot_lipid
<code>mass</code>	true
<code>profile</code>	true
<code>replace</code>	true
ITEM	END

Here, the keyword `ntotal` allows for setting the final number of sites to be simulated. The keyword `temperature` sets the temperature. The density of each phase is defined by keyword `density` through separation of each entry by either a comma or a tab. Chemically, each phase is defined by the keyword `phases`. Here, each phase is demarcated by the `+`-sign. Contributing clusters to each individual phase are separated by either commas, tabs, or spaces, e.g.

```
phases      water sds + oil
```

would create two phases: the middle phase would consist of a random mixture of water and sds, while the outer phase would consist of just oil. The fractions entered in the `SHORTHAND` are assumed to be by mass, which is expressed by setting the keyword `mass` to `true` in the `OPTIONS` section. Furthermore, analysis mass profiles is added to the generated LAMMPS input script `multiphase.in` by setting the keyword `profile` to `true`. Lastly, replacement of all scripts and files produced by executing `emc_setup.pl` are selected to be overwritten by setting keyword `replace` to `true`. Defining the chemical composition can be entered in the `SHORTHAND` section (see Section 4.6.2 [\[Chemistry File\]](#), page 35),

ITEM	SHORTHAND
------	-----------

```
water      0,0.4
oil        (C)8,0.4
sds        CCCCCCCCCC[O-]S([O-])([O-])[O-].[Na+],0.2

ITEM      END
```

Here, we select 40% by weight of both water and oil phases, while we add 20% by weight of SDS. Chemical identities are entered using the SMILES format (see Section 7.8 [SMILES], page 183).

Steps of execution are described in the usage section (see Section 4.2 [Setup Usage], page 9).

4.7.2.6 Surfaces

EMC allows you to set up surfaces and pack material between them. To this extend, the surface has to be identified by a **surface** keyword in the **CLUSTERS** section of your **emc_setup.pl** script. An example can be found in `${EMC_ROOT}/examples/setup/chemistry/surface`, where **surface.esh** provides the input information and **setup.sh** the manner in which **emc_setup.pl** should be called. In this example, we want to sandwich amorphous 100 decane molecules between two diamond surfaces, where the density of the decane phase should be 0.8 g/cc at a temperature of 298K. We have to first define the simulation conditions in the **OPTIONS** section (see [\[Chemistry Options\]](#), page [\(undefined\)](#)),

```

ITEM          OPTIONS

density       0.8
temperature   298
ntotal        1000
shape         1.5
build_dir     .
field         opls-ua
charge        false
profile       true
replace       true

ITEM          END

```

Here, we chose the OPLS-UA force field without charges, select a mass profile section to appear in the LAMMPS input script, and allow for **emc_setup.pl** output scripts to be overwritten. The definition of surfaces occurs through predefined EMC input files, of which a few examples can be found in `./lib`. One of these input files is a fully bonded diamond lattice called **diamond_lattice.emc**. EMC input files can contain variables, which can be redefined from an EMC input script. Specific variables need to be redefined, in order to turn the generic diamond lattice input file into a carbon diamond lattice. This redefinition occurs in the **VARIABLES** paragraph (see Section 4.6.2 [\[Chemistry File\]](#), page 35),

```

ITEM          VARIABLES

diamond_chemistry  "*C(*)(*)*"
diamond_name       "carbon"
diamond_atomistic  true

lbond              1.529

ITEM          END

```

Four variables are predefined in the diamond lattice input file: **diamond_chemistry**, **diamond_name**, **diamond_atomistic**, and **lbond**. The above definitions overwrite the

predefined values. The value for `lbond` has been taken from the OPLS-UA force field, where it represents the equilibrium bond distance between two `ct` types, which is 1.529 Å. With options and variables set, we now need to define the group describing decane,

```
ITEM      GROUPS

decane    (C)10

ITEM      END
```

Here, an extension of SMILES was used to depict decane. Proper SMILES would require `CCCCCCCCC`, but EMC allows for an extension to SMILES, for which a number after parenthesis is interpreted as the number of times, with which the snippet between the parenthesis is repeated. Finally, the `CLUSTERS` cluster section defines both decane and the surface,

```
ITEM      CLUSTERS

decane    decane      1
surface   import      4      $root+"lib/diamond_bonded"      emc

ITEM      END
```

Here, the fraction, 1, of decane molecules is defined, followed by the surface definition. Both lines start with the name of the cluster. This name is followed by a reference to the corresponding group for the decane molecules. The second keyword for the surface molecule is a reserved keyword, which indicates, that this molecule is a surface, which uses the library file `diamond_bonded` as a reference file. The different phases are built in the x-direction. The number 4 indicates the number of surface cells to build into this direction, which defines the value of internal variable `nx`.

4.7.3 Environment Mode

The environment mode allows creating a multi-simulation environment, conveniently packaged in one clear file. One example dealing with determining shear is included. Environment examples can be found in `${EMC_ROOT}/examples/setup/environment/`.

4.7.3.1 User-Defined Force Fields

This example demonstrates the concept of including a force field definition in a chemistry template, while adding loops in the environment section of the EMC Setup input script. The example can be found in `${EMC_ROOT}/examples/setup/environment/field`. EMC Setup provides the `FIELD` section for controlling user-defined force fields. The idea behind this example is to study the hypothetical influence of branching. To this extend, the number of branch points and the Lennard-Jones (LJ) interaction constant epsilon is varied. The work directory `${WORKDIR}` is defined by

```
${EMC_ROOT}/examples/setup/environment/field
```

later referred to by `${WORKDIR}`. The user is advised to maintain a standardized directory structure when using the environment mode of EMC Setup. This means, that EMC Setup environment .esh files are stored in `${WORKDIR}/setup`. Execution of the setup files should occur in `${WORKDIR}` and not in `${WORKDIR}/setup`. Once executed in `${WORKDIR}`, EMC Setup creates directories `analyze`, `build`, `chemistry`, `run`, and optionally `test` in `${WORKDIR}`. A typical convention for setup files is to name them by date, followed by a serial number, e.g. 2018070100 means, that the setup file was created on July 1, 2018 and is the first of the series of files created on that date.

Environment

All sections outside the `TEMPLATE` section define the environment and is considered to be the environment mode of EMC Setup. This includes the `ENVIRONMENT` and `LOOPS` sections as well as the `STAGE` designator. The `ENVIRONMENT` section defines the project name, queue settings, and high performance computing (HPC) cluster architecture parameters. Additionally, the name of a test directory can be defined by the option `name_testdir` (see the 'Execution' section below).

Loops

Loop variables are defined in the `LOOPS` section. Variables `stage` and `trial` are reserved keywords. The former indicates the stage of a project, e.g. one likes to study branched and linear structures. This could be divided in a `branched` stage and a `linear` stage. Here, only the `branched` variety is considered. Furthermore, the number of branches is indicated by `nbranches` and the settings for the LJ interaction constant by `epsilon`. These variables are later referred to in the `TEMPLATE` section by `@{NBRANCHES}` and `@{EPSILON}` respectively.

Template

The `TEMPLATE` section contains the definition of one simulation, here referred to as the chemistry mode of EMC Setup. In this chemistry template, one can use variables as defined in the `LOOPS` section of the environment. A template should at least contain an `OPTIONS`, `GROUPS`, and `CLUSTERS` section to fully define the simulation setup. Here, an additional `POLYMERS` section has been added to control the definition of the to be studied branched oligomers. Since polymers are requested, connectivity has to be defined in the `GROUPS` section. This connectivity is referred to with the `'*'` character. The numbers after the

chemical definition through a SMILES refer to the occurrence of the '*' character in the SMILES, e.g. the definition of group A

```
*a*,1,A:2,1,B:2,1,B:3,2,B:1
```

means, that the first '*' in '*a*' connects to the second of group A and the second and third of group B. The second star in '*a*' connects the first of group B.

The **CLUSTERS** section defines the molecule and the amount of each contributor, i.e. the solvent is defined by group S and has an 80% occurrence. The polymer is defined as a random copolymer and has a 20% occurrence.

The **POLYMERS** paragraph holds all definitions of polymers as referred to in the **CLUSTERS** paragraph. Here, only the polymer **polymer** exists with 10 repeat units of monomer A as defined by group A, and a variable amount of monomer B and terminator T. The amount is defined by @{NBRANCHES} which refers to **nbranches** in the **LOOPS** section.

The **FIELD** section defines the user-provided force fields and contains subsections related to the subsequent parameter definitions. Mandatory subsections are **MASS**, **NONBOND**, and – in case of connectivity – **BOND**. Additionally, in the case of this example angle terms are also defined in the **ANGLE** subsection. No torsions were needed. Wildcards in the form of '*' are allowed in all but the **MASS** section. In the case of the latter, each type can only contain one '*' character, which has to be at the end of the type. Note that the variable definition of the LJ epsilon is taken care of by @{EPSILON} in the **NONBOND** subsection.

Execution

Creation of the simulation environment occurs in \${WORKDIR} through

```
./setup/2018070100.esh
```

or, when the above is not executable or **emc_setup.pl** is not in your path,

```
${EMC_ROOT}/scripts/emc_setup.pl setup/2018070100.esh
```

Though the latter being possible, the user is advised to add both **bin** and **scripts** directory to their path, e.g.

```
export PATH=${EMC_ROOT}/bin:${EMC_ROOT}/scripts:${PATH}
```

Once executed, several bash scripts are generated in directories **analyze**, **build**, and **run** with name **2018070100.sh**. Subsequently, the provided test directory name appears as a subdirectory of \${WORKDIR}/test/, in which the script **setup.sh** is created. Execution of the latter creates an instance of the first occurrence of each loop variable, i.e. for **STAGE = branched**, **NBRANCHES = 2**, and **EPSILON = 0.9**. This test directory serves the purpose of testing the validity of the **TEMPLATE** section without having to submit to a queueing system.

Before submitting, an option which you would need to set is **queue_ppn** (procs or cores per node), since it is unknown how many cores can run on one of your nodes. EMC will

pack single core build jobs together on a node if it knows the ppn. Similarly, you can set the memory per core with `queue_memory`.

Submission to a queueing system occurs through `./build/2018070100.sh`, followed by `./run/2018070100.sh` once all builds as spawned by the former have finished. Chaining of builds and runs is also possible by executing the run script with build mode included, i.e.

```
./run/2018070100.sh -build
```

This spawns build and lets running of the systems LAMMPS wait for the building to finish.

4.7.3.2 Shear

The shear example, as found in `${EMC_ROOT}/examples/setup/environment/nemd`, illustrates the use of the EMC setup environment mode to define a set of simulations, calculating time dependent pressure tensors for a number of n-alkanes with various shear rates. The environment mode allows for scanning through parameter space, aiding the study of material response to various conditions. The example defines 40 simulations in total. An environment script contains multiple distinct sections. A shear simulation needs an equilibrated non-sheared configuration as a starting point. First, the environment mode needs to be triggered in the first occurring `OPTIONS` section,

```

ITEM      OPTIONS

project      pure
replace      true
environment   true
name_analyze date00
name_build   date00
name_run     date00
queue_build  default
queue_analyze default
shear        true, true
restart      fal
ncores       8
trun         -

ITEM      END

```

Besides toggling the environment mode, this section also defines the general project name, names associated with build, run, and analysis scripts, execution queues, and the number of cores to execute LAMMPS simulations on. Build, run, and analysis names use the base name of the chemistry script by as default. Zero shear simulations do not start from a restart file, but uses the output generated by EMC instead.

Loops can be defined over project stages, trials, and user defined variables in section `LOOPS`. This section controls parameter space for which phenomena are to be studied.

```

ITEM      LOOPS

stage      pure
trial      c-06   c-07   c-08   c-09   c-10   &
           c-11   c-12   c-13   c-14   c-16
shear      0.00e0
copy       1

ITEM      END

```

Reserved variables **stage**, **trial**, and **copy** are internally defined. Both **stage** and **trial** relate to other sections in the environment script. Each **stage** is associated with a specific chemistry file, which is defined in section **TEMPLATE** of the environment script. The definitions for variable **trial** are associated with section **STRUCTURES**. The reserved variable **copy** supports statistical sampling by creating independent structures (here, as an example, 1 structure is created). The use of the **copy** variable should be used with care, since it functions as a multiplier for the number of simulations, that are considered. In this example, the defined 11 trials and 4 shears comprise 44 simulations. This would increase to 440 simulations, when **copy** is set to 10 instead of 1.

The variable **shear** is a user defined variable. It is expected to be lower case. Its reference in the chemistry template is expected to be capitalized. Shell scripts are created after each **LOOPS** section. Build and run scripts – with names **date00.sh** and **date10.sh**, respectively, as defined in the previous **OPTIONS** section – are written in a newly created directory **run** in the current work directory. An analysis script with the name **date10.sh** is created in in a newly created directory **analyze**.

Subsequent shear simulations are defined after zero shear simulations. Here, only a few lines are needed. Non-defined parameters take the value of their definition in previous **OPTIONS** sections.

```

ITEM          OPTIONS

name_build    -
name_run      date10
name_analyze  date10
restart       true, ../../0.00e0

ITEM          END

```

These subsequent simulations will use equilibrated zero shear simulations as their starting point. A build script is not needed in this case, which is denoted by defining parameter **name_build** with a dash. Restart files are used as starting point, which is triggered by setting parameter **restart** and defining the source directory for the restart files, **../../0.00e0** in this case (the **0.00e0** value refers to the previously set zero shear).

Multiple shear rates are needed to see the shear behavior, intending to reach the Newtonian plateau at very low shear. Depending on the substance, this plateau might not be possible to be reached.

```

ITEM          LOOPS

shear         1.00e-4, 1.00e-5, 1.00e-6

ITEM          END

```

Shear rates are expressed in box length per time step. After the definition of this paragraph, a run shell script with name **2016103120.sh** is added to directory **run**, and an analysis shell

script with name `2016103120.sh` is added to directory `analyze`. Note, that the definition of `stage` and `trial` are inherited from the previous definition of section `LOOPS`.

With all shell scripts defined, now we can define each stage and trial separately by using sections `STAGE` and `TRIAL` respectively. After invoking `STAGE` and/or `trial`, all subsequent sections apply to the defined stage and/or trial. Here, we only need to define a section `STAGE` with name `pure`, relating back to the definition of variable `stage` in section `LOOPS`. Note, that both sections `STAGE` and `TRIAL` do not require an `ITEM END` designation.

```
ITEM          STAGE    pure
```

A template is used to define the actual simulation conditions. A template section is defined by `ITEM TEMPLATE` and needs an `ITEM END` to close. Variables appearing in section `LOOPS` can be referred to by starting with `@` followed by the variable name in all capitals. A simulation is defined as usual withing a template section.

OPLS-UA reflects the force field chosen for our n-alkanes. A total of 10000 beads is requested by setting `ntotal`. The equilibration time is set to 100000 time steps with `tequil`. The initial `density` is chosen to be 0.85 g/cc. The simulation is run at equal `pressure` of 1 atm and `temperature` of 300 K by using an NPT ensemble. Note, that it is possible, that `npt/sllod` is not available in all LAMMPS versions, in which case the pressure line should be omitted. Shear rates are set by the `shear` parameter. Here, `@SHEAR` refers back to variable `shear` in the `LOOPS` section. All existing scripts are overwritten by setting parameter `replace` to `true`. The usage of bond increments depend on the chosen force field. An empty bond increment is used when non-existing bond increments are encountered. Note, that OPLS-UA does not use bond increments.

```
ITEM          OPTIONS

replace          true
field            opls-ua
ntotal           10000
tequil           100000
density          0.85
pressure         1.0
temperature      300
shear            @SHEAR,erate
replace          true
increment        empty

ITEM            END
```

With simulation conditions set, substance definitions are in order. We chose to use a structure section to define chemical structures. As a consequence, the variable `structure` is defined and used to replace all occurring `@STRUCTURE` instances in the chemistry template.

```
ITEM          SHORTHAND
```

```
molecule           @STRUCTURE,1
```

```
ITEM                END
```

The template section is closed with an `ITEM END`. This template is used to define all simulations. It is stored with name `pure.esh` in a created directory `chemistry/stages` within the current work directory. Also, an adaptation shell script with name `pure.sh` is stored in created directory `chemistry/scripts`.

Finally, desired substances have to be defined. Doing so in section `STRUCTURES` suffices, since we are only interested in bulk properties of pure components.

```
ITEM                STRUCTURES

c-06                (C)6
c-07                (C)7
c-08                (C)8
c-09                (C)9
c-10                (C)10
c-11                (C)11
c-12                (C)12
c-13                (C)13
c-14                (C)14
c-15                (C)15
c-16                (C)16

ITEM                END
```

Each structure is stored in a separate file with names as occurring in the first column. These files are stored in directory `chemistry/structures`.

Execution of the whole simulation can be done on either the cluster on which the simulations are to run, or can be done from the machine on which the final results are to be analyzed. In the former case, each of the three scripts in directories `build`, `run`, and `analysis` are to be executed sequentially, i.e. `build/date00.sh` to build all simulations, subsequently followed by zero shear simulations through `run/date00.sh` when all build simulations are completed, and by finite shear simulations through `run/date10.sh` when all zero shear simulations are completed. Each script uses `run.sh` to submit jobs to the queueing system, which can be found in `${EMC_ROOT}/scripts`. The `run.sh` supports LSF and PBS queueing systems. The `run_host.sh` script can be found in the same directory allows remote submission of jobs. It assumes, that the execution directory structure is identical on both local and remote host machines. `run_host.sh` can be used to for instance execute the build shell script remotely through execution of

```
run_host.sh user@remote.machine.net build/date00.sh
```

in the current work directory. Log files are created both locally and remotely by `run_host.sh`, keeping records of simulation execution. The example above would create

`build/date00.log`. Remote execution of subsequent run scripts would follow each other sequentially upon completion of its predecessor.

4.8 Help Output

The following output appears when invoking `emc_setup.pl` either without any options, with a non-existing option, or with the `-help` option.

EMC Setup v4.1 (July 10, 2021), (c) 2004-2021 Pieter J. in 't Veld

Usage:

```
emc_setup.pl [-command[=#[,...]] project [phase 1 clusters [+ ...]]
```

Commands:

```
-analyze_archive  archive file names associated with analyzed data [true]
-analyze_data     create tar archive from exchange file list [true]
-analyze_last     include last trajectory frame (deprecated) [false]
-analyze_replace  replace already existing analysis results [true]
-analyze_skip     set the number of initial frames to skip [0]
-analyze_source   set data source directory for analysis scripts
-analyze_user     set directory for user analysis scripts
-analyze_window   set the number of frames in window average [1]
-angle           set DPD angle constants k and theta or set angle field
                 option (see below) [5,180]
-auto            add wildcard entry to mass and nonbond sections in DPD
                 .prm [false]
-binsize         set bin size for LAMMPS profiles [0.01]
-bond            set DPD bond constants k,l [25,1]
-build           set build script name [build]
-build_center    insert first site at the box center [false]
-build_dir       set build directory for LAMMPS script [../build]
-build_order     set build order of clusters [random]
-build_origin    set build order of clusters [x=0, y=0, z=0]
-build_replace   replace already existing build results [false]
-build_theta     set the minimal insertion angle [false]
-charge         chemistry contains charges [true]
-charge_cut      set charge interaction cut off [9.5]
-chunk          use chunk approach for profiles in LAMMPS script [true]
-communicate     use communicate keyword in LAMMPS script [false]
-core           set core diameter [-1]
-cross          include nonbond cross terms in LAMMPS params file [false]
-crystal        treat imported structure as a crystal [auto]
-cut            set pairwise interaction cut off [9.5]
-cutoff         set pairwise interaction cut off [center=-1, charge=9.5,
ghost=-1, inner=-1, outer=-1, pair=9.5, repulsive=0]
-debug          control debugging information [false]
-deform         deform system from given density [nblocks=1, ncycles=100,
type=relative, xx=1, yx=0, yy=1, zx=0, zy=0, zz=1]
-density        set simulation density in g/cc for each phase [1]
-dielectric      set charge medium dielectric constant [1]
-direction      set build direction of phases [x]
```

```

-dtdump          set LAMMPS trajectory file write frequency [100000]
-dtrestart       set LAMMPS restart file frequency [100000]
-dtthermo       set LAMMPS thermodynamic output frequency [1000]
-emc            create EMC build script [true]
-emc_depth      set ring recognition depth in groups paragraph [8]
-emc_exclude    set EMC section to exclude [build=false]
-emc_execute    execute EMC build script [false]
-emc_export     set EMC section to export [smiles=]
-emc_moves      set Monte Carlo moves for after build [displace=1]
-emc_output     set EMC output modes [debug=false, exit=true, info=true,
warning=true]

-emc_progress   set progress indicators [build=true, clusters=false]
-emc_run        set Monte Carlo run conditions for after build
                [clusters=all, groups=all, nblocks=100, ncycles=0,
nequil=0, sites=all]

-emc_test       test EMC build script [false]
-emc_traject    settings for EMC trajectory [append=true, frequency=0]
-environment    create project environment [false]
-ewald          set long range ewald summations [true]
-exclude        exclude previous phase during build process [true]
-expert         set expert mode to diminish error checking [false]
-extension     set environment script extension [.esh]
-extra          set extra interactions dest:src:offset (deprecated)
-field         set force field type and name based on root location
-field_angle    set angle field option (see below) [-]
-field_bond     set bond field option (see below) [-]
-field_charge   check system charge after applying force field [true]
-field_check    check force field compatibility [true]
-field_debug    set debug field option [false]
-field_dpd      set various DPD options [auto=false, bond=false]
-field_error    override field errors (used for debugging) [true]
-field_format   parameter format of generated force field [%15.10e]
-field_group    set group field option (see below) [-]
-field_id       set force field id [opls-ua]
-field_improper set improper field option (see below) [-]
-field_increment set increment field option (see below) [-]
-field_location set force field location [~/emc/v9.4.4/field]
-field_name     set force field name [opls/2012/opls-ua]
-field_nbonded  set number of excluded bonded interactions [0]
-field_reduced  set force field reduced units flag [false]
-field_torsion  set torsion field option (see below) [-]
-field_type     set force field type [opls]
-field_write    create field parameter file [true]
-focus         list of molecules to focus on [-]
-ghost_cut     set pairwise interaction cut off [-1]
-grace         (deprecated: use weight) set build relaxation grace
                [0.9999,0.9999,0]

```



```

-help                this message
-hexadecimal         set hexadecimal index output in PDB [false]
-host                set host on which to run EMC and LAMMPS [air]
-info                control runtime information [true]
-inner              set inner cut off [-1]
-insight             create InsightII CAR and MDF output [false]
-insight_compress    set InsightII CAR and MDF compression [false]
-insight_pbc         apply periodic boundary conditions [true]
-insight_unwrap      apply unwrapping [true]
-kappa               set electrostatics kappa [4]
-lammps              create LAMMPS input script or set LAMMPS version using
                    year, e.g. -lammps=2014 (new versions start at 2015)
                    [true]
-lammps_cutoff       generate output of pairwise cut off in parameter file
                    [false]
-lammps_dlimit       set LAMMPS nve/limit distance [0.1]
-lammps_pdamp        set LAMMPS barostat damping constant [1000]
-lammps_tdamp        set LAMMPS thermostat damping constant [100]
-location            prepend paths for various file locations
                    [analyze=~:/emc/v9.4.4/scripts/analyze,
                    field=~:/emc/v9.4.4/field, include=.]
-mass                assume mass fractions in chemistry file [false]
-memorypercore       set queue memory per core in gb [default]
-modules             swap modules to use in format old=new
-mol                 assume mol fractions in chemistry file [true]
-momentum            set zero total momentum in LAMMPS [100,1,1,1,angular]
-moves_cluster       define cluster move settings used to optimize build
                    [active=false, cut=0.05, frequency=1, limit=auto:auto,
                    max=0:0, min=auto:auto]
-msd                 set LAMMPS mean square displacement output [false]
-namd                create NAMD input script and parameter file [false]
-namd_dtcoulomb      set electrostatic interaction update frequency [1]
-namd_dtdcd          set output frequency of DCD file [10000]
-namd_dtnonbond      set nonbonded interaction update frequency [1]
-namd_dtrestart      set output frequency of restart files [100000]
-namd_dtthermo       set output frequency of thermodynamic quantities [1000]
-namd_dttiming       set timing frequency [10000]
-namd_dtupdate       set update frequency [20]
-namd_pres_decay     set pressure ensemble decay [50]
-namd_pres_period    set pressure ensemble period [100]
-namd_temp_damp      set temperature ensemble damping [3]
-namd_tminimize      set number of minimization timesteps [50000]
-namd_trun           set number of timesteps for running [10000000]
-name_analyze        set job analyze script name [chemistry]
-name_build          set job build script name [chemistry]
-name_run            set job run script name [chemistry]
-name_scripts        set analyze, job, and build script names simultaneously

```

```

-name_testdir      set job test directory as created in ./test/ [-]
-nchains           set number of chains for execution of LAMMPS jobs
-ncores            set number of cores for execution of LAMMPS jobs [-1]
-ncorespernode     set queue cores per node for packing jobs [default]
-niterations       set number of build insertion iterations [1000]
-norestart         control possibility of restarting when rerunning [false]
-nparallel         set number of surface parallel repeat unit cells [auto]
-nrelax            set number of build relaxation cycles [100]
-nsample           number of configuration in profile [1000]
-ntotal            set total number of atoms [10000]
-number            assume number of molecules in chemistry file [false]
-omit              omit fractions from chemistry file [false]
-options_perl       export options, comments, and default values in Perl
                    syntax [false]
-options_tcl        export options, comments, and default values in Tcl
                    syntax [false]
-outer             set outer cut off [-1]
-pair              set DPD pair constants [a=25, gamma=4.5, r=1]
-parameters        set parameters file name [parameters]
-params            create field parameter file [true]
-pdb               create PDB and PSF output [true]
-pdb_atom          set atom name behavior [index]
-pdb_compress      set PDB and PSF compression [true]
-pdb_connect       add connectivity information [false]
-pdb_cut           cut bonds spanning simulation box [false]
-pdb_extend        use extended format for PSF [false]
-pdb_fixed         do not unwrap fixed sites [true]
-pdb_parameters    generate NAMD parameter file [false]
-pdb_pbc           apply periodic boundary conditions [true]
-pdb_rank          apply rank evaluation for coarse-grained output [false]
-pdb_residue       set residue name behavior [index]
-pdb_rigid         do not unwrap rigid sites [true]
-pdb_segment       set segment name behavior [index]
-pdb_unwrap        apply unwrapping [clusters]
-pdb_vdw           add Van der Waals representation [false]
-percolate         import percolating InsightII structure [false]
-phases            sets which clusters to assign to each phase; each phase
                    is separated by a +-sign; default assigns all clusters to
                    phase 1 [all]
-polymer           default polymer settings for groups [bias=none,
                    fraction=number, niterations=-1, order=list]
-port              port EMC setup variables to other applications
-precision         set charge kspace precision [0.001]
-prefix            set project name as prefix to LAMMPS output files [false]
-pressure          set system pressure and invoke NPT ensemble; optionally
                    add direction and/or (un)couple for specifying
                    directional coupling [false,direction=x+y+z,couple]

```

```

-profile          set LAMMPS profile output [density=false,
                  density3d=false, pressure=false]
-project          set project name; slashes are used to create
                  subdirectories
-queue           queue settings [account=none, analyze=default,
                  build=default, memory=default, ncores=-1, ppn=default,
                  run=default, user=none]
-queue_account   set queue account for billing [none]
-queue_analyze   set job analyze script queue [default]
-queue_build     set job build script queue [default]
-queue_memory    set queue memory per core in gb [default]
-queue_ppn       set queue cores per node for packing jobs [default]
-queue_run       set job run script queue [default]
-queue_user      options to be passed directly to queuing system [none]
-quiet          turn off all information
-radius         set build relaxation radius [5]
-record         set record entry in build paragraph [cut=false,
                  frequency=1, inactive=true, name="", pbc=true,
                  unwrap=sites]
-references      set references file name [references]
-region_epsilon  set epsilon to use for exclusion regions [0.1]
-region_sigma    set sigma to use for exclusion regions [1]
-replace        replace all written scripts as produced by EMC setup
                  [false]
-replica        set replica interactions dest:src:offset
-restart        create LAMMPS restart script [false,...]
-rlength        set reference length
-rmass          set reference mass
-rtype          set reference type
-sample         set sampling options for LAMMPS input script
                  [energy=false, green-kubo=false, msd=false,
                  pressure=true, volume=false]
-script         set script file name [chemistry]
-script_ncolumns set number of columns in output scripts [80]
-seed           set initial random seed [-1]
-shake          set shake types
-shake_iterations set maximum number of shake iterations [20]
-shake_output    set shake output frequency [never]
-shake_tolerance set shake tolerance [0.0001]
-shape          set shape factor [1]
-shear          add shear paragraph to LAMMPS input script [false]
-skin           set LAMMPS skin [2]
-split          sets which clusters to partition; each split is separated
                  by a +-sign; default assigns no clusters to split
                  [clusters=all, fraction=0.5, groups=all, mode=random,
                  phase=1, sites=all, thickness=1, type=relative]
-suffix         set EMC and LAMMPS suffix [_air]

```

```

-system                system identification and checks during building
                        [charge=true, geometry=true, id=main, map=true, pbc=true]
-system_charge         check for charge neutrality after build [true]
-system_geometry       check geometry sizing upon building [true]
-system_id             check for charge neutrality after build [main]
-system_map            map system box before build [true]
-system_pbc            apply periodic boundary conditions after build [true]
-temperature           set simulation temperature in K [300]
-tequil               set LAMMPS equilibration time [1000]
-tfreq                set LAMMPS profile sampling frequency [10]
-thermo_multi          set LAMMPS thermo style to multi [false]
-tighten              set tightening of simulation box for imported structures
                        [3]
-time_analyze          set job analyze script wall time [00:30:00]
-time_build            set job build script wall time [00:10:00]
-time_run              set job run script wall time [24:00:00]
-timestep              set integration time step [2]
-triclinic             set LAMMPS triclinic mode
-trun                 set LAMMPS run time [10000000]
-units                 set units type [real]
-units_energy          set units for energetic scale [1]
-units_length          set units for length scale [1]
-version               output version information [false]
-volume                set recalculation based on molecular volume [false]
-warn                  control warning information [true]
-weight                set build relaxation energetic weights [bond=0.0001,
                        focus=1, nonbond=0.0001]
-width                 use double width in scripts [false]
-workdir                set work directory [~/emc/v9.4.4/texinfo/setup]

```

Notes:

- * This script comes with no warrenty of any kind. It is distributed under the same terms as EMC, which are described in the LICENSE file included in the EMC distribution.
- * Queue name 'default' refers to whichever queue is default; queue name 'local' executes all jobs sequentially on local machine
- * Reference and parameter file names are assumed to have .csv extensions
- * Chemistry and environment file names are assumed to have .esh extensions
- * File names with suffixes _chem can be taken as chemistry file names wild cards
- * Chemistry file format: mol id, smiles string, fraction, mol mass[, mol vol]
- * Reserved environment loop variables are: stage, trial, and copy
- * Densities for multiple phases are separated by commas
- * Shears are defined in terms of erate; values < 0 turns shear off
- * Inner and outer cut offs are interpreted as fractions for colloidal force fields

- * Valid field options are: ignore, complete, warn, empty, or error
- * A '+' sign demarcates clusters of each phase

5 Scripting Commands

EMC is driven through a scripting language that accesses the different predefined routines driving building and simulation of configurations. EMC requires each script to start with the text "`(* EMC: Script *)`" on the first line. Absence of this line will result in an error upon start up of EMC. The following sections describe the syntax of commands in the current version. Currently, commands use a variable representation similar to Mathematica. A change to a function representation is planned in future.

5.1 Build

5.1.1 Syntax

```

build      = {
  system    -> {id -> constant, temperature -> real,
               density -> real, geometry -> voigt,
               flag -> {charge -> boolean, map -> boolean}},
  ...
  select    -> {progress -> boolean, frequency -> long,
               message -> option, center -> boolean, origin -> vector,
               order -> constant, check -> boolean,
               cluster -> {constant, ...}, name -> string,
               relax -> {ncycles -> long, radius -> real},
               grow -> {method -> constant, check -> constant,
                       cutoff -> real, grace -> {real, real},
                       theta -> real, dphi -> real, nbonded -> long,
                       ntrials -> long, niterations -> long,
                       include -> {region, ...},
                       exclude -> {region, ...}}}
  ...
};

```

Directive	Parameters	Description
system	struct	Defines the system dimensions associated with a build.
id	constant	Sets identity of system to which to add the build; can be either textual or numerical.
density	real	Sets the system density; can either be set to <code>calculate</code> when choosing or forced when choosing values > 0.0 .
geometry	voigt	Sets the system geometry; used as a fractional size when density is set, otherwise used as system dimensions from which a density is calculated (see Section 7.15 [Voigt], page 335).
flag	boolean	Control system checks and optimizations.
charge	boolean	Perform a total charge check of the system and exit on a non-zero value; options are <code>true</code> or <code>false</code> .
map	boolean	Map a stretched triclinic cell geometry into its minimal representation; options are <code>true</code> or <code>false</code> .

Directive	Parameters	Description
select	struct	Selects building style and clusters
progress	boolean	Shows progress indicator; options are <code>bar</code> , <code>list</code> , or <code>none</code> .
frequency	long	Sets the progress frequency (in percentages), where $0 \leq \text{frequency} \leq 100$.

message	option	Sets the style for energetic output in progress report; valid options are none , raw , nkt , or n , which corresponds to energy in no particular units, internal units, units of nkT or units of n, respectively; internal units correspond to the units of the selected force field (kcal/mol for COMPASS, OPLS, GROMACS and MARTINI).
center	boolean	Place first inserted site at origin as defined origin option; options are true or false .
origin	{0,0,0}	Define origin for first inserted site; used only when center option is set to true .
order	constant	Insert clusters in sequence or at random .
check	boolean	Check for (semi)fixed sites and abort if present; options are true or false .
cluster	array	Sets the cluster(s) participating in the build.
name	string	Sets the output name for generation of an XYZ file upon error of all active sites of the built system; no output is generated, when the output name is not set.
relax	struct	Selects intermediate structure relaxation during the building process; Monte Carlo is performed on a spherical volume with a radius around the inserted site for ncycles after each successful insertion.

Directive	Parameters	Description
grow	struct	Describes growth method and settings.
method	option	Sets building style; options are overlap for hard sphere excluded volume considerations, energetic for energetic considerations, and minimum for energetic minimum considerations.
check	option	Sets overlap checking; options are bonded for checking overlaps up to four bonds away, cluster for checking intramolecular, all for intra- and intermolecular overlaps, and none for no overlap checking.
cutoff	real	Sets the energetic cutoff when using method -> minimum , where decisions are based on boltzmann sampling when cutoff <= 0.
grace	{real, real}	Sets the grace fraction, which correlates to how much overlap is allowed; 0 corresponds to no overlap, while 1 corresponds to full overlap (typically < 0.75); the first value influences the nonbonded acceptance, while the second value influences the bonded acceptance; an energetical building style requires grace < 0.9995 for coherent energies.
theta	real	Sets the minimum allowed bond angle.

dphi	real	Sets the maximum allowed torsion angle for for bonded construction using a Monte Carlo scheme; needed for all force field definitions beyond bond length interactions (i.e. it is neglected in case of the existence of bonds only or when a lower threshold is crossed).
nbonded	integer	Sets the number of trials moves for bonded construction using a Monte Carlo scheme.
ntrials	integer	Sets the number of trial overlap or energetic calculations.
niterations	integer	Sets the number of iterative trials.
include	region	Sets inclusion regions (see Section 7.7 [Region], page 182).
exclude	region	Sets exclusion regions (see Section 7.7 [Region], page 182).

5.1.2 Usage

The **build** command enables the build of previously defined clusters (see Section 5.4 [Clusters], page 89) into a desired system. All building modes apply energetic sampling of bonded potentials during the bond building process. Currently a hard sphere **overlap** and an **energetic** build is available. Note that a system needs to be defined before build mode can be executed (e.g. **define system** before **select**). Note that **hard** and **soft** modes regulate insertion behavior with respect to exclusion regions. The **hard** mode applies the region to all sites of inserted clusters, while the **soft** mode only applies the region to the first site of inserted clusters.

5.1.3 Default

The default is given by

```
build      = {
  system    -> {id -> 0, density -> calculate, geometry -> {1, 1, 1},
               flag -> {charge -> true, map -> true}},
  select    -> {progress -> bar, frequency -> 5, message -> nkt,
               center -> false, order -> random, check -> true,
               relax -> {
                 ncycles -> 0},
               grow -> {
                 method -> energetic, check -> all,
                 grace -> {0.99, 0}, theta -> 0.0,
                 nbonded -> 20, ntrials -> 20, niterations -> 1000}};
```

5.1.4 Examples

An example is given by

```
build      = {
  system    -> {
    id       -> main,
    density   -> 0.7,
    temperature -> 1
  },
  select    -> {
```

```
    grace      -> 0.5,  
    method     -> overlap,  
    theta      -> 0.3*pi,  
    cluster    -> {polymer, solvent}  
  }  
};
```

5.2 Cancel

5.2.1 Syntax

```
cancel          = {style -> option};
```

Directive	Parameters	Description
style	option	Selects cancellation style; options are abort and exit .

5.2.2 Usage

The **cancel** command is used to stop a running script at the selected position. Its main application serves debugging purposes. Script debugging can use either **abort** or **exit**, while source code debugging would use **abort**.

5.2.3 Default

The default is given by

```
cancel          = {style -> exit};
```

5.3 Carve

5.3.1 Syntax

```
carve          = {
  mode          -> option,
  fraction       -> {real, ...},
  nsites        -> {integer, ...},
  systems       -> {constant, ...},
  inside        -> region,
  outside       -> region
};
```

Directive	Parameters	Description
mode	option	Sets mode of operation; options are <code>include</code> and <code>exclude</code> .
fraction	real	Sets the fractions in each system of the groups to be carved out; the value of fraction lies between 0.0 and 1.0; chooses <code>nsites</code> when <code>fraction < 0.0</code> .
nsites	integer	Sets the number of sites remaining in the selected region in each system; number of sites in group sites increments.
systems	constant	Sets target systems; can be one or more systems.
inside	region	Sets the inner boundary of the region in which sites are to be deleted (see Section 7.7 [Region], page 182).
outside	region	Sets the outer boundary of the region in which sites are to be deleted (see Section 7.7 [Region], page 182).

5.3.2 Usage

The `carve` command carves a shape out all systems in the currently defined simulation. It will output the number of retained sites per selected system upon execution as info using the `nsites` directive when a fraction for the selected systems is defined. It will use `nsites` when the fraction for the selected systems is a number smaller than zero.

5.3.3 Default

The default is given by

```
carve          = {
  systems       -> 0,
  nsites        -> 0,
  mode          -> include,
  inside        -> {shape -> spheroid, type -> relative,
                    center -> {0,0,0}, radius -> {0,0,0}},
  outside       -> {shape -> spheroid, type -> relative,
                    center -> {0,0,0}, radius -> {0,0,0}}
};
```

5.3.4 Examples

An example is given by

```
variables      = {n -> 20, r1 -> 1/2, r2 -> 1/2-0.75/(2^(2/3)*n)};
carve          = {mode -> exclude,
  inside       -> {shape -> spheroid,
    center -> {0, 0, 0}, radius -> {r2, r2, r2}},
  outside      -> {shape -> spheroid,
    center -> {0, 0, 0}, radius -> {r1, r1, r1}}
};
```

5.4 Clusters

5.4.1 Syntax

```
clusters      = {
  cluster      -> {id -> constant, system -> constant, n -> integer,
                  [group -> constant | template -> constant]},
  ...
  polymer      -> {id -> constant, system -> constant, n -> integer,
                  type -> option, groups -> {constant, ...},
                  nrepeat -> {integer, ...}},
  ...
  graft        -> {core -> constant, type -> constant, polymer -> constant,
                  connect -> option},
  ...
  body         -> {id -> constant, system -> constant, n -> integer,
                  density -> real, [surface -> constant,]
                  core -> constant, region -> region},
  ...
};
```

Directive	Parameters	Description
cluster	struct	Defines multiple single group clusters.
polymer	struct	Defines multiple repeating group clusters.
graft	struct	Defines graft connection to a backbone cluster group.
body	struct	Defines multiple bodies constrained by a region; particles consist either one core with multiple surface sites or core sites only.
id	constant	Sets identifier, either textual or numerical.
system	constant	Sets destination system to construct in,
n	integer	Sets the number of molecules to create.
group	constant	Sets the contributing group; must be a group without branches for cluster.
template	constant	Sets the contributing template as predefined in force field files (see Section 5.10 [Field], page 101).
nrepeat	integer	Sets the number of repeat units of each contributing monomer in the polymer; one nrepeat can be given in case of multiple groups, which results in assignment of this one nrepeat to all contributing groups.
groups	constant	Sets contributing groups; must be branched and connected groups; groups will be terminated with their respective terminators as defined by the groups command (see Section 5.16 [Groups], page 112).

type	option	Sets the type of polymer; options are block for block copolymers, alternate for alternating copolymers, and random statistical copolymers.
core	constant	Sets the core cluster; must be a cluster without branches and only one contributing site; density describes core density when no surface groups are defined.
type	constant	Sets the grafting type from which to grow branch; only one grafting type per grafting core is allowed.
connect	option	Sets the growth starting point; options are head or tail .
surface	constant	Sets the surface group; must be a group without branches and only one contributing site; one core particle is assumed; density describes surface density.
region	region	Sets the region of the particle (see Section 7.7 [Region], page 182).

5.4.2 Usage

The **clusters** command creates clusters according to the desired schemes as outlined above, which populate the desired system as set by the **system** directive.

5.4.3 Default

The default is given by

```
clusters      = {
  cluster     -> {id -> 0, system -> 0, n -> 0},
  polymer     -> {id -> 0, system -> 0, n -> 0},
  body        -> {id -> 0, system -> 0, n -> 0}
};
```

5.4.4 Examples

Direct definition of molecules uses the keyword **cluster**, e.g. a configuration with 1000 water molecules is obtained through

```
groups        = {
  group        -> {id -> water, chemistry -> "O"}
};
clusters      = {
  cluster      -> {id -> water, system -> 0, n -> 1000, group -> water}
};
```

Polymers can either display a block, alternating, or random constellation of its constituting monomers. Assume three monomers, combined with one terminator,

```
groups        = {
  group        -> {
    id          -> A,
    chemistry   -> "*A*",
    connects    -> {
```

```

    {head, {A, tail}}, {head, {B, tail}}, {head, {C, tail}},
    {tail, {A, head}}, {tail, {B, head}}, {tail, {C, head}},
    {head, {T, head}}, {tail, {T, head}}
  }
},
group      -> {
  id        -> B,
  chemistry -> "*B*",
  connects  -> {
    {head, {A, tail}}, {head, {B, tail}}, {head, {C, tail}},
    {tail, {A, head}}, {tail, {B, head}}, {tail, {C, head}},
    {head, {T, head}}, {tail, {T, head}}
  }
},
group      -> {
  id        -> C,
  chemistry -> "*C*",
  connects  -> {
    {head, {A, tail}}, {head, {B, tail}}, {head, {C, tail}},
    {tail, {A, head}}, {tail, {B, head}}, {tail, {C, head}},
    {head, {T, head}}, {tail, {T, head}}
  }
},
group      -> {
  id        -> T,
  chemistry -> "*T*",
  connects  -> {
    {head, {A, head}}, {tail, {B, head}}, {tail, {C, head}},
    {head, {A, tail}}, {head, {B, tail}}, {head, {C, tail}}
  }
}
};

```

One hundred homopolymer molecules consisting of 10 A monomers are defined by

```

clusters = {
  polymer -> {id -> polymer, system -> main,
              n -> 100, nrepeat -> 10, type -> random, groups -> A}
};

```

which generates 100 clusters with TAAAAAAAAAAT as a resulting polymer sequence. Alternatively, these monomers can be combined into three types of copolymers, i.e. random, block, and alternating copolymers. A random copolymer consisting of 10 A monomers, 20 B monomers, and 30 C monomers, is defined by

```

clusters = {
  polymer -> {id -> polymer, system -> main,
              n -> 100, nrepeat -> {10, 20, 30}, type -> random,
              groups -> {A, B, C}}
};

```


in which all monomers are randomly distributed over the polymer backbone. Construction of a block copolymer consisting of 2 A, 3 B, and 4 C monomers, is obtained through

```
clusters      = {  
  polymer     -> {id -> polymer, system -> main,  
                  n -> 100, nrepeat -> {2, 3, 4}, type -> block,  
                  groups -> {A, B, C}}  
};
```

which generates TAABBBCCCCCT as a resulting polymer sequence. Finally, a alternating copolymer of 2 A, 3 B, and 4 C monomers is constructed through

```
clusters      = {  
  polymer     -> {id -> polymer, system -> main,  
                  n -> 100, nrepeat -> {2, 3, 4}, type -> alternate,  
                  groups -> {A, B, C}}  
};
```

which generates TABCABCBCCT as a resulting polymer sequence. Note the effect as a result of the unequal numbers when defining monomers.

5.5 Crystal

5.5.1 Syntax

```
crystal          = {n -> {x, y, z}, miller -> {h, k, l}, plane -> option,
                    periodic -> vector};
```

Directive	Parameters	Description
n	{x, y, z}	Sets the number of crystal repeat units in x, y, z integer increments.
miller	{h, k, l}	Sets miller crystal plane indices specifying the desired tilt, where h, k, and l are integers.
plane	option	Specifies crystalline base plane; this plane is the plane which creates a crystalline cut; options are a, b, and c, or x, y, and z, or m100, m010, and m001 respectively.
periodic	vector	Specifies whether to leave or cut connections through periodic boundaries, where vector elements x, y, and z are booleans, for which true leaves and false cuts connections in the specified direction.

5.5.2 Usage

The **crystal** command is used to build crystal copies of crystalline unit cells. These unit cells can either contain unconnected or connected molecules. Miller indices in combination with a base plane definition creates the possibility to rotate the final cell to any desired orientation using integer indices h, k, and l. These indices are defined using the usual conventions. Rotation allows for cleaving a crystal by a desired plane, e.g. **miller -> {2, 0, 1}** in combination with **plane -> m001** uses the plane defined by crystal base vectors \vec{a} and \vec{b} and rotates the cell to satisfy the miller index vector {2, 0, 1}. Box dimensions are altered to form a new unit cell which consequently can be used again as a crystalline building block. The **periodic** option allows for cutting connections (bonds) through periodic boundaries, which enables the creation of a free structure when using the **carve** command (see Section 5.3 [Carve], page 87). All vector elements are set to the given input when only one element is set, i.e. **periodic -> true** equals **periodic -> {true, true, true}**.

5.5.3 Default

The default is given by

```
crystal          = {
  plane          -> m100,
  n              -> {1, 1, 1},
  miller         -> {1, 0, 0},
  periodic       -> {false, false, false}
};
```

5.5.4 Examples

A few examples are

```
crystal          = {
```

```
plane      -> m001,  
n          -> {5, 4, 9},  
miller     -> {0, 0, 1},  
periodic   -> true  
};  
crystal    = {  
  n        -> {n, n, n},  
  miller   -> {0, 0, 1},  
  plane    -> m001,  
  periodic -> {true, true, false}  
};
```

5.6 Cut

5.6.1 Syntax

```
cut                = {n -> integer, mode -> option, nmin -> integer,
                      region -> region};
```

Directive	Parameters	Description
n	integer	Defines the number clusters to be cut in each active system.
mode	option	Selects cutting mode; currently the only option is random .
nmin	integer	Specifies the minimum allowable number of mobile repeat units that remain.
region	region	Specifies in which region in the simulation cell to cut (see Section 7.7 [Region], page 182).

5.6.2 Usage

The **cut** command finds its main application in cutting a set of crystalline clusters during the preparation of semi-crystalline structures. It is, however, not restricted to crystalline structures alone. The **cut** command cuts exactly one bond on the specified number of clusters that fall within the specified region. The cutting mode governs the choice of affected clusters.

5.6.3 Default

The default is given by

```
cut                = {
  n                -> 0,
  nmin             -> 0,
  mode             -> random,
  region           -> {shape -> spheroid, type -> relative,
                      center -> {0,0,0}, radius -> {0,0,0}}
};
```

5.6.4 Examples

An example is given by

```
cut                = {
  n                -> 20,
  nmin             -> 3,
  region           -> {shape -> cuboid, radius -> {1/2,1/2,1/4}}
};
```

5.7 Deform

5.7.1 Syntax

```
deform          = {mode -> constant, type -> constant, system -> constant,
                  frequency -> long, density -> real, geometry -> voigt};
```

Directive	Parameters	Description
mode	constant	Sets the deform mode; options are either none or affine for the site deformation mode.
type	constant	Sets the type of geometry; options are either relative for reduced units or absolute for absolute units.
system	constant	Sets identity of system to be deformed; can be either textual or numerical.
frequency	long	Sets the frequency with which to perform incremental deformations; can either be set to instant , forcing instantaneous deformation, or to a positive integer, expressing the step size (in system cycles) in which deformation is gradually applied over the course of a run.
density	real	Sets the final system density; can either be set to calculate when choosing or forced when choosing values > 0.0.
geometry	voigt	Sets the final system geometry; used as a fractional size when density is set, otherwise used as system dimensions from which a density is calculated (see Section 7.15 [Voigt], page 335).

5.7.2 Usage

The **deform** command is used to gradually deform a selected **system** to the desired target **density** over the course of a **run** (see Section 5.28 [Run], page 134). Mobile sites are affinely deformed, rigid sites are deformed with respect of the center of mass of the rigid object, and fixed sites stay undeformed.

5.7.3 Default

The default is given by

```
deform          = {mode -> none, system -> 0, frequency -> 1,
                  density -> calculate, geometry -> {1, 1, 1}};
```

5.7.4 Examples

Examples are given by

```
deform          = {system -> main, frequency -> 2, density -> 0.7};
deform          = {system -> main, frequency -> 2, geometry -> {1, 1, 3}};
```

5.8 Duplicate

5.8.1 Syntax

```
duplicate      = {source -> system, destination -> {start, end}};
```

Directive	Parameters	Description
source	system	Defines the source system.
destination	{start, end}	Defines the destination systems.

5.8.2 Usage

Take source system **system** and copy it to destination starting at **start** and ending at **end**. Destination systems overlapping with the source system is not allowed.

5.9 Export

5.9.1 Syntax

```
export          = {
  profile       -> {name -> string, type -> option, style -> option},
  ...
  sample       -> {name -> string, type -> option, style -> option},
  ...
};
```

Directive	Parameters	Description
profile	struct	Export settings for profiles.
sample	struct	Export settings for sampled data.
name	string	Defines the file name containing exported data.
type	option	Indicates the type of <code>profile</code> or <code>sample</code> to export; available profiles are <code>angle</code> , <code>bond</code> , <code>bridge</code> , <code>density</code> , <code>force</code> , <code>loop</code> , <code>mass</code> , <code>order</code> , and <code>tail</code> ; the available sample is <code>cavity</code> .
style	option	Export format; options are <code>csv</code> , <code>json</code> , and <code>math</code> for comma separated value, JSON, and Mathematica formats respectively.

5.9.2 Usage

The `export` command is used to export data resulting from structure analysis into individual files, using `mathematica` or comma separated value formats. Only distributions resulting from analyses are exported.

5.9.3 Default

The `export` command does not have any defaults.

5.9.4 Examples

Examples of the `export` command are

```
export          = {
  profile       -> {name -> "density", type -> density, style -> math},
  profile       -> {name -> "bond", type -> bond, style -> csv},
  sample       -> {name -> "cavity", type -> cavity, style -> math}
};
```

5.9.5 Data Interpretation

export a distribution by using the `export` command, e.g.

```
export          = {name -> "bridge", type -> bridge, style -> csv};
```

which exports the measured tail distribution to file name `bridge.csv` in comma separated format. An alternate style is `math`, which exports to `bridge.m` in Mathematica format. EMC output format is equivalent to Mathematica format. I typically use the latter, since it allows me to create notebooks for evaluation, which can be used within workflows.

For example, the bridge distribution under the profiles header in the output file contains of a header, followed by a distributions block:

```
profiles = {
  bridge -> {
    active -> true, convolute -> false, direction -> 0,
    nlevels -> 1, binsize -> 1, ndistributions -> 1, distributions -> {
      ncalls -> 101, offset -> 7, nbins -> 645, data -> {
        {n -> 101, accu -> 101, weight -> 101},
        ...
      }
    }
  }
};
```

The header contains general distribution information, i.e. if the distribution is actively sampled, whether to convolute the data (when applicable), in which direction was sampled, how many levels it has, the binsize, and the number of distributions (one for each system). Levels are used to allow for 3D density sampling.

The distributions also contain a header section, which is followed by a data section. The header records for each distribution how many calls a distribution has, what offset the data is recorded at and how many bins the distribution contains. One can construct the x coordinates by combination of the offset with the bin size and the number of bins, i.e.

$$x_i = (i + offset)binsize, \text{ with } 0 \leq i < nbins.$$

The data section entries contain the y coordinates in three contributions: the frequency of calls to this bin, the accumulated scalar data, and the weight with which this data data was recorded. These three numbers are always the same for histograms (e.g. bridge, loop and tail distributions) and represent the frequency of a certain length or size, i.e.

$$y_i = n$$

For order profiles and other slab-wise averages, the accumulated data needs to be divided by their corresponding weight in order to get a local density or order parameter respectively, i.e.

$$y_i = accu/weight$$

For density profiles, the accumulated data needs to be divided by the number of distribution calls (*ncalls*) in order to get the local density, i.e.

$$y_i = \textit{accu}/\textit{ncalls}$$

Interpretation of cavity distributions is identical as for any histogram distribution. See the above description for brigde distributions.

5.10 Field

5.10.1 Syntax

```
field          = {
  id           -> identifier,
  mode         -> option,
  name         -> {string, ...},
  compress     -> boolean,
  apply        -> boolean,
  error        -> boolean,
  debug        -> boolean,
  angle        -> option,
  torsion      -> option,
  improper      -> option,
  increment    -> option,
  allow        -> option
};
```

Directive	Parameters	Description
<code>id</code>	<code>constant</code>	Sets the subfield identifier, e.g. used in groups to identify the field relevant for typing.
<code>mode</code>	<code>option</code>	Sets execution, input, output, or import mode; EMC standard options are <code>apply</code> , <code>get</code> , and <code>put</code> ; options <code>cff</code> , <code>charmm</code> , <code>dpd</code> , <code>martini</code> , and <code>opls</code> import native and non-native force field formats.
<code>name</code>	<code>string</code>	Sets file names of either input or output; all import options expect a general parameter file followed by an optional topology file.
<code>compress</code>	<code>boolean</code>	Sets compression flag; options are <code>true</code> or <code>false</code> .
<code>apply</code>	<code>boolean</code>	Apply force field typing directly after reading; options are <code>true</code> or <code>false</code> .
<code>error</code>	<code>boolean</code>	Sets error flag; options are <code>true</code> or <code>false</code> ; exits on error when true.
<code>debug</code>	<code>boolean</code>	Set debug flag, which controls the output of rule interpretation debugging information; options are <code>true</code> or <code>false</code> .
<code>angle</code>	<code>option</code>	Sets angle field flag; options are <code>ignore</code> , <code>complete</code> , <code>warn</code> , <code>empty</code> , or <code>error</code> (see see Section 7.2 [Field Flags], page 175).
<code>torsion</code>	<code>option</code>	Sets torsion field flag; options are <code>ignore</code> , <code>complete</code> , <code>warn</code> , <code>empty</code> , or <code>error</code> (see see Section 7.2 [Field Flags], page 175).
<code>improper</code>	<code>option</code>	Set improper field flag; options are <code>ignore</code> , <code>complete</code> , <code>warn</code> , <code>empty</code> , or <code>error</code> (see see Section 7.2 [Field Flags], page 175).

increment	option	Set increment field flag; options are ignore , complete , warn , empty , or error (see Section 7.2 [Field Flags], page 175).
allow	option	Sets which selection is considered during typing; options are all or template .

5.10.2 Usage

The **field** command allows for force field typing. A native format governs storage through **get** and **put** modes. The **apply** mode executes force field typing and should be invoked after **clusters** assignment only. Currently **field** supports the Accelrys BIOSYM force field format (**mode** -> **cff**) for importing force field typing rules (**.dat** files) and parameter assignment tables (**.frc** files). CHARMM, DPD, MARTINI, and OPLS force fields (**mode** -> **charmm**, **dpd**, **martini**, or **opls** respectively) are imported using an EMC native textual format (**.prm** and **.top** files). A conversion script for CHARMM force fields is given by **./scripts/charmm.pl**, for MARTINI force fields by **./scripts/martini.pl**, and for OPLS force fields by **./scripts/opls.pl**. Force fields can contain templates, which are used to define groups (see Section 5.16 [Groups], page 112) and clusters (see Section 5.4 [Clusters], page 89). Force fields are found in **./field** and subsequent directories. Note, that each force field only includes chemistries, for which it was parameterized. This means, that not all chemistries can be typed by all force fields. Also be aware, that typing is not guaranteed to be correct. It is kindly requested to report bugs, so that improvements to force fields can be included in next versions.

Internally, force fields consist of a set of general and specific parameters. General parameters contain automatic parameters, which are used when specific parameters are not defined. Aside from parameters, templates can be defined for commonly used chemistries. Martini force fields use templates to preset the parametrized substances. Note, that the abovementioned Martini script can add types when doubles occur. Here, the use of templates is advised to avoid the selection of wrong types.

5.10.3 Formats

Modes **get** and **put** support an EMC-native textual format, which encompasses most possibilities, as needed by supported force fields. Support of leaner textual formats is provided as short hand for ease of use.

The CFF file format is identical to the **.frc** and **.dat** as provided by Accelrys' Materials Studio. It has been tested for PCFF and COMPASS formats.

The CHARMM, DPD, MARTINI, and OPLS file formats are created for EMC and function as intermediates between CHARMM, DPD, GROMACS, or OPLS force field families and EMC. Data blocks in the EMC **.prm** format are flanked by the keywords **ITEM** and **ITEM END**. The keyword **ITEM** is followed by an identifier to designate the following data block. This data block describes data in tab-separated columns. Note, that only exactly one tab is allowed between entries. Spaces and multiple tabs lead to erroneous behavior. The interpretation of the data depends on the **ITEM** identifier as described in the following table:

ITEM ID	Parameters	Description
DEFINE	See next table	Contains force field definitions and units.

MASS	type mass element nconnections charge comment	Defines masses; columns contain a textual type, a numerical mass value, a textual element, a numerical number of connections, a numerical charge, and a textual comment line respectively.
RULES	id type element {index residue atom} charge rule	Defines typing rules; columns contain a numerical id, a textual type, a textual element, a numerical index (e.g. OPLS index), a numerical charge, and a rule in SMILES format respectively; in case of the CHARMM force field, the single numerical index entry is replaced by two entries for residue and atom, respectively.
COMMENTS	id index element comment	Defines comments associated with rules; columns contain a numerical id, a numerical index (e.g. OPLS index), a textual element, and a textual comment respectively.
LITERATURE	year volume page journal	Lists literature references associated with force field; columns contain a numerical year, a textual cvolume, a textual page, and a textual journal reference respectively.
EQUIVALENCE	type pair bond angle torsion improper	Tabulates equivalences between types (used to reference a derived type to its original; pair types are considered unique); all columns are in a textual format.
TEMPLATE	name smiles	Lists predefined group and cluster templates; all columns are in a textual format; SMILES format follows the Daylight Chemical Information Systems definition.
PRECEDENCE		Precedences describe the order in which rules are interpreted; parenthesis indicate precedence hierarchy: (? (c (c1)(c2))) c followed by c1 and c2.
NONBOND	type1 type2 sigma epsilon	Defines the interaction parameters for Lennard-Jones 6-12 pairwise interactions; columns contain two textual types followed by two numerical values.
BOND, BOND_AUTO	type1 type2 k l0	Defines covalent bond length interaction parameters for a harmonic spring; columns contain two textual types followed by two numerical values.
ANGLE, ANGLE_AUTO	type1 type2 type3 k theta0	Defines bond angle interaction parameters for a harmonic spring; columns contain three textual types followed by two numerical values.
TORSION, TORSION_ AUTO	type1 type2 type3 type4 k n delta [...] [index]	Defines bond torsion interaction parameters for a fourier expansion; columns contain four textual types followed by up to four groups of three numerical values; the last column can contain an optional OPLS reference index.
IMPROPER, IMPROPER_ AUTO	type1 type2 type3 type4 k psi0	Defines improper interaction parameters; columns contain four textual types followed by two numerical values; each of the three possible angle combinations contribute 1/3 to the total improper energy.

Valid DEFINE IDs are given in the following table:

DEFINE ID	Type	Description
FFNAME	string	Defines force field name.
FFTYPE	string	Defines force field type; options are ATOMISTIC, UNITED, or COARSE.
VERSION	string	Defines force field version.
CREATED	string	Defines parameter file creation date.
LENGTH	string	Sets length unit of parameters in file; options are ANGSTROM, NANOMETER, MICROMETER, METER, or REDUCED.
ENERGY	string	Sets energy unit of parameters in file; options are J/MOL, KJ/MOL, CAL/MOL, KCAL/MOL, or REDUCED.
DENSITY	string	Sets density units of parameters in file; options are G/CC, KG/M ³ , or REDUCED.
MIX	string	Sets nonbonded mixing rule; options are NONE, BERTHELOT, ARTIHMETIC, GEOMETRIC, or SIXTH_POWER.
NBONDED	integer	Sets the number of bonds excluded from nonbonded pairwise calculations.
ANGLE	string	Include angle considerations during typing; options are IGNORE, COMPLETE, WARN, EMPTY or ERROR.
TORSION	string	Include torsion considerations during typing; same options as for ANLGE.
IMPROP	string	Include improper considerations during typing; same options as for ANLGE.

The distinction between general and specific force field entries is made by adding extension `_AUTO` to a parameter keyword. Allowed keywords for this extension are PRECEDENCE, BOND, ANGLE, TORSION, and IMPROPER.

A conversion script for the OPLS force field is provided by `./scripts/opls.pl`, which creates a `.prm` and `.top` from OPLS input formats `.par` and `.sb`. It is used through

```
opls.pl [-source=source] target
```

The script assumes the existence of `source.par`, `source.sb`, and `target.define`. The latter is an extra file providing typing rules and possible additions or redefinitions of the OPLS source. The `.define` files found in `./field/opls/2012/` are currently under development.

A conversion script for the MARTINI force field in GROMACS format is provided by `./scripts/martini.pl`. The script is called separately for each contribution through

```
martini.pl name[.itp]
```

The use of multiple topology files within one simulation can be obtained by using a master topology file, which consists of a concatenation of individual topology files. The `martini.pl` script will create extra types in case of redefinition of bonds, angles, torsions, or impropers.

MARTINI files can be found in `./field/martini/v2.0`. A converted copy of the nonbonded parameter file `martini.itp` is stored as `martini.prm` in this directory.

A conversion script for the CHARMM force field is provided by `./scripts/charmm.pl`, which creates a `.prm` and `.top` from CHARMM input formats `.prm` and `.rtf`. It is used through

```
charmm.pl [-source=source] target
```

The script assumes the existence of `par_source.prm`, `top_source.rtf`, and `target.define`. The latter is an extra file providing typing rules and possible additions or redefinitions of the OPLS source. A conversion of `all27_prot_lipid` with its accompanying `.define` file as can be found in `./field/charmm/c32b1/`. The CHARMM force field interpretation is still in beta phase.

5.10.4 Default

Unless otherwise stated, the default is given by

```
field = {  
  mode -> get, compress -> false, error -> true,  
  angle -> true, torsion -> true  
};
```

5.11 Flag

5.11.1 Syntax

```
flag                = {system -> id, oper -> mode, select -> mode,
                      flag -> {constant, ...},
                      site -> {constant, ...},
                      group -> {constant, ...},
                      cluster -> {constant, ...}, region -> region};
```

Directive	Parameters	Description
system	id	Sets system to operate on; identifiers as previously defined.
oper	mode	Sets how to operate on site flag; modes are set or unset .
select	mode	Sets selection (used for ignoring head-tail (i.e. ends) connections in crystal builds); modes are all or ends .
flag	constant	Sets site flag; input either one or more flags; options are fixed , semi , rigid , or branch .
site	constant	Sets the site types to include; all site types are implied when left blank.
group	constant	Sets the group to include; all groups are implied when left blank.
cluster	constant	Sets the cluster id to include; all cluster ids are implied when left blank.
region	region	Specifies in which region in the simulation cell to cut (see Section 7.7 [Region], page 182).

5.11.2 Usage

The **flag** command is used to define site-based functionalities through flags. Sites store these flags in a bitwise representation. Possible flag options are given in the table below.

Name	Bit	Description
fixed	0b0000000001	Prevents the site from being included in energy calculations and from being moved during simulation.
semi	0b0000000010	Prevents the site from being moved during simulation, however, it is included in energy calculations.
head	0b0000000100	Defines a site as a head site and is normally not set in a global fashion, but only on a per site basis (as used in data files).
tail	0b0000001000	Defines a site as a tail site and is normally not set in a global fashion, but only on a per site basis (as used in data files).
rigid	0b0000010000	Defines a site as part of a rigid body and causes a collection of sites to move as one object; only connected sites end up in one rigid body when the body flag is not set.

body	0b0000100000	Defines a site as being part of a body as defined by the region on the cluster as to which the site belongs; sites will either move inside or on the surface of this body; one cluster represents one body; all rigid sites within this body are treated as one rigid body.
surface	0b0001000000	Defines a site to be on the surface of a body rather than in its interior.
backbone	0b0010000000	Defines a site to be part of a cluster backbone; backbones are determined internally upon initialization and are able to take part in topology altering moves (e.g. endbridge and reptate moves).
branch	0b0100000000	Defines a site to be part of a cluster branch; branches are excluded from topology altering moves (e.g. endbridge and reptate moves).
unbound	0b1000000000	Defines a site to be part of a cluster branch; branches are excluded from topology altering moves (e.g. endbridge and reptate moves).

5.11.3 Default

The default is given by

```
flag          = {system -> 0, oper -> set, select -> all,
                 region -> {
                     shape -> cuboid, type -> relative,
                     center -> {0, 0, 0}, radius -> {1, 1, 1}};
```


5.12 Focus (Command)

5.12.1 Syntax

```
focus          = {
  sites         -> {constant, ...},
  groups        -> {constant, ...},
  clusters      -> {constant, ...}
};
```

Directive	Parameters	Description
sites	constant	Sets site or sites to focus on; refers to mass type constants.
groups	constant	Sets group or groups to focus on.
clusters	constant	Sets cluster or clusters to focus on.

5.12.2 Usage

The `focus` command performs a translation to the origin of the accumulative center of mass of all participating sites, groups, and/or clusters. Sites, groups, and clusters are defined by constants as referred to in the simulation constant definition paragraph (see Section 7.1 [Constants], page 174) and described in the focus section (see Section 7.3 [Focus], page 176).

5.12.3 Default

Unless otherwise stated, the default is given by

```
focus = {};
```

5.13 Force

5.13.1 Syntax

```
force                = {style -> option, message -> boolean};
```

Directive	Parameters	Description
style	option	Selects force style; valid options are none , init , or list
message	option	Selects output to screen; valid options are none , raw , nkt , or n , which corresponds to energy in no particular units, internal units, units of nkT or units of n , respectively; internal units correspond to the units of the selected force field (e.g. kcal/mol for COMPASS, CHARMM, OPLS, GROMACS, or MARTINI).

5.13.2 Usage

The **force** command is used to calculate and possibly output energies and virials associated with all systems in the active simulation. It allows for an internal check of consistency of energies after a simulation by calculation of energies either directly by using the **none** option, or using the neighbor list resulting from a simulation using the **list**, or through reinitialization of the neighbor lists by using the **init** option. The energy output is controlled by the **message** directive. Energies and virials are reported in either units as defined in the units paragraph when selecting **raw**, or normalized by the number of active sites when selecting **n**, or in relative units of $nk_B T$ when selecting **nkt**.

5.13.3 Default

The default is given by

```
force                = {style -> init, message -> nkt};
```

5.14 Former

5.14.1 Syntax

```
former          = {name -> string, mode -> option, forcefield -> option};
```

Directive	Parameters	Description
name	string	Defines the file name to be imported; an extension of .emc is assumed.
mode	option	Selects i/o mode; the only currently supported option is get .
forcefield	option	Selects the desired force field; valid options are none , standard , and charmm .

5.14.2 Usage

The **former** command is used to import data files generated by previous EMC versions. Currently only the import of version 8 data files is supported.

5.14.3 Default

The default is given by

```
former          = {mode -> get, forcefield -> none};
```

5.15 Get

5.15.1 Syntax

```
get                = {name -> string, compress -> boolean};
```

Directive	Parameters	Description
name	string	Defines the file name to be imported; an extension of .emc is assumed.
compress	boolean	Sets compression using Lempel-Ziv coding (LZ77); an extra extension of .gz is assumed; options are true or false .

5.15.2 Usage

The **get** command is used to import data files generated by the current version of EMC, using a structured data format. Data files function as a restart file and are expected to be in a textual format.

5.15.3 Default

The default is given by

```
get                = {compress -> false};
```

5.16 Groups

5.16.1 Syntax

```

groups          = {
  group          -> {
    id           -> constant,
    charge       -> real,
    charges      -> option,
    template     -> constant,
    terminator    -> boolean,
    chemistry    -> string,
    depth        -> integer,
    field        -> constant,
    connects     -> {{source -> integer,
                      destination -> {index -> constant, site -> integer}
                      }, ...}
  },
  delete        -> {
    id           -> constant,
    site        -> integer
  },
  ...
};

```

Directive	Parameters	Description
group	struct	Defines group parameters; repeating structure.
id	constant	Sets group identity; can be either numerical or textual.
charge	double	Sets an additional charge of the complete group, which value divided by the number of group sites is added to the site charges originating from force field or override charge assignments.
charges	option	Defines how partial charges defined in SMILES string are dealt with; valid options are forcefield for using partial charges assigned by a force field only (see Section 5.10 [Field], page 101), additive for adding specified partial charges per SMILES atom to values resulting from force field assignment, and override for only using the specified partial charges (non-specified charges are zero).
template	constant	Selects a template as predefined in force field files; can be either numerical or textual (see Section 5.10 [Field], page 101); templates replace the use of a chemistry string.
terminator	boolean	Indicates if this group can be used as a terminator, which is useful for coarse-grained polymers without termination groups; options are true or false .

chemistry	string	Sets the group's chemistry using the SMILES format (see Section 7.8 [SMILES], page 183).
depth	integer	Sets recursive depth for ring determination during SMILES interpretation; the depth should equal the maximum number of atoms participating in a ring within the given SMILES string; standard option is auto ; any set depth smaller than 3 reverts to the auto setting.
field	constant	Optionally set a specific subfield identifier to use when typing this group (see Section 5.10 [Field], page 101).
connects	struct	Defines connections to other groups; both source and site correspond to the position in the SMILES of where a connection occurs; '*', '>', or '<' characters mark connection points; shortcuts head , tail , and \$end1 through \$end10 denote connections in a group without having to state their exact positions within the corresponding SMILES; head and tail are analogous to \$end1 and \$end2 respectively (see example below).
source	integer	Sets the source element
destination	struct	Defines the destination group and element
index	constants	Sets the destination group; can be either numerical or textual
site	integer	Sets the destination element
delete	struct	Deletes a specified group site corresponding to a chemistry entry; all corresponding simulation sites are deleted accordingly.
id	constant	Target group identity; can be either numerical or textual.
site	integer	Site number corresponding to the entry position in the SMILES string; counting starts at 0.

5.16.2 Usage

The **groups** command creates the necessary groups used to identify full molecules or chemical repeat units within clusters, as represented by SMILES strings. Keyword **charges** is used to assign extra charge to a subset of atoms in case of ionic liquids, for which case a force field might not supply charged molecules. One can also use keyword **charge** to increase or decrease the charge of a full group. This option assigns a partial charge to each site, which is equal to the full charge divided by the number of sites in the group.

5.16.3 Default

The default is given by

```
groups      = {
  group      -> {
    id        -> 0,
    charge    -> 0,
    charges   -> forcefield,
```

```

    terminator -> false
  }
};

```

5.16.4 Examples

A few atomistic examples are

```

groups      = {
  group      -> {
    id        -> water,
    chemistry -> "HOH"
  },
  group      -> {
    id        -> dodecane,
    chemistry -> "(C)12"
  },
  group      -> {
    id        -> benzene,
    chemistry -> "c1ccccc1"
  },
  group      -> {
    id        -> acid,
    chemistry -> "C(=O)[O-]"
  }
};

```

An example of deleting a group site is given by

```

groups      = {
  delete     -> {
    id        -> dodecane,
    site      -> 0
  }
};

```

which deletes the first site of group dodecane, effectively changing dodecane into undecane. Note, that all corresponding simulation sites are deleted accordingly.

An example of a set of monomers for a coarse-grained polymer is given by

```

groups      = {
  group      -> {
    id        -> monomer1,
    chemistry -> "*ab*c(d)cc",
    connects  -> {
      {head, {monomer1, tail}}, {head, {monomer2, head}},
      {tail, {monomer1, head}}, {tail, {monomer2, head}},
      {head, {terminator, head}}, {tail, {terminator, head}}
    }
  },
  group      -> {

```

```

    id          -> monomer2,
    chemistry    -> "*c*",
    connects     -> {
        {head, {monomer1, head}}, {head, {monomer1, tail}},
        {head, {monomer2, head}}, {head, {monomer2, tail}},
        {head, {terminator, head}}}
    },
    group        -> {
        id          -> terminator,
        chemistry    -> "*t",
        connects     -> {
            {head, {monomer1, head}}, {head, {monomer2, head}}}
        },
    };

```

The '*' character implies the start and end or head and tail of a repeat unit of a polymer. By default, a group is not a terminator when more than one '*' character appears in the SMILES string. However, when such a group should also function as a terminator, the **terminator** flag can be used as indication of such. A group is automatically a terminator when a '*' character appears only once, which means, that the **terminator** flag need not be set. The keyword **connects** lists the possible connections of these start and end sites to sites in the same or other groups. The keywords **head** and **tail** represent internal short cuts and indicate the position of the '>' and '<' characters repectively.

5.17 Insight

5.17.1 Syntax

```
insight          = {name -> string, compress -> boolean,
                    system -> integer, mode -> option,
                    forcefield -> option, atomistic -> option,
                    charges -> boolean, formal -> boolean, depth -> integer,
                    detect -> boolean, cut -> boolean, pbc -> boolean,
                    unwrap -> boolean, percolate -> boolean,
                    crystal -> boolean, flag -> struct};
```

Directive	Parameters	Description
name	string	Defines the file name to be imported; an extension of .emc is assumed.
compress	boolean	Sets compression using Lempel-Ziv coding (LZ77); an extra extension of .gz is assumed; options are true or false .
system	integer	Selects system to be imported or exported
mode	option	Selects i/o mode; supported options are get or put .
forcefield	option	Selects the desired force field for writing; see Section 7.5 [Ports], page 180, for valid options; option auto auto-selects.
atomistic	option	Select the atomistic mode normally defined by the choice of forcefield ; options are none , united , or full .
charges	boolean	Use partial charges from imported structures; options are true or false .
formal	boolean	Use formal charges from imported structures; options are true or false .
detect	boolean	Detect and assign atom and amino acid residue types; options are true or false .
depth	integer	Sets recursive depth for ring determination during SMILES interpretation; the depth should equal the maximum number of atoms participating in a ring within the given SMILES string; standard option is auto ; any set depth smaller than 3 reverts to the auto setting.
cut	boolean	Selects cut mode: cut bonds that cross cell boundaries; options are true or false .
pbc	boolean	Applies periodic boundary conditions; options are true or false .
unwrap	boolean	Unwraps clusters to cross periodic boundaries with application of periodic boundary conditions to its center of mass when applicable; options are true or false .
crystal	boolean	Imported structure is crystalline; options are true or false .

<code>percolate</code>	<code>boolean</code>	Allow for percolating structures; options are <code>true</code> or <code>false</code> .
<code>flag</code>	<code>struct</code>	Allows for setting system flags while importing morphologies; see see Section 7.10 [System Flags], page 187, for the various available flags.

5.17.2 Usage

The `insight` command is used to import and export InsightII data files in CAR and MDF format for exchange of configurations between EMC and Materials Studio, a product of Accelrys¹.

5.17.3 Default

The default is given by

```
insight      = {compress -> false,  
                system -> 0, forcefield -> auto, detect -> false,  
                cut -> false, pbc -> true, unwrap -> true};
```

5.17.4 References

1. Materials Studio, '<http://www.accelrys.com/>'

5.18 Lammmps

5.18.1 Syntax

```
lammmps      = {name -> string, compress -> boolean,
                system -> integer, mode -> option,
                units -> option, length -> double, forcefield -> option,
                shake -> option, atomistic -> boolean, cutoff -> boolean,
                charges -> boolean, ewald -> boolean,
                bonds -> boolean, types -> boolean,
                parameters -> boolean, cross -> boolean,
                variables -> boolean, coefficients -> boolean,
                comment -> boolean};
```

Directive	Parameters	Description
name	string	Defines the file name to be imported; an extension of .data is assumed.
compress	boolean	Sets compression using Lempel-Ziv coding (LZ77); an extra extension of .gz is assumed; options are true or false .
system	integer	Selects system to be imported or exported
mode	option	Selects i/o mode; supported options are get or put .
length	double	Defines a scaling length to supercede the simulation variable units -> angstrom ; the latter is used when length is not specified (see Section 7.13 [Units], page 332).
units	option	Selects whether to represent site coordinates in reduced or real units, for which options are lj or real respectively; alternatively none is also defined.
forcefield	option	Selects the desired force field; see Section 7.5 [Ports], page 180, for valid options.
shake	option	Selects the SHAKE ^{1,2} mode for atomistic force fields; valid options are none , auto , hydrogen , water and all ; auto bases the choice on the selected force field (none by default).
atomistic	boolean	Selects atomistic mode: no charge column and no bonds; options are true or false .
cutoff	boolean	adds a column containing cut offs as used for EMC force field evaluation to the LAMMPS parameter file; helpful for transferring purely repulsive potentials; options are true or false .
charges	boolean	Selects charge mode: a charge column is present in the Atoms paragraph; options are true or false .
ewald	boolean	Selects the usage of Ewald summations for long-range charge treatment; options are true or false .

<code>bonds</code>	<code>boolean</code>	Selects bonds mode: all bonded contributions are present; options are <code>true</code> or <code>false</code> .
<code>types</code>	<code>boolean</code>	Selects types mode: all types parameter definitions are present; options are <code>true</code> or <code>false</code> .
<code>parameters</code>	<code>boolean</code>	Selects the separate output of variables and (non)bonded interaction parameters in LAMMPS input script format; an extension of <code>.params</code> is assumed; options are <code>true</code> or <code>false</code> .
<code>cross</code>	<code>boolean</code>	Includes nonbonded cross-terms in <code>.params</code> file; options are <code>true</code> or <code>false</code> .
<code>variables</code>	<code>boolean</code>	Selects adding simulation variables to the separate output of variables and interaction parameters; options are <code>true</code> or <code>false</code> .
<code>coefficients</code>	<code>boolean</code>	Selects adding interaction parameters to the separate output of variables and interaction parameters; options are <code>true</code> or <code>false</code> .
<code>comment</code>	<code>boolean</code>	Includes commented references to mass names in <code>.data</code> and <code>.params</code> files; options are <code>true</code> or <code>false</code> .

5.18.2 Usage

The `lammps` command is used to import and export data files generated by LAMMPS, a Molecular Dynamics code conceived at Sandia National Laboratories.^{3,4} Textual input formats for LAMMPS are ambiguous and are not self-defining, which creates the need for predefinition of the desired format. This port allows for commonly used force field modes and site (atom) definitions.

5.18.3 Default

The default is given by

```
lammps      = {compress -> false, system -> 0, mode -> put,
               units -> none, forcefield -> auto, shake -> auto,
               atomistic -> true, charges -> false, ewald -> false,
               bonds -> false, bonds -> false, types -> true,
               parameters -> false, cross -> false, variables -> true,
               coefficients -> true, comment -> true};
```

5.18.4 Examples

A few examples are given by

```
lammps      = {name -> "benzene", compress -> true,
               mode -> put, forcefield -> opls, types -> false,
               parameters -> true, charges -> true, ewald -> true};
lammps      = {name -> "polystyrene", compress -> true,
               mode -> put, forcefield -> coarse, types -> false,
               parameters -> true};
lammps      = {name -> "polyethylene", mode -> put};
```

5.18.5 References

1. Ryckaert, J.-P.; Ciccotti, G. and Berendsen, H.J.C., "", *J. Comp. Phys.* **1977**, *23*, 327-341
2. Andersen, H. "", *J. Comp. Phys.* **1983**, *52*, 24-34
3. LAMMPS - Molecular Dynamics Simulator, '<http://lammps.sandia.gov/>'
4. Plimpton, S., "Fast Parallel Algorithms for Short-Range Molecular Dynamics", *J. Comput. Phys.* **1995**, *117*, 1-19

5.19 Memory

5.19.1 Syntax

```
memory          = {style -> option};
```

Directive	Parameters	Description
style	option	Sets the output style; options are full or summary .

5.19.2 Usage

The **memory** command outputs the memory consumption of the current simulation to screen. Either a summary or a full description can be selected.

5.19.3 Default

The default is given by

```
memory          = {style -> full};
```

5.19.4 Examples

An example is given by

```
memory          = {style -> summary};
```

5.20 Moves

5.20.1 Syntax

```
moves          = {
  ncycles      -> integer,
  cycle        -> integer,
  move         -> integer,
  moves...
};
```

Directive	Parameters	Description
ncycles	integer	Sets the total number of simulation cycles.
cycle	integer	Sets the current simulation cycle
move	integer	Sets the current simulation move.
moves		Access to various move settings; see Section 7.4 [Moves], page 177, for further information.

5.20.2 Usage

This variable style describes types (see Section 7.4 [Moves], page 177, for further information) and allows direct access to all variables and parameters stored within the moves structure.

5.20.3 Default

Unless otherwise stated, the default is given by

Unless otherwise stated, the default is given by

@verbatim

```
moves          = {
  ncycles      -> 0,
  cycle        -> 0,
  move         -> 0
};
```

By default, all moves are activated with zero frequency, with the exception of the displacement move (see Section 7.4.5 [Displace], page 179), which has a frequency of one.

5.21 PDB

5.21.1 Syntax

```

pdb                = {name -> string,
                      compress -> boolean, mode -> option, system -> integer,
                      length -> real, forcefield -> option, atomistic -> option,
                      charges -> boolean, depth -> integer, detect -> boolean,
                      atom -> option, residue -> option, segment -> option,
                      vdw -> option, hexadecimal -> boolean,
                      cut -> boolean, pbc -> boolean, unwrap -> boolean,
                      rigid -> boolean, fixed -> boolean,
                      connectivity -> boolean, crystal -> boolean,
                      element -> option};

```

Directive	Parameters	Description
name	string	Defines the file name to be imported; an extension of .emc is assumed.
compress	boolean	Sets compression using Lempel-Ziv coding (LZ77); an extra extension of .gz is assumed; unpacking and packing is added to the VMD script; options are true or false .
mode	option	Selects i/o mode; supported options are get and put
system	integer	Selects system to be imported or exported.
length	real	Set the length scale with which to scale the resulting coordinates (normally results from choice of forcefield).
forcefield	option	Selects the desired force field; see Section 7.5 [Ports], page 180, for valid options; option auto auto-selects.
atomistic	option	Select the atomistic mode normally defined by the choice of forcefield ; options are none , united , or full .
charges	boolean	Use charges from imported structures; options are true or false .
depth	integer	Sets recursive depth for ring determination during SMILES interpretation; the depth should equal the maximum number of atoms participating in a ring within the given SMILES string; standard option is auto ; any set depth smaller than 3 reverts to the auto setting.
detect	boolean	Detect and assign atom and amino acid residue types; options are true or false .
atom	option	Sets options for atom name representation upon output; options are detect for detection of atom types, index for use of EMC mass element ids, or series for indexing using EMC element ids followed with sequential numbering per segment. The latter is useful for CHARMM tools expecting unique atom ids.

<code>residue</code>	<code>option</code>	Sets options for residue name representation upon output; options are <code>detect</code> for detection of amino acid residues, <code>index</code> for use of EMC group ids, or <code>series</code> for increments starting with the letter R.
<code>segment</code>	<code>option</code>	Sets options for segment name representation upon output; options are <code>detect</code> for detection, <code>index</code> for use of EMC cluster ids, or <code>series</code> for increments starting with the letter M.
<code>vdw</code>	<code>boolean</code>	Include the definition for switching on Van der Waals in the resulting VMD script; options are <code>true</code> or <code>false</code> .
<code>hexadecimal</code>	<code>boolean</code>	Use hexadecimal representations for residue sequence numbers; options are <code>true</code> or <code>false</code> .
<code>cut</code>	<code>boolean</code>	Selects whether cut bonds that cross cell boundaries upon output; options are <code>true</code> or <code>false</code> .
<code>vdw</code>	<code>boolean</code>	Add Van der Waals representation to the written VMD script; options are <code>true</code> or <code>false</code> .
<code>pb</code>	<code>boolean</code>	Applies periodic boundary conditions upon output; options are <code>true</code> or <code>false</code> .
<code>unwrap</code>	<code>boolean</code>	Unwraps clusters upon output to cross periodic boundaries with application of periodic boundary conditions to its center of mass when applicable; options are <code>true</code> or <code>false</code> .
<code>rigid</code>	<code>boolean</code>	Unwrap sites in clusters, that are flagged as rigid; options are <code>true</code> or <code>false</code> .
<code>fixed</code>	<code>boolean</code>	Unwrap sites in clusters, that are flagged as fixed; options are <code>true</code> or <code>false</code> .
<code>connectivity</code>	<code>boolean</code>	Include connectivity in the resulting PDB using keyword <code>CONNECT</code> ; options are <code>true</code> or <code>false</code> .
<code>crystal</code>	<code>boolean</code>	Selects whether a read structure is a crystal; needed for small crystal structures with box crossing bonds; options are <code>true</code> and <code>false</code> .
<code>element</code>	<code>option</code>	Selects location of element (internally used for defining masses); options are <code>auto</code> , <code>element</code> , and <code>type</code> ; <code>element</code> uses elements defined in PDB files; aromatic elements are all lower case (needed when reading united atom structures); <code>type</code> uses types as defined in PSF files, which is needed when reading coarse-grained structures; <code>auto</code> internally decides which of the previous options to use based on the choice of force field (see Section 5.10 [Field], page 101).

5.21.2 Usage

The `pdb` command is used to export data files in PDB and PSF format for convenient visualization with VMD (Visual Molecular Dynamics), a visualization conceived and supported by the Theoretical and Computational Biophysics Group at the University of Illinois at Urbana-Champaign. Textual PDB and PSF formats are ambiguous and are not self-

defining, which creates the need for predefinition of the desired format. This port allows for commonly used force field modes. Cell dimensions are defined by addition of a **CRYST1** keyword to the PDB file.

5.21.3 Default

The default is given by

```
pdb          = {compress -> false, mode -> put, system -> 0,  
               forcefield -> auto, atomistic -> full, depth -> auto,  
               detect -> false, atom -> series, residue -> index,  
               segment -> index, vdw -> false, hexadecimal -> false,  
               cut -> false, pbc -> true, unwrap -> true,  
               rigid -> true, fixed -> true,  
               connectivity -> false, crystal -> false, element -> auto};■
```

5.21.4 References

1. VMD - Visual Molecular Dynamics, '<http://www.ks.uiuc.edu/Research/vmd/>'
2. Humphrey, W., Dalke, A. and Schulten, K., "VMD - Visual Molecular Dynamics", *J. Molec. Graphics* **1996**, *14*, 33-38

5.22 Put

5.22.1 Syntax

```
put                = {name -> string, compress -> boolean, detail -> integer};
```

Directive	Parameters	Description
name	string	Defines the file name to be exported; an extension of .emc is assumed.
compress	boolean	Sets compression using Lempel-Ziv coding (LZ77); an extra extension of .gz is assumed; options are true or false .
detail	integer	Sets level of detail in output file; a non-zero setting overrides the already set detail.

5.22.2 Usage

The `put` command is used to export data files generated by the current version of EMC, using a structured data format. Data files contain all necessary information to function as a restart file and are in a textual format.

5.22.3 Default

The default is given by

```
put                = {compress -> false, detail -> 0};
```

5.23 Rename

5.23.1 Syntax

```

rename      = {
  site      -> {src -> constant, dest -> constant},
  ...
  group     -> {src -> constant, dest -> constant},
  ...
  cluster   -> {src -> constant, dest -> constant},
  ...
  system    -> {src -> constant, dest -> constant},
  ...
};

```

Directive	Parameters	Description
site	struct	Defines multiple sites.
group	struct	Defines multiple groups.
cluster	struct	Defines multiple clusters.
system	struct	Defines multiple systems.

5.23.2 Usage

The `rename` command allows for renaming already existing site, group, cluster, or system ids.

5.23.3 Default

No default.

5.23.4 Examples

A few examples are given by

```

rename      = {
  site      -> {src -> bcc, dest -> a},
  site      -> {src -> fcc, dest -> b}
};

rename      = {
  site      -> {src -> fcc, dest -> a},
  cluster   -> {src -> fcc, dest -> surface}
};

```

5.24 Remove

5.24.1 Syntax

```
remove          = {n -> integer, mode -> option, nmin -> integer,
                  target -> index, region -> region};
```

Directive	Parameters	Description
n	integer	Selects number repeat units to be removed in each active system.
mode	option	Selects removal mode; viable options are cluster , group , and site .
target	index	Specifies the targeted group to remove; additionally a group site can be specified; a value of -1 one indicates all .
nmin	integer	Specifies the minimum allowable number of mobile repeat units that remain.
region	region	Specifies in which region in the simulation cell to remove (see Section 7.7 [Region], page 182).

5.24.2 Usage

The **remove** command deletes a maximum of **n** and a minimum of **nmin** groups specified by **group**, which fall within a specified **region**. All sites within the specified region are deleted, when no group is provided.

5.24.3 Default

```
remove          = {
  n              -> 0,
  mode           -> cluster,
  nmin           -> 0,
  target         -> {index -> -1, site -> -1},
  region         -> {shape -> spheroid, type -> relative, mode -> hard,
                    center -> {0,0,0}, h -> {0,0,0,0,0,0}}
};
```

5.24.4 Examples

Remove 100 water molecules as defined by group **water** in the whole simulation box,

```
remove          = {
  n              -> 100,
  target         -> {index -> water},
  region         -> {shape -> cuboid, radius -> {1, 1, 1}}
};
```

Note, that overspecification by the region radius of a factor of two ensures the inclusion of all particles in that box direction.

Remove 10 methyl groups as defined by group `methyl` in the center halve portion in the x direction of the simulation box,

```
remove      = {  
  n          -> 10,  
  target     -> {index -> methyl},  
  region     -> {shape -> cuboid, radius -> {0.25, 1, 1}}  
};
```

5.25 Reset

5.25.1 Syntax

```
reset           = {style -> option};
```

Directive	Parameters	Description
style	integer	Possible options are simulation, statistics, profiles, or moves.

5.25.2 Usage

The **reset** command allows for resetting entries and counters of either the whole simulation structure, or of separate subsections statistics, profiles, or moves.

5.25.3 Default

The default is given by

```
reset           = {style -> simulation};
```

5.26 Restart

5.26.1 Syntax

```
restart          = {name -> string, compress -> boolean, format -> string,
                    frequency -> integer, reset -> boolean};
```

Directive	Parameters	Description
name	string	Defines the file name to be exported; an extension of .emc is assumed.
compress	boolean	Sets compression using Lempel-Ziv coding (LZ77); an extra extension of .gz is assumed; options are true or false .
format	string	Defines counter extension format.
frequency	integer	Selects output interval frequency in units of cycles.
reset	boolean	Resets simulation after writing restart file; options are true or false .

5.26.2 Usage

The **restart** command allows for writing of restart files at constant intervals during simulation, which safeguards long simulations in case of compute environment failure.

5.26.3 Default

The default is given by

```
restart          = {compress -> false, format -> "%08ld",
                    frequency -> 10000, reset -> false};
```


5.27 Retype

5.27.1 Syntax

```

retype          = {
  mode           -> option,
  fraction       -> real,
  charge        -> boolean,
  n             -> {integer, ...},
  system        -> {constant, ...},
  source        -> {constant, ...},
  destination    -> {constant, ...},
  group         -> {index -> constant, site -> integer},
  inside        -> region,
  outside       -> region
};

```

Directive	Parameters	Description
<code>mode</code>	<code>option</code>	Sets the selection criteria for the fraction of selected sites; valid modes are: random .
<code>charge</code>	<code>boolean</code>	Selects charge transfer from mass paragraph to sites.
<code>fraction</code>	<code>real</code>	Defines the fraction of sites within the selection to be changed; values smaller than zero will be set to zero; values larger than one will be set to one; non-zero number of sites overrides fractions.
<code>n</code>	<code>integer</code>	Sets the number of sites to delete per defined targeted system; the first entry is taken for all systems when targeted systems are omitted; fractions are taken when omitted.
<code>system</code>	<code>constant</code>	Sets targeted system; can one or more systems; all systems are implied when omitted.
<code>source</code>	<code>constant</code>	Sets source mass types; can be one or more types.
<code>destination</code>	<code>constant</code>	Sets destination mass types; can be one or more types; has to be either one element or the same number of elements as represented under the source directive.
<code>group</code>	<code>index</code>	Optional definition of group index and site for all destination types.
<code>inside</code>	<code>region</code>	Sets the inner boundary of the region in which mass types are to be changed (see Section 7.7 [Region], page 182); assumes domain center when omitted.
<code>outside</code>	<code>region</code>	Sets the outer boundary of the region in which mass types are to be changed (see Section 7.7 [Region], page 182); assumes whole domain when omitted.

5.27.2 Usage

The `retype` command allows for changing mass types from a source target consisting of either one or more types to a destination target consisting of one or more types. The destination target can either consist of one element or the same number of elements as the source target, in case the source target consists of more than one element. All groups, systems and regions are implied when none are supplied.

5.27.3 Default

The default is given by

```
retype          = {
  mode          -> random,
  fraction       -> 1
};
```

5.27.4 Examples

Note that the new site 'graft' needs to be defined in terms of mass and force field (see Section 5.31 [Sites], page 137, and Section 5.32 [Simulation], page 138). An example is given by

```
variables       = {r1 -> 1/2, r2 -> 1/2-0.75/(2^(2/3)*n)};
retype          = {
  source         -> fcc,
  destination    -> graft,
  inside         -> {shape -> spheroid,
                    center -> {0, 0, 0}, radius -> {r2, r2, r2}},
  outside        -> {shape -> spheroid,
                    center -> {0, 0, 0}, radius -> {r1, r1, r1}}
};
```

5.28 Run

5.28.1 Syntax

```
run                = {ncycles -> integer, nblocks -> integer,
                      cycle -> integer, seed -> integer};
```

Directive	Parameters	Description
<code>ncycles</code>	<code>integer</code>	Defines the number of cycles for which to run the simulation.
<code>nblocks</code>	<code>integer</code>	Defines the frequency with which to output intermediate energies in intervals of cycles.
<code>cycle</code>	<code>integer</code>	Defines the cycle counter at which to start the simulation; selects the cycle as defined by the input file when negative
<code>seed</code>	<code>integer</code>	Selects a random seed; the time is used when <code>seed</code> is greater than or equal to zero.

5.28.2 Usage

The `run` command is used to start a simulation. It allows for definition of simulation length and output frequency. Starting cycle and initial random seed can also be defined.

5.28.3 Default

The default is given by

```
run                = {cycle -> -1, seed -> 0};
```

5.29 Sample

5.29.1 Syntax

```
sample      = {
  cavity    -> struct,
  gr        -> struct,
  gyration  -> struct,
  ...
};
```

Directive	Parameters	Description
cavity	struct	Descriptor for cavity size sampling as defined by CESA (Cavity Energetic Sizing Algorithm) ¹ (see Section 6.2 [Cavity], page 155).
gr	struct	Descriptor for radial distribution functions sampling (see Section 6.3 [Gr], page 160).
gyration	struct	Descriptor for radius of gyration sampling (see Section 6.4 [Gyration], page 165).

5.29.2 Usage

This variable style describes the control of sampling options (see Chapter 6 [Sampling Tools], page 151, for applications).

5.29.3 Default

By default, all sampling is deactivated.

5.29.4 References

1. P.J. in 't Veld, M.T. Stone, T.M. Trustkett, and I.C. Sanchez, "Liquid Structure via Cavity Size Distributions", *J. Phys. Chem. B* **2000**, *104*, 12028

5.30 Shell

5.30.1 Syntax

```
shell          = {command -> string};
```

Directive	Parameters	Description
command	string	Execute a shell command as specified by string .

5.30.2 Usage

The **shell** command is used to execute a shell command, which allows for file manipulations during running. Any problems will be identified by a warning, thus not stopping the execution of the EMC script.

5.30.3 Default

The default is given by

```
shell          = {command -> ""};
```

5.31 Sites

5.31.1 Syntax

```
sites      = {
  site      -> {id -> constant,
               reference -> integer, name -> string, mass -> real},
  ...
};
```

Directive	Parameters	Description
site	struct	Identifies site parameters; repeating structure.
id	constant	Sets site identity; can be either numerical or textual.
reference	integer	Sets a numerical reference; not used internally.
name	string	Sets a longer identifying name.
mass	real	Sets the mass in units set in <code>units</code> .

5.31.2 Usage

The `sites` command creates mass entries in the `mass` section of the `types` section of `simulation`. Multiple sites can be entered by using multiple instances of the `site` directive, separated by commas.

5.31.3 Default

The default is given by

```
sites      = {site -> {id -> 0, reference -> 0}};
```

5.31.4 Examples

A few examples are

```
sites      = {
  site      -> {id -> a, reference -> 0, name -> "colloid", mass -> 10}
};
```

```
sites      = {
  site      -> {id -> c,
               reference -> 12, name -> "carbon", mass -> 12.011},
  site      -> {id -> h,
               reference -> 1, name -> "hydrogen", mass -> 1.008},
  site      -> {id -> o,
               reference -> 14, name -> "oxygen", mass -> 15.9994}
};
```

5.32 Simulation

5.32.1 Syntax

```
simulation      = {output -> {...}, units -> {...}, variables -> {...},
                  systems -> {...}, types -> {...}, moves -> {...},
                  profiles -> {...}};
```

Directive	Parameters	Description
output	{...}	Provides access to output parameters; possible directives include detail , wide , expand , math , reduced , info , strict , warning , message , and debug .
units	{...}	Provides access to internal units; possible directives include mass , length , angstrom , angle , energy , kb , nav , charge , permittivity , and seed (see Section 7.13 [Units], page 332).
variables	{...}	Provides access to changing predefined variables; used in conjunction with predefined variables in input files, allowing posteriori sizing of an input structure.
types	{...}	Provides access to all simulation-wide types; possible directives, which define force fields and their respective constants, include boltzmann , charmm , coarse , coulomb , spline , and standard (see Section 7.12 [Types], page 190).
systems	{...}	Provides access to all simulation-wide system settings; possible directives, which define e.g. system temperature or geometry, include properties ; most system properties are derived output variables (e.g. p , v , mass , nsites , nclusters) rather than input variables (see Section 7.11 [Systems], page 188).
moves	{...}	Provides access to all moves; possible directives include displace , endbridge , rebridge , reptate , rotate , and temper (see Section 7.4 [Moves], page 177).
profiles	{...}	Provides access to all profile definitions; possible directives include density , force , mass , and order (see Section 7.6 [Profiles], page 181).

5.32.2 Usage

The **simulation** command provides access to all variables and parameters defined within the simulation structure, of which the above table lists the main directives.

5.32.3 Examples

A few examples are (see data file for more suggestions)

```
simulation      = {output -> {detail -> 4}};
simulation      = {variables -> {lb -> 0.95*lb}};
```

```
simulation      = {types -> {standard -> {correct -> {active -> true}}}};
simulation      = {systems -> {
    properties -> {
        {id -> 0, t -> 298.15}, {id -> 1, t -> 315}}}};
simulation      = {moves -> {displace -> {frequency -> 1}}};
simulation      = {profiles -> {density -> {active -> true}}};
```


5.33 Split

5.33.1 Syntax

```
split          = {system -> constant, direction -> option,
                  mode -> option, unwrap -> boolean, fraction -> real,
                  focus -> struct, region -> struct};
```

Directive	Parameters	Description
system	constant	Sets identity of system to which to add the build; can be either textual or numerical.
direction	option	Sets the direction in which the system is grown; options are x , y , and z .
mode	option	Sets the selection algorithm; options are distance and random .
unwrap	boolean	Sets whether to unwrap already existing clusters (only the first instance before a build should be set to true); options are true and false .
fraction	real	Defines the distribution fraction; valid values are between 0 and 1, inclusive the extremes.
focus	struct	Sets the selection to focus on (see see Section 7.3 [Focus], page 176).
region	struct	Sets the region to consider (see see Section 7.7 [Region], page 182).

5.33.2 Usage

The **split** command is used to redistribute clusters at the growing sides of a simulation box. Specific selection criteria can be set by defining a region in which the center of mass of targetted clusters need to reside. Additionally, a focussing selection can be specified based on site types, group ids, and cluster ids. Undefined **focus** and **region** directives imply considering all active sites in the system.

5.33.3 Default

The default is given by

```
split          = {system -> 0, direction -> x,
                  mode -> random, unwrap -> true, fraction -> 0.5};
```

5.34 Terminate

5.34.1 Syntax

```
terminate          = {mode -> option};
```

Directive	Parameters	Description
mode	option	Selects termination mode; currently only the option <code>all</code> is supported.

5.34.2 Usage

The `terminate` command is used to terminate free cluster ends created by `cut` and `remove` commands. Currently, terminators existing of only one site are supported. Future releases will include a group-based terminator.

5.34.3 Default

The default is given by

```
terminate          = {mode -> all};
```

5.35 Timing

5.35.1 Syntax

```
timing          = {style -> option};
```

Directive	Parameters	Description
style	option	Possible options are <code>none</code> , <code>show</code> , or <code>reset</code> .

5.35.2 Usage

The `timing` command allows for showing current timing, after which timing can be reset or allowed to accumulate further.

5.35.3 Default

The default is given by

```
timing          = {style -> show};
```

5.36 Traject

5.36.1 Syntax

```
traject      = {mode -> option, system -> integer,
                name -> string, compress -> boolean, format -> string,
                start -> integer, end -> integer, frequency -> integer,
                append -> boolean, reset -> boolean, scale -> boolean,
                unwrap -> boolean, pbc -> boolean};
```

Directive	Parameters	Description
mode	option	Sets the operation mode; viable options are <code>get</code> , <code>put</code> , and <code>sample</code> .
system	integer	Selects the system to export.
name	string	Defines the file name to be exported; an extension of <code>.traject</code> is assumed.
compress	boolean	Sets compression using Lempel-Ziv coding (LZ77); an extra extension of <code>.gz</code> is assumed; options are <code>true</code> or <code>false</code> .
format	string	Defines counter extension format.
start	integer	Selects the frame at which to start reading; -1 indicates the last frame.
end	integer	Selects the frame at which to end reading; -1 indicates the last frame.
frequency	integer	Selects output interval frequency in units of cycles.
append	boolean	Appends each selected cycle to the end of the trajectory file; options are <code>true</code> or <code>false</code> .
reset	boolean	Resets output file; options are <code>true</code> or <code>false</code> .
scale	boolean	Scales the output using system box dimensions; options are <code>true</code> or <code>false</code> .
unwrap	boolean	Unwraps clusters in output; options are <code>true</code> or <code>false</code> .
pbc	boolean	Applies periodic boundary conditions; options are <code>true</code> or <code>false</code> .

5.36.2 Usage

The `traject` command allows for reading, writing, or analyzing of LAMMPS-style trajectory files. In `read` mode, a configuration from a trajectory can be loaded over already existing positions. In `write` mode, configurations are written at constant intervals during simulation. Output can be directed to either one or separate files by setting the `append` option to `true` or `false` respectively. Cycle counts are added to the file name when separate files are chosen, however they are omitted when the `append` option is set. In `analyze` mode, a full trajectory is analyzed using preset sampling settings (see Section 5.29 [Sample], page 135). An example can be found in `./examples/sample/cavity/traject.emc`.

5.36.3 Default

The default is given by

```
traject      = {mode -> put,  
               compress -> false, format -> "%08ld", system -> 0,  
               start -> 0, end -> -1, frequency -> 1000, append -> true,  
               reset -> true, scale -> false};
```

5.36.4 Examples

A few examples are given by

```
traject      = {name -> "test/polyethylene", compress -> true};  
traject      = {name -> "water", format -> "%ld", system -> 1,  
               frequency -> 10000, append -> false, scale -> false};  
traject      = {name -> "cavity", frequency -> 1, mode -> analyze};
```

5.37 Translate

5.37.1 Syntax

```

translate      = {
  site         -> {constant, ...},
  group        -> {constant, ...},
  cluster      -> {constant, ...}
  delta        -> vector
};

```

Directive	Parameters	Description
sites	constant	Sets site or sites to focus on; refers to mass type constants.
groups	constant	Sets group or groups to focus on.
clusters		
delta	vector	Sets the displacement vector

5.37.2 Usage

The `translate` command facilitates translation of a selection of sites over a displacement vector `delta`. This scripting command can be helpful in translating surfaces constructed in the center of the simulation box to its edges.

5.37.3 Default

The default is given by

```

translate      = {
  vector       -> {0, 0, 0}
};

```

5.37.4 Examples

An example is given by

```

translate      = {
  cluster      -> surface,
  delta        -> {-lx/2, 0, 0}
};

```

which translates clusters called 'surface' to the lower edge of the simulation box. Note, that `lx` has to be provided by the user.

5.38 Types

5.38.1 Syntax

```
types          = {
  merge        -> boolean,
  virial        -> boolean,
  periodic      -> vector,
  neighbor      -> constant,
  stencil       -> constant,
  skin          -> real,
  shake         -> constant
  depth        -> integer,
  mass         -> struct,
  fields...
};
```

Directive	Parameters	Description
<code>merge</code>	<code>boolean</code>	Allows for merging force field constants upon input when <code>true</code> ; options are <code>true</code> or <code>false</code> .
<code>virial</code>	<code>boolean</code>	Describes if virial calculations are included; options are <code>true</code> or <code>false</code> .
<code>periodic</code>	<code>vector</code>	Indicate periodicity with a three-element boolean vector with options <code>true</code> or <code>false</code> .
<code>neighbor</code>	<code>constant</code>	Describes what kind of neighbor list algorithm is used during pair interaction calculations; options are <code>sector</code> or <code>pair</code> .
<code>stencil</code>	<code>constant</code>	Describes the kind of stencil used during pair interaction calculations; options are <code>standard</code> or <code>multi</code> .
<code>skin</code>	<code>real</code>	Describes the skin used during pair interaction calculations; the skin is added to the pairwise cutoff.
<code>shake</code>	<code>constant</code>	Indicates the use of the SHAKE algorithm in subsequent codes (e.g. LAMMPS); valid options are <code>none</code> , <code>auto</code> , <code>hydrogen</code> , <code>water</code> , or <code>all</code> .
<code>depth</code>	<code>integer</code>	Maximum depth used for construction of ring structures during typing; allowed values are positive, where a value of 8 works in most ring cases; alternatively, an <code>auto</code> keyword allows for checking rings of unknown size; please note, that significant slow down occurs with the latter options for intricate ring systems.
<code>mass</code>	<code>struct</code>	Describes the site masses.
<code>fields</code>		Access to various field settings; see Section 7.12 [Types], page 190, for further information.

5.38.2 Usage

This variable style describes types (see Section 7.12 [Types], page 190, for further information) and allows direct access to all variables and parameters stored within the types structure.

5.38.3 Default

Unless otherwise stated, the default is given by

```
types          = {  
  merge        -> false,  
  virial        -> false,  
  periodic      -> {true, true, true}  
  neighbor      -> sector,  
  stencil       -> standard,  
  shake         -> none,  
  depth        -> auto,  
  skin         -> 0  
};
```

By default, all force fields are deactivated.

5.39 Variables

5.39.1 Syntax

```
variables      = {variable -> string|integer|real, ...};
```

Directive	Parameters	Description
variable	string, integer, real	Sets or redefine a variable; currently supported format only include scalars of types string , integer , and real .

5.39.2 Usage

The **variable** command is used to define or override previously defined variables, which allows for e.g. resizing of simulation cells as defined in the original data file. Also, simulation-wide string variables can be set, which later can be used as file names, etc. Currently, only scalar variables are supported. Future releases will include structured variables as well.

5.39.3 Examples

Possible definitions include

```
variables      = {la -> 0.95*la, lb -> 1.05*lb};  
variables      = {name -> "test/polyethylene"};
```

5.40 XYZ

5.40.1 Syntax

```
xyz          = {name -> string, compress -> boolean,
                system -> integer, forcefield -> option,
                cut -> boolean, detect -> boolean,
                segment -> option, residue -> option,
                pbc -> boolean, unwrap -> boolean};
```

Directive	Parameters	Description
name	string	Defines the file name to be imported; an extension of .emc is assumed.
compress	boolean	Sets compression using Lempel-Ziv coding (LZ77); an extra extension of .gz is assumed; options are true or false .
system	integer	Selects system to be imported or exported
forcefield	option	Selects the desired force field; valid options are none , standard , or charmm .
cut	boolean	Selects cut mode: cut bonds that cross cell boundaries; options are true or false .
detect	boolean	Detect and assign atom and amino acid residue types; options are true or false .
segment	option	Sets options for segment representation; options are detect for detection, index for use of EMC cluster ids, or series for increments starting with the letter M.
residue	option	Sets options for residue representation; options are detect for detection of amino acid residues, index for use of EMC group ids, or series for increments starting with the letter R.
pbc	boolean	Applies periodic boundary conditions; options are true or false .
unwrap	boolean	Unwraps clusters to cross periodic boundaries with application of periodic boundary conditions to its center of mass when applicable; options are true or false .

5.40.2 Usage

The xyz command is used to export data files in XYZ format for convenient visualization with VMD (Visual Molecular Dynamics), a visualization conceived and supported by the Theoretical and Computational Biophysics Group at the University of Illinois at Urbana-Champaign. Textual XYZ formats are ambiguous and are not self-defining, which creates the need for predefinition of the desired format. This port allows for commonly used force field modes.

5.40.3 Default

The default is given by

```
xyz                = {compress -> false,  
                      system -> 0, forcefield -> standard,  
                      cut -> false, detect -> false,  
                      segment -> index, residue -> index,  
                      pbc -> true, unwrap -> true};
```

5.40.4 References

1. VMD - Visual Molecular Dynamics, '<http://www.ks.uiuc.edu/Research/vmd/>'
2. Humphrey, W., Dalke, A. and Schulten, K., "VMD - Visual Molecular Dynamics", *J. Molec. Graphics* **1996**, *14*, 33-38

6 Sampling Tools

EMC offers the possibility of applying sampling either on the fly or by application to pregenerated trajectories (in `.traject` format). Available sampling algorithms include radii of gyration, pair correlation functions, cavity size distributions, energy and density profiles. The latter profiles offer a convolution option, which promotes smoothing out results by spreading point measurables over spherical volumes. Sampling is accessed through the scripting command `sample` (see Section 5.29 [Sample], page 135). Examples can be found in `./examples/build/sample`.

6.1 Bond

6.1.1 Syntax

```
gyration      = {
  id           -> integer,
  active       -> boolean,
  frequency    -> integer,
  source       -> struct,
  target       -> struct,
  binsize      -> real,
  distributions -> struct
};
```

Directive	Parameters	Description
id	integer	Method identifier for referencing subsequent alterations.
active	boolean	Indicates the state of the method; options are true or false .
frequency	integer	Sets the frequency of the method, which is assumed to be a positive integer.
source, target	struct	Definition of clusters, groups, and sites as to which to focus analysis on (see Section 7.3 [Focus], page 176); both contributing types need to be included for a specific bond; all mobile and active sites are taken when not defined.
binsize	real	Sets the binsize of the resulting distributions.
distributions	struct	Distributions resulting from analysis; these distributions - only shown in .emc and exported .m files - cannot be influenced with scripting command sample .

6.1.2 Usage

The **bond** command is used to sample bond lengths, which are indicated by defining pairs withing **focus**. Resulting distributions are reported as frequency distributions, which can be exported by using the scripting command **export** (see Section 5.9 [Export], page 98). Averages are reported as part of the resulting distribution as shown in .emc and .m files. An example can be found in `./examples/build/sample/bond/` by running `setup.sh`.

6.1.3 Default

The default is given by

```
sample      = {
  id         -> 0,
  active     -> false,
  frequency  -> 1,
  source     -> {},
  target     -> {},
  binsize    -> 0.01
};
```

6.1.4 Example

The example as provided in `./examples/sample/bond/` creates a system of octane molecules using the TraPPE-UA force field (see Section 7.12.19 [TraPPE], page 324). The EMC build script is generated by `emc_setup.pl`, which uses `chemistry.csv` to define the octane chemistry (see Chapter 4 [Simulation Setup], page 8). The final system contains a total of 1000 beads. The `setup.sh` calls EMC to build and sample. The sampling script `sample.emc` is explained in the following. First, a set of variables are set, which are used later in the script.

```
variables      = {
  input        -> "example",
  output       -> "test",
  ncycles      -> 1000,
  nblocks      -> 100,
  frequency    -> 1
};
```

Next, the created structure `example.emc.gz` is loaded.

```
get            = {name -> input, compress -> true};
```

The structure should first be equilibrated before sampling commences. To this effect, a simulation is executed for `ncycles` cycles.

```
run            = {
  ncycles      -> ncycles,
  nblocks      -> nblocks
};
```

The bond lengths of bonds between CH3 and CH2 types and between CH2 types are to be sampled. To this end, sampling of bonds is defined in the `sample` paragraph. The TraPPE type for CH3 groups is `c4h3` and for CH2 groups `c4h2`.

```
sample         = {
  bond         -> {
    {
      id        -> 0,
      source    -> {sites -> c4h3},
      target    -> {sites -> c4h2},
      active    -> true, frequency -> frequency, binsize -> 0.001
    },
    {
      id        -> 1,
      source    -> {sites -> c4h2},
      target    -> {sites -> c4h2},
      active    -> true, frequency -> frequency, binsize -> 0.001
    }
  }
};
```

```

    }
  }
};

```

The simulation is continued for another `ncycles`. Sampling occurs during the execution of the simulation.

```

run          = {
  ncycles    -> ncycles,
  nblocks    -> nblocks
};

```

Trajectories can also be used for sampling analysis. To this effect, a trajectory can be loaded by using the `traject` keyword.

```

traject      = {name -> name, frequency -> 1, mode -> analyze};

```

Sampling of trajectories occurs when the `mode` option `analyze` is selected. The resulting distributions can be exported to either `math` and `csv` formats.

```

export       = {
  sample     -> {name -> "bond", type -> bond, style -> math},
  sample     -> {name -> "bond", type -> bond, style -> csv}
};

```

The equilibrated and resulting structure is stored in EMC format by means of

```

put          = {name -> output, compress -> true};

```

6.2 Cavity

6.2.1 Syntax

```
cavity      = {
  id         -> integer,
  active     -> boolean,
  frequency  -> integer,
  focus      -> struct,
  solver     -> option,
  record     -> string,
  separate   -> boolean,
  ninherits  -> integer,
  ntrials    -> integer,
  npoints    -> integer,
  tolerance  -> real,
  cutoff     -> real,
  binsize    -> real,
  distributions -> struct,
  nerrors    -> struct,
  zero       -> boolean,
  negative   -> boolean
};
```

Directive	Parameters	Description
id	integer	Method identifier for referencing subsequent alterations.
active	boolean	Indicates the state of the method; options are true or false .
frequency	integer	Sets the frequency of the method, which is assumed to be a positive integer.
focus	struct	Defines sites, groups, or clusters to focus on (see Section 7.3 [Focus], page 176).
solver	option	Selects the solver used in finding local minima or saddle points in the energy surface; available options are broyden or newton .
record	string	Sets the output file name for the positions and sizes of found cavities; ignored when no file name is set.
separate	boolean	Toggles whether to write recorded output in one single or separate files; options are true or false .
ninherits	integer	Sets the number of initially inserted points for finding local minima or saddle points.
niterations	integer	Sets the maximum number of iteration steps allowed in finding the local minima or saddle points.
npoints	integer	Sets the number of points used to determine the cavity size by means of the root of the local energy.

tolerance	real	Sets the tolerance for the zero force at the local minimum or saddle point.
cutoff	real	Sets the cutoff in Jacobian elements used to discriminate between minima and saddle points.
binsize	real	Sets the binsize of the resulting distributions.
distributions	struct	Distributions resulting from analysis; local minima and saddle points distributions are reported per system in alternating fashion.
nerrors	struct	Resulting bookkeeping of algorithm performance.
zero	boolean	Include negative sizes as zero size in distribution; negative sizes are ignored when false ; superceedes negative when true ; options are true or false .
negative	boolean	Include negative sizes in distribution; options are true or false .

6.2.2 Usage

The `cavity` command is used to sample cavity size distributions of all morphologies entailed by a simulation according to the algorithm as described by In 't Veld et al.[1] The algorithm uses either a Newton or Broyden iterative scheme in combination with a line search to determine local minima or saddle point in the surface of the energy landscape. This point is then used to grow a bead to such an extent, that the resulting energy equals zero. Distinction between minima and saddle points is made by requiring the Jacobian to be definate positive for minima. The definition of "definate positive" can be adjusted by means of the `cutoff` option. An example can be found in `./examples/build/sample/cavity/` by running `build.emc`. An example of trajectory analysis is given by `traject.emc` in the same examples directory.

6.2.3 Default

The default is given by

```
sample      = {
  active      -> false,
  frequency   -> 1,
  focus       -> {},
  solver      -> newton,
  record      -> "",
  separate    -> false,
  ninsersts   -> 1000,
  ntrials     -> 200,
  npoints     -> 10,
  tolerance   -> 1e-8,
  cutoff      -> 1e-6,
  binsize     -> 0.01,
  zero        -> false,
  negative    -> false
};
```

6.2.4 Example

The following example samples the cavity size distribution of an FCC lattice, which consists of a tetrahedral and an octahedral cavity. Its scripting equivalent can be found in `./examples/sample/cavity/build.emc`. This example will be discussed in parts in the following. First, a number of variables are set,

```
variables      = {
  lattice      -> $root+"lib/fcc",
  output       -> "lattice",
  n            -> 4
};
```

Here the variable `lattice` describes the location of a predefined FCC lattice in EMC format. The final resulting positions are stored in filenames with starting with `lattice`. The variable `n` holds the number of replicas. An FCC lattice is obtained from the library directory by

```
get            = {name -> lattice};
```

after which it is replicated `n` times in all directions by

```
crystal       = {n -> {n, n, n}};
```

thus creating a superlattice. Subsequently, sampling of cavities using a Newton solver is turned on by

```
sample        = {
  cavity       -> {
    active     -> true,
    solver     -> newton,
    record     -> "cavity",
    binsize    -> 0.001,
    ninsets    -> 1000
  }
};
```

One thousand inserts are executed for each sampled structure. Cavity and saddle point positions are stored in file "cavity.pdb" by defining `record`. The resulting distribution has a bin size of 0.001. All omitted parameter definitions will internally be set to the above mentioned defaults. The selected FCC lattice does not include a force field definition, Force field parameters – which are not included in the selected FCC lattice – can be defined by using the `simulation` scripting command,

```
simulation     = {
  output       -> {debug -> false},
  units       -> {seed -> -1},
  types       -> {
```

```

    standard    -> {
      pair      -> {
        active   -> true,
        cutoff   -> 2.5,
        data     -> {
          {i0     -> 0, i1 -> 0, sigma -> 1, epsilon -> 1}
        }
      }
    },
    moves        -> {
      displace   -> {active -> false}
    }
  };

```

Here, the standard Lennard-Jones force field is chosen. Subsequently, displacement moves are switched off, thus not allowing particles to move. The seed value -1 triggers the use of the system clock as a seed value. A trajectory file is created through

```
traject          = {name -> "cavity", frequency -> 1};
```

The resulting configuration is run for 0 cycles in order to sample the cavity size distribution,

```
run              = {ncycles -> 0};
```

Once the cavity size distribution has been sampled, it is stored in both a Mathematica and a comma separated value format through

```

export          = {
  sample        -> {name -> "cavity", type -> cavity, style -> math},
  sample        -> {name -> "cavity", type -> cavity, style -> csv}
};

```

The superlattice is stored in both EMC native and PDB formats by means of

```

put              = {name -> output};
pdb              = {name -> output};

```

The above described script is run with

```
emc_linux build.emc 2>&1 | tee build.out
```

A VMD script called `cavity.vmd` has been provided in the same directory. This script allows for simultaneous visualization of particles, octahedral and tetrahedral cavities, demarcated with types A and B respectively. The script is called with

```
vmd -e cavity.vmd
```

The color of the solid particles identifies the size of the cavity. Note, that due to the statistical nature of the algorithm, not all cavities necessarily will be found. Thus, possibly not all cavities will be displayed.

6.2.5 References

1. P.J. in 't Veld, M.T. Stone, T.M. Trustkett, and I.C. Sanchez, "Liquid Structure via Cavity Size Distributions", *J. Phys. Chem. B* **2000**, *104*, 12028

6.3 Gr

6.3.1 Syntax

```

gr          = {
  id         -> integer,
  active     -> boolean,
  frequency  -> integer,
  source     -> struct,
  target     -> struct,
  cutoff     -> real,
  binsize    -> real,
  intra     -> struct,
  inter     -> struct,
  total     -> struct
};

```

Directive	Parameters	Description
id	integer	Method identifier for referencing subsequent alterations.
active	boolean	Indicates the state of the method; options are true or false .
frequency	integer	Sets the frequency of the method, which is assumed to be a positive integer.
source, target	struct	Definition of source and target clusters, groups, and sites (see Section 7.3 [Focus], page 176); all mobile and active sites are taken when not defined.
cutoff	real	Sets the extent to which distances are evaluated; a zero value will result in initialization with the maximum cutoff resulting from active pairwise potential definitions.
binsize	real	Sets the binsize of the resulting distributions.
intra, inter, total	struct	Distributions resulting from analysis, reporting intramolecular, intermolecular, and all distances resulting from pairwise combinations as specified by source and target structures.

6.3.2 Usage

The **gr** command is used to sample radial distribution functions of all morphologies entailed by a simulation. Intramolecular, intermolecular, and total distributions are sampled based on the specified source and target definitions. Resulting distributions are reported as frequency distributions, which can be exported by using the scripting command **export** (see Section 5.9 [Export], page 98). An example can be found in `./examples/build/sample/gr/` by running `setup.sh`.

6.3.3 Default

The default is given by

```

sample      = {

```

```

id          -> 0,
active      -> false,
frequency   -> 1,
source      -> {},
target      -> {},
cutoff      -> 0,
binsize     -> 0.01
};

```

6.3.4 Pair Correlation Functions

The following has been borrowed from Wikipedia¹. In statistical mechanics, the radial distribution function, (or pair correlation function) $g(r)$ in a system of particles (atoms, molecules, colloids, etc.), describes how density varies as a function of distance from a reference particle.

If a given particle is taken to be at the origin O , and if $\rho = N/V$ is the average number density of particles, then the local time-averaged density at a distance r from O is $\rho g(r)$. This simplified definition holds for a homogeneous and isotropic system. A more general case will be considered below.

In simplest terms it is a measure of the probability of finding a particle at a distance of r away from a given reference particle, relative to that for an ideal gas. The general algorithm involves determining how many particles are within a distance of r and $r + dr$ away from a particle. This general theme is depicted to the right, where the red particle is our reference particle, and blue particles are those which are within the circular shell, dotted in orange.

The RDF is usually determined by calculating the distance between all particle pairs and binning them into a histogram. The histogram is then normalized with respect to an ideal gas, where particle histograms are completely uncorrelated. For three dimensions, this normalization is the number density of the system multiplied by the volume of the spherical shell, which mathematically can be expressed as $g(r)_I = 4\pi r^2 \rho dr$, where ρ is the number density.

Given a potential energy function, the radial distribution function can be computed either via computer simulation methods like the Monte Carlo method, or via the Ornstein-Zernike equation, using approximative closure relations like the Percus-Yevick approximation or the Hypernetted Chain Theory. It can also be determined experimentally, by radiation scattering techniques or by direct visualization for large enough (micrometer-sized) particles via traditional or confocal microscopy.

The radial distribution function is of fundamental importance since it can be used, using the Kirkwood–Buff solution theory, to link the microscopic details to macroscopic properties. Moreover, by the reversion of the Kirkwood–Buff theory, it is possible to attain the microscopic details of the radial distribution function from the macroscopic properties.

6.3.5 Definition

Consider a system of N particles in a volume V (for an average number density $\rho = N/V$) and at a temperature T (let us also define $\beta = \frac{1}{kT}$). The particle coordinates are \vec{r}_i , with $i = 1, \dots, N$. The potential energy due to the interaction between particles is $U_N(\vec{r}_1, \dots, \vec{r}_N)$ and we do not consider the case of an externally applied field.

The appropriate averages are taken in the canonical ensemble (N, V, T) , with $Z_N = \int \dots \int e^{-\beta U_N} d\vec{r}_1 \dots d\vec{r}_N$ the configurational integral, taken over all possible combinations of particle positions. The probability of an elementary configuration, namely finding particle 1 in $d\vec{r}_1$, particle 2 in $d\vec{r}_2$, etc. is given by

$$P^{(N)}(\vec{r}_1, \dots, \vec{r}_N) d\vec{r}_1 \dots d\vec{r}_N = \frac{e^{-\beta U_N}}{Z_N} d\vec{r}_1 \dots d\vec{r}_N.$$

The total number of particles is huge, so that $P^{(N)}$ in itself is not very useful. However, one can also obtain the probability of a reduced configuration, where the positions of only $n < N$ particles are fixed, in $\vec{r}_1, \dots, \vec{r}_n$, with no constraints on the remaining $N - n$ particles. To this end, one has to integrate the above equation over the remaining coordinates $\vec{r}_{n+1}, \dots, \vec{r}_N$:

$$P^{(n)}(\vec{r}_1, \dots, \vec{r}_n) = \frac{1}{Z_N} \int \dots \int e^{-\beta U_N} d\vec{r}_{n+1} \dots d\vec{r}_N.$$

The particles being identical, it is more relevant to consider the probability that any n of them occupy positions $\vec{r}_1, \dots, \vec{r}_n$ in any permutation, thus defining the n -particle density

$$\rho^{(n)}(\vec{r}_1, \dots, \vec{r}_n) = \frac{N!}{(N-n)!} P^{(n)}(\vec{r}_1, \dots, \vec{r}_n).$$

For $n = 1$, this equation gives the one-particle density which, for a crystal, is a periodic function with sharp maxima at the lattice sites. For a (homogeneous) liquid, it is independent of the position \vec{r}_1 and equal to the overall density of the system:

$$\frac{1}{V} \int \rho^{(1)}(\vec{r}_1) d\vec{r}_1 = \rho^{(1)} = \frac{N}{V} = \rho.$$

It is now time to introduce a correlation function $g^{(n)}$ by

$$\rho^{(n)}(\vec{r}_1, \dots, \vec{r}_n) = \rho^n g^{(n)}(\vec{r}_1, \dots, \vec{r}_n).$$

$g^{(n)}$ is called a correlation function, since if the atoms are independent from each other $\rho^{(n)}$ would simply equal ρ^n and therefore $g^{(n)}$ corrects for the correlation between atoms. From the latter two equations it follows that

$$g^{(n)}(\vec{r}_1, \dots, \vec{r}_n) = \frac{V^n N!}{N^n (N-n)!} \cdot \frac{1}{Z_N} \int \dots \int e^{-\beta U_N} d\vec{r}_{n+1} \dots d\vec{r}_N.$$

6.3.6 Relations Involving $g(\mathbf{r})$

6.3.6.1 Structure Factor

The second-order correlation function $g^{(2)}(\vec{r}_1, \vec{r}_2)$ is of special importance, as it is directly related (via a Fourier transform) to the structure factor of the system and can thus be determined experimentally using X-ray diffraction or neutron diffraction. If the system consists of spherically symmetric particles, $g^{(2)}(\vec{r}_1, \vec{r}_2)$ depends only on the relative distance between them, $\vec{r}_{12} = \vec{r}_2 - \vec{r}_1$. We will drop the sub- and superscript: $g(\vec{r}) \equiv g^{(2)}(\vec{r}_{12})$. Taking particle 0 as fixed at the origin of the coordinates, $\rho g(\vec{r}) dr = dn(\vec{r})$ is the number

of particles (among the remaining $N-1$) to be found in the volume $d\vec{r}$ around the position \vec{r} . We can formally count these particles as $dn(\vec{r}) = \langle \sum_{i \neq 0} \delta(\vec{r} - \vec{r}_i) \rangle d\vec{r}$, with $\langle \dots \rangle$ the ensemble average, yielding

$$g(\vec{r}) = \frac{1}{\rho} \langle \sum_{i \neq 0} \delta(\vec{r} - \vec{r}_i) \rangle = V \frac{N-1}{N} \langle \delta(\vec{r} - \vec{r}_1) \rangle,$$

where the second equality requires the equivalence of particles 1, ..., $N-1$. The formula above is useful for relating $g(\vec{r})$ to the static structure factor $S(\vec{q})$, defined by $S(\vec{q}) = 1/N \langle \sum_{ij} e^{-i\vec{q}(\vec{r}_i - \vec{r}_j)} \rangle$, since we have

$$\begin{aligned} S(\vec{q}) &= 1 + \frac{1}{N} \langle \sum_{i \neq j} e^{-i\vec{q}(\vec{r}_i - \vec{r}_j)} \rangle = \\ &= 1 + \frac{1}{N} \left\langle \int_V e^{-i\vec{q}\vec{r}} \sum_{i \neq j} \delta[\vec{r} - (\vec{r}_i - \vec{r}_j)] d\vec{r} \right\rangle = \\ &= 1 + \frac{N(N-1)}{N} \int_V e^{-i\vec{q}\vec{r}} \langle \delta(\vec{r} - \vec{r}_1) \rangle d\vec{r}, \end{aligned}$$

and thus

$$S(\vec{q}) = 1 + \rho \int_V e^{-i\vec{q}\vec{r}} g(\vec{r}) d\vec{r},$$

proving the Fourier relation alluded to above. This equation is only valid in the sense of distributions, since $g(\vec{r})$ is not normalized: $\lim_{r \rightarrow \infty} g(\vec{r}) = 1$, so that $\int_V d\vec{r} g(\vec{r})$ diverges as the volume V , leading to a Dirac peak at the origin for the structure factor. Since this contribution is inaccessible experimentally we can subtract it from the equation above and redefine the structure factor as a regular function,

$$S'(\vec{q}) = S(\vec{q}) - \rho \delta(\vec{q}) = 1 + \rho \int_V e^{-i\vec{q}\vec{r}} [g(\vec{r}) - 1] d\vec{r}.$$

Finally, we rename $S(\vec{q}) \equiv S'(\vec{q})$ and, if the system is a liquid, we can invoke its isotropy,

$$S(q) = 1 + \rho \int_V e^{-i\vec{q}\vec{r}} [g(r) - 1] d\vec{r} = 1 + 4\pi\rho \frac{1}{q} \int r \sin(qr) [g(r) - 1] dr.$$

6.3.6.2 Compressibility Equation

Evaluating the latter equation for $q = 0$ while using the relation between the isothermal compressibility χ_T and the structure factor at the origin yields the compressibility equation:

$$\rho kT \chi_T = kT \left(\frac{\partial \rho}{\partial p} \right) = 1 + \rho \int_V [g(r) - 1] d\vec{r}.$$

6.3.6.3 Potential of Mean Force

It can be shown that the radial distribution function is related to the two-particle potential of mean force $w^{(2)}(r)$ by²

$$g(r) = \exp \left[-\frac{w^{(2)}(r)}{kT} \right].$$

6.3.6.4 Energy Equation

If the particles interact via identical pairwise potentials³,

$$U_N = \sum_{i>j=1}^N u(|\vec{r}_i - \vec{r}_j|),$$

the average internal energy per particle is

$$\frac{\langle E \rangle}{N} = \frac{3}{2}kT + \frac{\langle U_N \rangle}{N} = \frac{3}{2}kT + \frac{1}{2}\rho \int_V d\vec{r} u(r)g(r, \rho, T).$$

6.3.6.5 Pressure Equation of State

Developing the virial equation yields the pressure equation of state,

$$p = \rho kT - \frac{1}{6}\rho^2 \int_V r g(r, \rho, T) \frac{du(r)}{dr} d\vec{r}.$$

6.3.6.6 Thermodynamic Properties

The radial distribution function is an important measure because several key thermodynamic properties, such as potential energy and pressure can be calculated from it. For a 3-D system where particles interact via pairwise potentials, the potential energy of the system can be calculated as follows⁴

$$PE = 2\pi\rho N \int_0^\infty r^2 u(r)g(r)dr,$$

where N is the number of particles in the system, ρ is the number density, $u(r)$ is the pair potential. The pressure of the system can also be calculated by relating the 2nd virial coefficient to $g(r)$. The pressure can be calculated as follows

$$P = \rho k_B T - \frac{2}{3}\pi\rho^2 \int_0^\infty r^3 g(r) \frac{du(r)}{dr} dr,$$

where T is the temperature and k_B is Boltzmann's constant. Note that the results of potential and pressure will not be as accurate as directly calculating these properties because of the averaging involved with the calculation of $g(r)$.

6.3.7 References

1. N. N., Radial Distribution Function, Wikipedia (1/1/2016).
2. Chandler, D., Introduction to Modern Statistical Mechanics. Chapter 7.3, Oxford University Press (1987).
3. Hansen, J. P. and McDonald, I. R., Theory of Simple Liquids (3rd ed.), Academic Press (2005).
4. Frenkel, D. and Smit, B., Understanding Molecular Simulation from Algorithms to Applications (2nd ed.), Academic Press (2002).

6.4 Gyration

6.4.1 Syntax

```
gyration      = {
  id           -> integer,
  active       -> boolean,
  frequency    -> integer,
  focus        -> struct,
  cutoff       -> real,
  binsize      -> real,
  distributions -> struct
};
```

Directive	Parameters	Description
id	integer	Method identifier for referencing subsequent alterations.
active	boolean	Indicates the state of the method; options are true or false .
frequency	integer	Sets the frequency of the method, which is assumed to be a positive integer.
focus	struct	Definition of clusters, groups, and sites as to which to focus analysis on (see Section 7.3 [Focus], page 176); all mobile and active sites are taken when not defined.
cutoff	real	Sets the extent to which distances are evaluated; a zero value will result in initialization with the maximum cutoff resulting from active pairwise potential definitions.
binsize	real	Sets the binsize of the resulting distributions.
distributions	struct	Distributions resulting from analysis; these distributions - only shown in .emc and exported .m files - cannot be influenced with scripting command sample .

6.4.2 Usage

The **gyration** command is used to sample radius of gyration distributions of all morphologies entailed by a simulation. Resulting distributions are reported as frequency distributions, which can be exported by using the scripting command **export** (see Section 5.9 [Export], page 98). Averages are reported as part of the resulting distribution as shown in .emc and .m files. An example can be found in `./examples/build/sample/gyration/` by running `setup.sh`.

6.4.3 Default

The default is given by

```
sample      = {
  id         -> 0,
  active     -> false,
  frequency  -> 1,
  focus      -> {},
```

```

    cutoff      -> 0,
    binsize     -> 0.01
};

```

6.4.4 Theory

The following has been borrowed from Wikipedia¹. In polymer physics, the radius of gyration is used to describe the dimensions of a polymer chain. The radius of gyration of a particular molecule at a given time is defined as:

$$R_g^2 = \frac{1}{N^2} \sum_{i=1, j>i}^N (\vec{r}_i - \vec{r}_j)^2,$$

where N represents the number of particles in the molecule and vectors \vec{r}_i and \vec{r}_j denote particle positions. Since the chain conformations of a polymer sample are quasi infinite in number and constantly change over time, the "radius of gyration" discussed in polymer physics must usually be understood as a mean over all polymer molecules of the sample and over time. That is, the radius of gyration which is measured as an average over time or ensemble:

$$\langle R_g^2 \rangle = \frac{1}{N^2} \langle \sum_{i=1, j>i}^N (\vec{r}_i - \vec{r}_j)^2 \rangle,$$

where the angular brackets $\langle \dots \rangle$ denote the ensemble average. An entropically governed polymer chain (i.e. in so called theta conditions) follows a random walk in three dimensions. The radius of gyration for this case is given by

$$R_g = \frac{1}{\sqrt{6}} \sqrt{N} a$$

Note that although aN represents the contour length of the polymer, a is strongly dependent of polymer stiffness and can vary over orders of magnitude. N is reduced accordingly. One reason that the radius of gyration is an interesting property is that it can be determined experimentally with static light scattering as well as with small angle neutron- and x-ray scattering. This allows theoretical polymer physicists to check their models against reality. The hydrodynamic radius is numerically similar, and can be measured with Dynamic Light Scattering (DLS).

6.4.5 References

1. N. N., Radius of Gyration, Wikipedia (1/1/2016).

6.5 Interaction

6.5.1 Syntax

```

interaction      = {
  id             -> integer,
  active         -> boolean,
  frequency      -> integer,
  cutoff         -> real,
  mode           -> option,
  coulomb        -> option,
  bias           -> option,
  weight         -> option,
  ntrials        -> integer,
  nvolumes       -> integer,
  ninit          -> integer,
  nwidth         -> integer,
  source         -> struct,
  target         -> struct,
  mass           -> real,
  dvolume        -> real,
  volume1        -> struct,
  volume2        -> struct,
  denenergy      -> real,
  energy         -> struct
};

```

Directive	Parameters	Description
id	integer	Method identifier for referencing subsequent alterations.
active	boolean	Indicates the state of the method; options are true or false .
frequency	integer	Sets the frequency of the method, which is assumed to be a positive integer.
mode	global	Sets the desired pairwise potential cut off mode; options are global , individual , and repulsive
coulomb	none	Sets the desired type of coulombic interactions; options are none , cut , switch , and long .
bias	none	Sets the bias used during interaction sampling; options are none , energy , and frequency .
weight	none	Sets the weighting used for populating the resulting interaction distribution; options are none , boltzmann , and montecarlo .
ntrials	0	Sets the number of energetic interaction sampling trials.
nvolumes	100	Sets the number of volume sampling trials.

ninit	1000	Sets the number of trials for populating the initial distribution as used by energy and frequency biased sampling.
nwidth	1	Sets the spread used during sampling for accessing non-initialized bins resulting from the previous option.
source, target	struct	Definition of source and target clusters, groups, and sites (see Section 7.3 [Focus], page 176) setting the two interaction species; all mobile and active sites are taken when not defined.
cutoff	real	Sets the extent to which distances are evaluated; a zero value will result in initialization with the maximum cutoff resulting from active pairwise potential definitions.
dvolume	real	Sets the binsize of the resulting distributions.
volume1, volume2	real	Resulting distribution of source and target volume analysis; produces the volume distribution of the two interacting species.
denergy	real	Sets the binsize of the resulting energy distributions.
energy	struct	Distributions resulting from analysis, reporting pairwise interaction as specified by source and target structures.

6.5.2 Usage

The **interaction** command is used to sample the pairwise potential of mean force, which can either be calculated from provided or from internally generated structures. Internally generated structures are sampled over distances within the specified **cutoff** and over randomly chosen orientations. Resulting pairwise interactions can be weighted by several weighting techniques. Resulting samples per bin can be controlled by different biasing schemes. Volumetric and energetic distributions are sampled based on the specified source and target definitions. Resulting distributions are reported as frequency distributions, which can be exported by using the scripting command **export** (see Section 5.9 [Export], page 98).

6.5.3 Default

The default is given by

```
sample      = {
  id         -> 0,
  active     -> false,
  frequency  -> 1,
  mode       -> option,
  coulomb    -> option,
  bias       -> none,
  weight     -> none,
  ntrials    -> 0,
  nvolumes   -> 100,
  ninit      -> 1000,
  nwidth     -> 1,
  source     -> {},
  target     -> {},
```

```
    mass      -> 0,  
    cutoff    -> 0,  
    dvolume   -> 0.01,  
    denenergy -> 0.01  
};
```

6.6 Profiles

Density, pressure, and energy profile explanation.

6.7 Examples

Choice examples.

7 Variable Descriptions

EMC uses hierarchical variables, which can be recursive in character. This chapter describes the syntax of variables that are used often within the context of the EMC scripting language. Currently, variables use a representation similar to Mathematica.

Mathematical operations supported for scalar variable types `integer` and `real` are

Operator	Application	Description
+	$a + b$	Addition
-	$a - b$	Subtraction
*	$a * b$	Multiplication
/	a / b	Division
^	$a ^ b$	Power

Mathematical functions supported for scalar variable types `integer` and `real` are

Operator	Application	Description
<code>acos</code>	<code>acos(x)</code>	Arccosine using a radial basis
<code>asin</code>	<code>asin(x)</code>	Arcsine using a radial basis
<code>atan</code>	<code>atan(x)</code>	Arctangent using a radial basis
<code>cos</code>	<code>cos(x)</code>	Cosine using a radial basis
<code>eval</code>	<code>eval(expr)</code>	Evaluate a string expression into a value
<code>exp</code>	<code>exp(x)</code>	Exponential
<code>int</code>	<code>int(x)</code>	Floor of a real number
<code>log</code>	<code>log(x)</code>	Natural log
<code>sin</code>	<code>sin(x)</code>	Sine using a radial basis
<code>sqrt</code>	<code>sqrt(x)</code>	Square root
<code>tan</code>	<code>tan(x)</code>	Tangent using a radial basis
<code>mass</code>	<code>mass(index)</code>	Mass of a predefined group index
<code>mtotal</code>	<code>mtotal(focus)</code>	Mass of sites in selection as defined by focus (See Section 7.3 [Focus], page 176)
<code>nclusters</code>	<code>nclusters(focus)</code>	Number of clusters in selection as defined by focus (See Section 7.3 [Focus], page 176)
<code>nsites</code>	<code>nsites(index)</code>	Number of sites of a predefined group index
<code>ntotal</code>	<code>ntotal(focus)</code>	Number of sites in selection as defined by focus (See Section 7.3 [Focus], page 176)
<code>type</code>	<code>type(index)</code>	Determine type number of site index (used for transfer to e.g. LAMMPS input scripts)
<code>vsites</code>	<code>vtotal(focus)</code>	Volume captured by selection as defined by focus (See Section 7.3 [Focus], page 176)
<code>vtotal</code>	<code>vtotal(focus)</code>	Total system volume as defined by system selection (See Section 7.3 [Focus], page 176)

Note, that groups must have been defined (see Section 5.16 [Groups], page 112) and a force field must have been applied (see Section 5.10 [Field], page 101) in order for `mass(index)` and `nsites(index)` to return correct values. See also `scripts/emc_setup.pl`.

Reverved variable names are represented by the following constants

Constant	Type	Description
<code>\$root</code>	<code>string</code>	EMC root directory
<code>\$arg#</code>	<code>varies</code>	command line argument
<code>pi</code>	<code>real</code>	number: 3.14159265358979323846264338327950288
<code>e</code>	<code>real</code>	number: 2.71828182845904523536028747135266250
<code>true</code>	<code>integer</code>	number: 0
<code>false</code>	<code>integer</code>	number: 1
<code>null</code>	<code>integer</code>	number: -1

Internally defined constants are booleans `true` and `false`, mathematical constants `pi` and `e`, internal variable `null`, and the program's root directory `$root`. For the latter, it is assumed, that the `emc` executable resides in `$root/bin`. All other locations will result in erroneous behavior of `$root`. Command line arguments not preceeded by a "-" are accessed through the prefix `$arg` followed by a number, starting at 0. For example

```
variables = {name -> $arg0};
```

sets the variable `name` to the first command line argument. In case of the command

```
emc_${HOST} build.emc atoms
```

the variable `name` in the above example would be set to the text `"atoms"`. Note, that – in this example – calling `$arg1` and subsequent non-exisiting arguments as a variable will result in an error due to the fact, that `$arg1` is not defined on the command line and therefore also not internally. Addition operations using `a + b`, where either `a` or `b` is of variable type `string`, will result in a string. For example

```
"text" + 1 + 1 := "text11"
```

but

```
"text" + (1 + 1) := "text2"
```

Previously defined variables can also be included

```
a = 2^2
```

```
"text" + a := "text4"
```

or

```
a = 2*3
```

```
dir = "/home/user/text"
```

```
dir + "_" + a := "/home/user/text_6"
```

7.1 Constants

7.1.1 Syntax

```
constants      = {
  systems      -> {constant, ...},
  clusters     -> {constant, ...},
  groups       -> {constant, ...},
  sites        -> {constant, ...}
};
```

Directive	Parameters	Description
<code>systems</code>	<code>constant</code>	Adds a constant to the systems category; constants are expected to be alpha-numerical.
<code>clusters</code>	<code>constant</code>	Adds a constant to the clusters category; constants are expected to be alpha-numerical.
<code>groups</code>	<code>constant</code>	Adds a constant to the groups category; constants are expected to be alpha-numerical.
<code>sites</code>	<code>constant</code>	Adds a constant to the sites category; constants are expected to be alpha-numerical.

7.1.2 Usage

This variable style describes constants. The style is additive when used in combinations with see Section 5.39 [Variables], page 148.

7.1.3 Default

Unless otherwise stated, the default is given by

```
constants      = {
  systems      -> {},
  clusters     -> {},
  groups       -> {},
  sites        -> {}
};
```

7.2 System Flags

7.2.1 Syntax

flag -> option

Valid options are, when set,

Directive	Parameters
ignore	Ignore any missing contribution and do not generate any output
complete	Complete any missing contribution
warn	Generate a warning when a contribution is missing
empty	Create an empty entry when a contribution is missing
error	Exit on errors resulting from missing contributions; all missing contributions will be listed before an exit on error occurs

7.2.2 Default

Unless otherwise stated, the default is given by

flag -> error

7.3 Focus

7.3.1 Syntax

```
focus      = {
  clusters  -> {constant, ...},
  groups    -> {constant, ...},
  sites     -> {constant, ...},
  systems   -> {constant, ...},
  ntrials   -> integer
};
```

Directive	Parameters	Description
clusters	constant	Refers to cluster constants as defined by the <code>constants</code> table.
groups	constant	Refers to group constants as defined by the <code>constants</code> table.
ntrials	integer	Defines the number of trials used to determine volume; volume is determined by means of Monte Carlo integration.
sites	constant	Refers to site constants as defined by the <code>constants</code> table.
systems	constant	Refers to system constants as defined by the <code>constants</code> table.

7.3.2 Usage

Defines a system sites subset by means of their constants identifiers (see Section 7.1 [Constants], page 174).

7.3.3 Default

The default is described by an empty list for all three contributors,

```
focus      = {
  clusters  -> {},
  groups    -> {},
  sites     -> {},
  systems   -> {},
  ntrials   -> 10000
};
```

7.4 Moves

7.4.1 Syntax

```

moves          = {
  ncycles      -> integer,
  cycle        -> integer,
  move         -> integer,
  body         -> struct,
  deform       -> struct,
  displace     -> struct,
  endbridge    -> struct,
  rebridge     -> struct,
  reptate      -> struct,
  rotate       -> struct,
  surface      -> struct,
  temper       -> struct
};

```

Directive	Parameters	Description
ncycles	integer	Sets the total number of simulation cycles.
cycle	integer	Sets the current simulation cycle
move	integer	Sets the current simulation move.
body	struct	Settings for displacing sites inside a body.
deform	struct	Settings for system box deformation (see Section 7.4.5 [Displace], page 179).
displace	struct	Settings for displacing sites in a system box.
endbridge	struct	Settings for recombining cluster ends in a system box.
rebridge	struct	Settings for rebridging clusters in a system box.
reptate	struct	Settings for reptating cluster ends in a system box.
rotate	struct	Settings for rotating cluster ends in a system box.
surface	struct	Settings for displacing sites on a body surface.
temper	struct	Settings for parallel tempering between systems.

7.4.2 Usage

This variable style describes moves. These moves encompass standard and advanced Monte Carlo moves.¹

7.4.3 Default

Unless otherwise stated, the default is given by

```

moves          = {
  ncycles      -> 0,
  cycle        -> 0,
  move         -> 0
};

```

By default, all moves are activated with zero frequency, with the exception of the displacement move (see Section 7.4.5 [Displace], page 179), which has a frequency of one.

7.4.4 References

1. P.J. in 't Veld, M. Hütter, and G.C. Rutledge, "Temperature-Dependent Thermal and Elastic Properties of the Interlamellar Phase of Semicrystalline Polyethylene by Molecular Simulation", *Macromolecules* **2006**, *39*, 439

7.4.5 Displace

7.4.5.1 Syntax

```
displace      = {
  active      -> boolean,
  frequency   -> integer,
  dlimit      -> real,
  nsites      -> integer,
  couple      -> boolean,
  n           -> integer,
  dmax        -> real,
  accept      -> struct
};
```

Directive	Parameters	Description
active	boolean	Indicates the state of the move; options are true or false .
frequency	integer	Sets the frequency of the move, which is assumed to be a positive integer.
dlimit	real	Set the maximum displacement limit.
couple	boolean	Couple all individual acceptances when internally determining displacement limits.
nsites	integer	Set the number of sites to be displaced within one move.
n	integer	Number of masses available in the simulation; controlled internally.
dmax	real	Array of adapted displacement limits based on simulation progress; internally determined using a feedback loop.
accept	struct	Array of accumulative accepted and total trials.

7.4.5.2 Usage

The **displace** move is used to displace sites by means of perturbation. The target acceptance is set to 50%.

7.4.5.3 Default

The default is given by

```
sample      = {
  active     -> true,
  couple     -> false,
  frequency  -> 1,
  nsites     -> 1
};
```


7.5 Port

7.5.1 Syntax

```
port                = {forcefield -> option};
```

Directive	Parameters	Description
forcefield	option	Sets the ported force field type.

Option	Description
none	No force field interpretation; bonded contributions are not included in the ported format.
auto	Automated guess of ported format; based on which force field family is currently active; only works correctly when just one force field family is activated.
boltzmann	Assumes Boltzmann force fields (see Section 7.12.4 [Boltzmann], page 193).
cff	Assumes CFF force fields (see Section 7.12.6 [CFF], page 206).
charmm	Assumes CHARMM force fields (see Section 7.12.7 [CHARMM], page 222).
dpd	Assumes DPD coarse-grained force fields (see Section 7.12.11 [DPD], page 261).
martini	Assumes MARTINI coarse-grained force fields (see Section 7.12.13 [MARTINI], page 279).
opls	Assumes OPLS all-atom and united-atom force fields (see Section 7.12.14 [OPLS], page 287).
sdk	Assumes SDK coarse-grained force fields (see Section 7.12.15 [SDK], page 296).
standard	Assumes standard force fields (see Section 7.12.17 [Standard], page 309).
table	Assumes tabulated force fields (see Section 7.12.18 [Table], page 318).
trappe	Assumes TraPPE all-atom and united-atom force fields (see Section 7.12.19 [TraPPE], page 324).
coarse	Assumes coarse-grained force fields.
colloid	Assumes colloidal force fields.
fene	Assumes FENE force fields.

7.5.2 Default

Unless otherwise stated, the default is given by

```
port                = {forcefield -> auto};
```

7.6 Profiles

7.6.1 Syntax

```
profiles      = {
  bond        -> struct,
  density      -> struct,
  force        -> struct,
  mass         -> struct,
  order        -> struct
};
```

Directive	Parameters	Description
bond	struct	Settings for bond length distributions of all bonds within all systems; no discrimination is made between bond types.
density	struct	Settings for mass density profiles; no discrimination is made between site masses.
force	struct	Settings for energy and virial density profiles; resulting binned contributions reflect the interaction of that bin with its surroundings.
mass	struct	Settings for mass profiles; distinction is made between contributing site masses.
order	struct	Settings for order profiles; no distinction is made between contributing bond types.

7.6.2 Usage

This variable style describes profiles.

7.6.3 Default

By default, all profiles are deactivated and all entries are zeroed out.

7.7 Region

7.7.1 Syntax

```
region          = {shape -> option, type -> option, mode -> option,
                  center -> vector, h -> voigt[, radius -> vector]}
```

Directive	Parameters	Description
shape	option	Sets region shape; possibilities are cuboid or spheroid .
type	option	Sets region type; possibilities are relative or absolute .
mode	option	Sets region mode; possibilities are hard or soft ; used in conjunction with sites growth.
center	vector	Defines the center of a region.
h	voigt	Defines the extent of a region as a voigt notation shape (see Section 7.15 [Voigt], page 335).
radius	vector	Defines the extent of a region as a vector (see Section 7.14 [Vector], page 334); the radius is a derived alternative to h and translated into h upon execution.

7.7.2 Usage

Specifies a region of a certain shape within system simulation cell. Units of the shape center and radius can either be expressed in a relative or an absolute fashion. Relative units express both shape center and radius in units of base vectors \vec{a} , \vec{b} , and \vec{c} and lengths thereof. Absolute units make use of the simulation-wide settings as expressed under in the 'Units' section (see Section 7.13 [Units], page 332) of the simulation structure. The region's mode influences the growth of sites by either **hard** or **soft** exclusion or inclusion. The **hard** mode applies the region to all sites of inserted clusters, while the **soft** mode only applies the region to the first site of inserted clusters.

7.7.3 Default

Unless otherwise stated, the default is given by

```
region          = {shape -> spheroid, type -> relative, mode -> hard,
                  center -> {0,0,0}, h -> {0,0,0,0,0,0}};
```

7.8 SMILES

7.8.1 Syntax

```
chemistry      = string;
```

7.8.2 Usage

The Simplified Molecular Input Line Entry System (or SMILES for short) is used to describe chemistry as reflected in the definitions of **groups** (see Section 5.16 [Groups], page 112). It follows the syntax as described Daylight Chemical Information Systems, Inc., but is extended to also capture non-periodic system representations as are used in defining coarse-grained systems. The distinction between periodic and non-periodic representations is identified by keywords **atomistic** and **coarse** respectively. In **atomistic** mode, hydrogen completion is implicit and lower character case signifies aromaticity, where allowed characters are **c**, **n**, **o**, **p**, and **s**. A short summary is listed in the table below. Extensions to the original SMILES format are marked with (extension).

Modifier	Example	Description
>X	>C	First branch point, connecting to chains in the down direction (extension).
X<	C<	Second branch point, connecting to chains in the up direction (extension).
*	*	General branch point, connecting chains (extension).
~X	~C	Represents any bond (extension).
-X	-C	Represents a single bond (extension).
:X	:C	Represents a partial double bond (extension).
=X	=C	Represents a double bond.
#X	#N	Represents a triple bond.
x	c	Lower case indicate aromatics in atomistic mode; allowed characters are c , n , o , p , and s .
[XX]	[He]	Needed for usage of symbols with more than one character.
()	CC(C)C	Indicates branching from backbone.
(X)1	(C)3	Indicates repeating sequences (here (C)3 = CCC); note, that CC(C)3C = CCCCCC and CC((C)3)C = CC(CCC)C; multi-digits are allowed (extension).
1X	2H	Assigns an aberrant mass to a symbol (here 2).
X1	c1ccccc1	Indicates a link number to create ring structures (here a benzene ring); duplicate link numbers are allowed.
%	c%11ccccc%11	Used for multi-digit link numbers.
\X or /X	C/C=C/C	Identifies cis (C/C=C\C) or trans (C/C=C/C) isomers.
@	[C@H]	Identifies L- ([C@@H]) or R- ([C@H]) stereoisomers.
+1	[Na+1]	Indicates an assigned positive charge; charges can be partial (e.g. 0.5); a + with omitted digits indicates an increase by +1, e.g. [Na+] equals [Na+1] or [Ca++] represents a calcium atom with charge +2.

-1 [C1-1] Indicates an assigned negative charge; charges can be partial (e.g. 0.5); a - with omitted digits indicates an decrease by -1.

7.8.3 Examples

The following table shows a few example chemistry and their corresponding SMILES strings.

Chemistry	SMILES
water	HOH
cyclohexanone	O=C1CCCCC1
dodecane	(C)12 CCCCCCCCCCCC
iso-octane	CC(C)CC(C)(C)C
diphenylmethane	c1ccccc1Cc1ccccc1
Methylene diphenyl 4,4'-diisocyanate	O=C=Nc1ccc(cc1)Cc2ccc(N=C=O)cc2
trans-2-butene	C/C=C/C
L-tryptophan	c1ccc2c(c1)c(c[nH]2)C[C@@H](C(=O)O)N

7.9 Splines

7.9.1 Introduction¹

In mathematics, a spline is a sufficiently smooth polynomial function that is piecewise-defined, and possesses a high degree of smoothness at the places where the polynomial pieces connect (which are known as knots).^{2,3}

In interpolating problems, spline interpolation is often referred to as polynomial interpolation because it yields similar results, even when using low-degree splines, to interpolating with higher degree polynomials while avoiding instability due to Runge's phenomenon. In computer graphics splines are popular curves because of the simplicity of their construction, their ease and accuracy of evaluation, and their capacity to approximate complex shapes through curve fitting and interactive curve design.

The most commonly used splines are cubic spline, i.e., of order 3 – particular, cubic B-spline and cubic Bezier spline. They are common, in particular, in spline interpolation simulating the function of flat splines. The term spline is adopted from the name of a flexible strip of metal commonly used by draftsmen to assist in drawing curved lines.⁴

Splines are curves, which are usually required to be continuous and smooth. Splines are usually defined as piecewise polynomials of degree n with function values and first $n-1$ derivatives that agree at the points where they join. The abscissa values of the join points are called knots. The term "spline" is also used for polynomials (splines with no knots) and piecewise polynomials with more than one discontinuous derivative. As such, splines with no knots are generally smoother than splines with knots, which are generally smoother than splines with multiple discontinuous derivatives. Splines with few knots are generally smoother than splines with many knots; however, increasing the number of knots usually increases the fit of the spline function to the data. Knots give the curve freedom to bend to more closely follow the data.

It is commonly accepted that the first mathematical reference to splines is the 1946 paper by Schoenberg,⁵ which is probably the first place that the word "spline" is used in connection with smooth, piecewise polynomial approximation. However, the ideas have their roots in the aircraft and shipbuilding industries.

7.9.2 Linear Spline

The linear splined applied by EMC follows

$$\begin{aligned}\Delta x_i &= x - x_i, \\ S_i(x) &= \sum_{j=0}^1 k_{j,i} \Delta x_i^j, \\ k_{0,i} &= y_i, \\ k_{1,i} &= (y_i - y_{i-1}) / (x_i - x_{i-1}).\end{aligned}$$

Linear splines find their application in cases of sudden large gradients or steps in y data with respect to x .

7.9.3 Cubic Spline

Internally, EMC applies a natural cubic spline by default. The algorithm behind a natural cubic spline is given by

$$\begin{aligned}
\Delta x_i &= x - x_i, \\
S_i(x) &= \sum_{j=0}^3 k_{j,i} \Delta x_i^j, \\
S_i(x_i) &= S_{i-1}(x_i) = y_i, \\
S'_i(x_i) &= S'_{i-1}(x_i), \\
S''_i(x_i) &= S''_{i-1}(x_i), \\
S''_i(x_0) &= S''_{n-1}(x_n) = 0,
\end{aligned}$$

for which the above set of equations defines all constants k needed to describe the full spline function. Cubic splines are useful for capturing the behavior of reasonably well-behaved data.

7.9.4 References

1. Wikipedia on "*Cubic Splines*"
2. K. L. Judd, "*Numerical Methods in Economics*", *MIT Press* **1998**, 225 (ISBN 978-0-262-10071-7).
3. W.-K. Chen, Wai-Kai (2009 "*Feedback, Nonlinear, and Distributed Circuits*", *CRC Press* **2009**, 9-20 (ISBN 978-1-4200-5881-9).
4. M. H. Katz, "*Multivariable Analysis: A Practical Guide for Clinicians and Public Health Researchers*", *Cambridge University Press* **2011**, 82 (ISBN 978-0-521-14107-9).
5. Schoenberg, "*Contributions to the problem of approximation of equidistant data by analytic functions*", *Quart. Appl. Math.* **1946**, 4, 45-99 and 112-141.

7.10 System Flags

7.10.1 Syntax

```
flag                = {charge -> boolean, map -> boolean, pbc -> boolean};
```

Directive	Parameters	Description
charge	boolean	Check for charge neutrality; options are <code>true</code> or <code>false</code> .
map	boolean	Map configurations into the system box; options are <code>true</code> or <code>false</code> .
pbc	boolean	Apply periodic conditions to system geometry; options are <code>true</code> or <code>false</code> .

7.10.2 Default

Unless otherwise stated, the default is given by

```
flag                = {charge -> true, map -> true, pbc -> true};
```


7.11 Systems

7.11.1 Syntax

```

systems      = {
  n          -> integer,
  properties  -> {
    {
      id      -> constant,
      p       -> real,
      v       -> real,
      t       -> real,
      mass    -> real,
      nclusters -> integer,
      nsites  -> struct,
      plane   -> integer,
      geometry -> voigt
    },
    ...
  }
};

```

Directive	Parameters	Description
n	integer	Sets the number of contributing systems to the total simulation.
properties		Describes the properties of each individual system.
id	constant	Identifies the system; refers to the systems paragraph in constants (see Section 7.1 [Constants], page 174).
p	real	Defines the system pressure; only used when deformation moves are active.
v	real	Defines the system volume.
t	real	Defines the system temperature.
mass	real	Reflects the total system mass; cannot be altered.
nclusters	integer	Reflects the number of clusters in this system; cannot be altered.
nsites	struct	Reflects the number of sites in this system; cannot be altered.
plane	integer	Reflects the original crystal plane the system was constructed with.
geometry	voigt	Defines the system geometry (see Section 7.15 [Voigt], page 335).

7.11.2 Usage

This variable style describes the definition of systems.

7.11.3 Default

Unless otherwise stated, the default is given by

`systems` `= {n -> 0};`

7.12 Types

7.12.1 Syntax

```
types          = {
  merge        -> boolean,
  virial        -> boolean,
  periodic      -> vector,
  neighbor      -> constant,
  stencil       -> constant,
  skin          -> real,
  shake         -> constant
  depth        -> integer,
  mass          -> struct,
  boltzmann    -> struct,
  charmm        -> struct,
  cff           -> struct,
  coarse        -> struct,
  colloid       -> struct,
  coulomb       -> struct,
  dpd           -> struct,
  gromacs       -> struct,
  inverse       -> struct,
  martini       -> struct,
  opl           -> struct,
  sdk           -> struct,
  spline        -> struct,
  standard      -> struct,
  table         -> struct,
  trappe        -> struct
};
```

Directive	Parameters	Description
merge	boolean	Allows for merging force field constants upon input when true; options are true or false .
virial	boolean	Describes if virial calculations are included; options are true or false .
periodic	vector	Indicate periodicity with a three-element boolean vector with options true or false .
neighbor	constant	Describes what kind of neighbor list algorithm is used during pair interaction calculations; options are sector or pair .
stencil	constant	Describes the kind of stencil used during pair interaction calculations; options are standard or multi .
skin	real	Describes the skin used during pair interaction calculations; the skin is added to the pairwise cutoff.

shake	constant	Indicates the use of the SHAKE algorithm in subsequent codes (e.g. LAMMPS); valid options are none , auto , hydrogen , water , or all .
depth	integer	Maximum depth used for construction of ring structures during typing; allowed values are positive, where a value of 8 works in most ring cases; alternatively, an auto keyword allows for checking rings of unknown size; please note, that significant slow down occurs with the latter options for intricate ring systems.
mass	struct	Describes the site masses.
boltzmann	struct	Describes Boltzmann force fields (see Section 7.12.4 [Boltzmann], page 193).
born	struct	Describes Born force fields (see Section 7.12.5 [Born], page 198).
cff	struct	Describes CFF (Class2) force field families (see Section 7.12.6 [CFF], page 206).
charmm	struct	Describes CHARMM force fields (see Section 7.12.7 [CHARMM], page 222).
coarse	struct	Describes multiple coarse-grained force field definitions (see Section 7.12.8 [Coarse], page 233).
colloid	struct	Describes colloidal force fields (see Section 7.12.9 [Colloid], page 249).
coulomb	struct	Describes coulombic contribution definitions (see Section 7.12.10 [Coulomb], page 257).
martini	struct	Describes MARTINI coarse-grained force fields (see Section 7.12.13 [MARTINI], page 279).
gromacs	struct	Describes GROMACS force fields (see Section 7.12.12 [GROMACS], page 270).
inverse	struct	Describes settings used in applications of inverse bonded interactions.
dpd	struct	Describes DPD coarse-grained force fields (see Section 7.12.11 [DPD], page 261).
opls	struct	Describes OPLS force fields (see Section 7.12.14 [OPLS], page 287).
sdk	struct	Describes SDK force fields (see Section 7.12.15 [SDK], page 296).
spline	struct	Describes spline-based force fields (see Section 7.12.16 [Spline], page 305).
standard	struct	Describes standard force fields (see Section 7.12.17 [Standard], page 309).
table	struct	Describes tabular force fields (see Section 7.12.18 [Table], page 318).
trappe	struct	Describes tabular force fields (see Section 7.12.19 [TraPPE], page 324).

7.12.2 Usage

This variable style describes types.

7.12.3 Default

Unless otherwise stated, the default is given by

```
types          = {  
  merge        -> false,  
  virial        -> false,  
  periodic      -> {true, true, true},  
  neighbor      -> sector,  
  stencil       -> standard,  
  shake         -> none,  
  depth         -> auto,  
  skin         -> 0  
};
```

By default, all force fields are deactivated.

7.12.4 Boltzmann

7.12.4.1 Syntax

```
boltzmann      = {
  bond         -> struct,
  angle        -> struct,
  pair         -> struct
};
```

Directive	Parameters	Description
bond	struct	Bond interaction descriptors
angle	struct	Angle interaction descriptors
pair	struct	Pair interaction descriptors

The boltzmann force field uses exponential functions for bonded interactions and tabulated functions for nonbonded interactions. Its functional form is a summation of bonded and nonbonded interactions given by

$$E_{\text{boltzmann}} = E_{\text{bond}} + E_{\text{angle}} + E_{\text{pair}}$$

for angle, bond and pair contributions respectively. The following paragraphs describe each contribution in detail. See `./examples/nanoparticle/coarse/` for an application.

7.12.4.2 Example

```
variables      = {
  a            -> 1*^-10,
  prefix       -> "~/emc/version/force/polystyrene/smoothed_"
};

simulation     = {
  types        -> {
    merge      -> true,
    boltzmann  -> {
      bond     -> {
        active  -> true,
        data    -> {
          {i0 -> m, i1 -> m, coefficients -> {
            a -> 0.015e-10, w -> 0.09e-10, length -> 2.46e-10}},
          }
        },
      angle    -> {
        active  -> true,
        data    -> {
          {i0 -> m, i1 -> m, i2 -> m, coefficients -> {
            {a -> 0.140, w -> 14.2, theta -> 147.3},
            {a -> 0.030, w -> 15.5, theta -> 158.0}}}
          }
        }
      }
    }
  }
};
```

```
    }  
  },  
  pair      -> {  
    active  -> true,  
    nbonded -> 2,  
    data    -> {  
      {i0 -> m, i1 -> m, order -> 2, name -> prefix+"m-m.m"}  
    }  
  }  
}  
}  
};
```

7.12.4.3 References

1. T. Spyriouni, C. Tzoumanekas, D. Theodorou, F. Mueller-Plathe, and G. Milano, "*Coarse-Grained and Reverse-Mapped United-Atom Simulations of Long-Chain Atactic Polystyrene Melts: Structure, Thermodynamic Properties, Chain Conformation, and Entanglements*", *Macromolecules* **2007**, *40*, 3876

7.12.4.4 Pair

```

bond          = {
  active      -> boolean,
  nbonded     -> integer,
  n           -> integer,
  data        ->
  {
    {
      i0       -> id,
      i1       -> id,
      cutoff   -> real,
      order    -> integer,
      name     -> string
    },
    ...
  }
};

```

Directive	Parameters	Description
active	boolean	Interaction activator; either <code>true</code> or <code>false</code> .
nbonded	integer	Number of bonded sites to exclude from nonbonded interactions; the given number is expected to be larger or equal to zero.
n	integer	Number of data entries; can be omitted when <code>type = {merge -> true}</code> .
data	struct	Summary of interaction constants
i0, i1	id	Site id, referring to <code>mass</code> paragraph in <code>types</code>
cutoff	real	Defines the cutoff of the potential; takes the last data file entry when <code>cutoff <= 0</code> .
order	integer	Interpolation order; either 1 for linear interpolation or 2 for cubic splines.
name	string	Defines the data file name; <code>./force/polystyrene/</code> shows examples of the data file format (i.e. <code>{x, y, ...}</code>).

The energetic functional form of the pair contributions to the total potential is formed by a linear interpolation or a cubic spline through the provided data.

7.12.4.5 Bond

```

bond          = {
  active      -> boolean
  n           -> integer,
  data        ->
  {
    {
      i0       -> id,
      i1       -> id,
      coefficients ->
      {
        {
          a     -> real,
          w     -> real,
          length-> real
        },
        ...
      }
    },
    ...
  }
};

```

Directive	Parameters	Description
active	boolean	Interaction activator; either <code>true</code> or <code>false</code> .
n	integer	Number of data entries; can be omitted when <code>type = {merge -> true}</code> .
data	struct	Summary of interaction constants
i0, i1	id	Site id, referring to <code>mass</code> paragraph in <code>types</code>
coef	struct	Describes individual coefficient contributions
a, w	real	Force constants
length	real	Equilibrium length

The energetic functional form of the bond contributions to the total potential is described by

$$E_{bond} = \sum_{i=1}^n a_i / (2\pi w_i) \exp[-2(l - l_{0,i})^2 / w_i^2],$$

where l represents the bond length.

7.12.4.6 Angle

```

angle      = {
  active    -> boolean
  n         -> integer,
  data      ->
  {
    {
      i0      -> id,
      i1      -> id,
      i2      -> id,
      coefficients ->
      {
        {
          a      -> real,
          w      -> real,
          theta  -> real
        },
        ...
      }
    },
    ...
  }
};

```

Directive	Parameters	Description
active	boolean	Interaction activator; either true or false .
n	integer	Number of data entries; can be omitted when type = {merge -> true} .
data	struct	Summary of interaction constants
i0, i1, i2	id	Site id, referring to mass paragraph in types
coef	struct	Describes individual coefficient contributions
a, w	real	Force constants
theta	real	Equilibrium angle

The energetic functional form of the angle contributions to the total potential is described by

$$E_{angle} = \sum_{i=1}^n a_i / (2\pi w_i) \exp[-2(\theta - \theta_{0,i})^2 / w_i^2],$$

where θ represents the bond angle.

7.12.5 Born

7.12.5.1 Syntax

```
born          = {
  bond        -> struct,
  angle       -> struct,
  torsion     -> struct,
  improper     -> struct,
  pair14      -> struct,
  pair        -> struct
};
```

Directive	Parameters	Description
bond	struct	Bond interaction descriptors
angle	struct	Angle interaction descriptors
torsion	struct	Torsion interaction descriptors
improper	struct	Improper interaction descriptors
pair14	struct	1-4 intra-molecular pair interaction activator
pair	struct	Pair interaction descriptors

The OPLS force field is a compounded force field based on the Born force field.¹⁻² Bonded terms are equivalent to OPLS force fields³. Its functional form is a summation of bonded and nonbonded interactions given by

$$E_{opls} = E_{pair} + E_{bond} + E_{angle} + E_{torsion} + E_{improper}$$

for bond, angle, torsion, and pair contributions respectively. The following paragraphs describe each contribution in detail. See `./examples/field/born/` for applications.

7.12.5.2 Example

```
simulation    = {
  types       -> {
    merge     -> true,
    born      -> {
      bond    -> {
        active -> true,
        data   -> {
          {i0   -> a, i1 -> b, k -> 100, l -> 2}
        }
      },
      angle   -> {
        active -> true,
        data   -> {
          {i0   -> a, i1 -> b, i2 -> c, k -> 100, theta -> 110}
        }
      },
    },
  },
};
```

```

torsion    -> {
  active    -> true,
  data      -> {
    {i0      -> a, i1 -> b, i2 -> c, i3 -> d, coefficients -> {
      {k -> 1.5, n -> 1, delta -> 0},
      {k -> 2.5, n -> 2, delta -> 0}
    }
  }
},
improper    -> {
  active    -> true,
  data      -> {
    {i0      -> a, i1 -> b, i2 -> c, i3 -> d, k -> 100, psi -> 0}
  }
},
pair        -> {
  active    -> true,
  nbonded   -> 3,
  mode      -> global,
  cutoff    -> 10,
  mix       -> berthelot,
  data      -> {
    {i0      -> a, i1 -> a, sigma -> 4.0, epsilon -> 0.4},
    {i0      -> b, i1 -> b, sigma -> 3.7, epsilon -> 0.5},
    {i0      -> c, i1 -> c, sigma -> 4.1, epsilon -> 0.22},
    {i0      -> d, i1 -> d, sigma -> 2.4, epsilon -> 0.09},
  }
}
};

```

7.12.5.3 References

1. F.G. Fumi and M.P. Tosi, *Ionic sizes and born repulsive parameters in the NaCl-type alkali halides—I: The Huggins-Mayer and Pauling forms*, *J. Phys. Chem. Solids* **1964**, 25, 31-44.
2. F.G. Fumi and M.P. Tosi, *Ionic sizes and born repulsive parameters in the NaCl-type alkali halides—II: The generalized Huggins-Mayer form*, *J. Phys. Chem. Solids* **1964**, 25, 31-44.
3. W. L. Jorgensen and J. Tirado-Rives, "The OPLS Potential Functions for Proteins. Energy Minimizations for Crystals of Cyclic Peptides and Crambin", *J. Am. Chem. Soc.* **1988**, 110, 1657-1666.

7.12.5.4 Pair

```
pair          = {
  active      -> boolean,
  nbonded     -> integer,
  mix         -> option,
  shift       -> boolean,
  coulomb     -> option,
  cutoff      -> real,
  mode        -> option,
  n           -> integer,
  data        ->
  {
    {
      i0      -> id,
      i1      -> id,
      sigma   -> real,
      rho     -> real,
      a       -> real,
      c       -> real,
      d       -> real,
      core    -> real,
      cutoff  -> real
    },
    ...
  }
};
```

Directive	Parameters	Description
active	boolean	Interaction activator; either true or false .
nbonded	integer	Number of bonded sites to exclude from nonbonded interactions; the given number is expected to be larger or equal to zero.
mix	option	Sets mixing rule; options are none , berthelot , arithmetic , geometric , and sixth .
shift	boolean	Sets shifting of potential at cutoff; options are true and false .
coulomb	option	Set type of coulomb treatment; options are none , cut , and long .
cutoff	real	Sets global cut off for global mode
mode	option	Sets cut off mode; options are global using the value of cutoff for all contributions, individual defining cut offs per contribution, and repulsive defining the well location for each separate contribution as the cut off distance.
n	integer	Number of data entries; can be omitted when type = {merge -> true} .
data	struct	Summary of interaction constants

<code>i0, i1</code>	<code>id</code>	Site id, referring to types as defined by the <code>mass</code> paragraph.
<code>sigma</code>	<code>real</code>	Interaction dependent parameter.
<code>rho</code>	<code>real</code>	Ionic pair dependent parameter.
<code>a</code>	<code>real</code>	Exponential prefactor.
<code>c</code>	<code>real</code>	Attractive parameter.
<code>d</code>	<code>real</code>	Repulsive parameter.
<code>core</code>	<code>real</code>	Distance at which the potential is treated as a purely repulsive potential; needed to avoid singular behavior at small distances.
<code>cutoff</code>	<code>real</code>	Cut off; distance at which the pair-wise contribution is set to zero; can be omitted.

The energetic functional form of the pair contributions to the total potential is described by a 6-12 Lennard-Jones potential,

$$E_{pair} = A \exp\left(\frac{\sigma - r_{ij}}{\rho}\right) - \frac{C}{r_{ij}^6} + \frac{D}{r_{ij}^8},$$

for $r_{ij} > r_{core}$, where r_{ij} represents the distance between site i and site j .

7.12.5.5 Bond

```

bond          = {
  active      -> boolean
  n           -> integer,
  data        ->
  {
    {
      i0       -> id,
      i1       -> id,
      k        -> real,
      l        -> real
    },
    ...
  }
};

```

Directive	Parameters	Description
active	boolean	Interaction activator; either <code>true</code> or <code>false</code> .
n	integer	Number of data entries; can be omitted when <code>type = {merge -> true}</code> .
data	struct	Summary of interaction constants
i0, i1	id	Site id, referring to types as defined by the <code>mass</code> paragraph in <code>types</code>
k	real	Bond spring constants
l	real	Equilibrium bond length

The energetic functional form of the bond contributions to the total potential is described by

$$E_{bond} = k_{bond}(l - l_0)^2,$$

where k_{bond} represents a spring constant, l the bond length of bond $\{i_0, i_1\}$, and l_0 the equilibrium bond length.

7.12.5.6 Angle

```

angle          = {
  active       -> boolean
  n            -> integer,
  data         ->
  {
    {
      i0        -> id,
      i1        -> id,
      i2        -> id,
      k         -> real,
      theta     -> real
    },
    ...
  }
};

```

Directive	Parameters	Description
active	boolean	Interaction activator; either <code>true</code> or <code>false</code> .
n	integer	Number of data entries; can be omitted when <code>type = {merge -> true}</code> .
data	struct	Summary of interaction constants
i0, i1, i2	id	Site id, referring to types as defined by the <code>mass</code> paragraph in <code>types</code>
k	real	Angle spring constants
theta	real	Equilibrium bond angle

The energetic functional form of the angle contributions to the total potential is described by

$$E_{angle} = k_{angle}(\theta - \theta_0)^2,$$

where k_{angle} represents a spring constant, θ the angle between bonds $\{i_0, i_1\}$ and $\{i_1, i_2\}$, and θ_0 the equilibrium angle.

7.12.5.7 Torsion

```

torsion      = {
  active      -> boolean
  n           -> integer,
  data        ->
  {
    {
      i0       -> id,
      i1       -> id,
      i2       -> id,
      i3       -> id,
      coefficients ->
      {
        {
          k      -> real,
          n      -> integer,
          delta  -> real
        }
        ...
      }
    },
    ...
  }
};

```

Directive	Parameters	Description
active	boolean	Interaction activator; either <code>true</code> or <code>false</code> .
n	integer	Number of data entries; can be omitted when <code>type = {merge -> true}</code> .
data	struct	Summary of interaction constants
i0, i1, i2, i3	id	Site id, referring to types as defined by the <code>mass</code> paragraph in <code>types</code>
k	real	Force constant
n	integer	Torsional angle prefactor; values range between 1 and 4
delta	real	Offset angle, mainly used to change the sign of the cosine function

The energetic functional form of the torsion contributions to the total potential is described by

$$E_{torsion} = \sum_{i=1}^m 1/2 k_i (1 + \text{sign}(n_i) \cos(n_i \phi - \delta_i)),$$

where ϕ represents the bond torsion constructed by bonds $\{i_0, i_1\}$, $\{i_1, i_2\}$, and $\{i_2, i_3\}$. Function $\text{sign}(n_i)$ is +1 for odd and -1 for even values of n_i .

7.12.5.8 Improper

```
improper      = {
  active      -> boolean
  n           -> integer,
  data        ->
  {
    {
      i0       -> id,
      i1       -> id,
      i2       -> id,
      i3       -> id,
      k        -> real,
      psi      -> real
    },
    ...
  }
};
```

Directive	Parameters	Description
active	boolean	Interaction activator; either <code>true</code> or <code>false</code> .
n	integer	Number of data entries; can be omitted when <code>type = {merge -> true}</code> .
data	struct	Summary of interaction constants
i0, i1, i2,	id	Site id, referring to types as defined by the <code>mass</code> paragraph in <code>types</code>
i3		
k	real	Force constant
psi	real	Improper equilibrium angle

The energetic functional form of the torsion contributions to the total potential is described by

$$E_{improper} = k_{improper}(\psi - \psi_0)^2,$$

where ψ represents the improper angle, which is given by the average of torsion angles formed by planes out of permutations of any of the following two bonds: $\{i_1, i_0\}$, $\{i_2, i_0\}$, and $\{i_3, i_0\}$. Note, that site i_0 is the central site, as opposed to some representations, where i_1 is the central site.

7.12.6 CFF

7.12.6.1 Syntax

```

cff                                = {
  pair                            -> struct,
  bond                            -> struct,
  angle                           -> struct,
  bond_bond                       -> struct,
  bond_angle                      -> struct,
  torsion                         -> struct,
  end_bond_torsion                -> struct,
  middle_bond_torsion             -> struct,
  bond_bond_13                   -> struct,
  angle_torsion                   -> struct,
  angle_angle_torsion            -> struct,
  improper                        -> struct,
  angle_angle                    -> struct
};

```

Directive	Parameters	Description
pair	struct	Pair interaction descriptors
bond	struct	Bond interaction descriptors
angle	struct	Angle interaction descriptors
bond_bond	struct	Bond-bond interaction descriptors
bond_angle	struct	Bond-angle interaction descriptors
torsion	struct	Torsion interaction descriptors
end_bond_ torsion	struct	End bond torsion interaction descriptors
middle_ bond_ torsion	struct	Middle bond torsion interaction descriptors
bond_bond_ 13	struct	Bond-bond-13 interaction descriptors
angle_ torsion	struct	Angle-torsion interaction descriptors
angle_ angle_ torsion	struct	Angle-angle-torsion interaction descriptors
improper	struct	Improper interaction descriptors
angle_angle	struct	Angle-angle interaction descriptors

The CFF force field, as described by Sun,¹ uses a combination of standard and cross-coupled bonded interactions combined with Lennard-Jones 6-9 nonbonded interactions. Its functional form is a summation of bonded and nonbonded interactions given by

$$\begin{aligned}
E_{cff} &= E_{bond} + E_{angles} + E_{torsions} + E_{improvers} + E_{pair} \\
E_{angles} &= E_{angle} + E_{bond-bond} + E_{bond-angle} \\
E_{torsions} &= E_{torsion} + E_{end-bond-torsion} + E_{middle-bond-torsion} + E_{bond-bond-13} + E_{angle-torsion} \\
&\quad + E_{angle-angle-torsion} \\
E_{improvers} &= E_{improper} + E_{angle-angle}
\end{aligned}$$

for bond, angle, torsion, improper, and pair contributions respectively. The following paragraphs describe each contribution in detail. Note, that improper contributions are calculated differently (see Improper). Force field typing is available for the PCFF force field.

7.12.6.2 Examples

The following example creates topology for water and types these sites with the PCFF force field as included with EMC,

```

field          = {
  mode         -> cff,
  name         -> {$root+"pcff/pcff_templates.dat", $root+"pcff/pcff.frc"
};

groups         = {
  group        -> {id -> water, chemistry -> "O"}
};

clusters       = {
  cluster      -> {id -> water, system -> main, group -> water, n -> 1000}
};

field          = {
  mode         -> apply
};

```

Application of the `field` scripting command saves the need for manual force field typing (see Section 5.10 [Field], page 101). However, CFF-type force field alterations or additions can also be entered directly through

```

simulation     = {
  types        -> {
    merge      -> true,
    cff -> {active -> true,
      angle    -> {
        active -> true, n -> 1, data ->
          {i0 -> hw, i1 -> o*, i2 -> hw, theta -> 103.7, k -> {49.84,
            -11.6, -8}}},
        angle_angle -> {
          active -> true, n -> 0},
        angle_angle_torsion -> {

```

```

    active  -> true, n -> 0},
angle_torsion -> {
    active  -> true, n -> 0},
bond       -> {
    active  -> true, n -> 1, data ->
    {i0     -> o*, i1 -> hw, l -> 0.97, k -> {563.28, -1428.22,
    1902.12}}}},
bond_angle -> {
    active  -> true, n -> 1, data ->
    {i0     -> hw, i1 -> o*, i2 -> hw, k -> {22.35, 22.35}}}},
bond_bond  -> {
    active  -> true, n -> 1, data ->
    {i0 -> hw, i1 -> o*, i2 -> hw, k -> -9.5}}},
bond_bond_13 -> {
    active  -> true, n -> 0},
end_bond_torsion -> {
    active  -> true, n -> 0},
improper   -> {
    active  -> true, n -> 0},
middle_bond_torsion -> {
    active  -> true, n -> 0},
pair -> {
    active  -> true, mix -> sixth, shift -> false, coulomb -> cut,
    nbonded -> 2, mode -> repulsive, cutoff -> 9.5, n -> 3, data -> {
    {i0     -> o*, i1 -> o*, epsilon -> 0.274, sigma -> 3.608},
    {i0     -> o*, i1 -> hw, epsilon -> 0.00336154972409,
    sigma -> 3.21478799199},
    {i0     -> hw, i1 -> hw, epsilon -> 0.013, sigma -> 1.098}}}},
torsion    -> {
    active  -> true, n -> 0}},
}
};

```

The above definition would result from application of the PCFF force field.

7.12.6.3 References

1. H. Sun, "*COMPASS: An ab Initio Force-Field Optimized for Condensed-Phase Applications - Overview with Details on Alkane and Benzene Compounds*", *J. Phys. Chem. B* **1998**, *102*, 7338.

7.12.6.4 Pair

```
pair          = {
  active      -> boolean,
  nbonded     -> integer,
  mix         -> option,
  shift       -> boolean,
  coulomb     -> option,
  cutoff      -> real,
  mode        -> option,
  n           -> integer,
  data        ->
  {
    {
      i0      -> id,
      i1      -> id,
      epsilon -> real,
      sigma   -> real,
      cutoff  -> real
    },
    ...
  }
};
```

Directive	Parameters	Description
active	boolean	Interaction activator; either true or false .
nbonded	integer	Number of bonded sites to exclude from nonbonded interactions; the given number is expected to be larger or equal to zero.
mix	option	Sets mixing rule; options are none , berthelot , arithmetic , geometric , and sixth .
shift	boolean	Sets shifting of potential at cutoff; options are true and false .
coulomb	option	Set type of coulomb treatment; options are none , cut , and long .
cutoff	real	Sets global cut off for global mode
mode	option	Sets cut off mode; options are global using the value of cutoff for all contributions, individual defining cut offs per contribution, and repulsive defining the well location for each separate contribution as the cut off distance.
n	integer	Number of data entries; can be omitted when type = {merge -> true} .
data	struct	Summary of interaction constants.
i0, i1	id	Site id, referring to types as defined by the mass paragraph.

epsilon	real	Force interaction constant; identical to the well depth for a Lennard-Jones 6-9 potential.
sigma	real	Site size; the location of the potential well.
cutoff	real	Cut off; distance at which the pair-wise contribution is set to zero; can be omitted.

The energetic functional form of the pair contributions to the total potential is described by a 6-9 Lennard-Jones potential,

$$E_{pair} = \epsilon \left(\frac{\sigma}{r_{ij}} \right)^6 \left[2 \left(\frac{\sigma}{r_{ij}} \right)^3 - 3 \right],$$

where r_{ij} represents the distance between site i and site j . The nonbond interactions, which include Lennard-Jones and coulombic interactions, encompass all pairs between atoms which are separated by two or more bonded atoms, i.e. **nbonded** = 2.

7.12.6.5 Bond

```

bond          = {
  active      -> boolean
  n           -> integer,
  data        ->
  {
    {
      i0       -> id,
      i1       -> id,
      l        -> real,
      k        -> {real, real, real}
    },
    ...
  }
};

```

Directive	Parameters	Description
active	boolean	Interaction activator; either <code>true</code> or <code>false</code> .
n	integer	Number of data entries; can be omitted when <code>type = {merge -> true}</code> .
data	struct	Summary of interaction constants
i0, i1	id	Site id, referring to <code>mass</code> paragraph in <code>types</code>
l	real	Equilibrium length
k	real	Defines up to three bond constants

The energetic functional form of the bond contributions to the total potential is described by

$$E_{bond} = \sum_{i=1}^3 k_i (l - l_{0,i})^{i+1},$$

where l represents the bond length of bond $\{i_0, i_1\}$.

7.12.6.6 Angle

```

angle          = {
  active        -> boolean
  n             -> integer,
  data          ->
  {
    {
      i0         -> id,
      i1         -> id,
      i2         -> id
      theta      -> real,
      k          -> {real, real, real}
    },
    ...
  }
};

```

Directive	Parameters	Description
active	boolean	Interaction activator; either <code>true</code> or <code>false</code> .
n	integer	Number of data entries; can be omitted when <code>type = {merge -> true}</code> .
data	struct	Summary of interaction constants
i0, i1, i2	id	Site id, referring to <code>mass</code> paragraph in <code>types</code>
theta	real	Equilibrium angle
k	real	Defines up to three angle constants

The energetic functional form of the angle contributions to the total potential is described by

$$E_{angle} = \sum_{i=1}^3 k_i (\theta - \theta_{0,i})^{i+1},$$

where θ represents the angle between bonds $\{i_0, i_1\}$ and $\{i_1, i_2\}$.

7.12.6.7 Bond-Bond

```

bond_bond      = {
  active        -> boolean
  n             -> integer,
  data          ->
  {
    {
      i0        -> id,
      i1        -> id,
      i2        -> id
      k         -> real
    },
    ...
  }
};

```

Directive	Parameters	Description
active	boolean	Interaction activator; either <code>true</code> or <code>false</code> .
n	integer	Number of data entries; can be omitted when <code>type = {merge -> true}</code> .
data	struct	Summary of interaction constants
i0, i1, i2	id	Site id, referring to <code>mass</code> paragraph in <code>types</code>
k	real	Defines the bond-bond cross term constant

The energetic functional form of the bond-bond cross-coupling contributions to the total potential is described by

$$E_{bond-bond} = k(l_1 - l_{0,1})(l_2 - l_{0,2}),$$

where l_1 represents the length of bond $\{i_0, i_1\}$ with equilibrium value $l_{0,1}$ and l_2 the length of bond $\{i_1, i_2\}$ with equilibrium value $l_{0,2}$. Equilibrium values reference the bond contributions.

7.12.6.8 Bond-Angle

```

bond_angle      = {
  active         -> boolean
  n              -> integer,
  data           ->
  {
    {
      i0          -> id,
      i1          -> id,
      i2          -> id
      k           -> {real, real}
    },
    ...
  }
};

```

Directive	Parameters	Description
active	boolean	Interaction activator; either <code>true</code> or <code>false</code> .
n	integer	Number of data entries; can be omitted when <code>type = {merge -> true}</code> .
data	struct	Summary of interaction constants
i0, i1, i2	id	Site id, referring to <code>mass</code> paragraph in <code>types</code>
k	real	Defines the bond-angle cross term constants

The energetic functional form of the bond-angle cross-coupling contributions to the total potential is described by

$$E_{bond-angle} = (k_1(l_1 - l_{0,1}) + k_2(l_2 - l_{0,2}))(\theta - \theta_{0,i}),$$

where l_1 represents the length of bond $\{i_0, i_1\}$ with equilibrium value $l_{0,1}$, l_2 the length of bond $\{i_1, i_2\}$ with equilibrium value $l_{0,2}$, and θ the angle between bonds $\{i_0, i_1\}$ and $\{i_1, i_2\}$. Equilibrium values reference both bond and angle contributions.

7.12.6.9 Torsion

```

torsion      = {
  active      -> boolean
  n           -> integer,
  data        ->
  {
    {
      i0       -> id,
      i1       -> id,
      i2       -> id,
      i3       -> id,
      coefficients ->
      {
        {k      -> real, n -> integer, delta -> real},
        {k      -> real, n -> integer, delta -> real},
        {k      -> real, n -> integer, delta -> real}
      }
    },
    ...
  }
};

```

Directive	Parameters	Description
active	boolean	Interaction activator; either <code>true</code> or <code>false</code> .
n	integer	Number of data entries; can be omitted when <code>type = {merge -> true}</code> .
data	struct	Summary of interaction constants
i0, i1, i2, i3	id	Site id, referring to <code>mass</code> paragraph in <code>types</code>
coefficients	struct	List of up to three sets of coefficients
k	real	Torsion constants
n	integer	Torsion prefactor
delta	real	Torsion offset

The energetic functional form of the torsion contributions to the total potential is described by

$$E_{torsion} = \sum_{i=1}^3 k_i (1 - \cos[n_i \phi - \delta]),$$

where ϕ represents the bond torsion constructed by bonds $\{i_0, i_1\}$, $\{i_1, i_2\}$, and $\{i_2, i_3\}$.

7.12.6.10 End-Bond-Torsion

```

end_bond_torsion = {
  active      -> boolean
  n           -> integer,
  data        ->
  {
    {
      i0       -> id,
      i1       -> id,
      i2       -> id,
      i3       -> id,
      k        -> {real, real, real, real, real, real}
    },
    ...
  }
};

```

Directive	Parameters	Description
active	boolean	Interaction activator; either <code>true</code> or <code>false</code> .
n	integer	Number of data entries; can be omitted when <code>type = {merge -> true}</code> .
data	struct	Summary of interaction constants
i0, i1, i2, i3	id	Site id, referring to <code>mass</code> paragraph in <code>types</code>
k	real	Three left (k_1 through k_3) and three right (k_4 through k_6) interaction constants

The energetic functional form of the end-bond-torsion cross-coupling contributions to the total potential is described by

$$E_{\text{end-bond-torsion}} = \sum_{i=1}^3 (k_i(l_1 - l_{0,1}) + k_{i+3}(l_2 + l_{0,2})) \cos[i\phi],$$

where l_1 represents the length of bond $\{i_0, i_1\}$ with equilibrium value $l_{0,1}$, l_2 represents the length of bond $\{i_2, i_3\}$ with equilibrium value $l_{0,2}$, and ϕ represents the bond torsion constructed by bonds $\{i_0, i_1\}$, $\{i_1, i_2\}$, and $\{i_2, i_3\}$.

7.12.6.11 Middle-Bond-Torsion

```
middle_bond_torsion = {
  active      -> boolean
  n           -> integer,
  data        ->
  {
    {
      i0       -> id,
      i1       -> id,
      i2       -> id,
      i3       -> id,
      k        -> {real, real, real}
    },
    ...
  }
};
```

Directive	Parameters	Description
active	boolean	Interaction activator; either true or false .
n	integer	Number of data entries; can be omitted when type = {merge -> true} .
data	struct	Summary of interaction constants
i0, i1, i2, i3	id	Site id, referring to mass paragraph in types
k	real	Three (k_1 through k_3) interaction constants

The energetic functional form of the middle-bond-torsion cross-coupling contributions to the total potential is described by

$$E_{\text{end-bond-torsion}} = \sum_{i=1}^3 k_i (l_i - l_{0,i}) \cos[i\phi],$$

where l_i represents the length of bond $\{i_1, i_2\}$ with equilibrium value $l_{0,i}$ and ϕ represents the bond torsion constructed by bonds $\{i_0, i_1\}$, $\{i_1, i_2\}$, and $\{i_2, i_3\}$.

7.12.6.12 Bond-Bond-13

```

bond_bond_13    = {
  active        -> boolean
  n             -> integer,
  data          ->
  {
    {
      i0        -> id,
      i1        -> id,
      i2        -> id,
      i3        -> id,
      k         -> real
    },
    ...
  }
};

```

Directive	Parameters	Description
active	boolean	Interaction activator; either <code>true</code> or <code>false</code> .
n	integer	Number of data entries; can be omitted when <code>type = {merge -> true}</code> .
data	struct	Summary of interaction constants.
i0, i1, i2, i3	id	Site id, referring to <code>mass</code> paragraph in <code>types</code> .
k	real	Bond-Bond-13 interaction constant.

The energetic functional form of the bond-bond-13 cross-coupling contributions to the total potential is described by

$$E_{bond-bond-13} = k_{bond-bond-13}(l_1 - l_{0,1})(l_2 - l_{0,2}),$$

where l_1 represents the length of bond $\{i_0, i_1\}$ with equilibrium value $l_{0,1}$, l_2 represents the length of bond $\{i_2, i_3\}$ with equilibrium value $l_{0,2}$.

7.12.6.13 Angle-Torsion

```

angle_torsion      = {
  active           -> boolean
  n                -> integer,
  data             ->
  {
    {
      i0           -> id,
      i1           -> id,
      i2           -> id,
      i3           -> id,
      k            -> {real, real, real, real, real, real}
    },
    ...
  }
};

```

Directive	Parameters	Description
active	boolean	Interaction activator; either <code>true</code> or <code>false</code> .
n	integer	Number of data entries; can be omitted when <code>type = {merge -> true}</code> .
data	struct	Summary of interaction constants.
i0, i1, i2, i3	id	Site id, referring to <code>mass</code> paragraph in <code>types</code> .
k	real	Six (k_1 through k_6) interaction constants.

The energetic functional form of the angle-torsion cross-coupling contributions to the total potential is described by

$$E_{angle-torsion} = \sum_{i=1}^2 (\theta_i - \theta_{0,i}) \sum_{j=1}^3 k_{3(i-1)+j} \cos[j\phi],$$

where θ_1 the angle between bonds $\{i_0, i_1\}$ and $\{i_1, i_2\}$, θ_2 the angle between bonds $\{i_1, i_2\}$ and $\{i_2, i_3\}$, and ϕ the bond torsion constructed by bonds $\{i_0, i_1\}$, $\{i_1, i_2\}$, and $\{i_2, i_3\}$.

7.12.6.14 Angle-Angle-Torsion

```

angle_torsion      = {
  active           -> boolean
  n                -> integer,
  data             ->
  {
    {
      i0           -> id,
      i1           -> id,
      i2           -> id,
      i3           -> id,
    },
    ...
  }
};

```

Directive	Parameters	Description
active	boolean	Interaction activator; either <code>true</code> or <code>false</code> .
n	integer	Number of data entries; can be omitted when <code>type = {merge -> true}</code> .
data	struct	Summary of interaction constants
i0, i1, i2, i3	id	Site id, referring to <code>mass</code> paragraph in <code>types</code>

The energetic functional form of the angle-angle-torsion cross-coupling contributions to the total potential is described by

$$E_{angle-angle-torsion} = k_{angle-angle-torsion} ((\theta_1 - \theta_{0,1})(\theta_2 - \theta_{0,2}) \cos[\phi],$$

where θ_1 the angle between bonds $\{i_0, i_1\}$ and $\{i_1, i_2\}$, θ_2 the angle between bonds $\{i_1, i_2\}$ and $\{i_2, i_3\}$, and ϕ the bond torsion constructed by bonds $\{i_0, i_1\}$, $\{i_1, i_2\}$, and $\{i_2, i_3\}$.

7.12.6.15 Improper

```

angle          = {
  active        -> boolean
  n             -> integer,
  data          ->
  {
    {
      i0         -> id,
      i1         -> id,
      i2         -> id,
      i3         -> id,
      k          -> real,
      psi        -> real
    },
    ...
  }
};

```

Directive	Parameters	Description
active	boolean	Interaction activator; either <code>true</code> or <code>false</code> .
n	integer	Number of data entries; can be omitted when <code>type = {merge -> true}</code> .
data	struct	Summary of interaction constants
i0, i1, i2,	id	Site id, referring to types as defined by the <code>mass</code> paragraph in <code>types</code>
i3		
k	real	Force constant
psi	real	Improper equilibrium angle

The energetic functional form of the torsion contributions to the total potential is described by

$$E_{improper} = k_{improper}(\psi - \psi_0)^2,$$

where ψ represents the improper angle, which is given by the average of torsion angles formed by planes out of permutations of any of the following two bonds: $\{i_1, i_0\}$, $\{i_2, i_0\}$, and $\{i_3, i_0\}$. EMC represents site i_0 as the central site, as opposed to some representations, for which i_1 is the central site. Note, that each improper contributes to the total improper energy individually. Note, that Discover and LAMMPS use the average of the three contributing angles ψ and calculate the improper energy accordingly. EMC, however, uses the average of the energy for all three contributing factors.

7.12.7 CHARMM

7.12.7.1 Syntax

```
charmm      = {
  pair      -> struct,
  bond      -> struct,
  angle     -> struct,
  urey      -> struct,
  torsion   -> struct,
  pair14    -> struct,
  improper   -> struct
};
```

Directive	Parameters	Description
pair	struct	Pair interaction descriptors
bond	struct	Bond interaction descriptors
angle	struct	Angle interaction descriptors
urey	struct	Urey 1-3 interaction descriptors
torsion	struct	Torsion interaction descriptors
pair14	struct	Pair 1-4 interaction descriptors
improper	struct	Improper interaction descriptors

The CHARMM force field as included in EMC is based on the CHARMM description as defined by MacKerrell et al.¹ EMC provides force field files, which are derived from the original CHARMM force field files, but are governed by typing rules, rather than residue templates, as is the case in the original definition. An included script (`./scripts/charmm.pl`) derives typing rules from the original CHARMM force field files in an automated fashion. The force field files are found in `./field/charmm`. Its functional form is a summation of bonded and nonbonded interactions given by

$$E_{charmm} = E_{bond} + E_{angle} + E_{urey} + E_{torsion} + E_{improper} + E_{pair14} + E_{pair}$$

for bond, angle, urey, torsion, pair14, and pair contributions respectively. The following paragraphs describe each contribution in detail. See `./examples/field/charmm/` for applications.

7.12.7.2 Examples

The following example creates topology for water and types these sites with the CHARMM force field as included with EMC,

```
field      = {
  mode     -> charmm,
  name     -> {$root+"field/charmm/c32b1/all127_prot_lipid.prm",
              $root+"field/charmm/c32b1/all127_prot_lipid.top"}
};
groups    = {
```

```

    group          -> {id -> water, chemistry -> "O"}
}

clusters          = {
  cluster          -> {id -> water, system -> main, group -> water, n -> 1000}
};

field              = {
  mode             -> apply
};

```

Application of the `field` scripting command saves the need for manual force field typing (see Section 5.10 [Field], page 101). However, CHARMM force field alterations or additions can also be entered directly through

```

simulation          = {
  types             -> {
    merge           -> true,
    charmm          -> {active -> true,
      angle         -> {active -> true, n -> 1, data ->
        {i0         -> HT, i1 -> OT, i2 -> HT, k -> 55, theta -> 104.52}}},
      bond          -> {active -> true, n -> 1, data ->
        {i0         -> OT, i1 -> HT, k -> 450, l -> 0.9572}}},
      improper       -> {active -> true, n -> 0},
      pair          -> {active -> true, mix -> berthelot, shift -> false,
        coulomb     -> none, nbonded -> 3, mode -> individual, inner -> 0,
        cutoff      -> 9.5, n -> 3, data -> {
          {i0       -> OT, i1 -> OT, epsilon -> 0.1521, sigma -> 3.15057422683},
          {i0       -> OT, i1 -> HT, epsilon -> 0.0836456812992,
            sigma -> 1.77529387564},
          {i0       -> HT, i1 -> HT, epsilon -> 0.046,
            sigma -> 0.400013524445}}}},
    pair14          -> {active -> true, n -> 3, data -> {
      {i0          -> OT, i1 -> OT, epsilon -> 0.1521, sigma -> 3.15057422683},
      {i0          -> OT, i1 -> HT, epsilon -> 0.0836456812992,
        sigma -> 1.77529387564},
      {i0          -> HT, i1 -> HT, epsilon -> 0.046,
        sigma -> 0.400013524445}}}},
    torsion         -> {active -> true, n -> 0},
    urey            -> {active -> true, n -> 1, data ->
      {i0          -> HT, i1 -> OT, i2 -> HT, k -> 0, l -> 0}}},
  }
};

```

7.12.7.3 References

1. A. D. MacKerrell, Jr. et al., "All-Atom Empirical Potential for Molecular Modeling

and Dynamics Studies of Proteins", J. Phys. Chem. B **1998**, *102*, 3586-3616.

7.12.7.4 Pair

```
pair          = {
  active      -> boolean,
  nbonded     -> integer,
  mix         -> option,
  shift       -> boolean,
  coulomb     -> option,
  inner       -> real,
  cutoff      -> real,
  mode        -> option,
  n           -> integer,
  data        ->
  {
    {
      i0      -> id,
      i1      -> id,
      epsilon -> real,
      sigma   -> real,
      inner   -> real,
      cutoff  -> real
    },
    ...
  }
};
```

Directive	Parameters	Description
active	boolean	Interaction activator; either true or false .
nbonded	integer	Number of bonded sites to exclude from nonbonded interactions; the given number is expected to be larger or equal to zero.
mix	option	Sets mixing rule; options are none , berthelot , arithmetic , geometric , and sixth .
shift	boolean	Sets shifting of potential at cutoff; options are true and false .
coulomb	option	Set type of coulomb treatment; options are none , cut , and long .
inner	real	Sets global inner start of cut off for global mode.
cutoff	real	Sets global cut off for global mode.
mode	option	Sets cut off mode; options are global using the value of cutoff for all contributions, individual defining cut offs per contribution, and repulsive defining the well location for each separate contribution as the cut off distance.
n	integer	Number of data entries; can be omitted when type = {merge -> true} in which case the entered entries are merged with already existing ones.
data	struct	Summary of interaction constants.

<code>i0, i1</code>	<code>id</code>	Site id, referring to types as defined by the <code>mass</code> paragraph.
<code>epsilon</code>	<code>real</code>	Force interaction constant; identical to the well depth for a Lennard-Jones 6-12 potential.
<code>sigma</code>	<code>real</code>	Site size; the point where the potential equals zero.
<code>inner</code>	<code>real</code>	Sets pairwise inner start of cut off; can be omitted upon global <i>inner</i> definition.
<code>cutoff</code>	<code>real</code>	Sets pairwise cut off; can be omitted upon global <i>cutoff</i> definition.

The energetic functional form of the pair contributions to the total potential is described by a 6-12 Lennard-Jones potential,

$$E_{pair} = 4\epsilon \left(\frac{\sigma}{r_{ij}} \right)^6 \left[\left(\frac{\sigma}{r_{ij}} \right)^6 - 1 \right],$$

where r_{ij} represents the distance between sites i and j .

7.12.7.5 Bond

```

bond          = {
  active      -> boolean
  n           -> integer,
  data        ->
  {
    {
      i0       -> id,
      i1       -> id,
      k        -> real,
      l        -> real
    },
    ...
  }
};

```

Directive	Parameters	Description
active	boolean	Interaction activator; either <code>true</code> or <code>false</code> .
n	integer	Number of data entries; can be omitted when <code>type = {merge -> true}</code> .
data	struct	Summary of interaction constants
i0, i1	id	Site id, referring to types as defined by the <code>mass</code> paragraph in <code>types</code>
k	real	Bond spring constants
l	real	Equilibrium bond length

The energetic functional form of the bond contributions to the total potential is described by

$$E_{bond} = k_{bond}(l - l_0)^2,$$

where k_{bond} represents a spring constant, l the bond length of bond $\{i_0, i_1\}$, and l_0 the equilibrium bond length.

7.12.7.6 Angle

```

angle          = {
  active       -> boolean
  n            -> integer,
  data         ->
  {
    {
      i0        -> id,
      i1        -> id,
      i2        -> id,
      k         -> real,
      theta     -> real
    },
    ...
  }
};

```

Directive	Parameters	Description
active	boolean	Interaction activator; either <code>true</code> or <code>false</code> .
n	integer	Number of data entries; can be omitted when <code>type = {merge -> true}</code> .
data	struct	Summary of interaction constants
i0, i1, i2	id	Site id, referring to types as defined by the <code>mass</code> paragraph in <code>types</code>
k	real	Angle spring constants
theta	real	Equilibrium bond angle

The energetic functional form of the angle contributions to the total potential is described by

$$E_{angle} = k_{angle}(\theta - \theta_0)^2,$$

where k_{angle} represents a spring constant, θ the angle between bonds $\{i_0, i_1\}$ and $\{i_1, i_2\}$, and θ_0 the equilibrium angle.

7.12.7.7 Urey

```

urey          = {
  active      -> boolean
  n           -> integer,
  data        ->
  {
    {
      i0       -> id,
      i1       -> id,
      i2       -> id,
      k        -> real,
      l        -> real
    },
    ...
  }
};

```

Directive	Parameters	Description
active	boolean	Interaction activator; either <code>true</code> or <code>false</code> .
n	integer	Number of data entries; can be omitted when <code>type = {merge -> true}</code> .
data	struct	Summary of interaction constants
i0, i1, i2	id	Site id, referring to types as defined by the <code>mass</code> paragraph in <code>types</code>
k	real	Urey bond spring constants
l	real	Equilibrium urey bond length

The energetic functional form of the Urey-Bradley contributions to the total potential is described by

$$E_{urey} = k_{urey}(l - l_0)^2,$$

where k_{urey} represents a spring constant, l the urey bond length of bond $\{i_0, i_2\}$, and l_0 the equilibrium bond length.

7.12.7.8 Torsion

```
torsion      = {
  active      -> boolean
  n           -> integer,
  data        ->
  {
    {
      i0       -> id,
      i1       -> id,
      i2       -> id,
      i3       -> id,
      coefficients ->
      {
        {
          k      -> real,
          n      -> integer,
          delta  -> real
        }
        ...
      }
    },
    ...
  }
};
```

Directive	Parameters	Description
active	boolean	Interaction activator; either <code>true</code> or <code>false</code> .
n	integer	Number of data entries; can be omitted when <code>type = {merge -> true}</code> .
data	struct	Summary of interaction constants
i0, i1, i2, i3	id	Site id, referring to <code>mass</code> paragraph in <code>types</code>

The energetic functional form of the torsion contributions to the total potential is described by

$$E_{torsion} = \sum_{i=1}^m k_i (1 + \cos(n_i \phi - \delta_i)),$$

where k_i represents a set of torsion constants, δ_i a torsion offset, n_i the torsion pre-factor, and ϕ the bond torsion constructed by bonds $\{i_0, i_1\}$, $\{i_1, i_2\}$, and $\{i_2, i_3\}$. The maximum allowed value of n_i is 6.

7.12.7.9 Pair14

```

pair          = {
  active      -> boolean,
  mix         -> option,
  n           -> integer,
  data        ->
  {
    {
      i0       -> id,
      i1       -> id,
      epsilon  -> real,
      sigma    -> real
    },
    ...
  }
};

```

Directive	Parameters	Description
active	boolean	Interaction activator; either <code>true</code> or <code>false</code> .
mix	option	Sets mixing rule; options are <code>none</code> , <code>berthelot</code> , <code>arithmetic</code> , <code>geometric</code> , and <code>sixth</code> .
n	integer	Number of data entries; can be omitted when <code>type = {merge -> true}</code> in which case the entered entries are merged with already existing ones.
data	struct	Summary of interaction constants.
i0, i1	id	Site id, referring to types as defined by the <code>mass</code> paragraph.
epsilon	real	Force interaction constant; identical to the well depth for a Lennard-Jones 6-12 potential.
sigma	real	Site size; the point where the potential equals zero.

The energetic functional form of the pair contributions to the total potential is described by a 6-12 Lennard-Jones potential,

$$E_{pair} = 4\epsilon \left(\frac{\sigma}{l}\right)^6 \left[\left(\frac{\sigma}{l}\right)^{12} - 1\right],$$

where l represents the distance between sites $i0$ and $i1$.

7.12.7.10 Improper

```
improper      = {
  active      -> boolean
  n           -> integer,
  data        ->
  {
    {
      i0       -> id,
      i1       -> id,
      i2       -> id,
      i3       -> id,
      k        -> real,
      psi      -> real
    },
    ...
  }
};
```

Directive	Parameters	Description
active	boolean	Interaction activator; either <code>true</code> or <code>false</code> .
n	integer	Number of data entries; can be omitted when <code>type = {merge -> true}</code> .
data	struct	Summary of interaction constants
i0, i1, i2,	id	Site id, referring to types as defined by the <code>mass</code> paragraph in <code>types</code>
i3		
k	real	Force constant
psi	real	Improper equilibrium angle

The energetic functional form of the torsion contributions to the total potential is described by

$$E_{improper} = k_{improper}(\psi - \psi_0)^2,$$

where ψ represents the improper angle, which is given by the average of torsion angles formed by planes out of permutations of any of the following two bonds: $\{i_1, i_0\}$, $\{i_2, i_0\}$, and $\{i_3, i_0\}$. Note, that site i_0 is the central site, as opposed to some representations, where i_1 is the central site.

7.12.8 Coarse

7.12.8.1 Syntax

```
coarse      = {
  lj         -> struct,
  repulsive  -> struct,
  sphere     -> struct,
  colloid    -> struct,
  dpd        -> struct,
  charge     -> struct,
  fene       -> struct,
  angle      -> struct
};
```

Directive	Parameters	Description
lj	struct	Hard sphere core with Lennard-Jones shell interaction descriptors
repulsive	struct	Hard sphere core with r^{-6} repulsive shell interaction descriptors
sphere	struct	Coarse-grained sphere interaction descriptors (Girifalco ¹)
colloid	struct	Colloidal interaction descriptors (Everaers and Ejtehadi ²)
dpd	struct	DPD interaction descriptors (Groot and Warren ³)
charge	struct	Coarse-grained charge interaction descriptors (internal form)
fene	struct	FENE interaction descriptors (Kremer and Grest ⁴)
angle	struct	Angle interaction descriptors

The coarse force field represents a set of coarse-grained interactions functions. Its functional form is a summation of bonded and nonbonded interactions given by

$$E_{coarse} = E_{LJ} + E_{repulsive} + E_{sphere} + E_{colloid} + E_{DPD} + E_{charge} + E_{FENE} + E_{angle}$$

for hard core Lennard-Jones, hard core repulsive, colloidal, coarse-grained sphere angle, DPD, charge, FENE, and angle contributions respectively. The following paragraphs describe each contribution in detail. Typically, all contributions are not used simultaneously. Note, that the DPD contributions are included for backwards compatibility. Use the DPD force field when typing and transferring to e.g. LAMMPS. See `./examples/nanoparticle/coarse/` for applications.

7.12.8.2 Examples

```
variables    = {
};
```

```
simulation   = {
```

```

types      -> {
  merge     -> true,
  coarse    -> {
    dpd      -> {
      active -> true,
      data   -> {
      }
    }
  }
}
};

```

```

simulation = {
  types      -> {
    merge     -> true,
    coarse    -> {
      colloid  -> {
        active -> true,
        data   -> {
        }
      }
    }
  }
}
};

```

7.12.8.3 References

1. L. A. Girifalco, "Molecular Properties of C_{60} in the Gas and Solid Phases" *J. Chem. Phys.* **1992**, *96*, 858-861.
2. R. Everaers and M. R. Ejtehadi, "Interaction potentials for soft and hard ellipsoids", *Phys. Rev. E* **2003**, *67*, 041710.
3. R. D. Groot and P. B. Warren, "Dissipative particle dynamics: Bridging the gap between atomistic and mesoscopic simulation", *J. Chem. Phys.* **1997**, *107*, 4423-4435.
4. K. Kremer and G. S. Grest, *J. Chem. Phys.* **1990**, *92*, 5057.

7.12.8.4 LJ

```

lj          = {
  active     -> boolean,
  nbonded    -> integer,
  mix        -> option,
  shift      -> boolean,
  cutoff     -> real,
  mode       -> option,
  n          -> integer,
  data       ->
  {
    {
      i0      -> id,
      i1      -> id,
      epsilon -> real,
      sigma   -> real,
      core    -> real,
      cutoff  -> real
    },
    ...
  }
};

```

Directive	Parameters	Description
active	boolean	Interaction activator; either true or false .
nbonded	integer	Number of bonded sites to exclude from nonbonded interactions; the given number is expected to be larger or equal to zero.
mix	option	Sets mixing rule; options are none , berthelot , arithmetic , geometric , and sixth .
shift	boolean	Sets shifting of potential at cutoff; options are true and false .
cutoff	real	Sets global cut off for global mode
mode	option	Sets cut off mode; options are global using the value of cutoff for all contributions, individual defining cut offs per contribution, and repulsive defining the well location for each separate contribution as the cut off distance.
n	integer	Number of data entries; can be omitted when type = {merge -> true} .
data	struct	Summary of interaction constants
i0, i1	id	Site id, referring to types as defined by the mass paragraph.
epsilon	real	Force interaction constant; identical to the well depth for a Lennard-Jones 6-12 potential
sigma	real	Site size; the point where the potential equals zero

core	real	Site hard core size
cutoff	real	Sets the local pairwise cut off

The energetic functional form of the pair contributions to the total potential is described by a hard core, extended with a 6-12 Lennard-Jones potential,

$$E_{LJ} = 4\epsilon \left(\frac{\sigma}{r_{ij} - d_{core}} \right)^6 \left[\left(\frac{\sigma}{r_{ij} - d_{core}} \right)^6 - 1 \right] \quad \text{for } r_{ij} \geq d_{core},$$

where r_{ij} represents the distance between site i and site j , and d_{core} its core size.

7.12.8.5 Repulsive

```
repulsive      = {
  active       -> boolean,
  nbonded      -> integer,
  mix          -> option,
  shift        -> boolean,
  cutoff       -> real,
  mode         -> option,
  n            -> integer,
  data         ->
  {
    {
      i0        -> id,
      i1        -> id,
      epsilon   -> real,
      sigma     -> real,
      core      -> real
      cutoff    -> real
    },
    ...
  }
};
```

Directive	Parameters	Description
active	boolean	Interaction activator; either true or false .
nbonded	integer	Number of bonded sites to exclude from nonbonded interactions; the given number is expected to be larger or equal to zero.
mix	option	Sets mixing rule; options are none , berthelot , arithmetic , geometric , and sixth .
shift	boolean	Sets shifting of potential at cutoff; options are true and false .
cutoff	real	Sets global cut off for global mode
mode	option	Sets cut off mode; options are global using the value of cutoff for all contributions, individual defining cut offs per contribution, and repulsive defining the well location for each separate contribution as the cut off distance.
n	integer	Number of data entries; can be omitted when type = {merge -> true} .
data	struct	Summary of interaction constants
i0, i1	id	Site id, referring to types as defined by the mass paragraph.
epsilon	real	Force interaction constant; identical to the well depth for a Lennard-Jones 6-12 potential
sigma	real	Site size; the point where the potential equals zero

core **real** Site hard core size

The energetic functional form of the pair contributions to the total potential is described by a hard core, extended with a repulsive potential,

$$E_{repulsive} = 4\epsilon \left(\frac{\sigma}{r_{ij} - d_{core}} \right)^{12} \quad \text{for } r_{ij} \geq d_{core},$$

where r_{ij} represents the distance between site i and site j , and d_{core} its core size.

7.12.8.6 Sphere

```

lj          = {
  active     -> boolean,
  nbonded    -> integer,
  mix        -> option,
  cutoff     -> real,
  mode       -> option,
  n          -> integer,
  data       ->
  {
    {
      i0      -> id,
      i1      -> id,
      epsilon -> real,
      sigma   -> real,
      d       -> real,
      n       -> real,
      cutoff  -> real
    },
    ...
  }
};

```

Directive	Parameters	Description
active	boolean	Interaction activator; either true or false .
nbonded	integer	Number of bonded sites to exclude from nonbonded interactions; the given number is expected to be larger or equal to zero.
mix	option	Sets mixing rule; options are none , berthelot , arithmetic , geometric , and sixth .
cutoff	real	Sets global cut off for global mode
mode	option	Sets cut off mode; options are global using the value of cutoff for all contributions, individual defining cut offs per contribution, and repulsive defining the well location for each separate contribution as the cut off distance
n	integer	Number of data entries; can be omitted when type = {merge -> true} .
data	struct	Summary of interaction constants
i0, i1	id	Site id, referring to types as defined by the mass paragraph.
epsilon	real	Force interaction constant; identical to the well depth for a Lennard-Jones 6-12 potential
d	real	Sets the diameter of both particles
n	real	Sets the number of contributing Lennard-Jones beads to each spherical FCC particle

cutoff **real** Sets the local pairwise cut off

The energetic functional form of an integration of two interacting spherical FCC lattices consisting of n Lennard-Jones particles,

$$E_{LJ} = 4 \epsilon n^2 \left(\frac{\sigma}{d} \right)^6 \left[\frac{1}{12} \left(\frac{1}{s(s-1)^3} + \frac{1}{s(s+1)^3} - \frac{2}{s^4} \right) + \frac{1}{90} \left(\frac{\sigma}{d} \right)^6 \left(\frac{1}{s(s-1)^9} + \frac{1}{s(s+1)^9} - \frac{2}{s^{10}} \right) \right],$$

where ϵ and σ are the standard Lennard-Jones constants, $s = r_{ij}/d$, and r_{ij} represents the distance between the particle centers.

7.12.8.7 Colloid

```
colloid      = {
  active     -> boolean,
  nbonded    -> integer,
  n          -> integer,
  data       ->
  {
    {
      i0      -> id,
      i1      -> id,
      A       -> real,
      sigma   -> real,
      d1      -> real,
      d2      -> real,
      cutoff  -> real
    },
    ...
  }
};
```

Directive	Parameters	Description
active	boolean	Interaction activator; either true or false .
nbonded	integer	Number of bonded sites to exclude from nonbonded interactions; the given number is expected to be larger or equal to zero.
n	integer	Number of data entries; can be omitted when type = {merge -> true} .
data	struct	Summary of interaction constants
i0, i1	id	Site id, referring to mass paragraph in types
A	real	Pairwise Hamaker constants
sigma	real	Pairwise Lennard-Jones sigma
d1, d2	real	Respective colloidal particle diameter
cutoff	real	Pairwise potential cutoff

The energetic functional form of the pair contributions to the total potential, as described by Everaers and Ejtehadi, is formed by the sum of attractive and repulsive energy:

$$E = E_A + E_R$$

Attractive energy is expressed by

$$E_A = -\frac{A}{6} \left[\frac{2d_i d_j}{r_{ij}^2 - (d_i + d_j)^2} + \frac{2d_i d_j}{r_{ij}^2 - (d_i - d_j)^2} + \ln \left(\frac{r_{ij}^2 - (d_i + d_j)^2}{r_{ij}^2 - (d_i - d_j)^2} \right) \right],$$

where A is referred to as Hamaker's constant, d_i and d_j represent the diameter sites i and j , and r_{ij} represents the distance between sites. Repulsive energy is expressed by

$$E_R = \frac{A}{37800} \frac{\sigma^6}{r_{ij}} \left[\frac{r_{ij}^2 - 7r_{ij}(d_i + d_j) + 6(d_i^2 + 7d_i d_j + d_j^2)}{(r_{ij} - d_i - d_j)^7} + \frac{r_{ij}^2 + 7r_{ij}(d_i + d_j) + 6(d_i^2 + 7d_i d_j + d_j^2)}{(r_{ij} + d_i + d_j)^7} - \frac{r_{ij}^2 + 7r_{ij}(d_i - d_j) + 6(d_i^2 - 7d_i d_j + d_j^2)}{(r_{ij} + d_i - d_j)^7} - \frac{r_{ij}^2 - 7r_{ij}(d_i - d_j) + 6(d_i^2 - 7d_i d_j + d_j^2)}{(r_{ij} - d_i + d_j)^7} \right],$$

where σ represents the Lennard-Jones (LJ) constant sigma. Hamaker's constant A in LJ units is given by $A = 4\pi\epsilon_{LJ}(\rho\sigma^3)^2$, with ρ representing reduced density.

7.12.8.8 DPD

```

pair          = {
  active      -> boolean,
  mix         -> option,
  nbonded     -> integer,
  mode        -> option,
  cutoff      -> real,
  n           -> integer,
  data        ->
  {
    {
      i0       -> id,
      i1       -> id,
      a        -> real,
      gamma    -> real,
      cutoff   -> real
    },
    ...
  }
};

```

Directive	Parameters	Description
active	boolean	Interaction activator; either <code>true</code> or <code>false</code> .
mix	option	Sets mixing rule; options are <code>none</code> , <code>berthelot</code> , <code>arithmetic</code> , <code>geometric</code> , and <code>sixth</code> .
nbonded	integer	Number of bonded sites to exclude from nonbonded interactions; the given number is expected to be larger or equal to zero.
mode	option	Sets cut off mode; options are <code>global</code> using the value of <code>cutoff</code> for all contributions, <code>individual</code> defining cut offs per contribution, and <code>repulsive</code> defining the well location for each separate contribution as the cut off distance.
cutoff	real	Sets global cut off for <code>global</code> mode
n	integer	Number of data entries; can be omitted when <code>type = {merge -> true}</code> .
data	struct	Summary of interaction constants
i0, i1	id	Site id, referring to types as defined by the <code>mass</code> paragraph.
a	real	Force interaction constant.
gamma	real	Force damping constant.
cutoff	real	Site size; the point where the potential equals zero.

The energetic functional form of pair contributions to the total soft potential is described by a repulsive Hertzian spring,

$$E_{pair} = 1/2 \ a \ r_{ij,cutoff} \left(1 - r_{ij}/r_{ij,cutoff}\right)^2, \quad r_{ij} < r_{ij,cutoff}$$

where a represents the pairwise interaction paramater, r_{ij} the distance between site i and site j , and $r_{ij,cutoff}$ the cutoff distance between these two sites.

7.12.8.9 Charge

```

charge      = {
  active     -> boolean,
  nbonded    -> integer,
  mix        -> option,
  shift      -> boolean,
  n          -> integer,
  data       ->
  {
    {
      i0      -> id,
      i1      -> id,
      a       -> real,
      kappa   -> real,
      d1      -> real,
      d2      -> real,
      cutoff  -> real
    },
    ...
  }
};

```

Directive	Parameters	Description
active	boolean	Interaction activator; either <code>true</code> or <code>false</code> .
nbonded	integer	Number of bonded sites to exclude from nonbonded interactions; the given number is expected to be larger or equal to zero.
mix	option	Sets mixing rule; options are <code>none</code> , <code>berthelot</code> , <code>arithmetic</code> , <code>geometric</code> , and <code>sixth</code> .
shift	boolean	Sets shifting of potential at cutoff; options are <code>true</code> and <code>false</code> .
n	integer	Number of data entries; can be omitted when <code>type = {merge -> true}</code> .
data	struct	Summary of interaction constants
i0, i1	id	Site id, referring to <code>mass</code> paragraph in <code>types</code>
a	real	Model constant
kappa	real	Electrostatic screening factor
d1	real	Sets the diameter of particle <code>i0</code>
d2	real	Sets the diameter of particle <code>i1</code>
cutoff	real	Sets the local pairwise cut off

The energetic functional form of the charge contributions to the total potential, which is based on a Yukawa potential. It is described by

$$E_{charge} = A \ln(1 + e^{-\kappa H}) (4 - 2H/d_1 - 2H/d_2),$$

where d_1 and d_2 represent particle diameters, $H = r_{ij} - \frac{1}{2}(d_1 + d_2)$, and A the electrostatic constant, which can be expressed in terms of surface potentials ψ_1 and ψ_2 ,

$$A = \pi \epsilon_0 \epsilon_r (\psi_1 + \psi_2)^2 d_1 d_2 / (d_1 + d_2) / 8,$$

assuming, that both surface potentials are approximately the same.

7.12.8.10 FENE

```

bond          = {
  active      -> boolean
  n           -> integer,
  data        ->
  {
    {
      i0       -> id,
      i1       -> id,
      k        -> real,
      l        -> real
    },
    ...
  }
};

```

Directive	Parameters	Description
active	boolean	Interaction activator; either <code>true</code> or <code>false</code> .
n	integer	Number of data entries; can be omitted when <code>type = {merge -> true}</code> .
data	struct	Summary of interaction constants
i0, i1	id	Site id, referring to types as defined by the <code>mass</code> paragraph in <code>types</code>
k	real	Bond spring constants
l	real	Equilibrium bond length

The energetic functional form of the bond contributions to the total potential is described by

$$E_{bond} = -\frac{1}{2} k l_0^2 \ln [1 - (l/l_0)^2],$$

where k represents a spring constant, l the bond length of bond $\{i_0, i_1\}$, and l_0 the equilibrium bond length.

7.12.8.11 Angle

```

angle          = {
  active        -> boolean
  n             -> integer,
  data          ->
  {
    {
      i0         -> id,
      i1         -> id,
      i2         -> id
    },
    ...
  }
};

```

Directive	Parameters	Description
active	boolean	Interaction activator; either true or false .
n	integer	Number of data entries; can be omitted when type = {merge -> true} .
data	struct	Summary of interaction constants
i0, i1	id	Site id, referring to mass paragraph in types

The energetic functional form of the bond contributions to the total potential is described by

$$E_{angle} =$$

where

7.12.9 Colloid

7.12.9.1 Syntax

```
colloid      = {
  pair       -> struct,
  charge     -> struct,
  bond       -> struct,
  angle      -> struct
};
```

Directive	Parameters	Description
pair	struct	Pairwise colloidal interaction descriptors
charge	struct	Pairwise charge interaction descriptors
bond	struct	Bond interaction descriptors
angle	struct	Angle interaction descriptors

The colloid force field is a compounded force field, that uses Hamaker and charge screening interactions. Its functional form is a summation of nonbonded and bonded interactions given by

$$E_{colloid} = E_{pair} + E_{charge} + E_{bond} + E_{angle}$$

for pair, charge, bond, and angle contributions respectively. The following paragraphs describe each contribution in detail.

7.12.9.2 Example

```
simulation    = {
  types       -> {
    merge     -> true,
    standard  -> {
      bond    -> {
        active -> true,
        data   -> {
          {i0   -> a, i1 -> b, k -> 100, l -> 2}
        }
      },
      angle   -> {
        active -> true,
        data   -> {
          {i0   -> a, i1 -> b, i2 -> c, k -> 400, theta -> 110}
        }
      },
      pair    -> {
        active -> true,
        inner  -> 1.0001,
        outer  -> 1.25,
      }
    }
  }
};
```

```

nbonded -> 2,
data    -> {
  {i0    -> a, i1 -> a,
    A    -> 1e-21, d1 -> 1e-7, d2 -> 1e-7, cutoff -> 2.5e-7},
  {i0    -> a, i1 -> b,
    A    -> 1.1e-21, d1 -> 1e-7, d2 -> 2e-7, cutoff -> 3.75e-7},
  {i0    -> a, i1 -> c,
    A    -> 1.2e-21, d1 -> 1e-7, d2 -> 3e-7, cutoff -> 5e-7},
  {i0    -> b, i1 -> b,
    A    -> 1.2e-21, d1 -> 2e-7, d2 -> 2e-7, cutoff -> 5e-7},
  {i0    -> b, i1 -> c,
    A    -> 1.3e-21, d1 -> 2e-7, d2 -> 3e-7, cutoff -> 6.25e-7},
  {i0    -> c, i1 -> c,
    A    -> 1.4e-21, d1 -> 3e-7, d2 -> 3e-7, cutoff -> 7.5e-7}
  }
}
}
};

```

7.12.9.3 References

1. R. Everaers and M. R. Ejtehadi, "*Interaction potentials for soft and hard ellipsoids*", *Phys. Rev. E*. **2003**, 67, 041710.

7.12.9.4 Pair

```

pair          = {
  active      -> boolean,
  nbonded     -> integer,
  n           -> integer,
  inner       -> real,
  outer       -> real,
  data        ->
  {
    {
      i0       -> id,
      i1       -> id,
      A        -> real,
      sigma    -> real,
      d1       -> real,
      d2       -> real,
      cutoff   -> real
    },
    ...
  }
};

```

Directive	Parameters	Description
active	boolean	Interaction activator; either true or false .
nbonded	integer	Number of bonded sites to exclude from nonbonded interactions; the given number is expected to be larger or equal to zero.
inner	real	Fraction of diameter defining the inner interaction rim; must be a number larger than 1.
outer	real	Fraction of diameter defining the outer interaction rim; must be a number larger than 1.
n	integer	Number of data entries; can be omitted when type = {merge -> true} .
data	struct	Summary of interaction constants
i0, i1	id	Site id, referring to mass paragraph in types
A	real	Pairwise Hamaker constants
sigma	real	Pairwise Lennard-Jones sigma
d1, d2	real	Respective colloidal particle diameter
cutoff	real	Pairwise potential cutoff

The energetic functional form of the pair contributions to the total potential, as described by Everaers and Ejtehadi, is formed by the sum of attractive and repulsive energy:

$$E = E_A + E_R$$

Attractive energy is expressed by

$$E_A = -\frac{A}{6} \left[\frac{2d_i d_j}{r_{ij}^2 - (d_i + d_j)^2} + \frac{2d_i d_j}{r_{ij}^2 - (d_i - d_j)^2} + \ln \left(\frac{r_{ij}^2 - (d_i + d_j)^2}{r_{ij}^2 - (d_i - d_j)^2} \right) \right],$$

where A is referred to as Hamaker's constant, d_i and d_j represent the diameter sites i and j , and r_{ij} represents the distance between sites. Repulsive energy is expressed by

$$E_R = \frac{A}{37800} \frac{\sigma^6}{r_{ij}} \left[\frac{r_{ij}^2 - 7r_{ij}(d_i + d_j) + 6(d_i^2 + 7d_i d_j + d_j^2)}{(r_{ij} - d_i - d_j)^7} + \frac{r_{ij}^2 + 7r_{ij}(d_i + d_j) + 6(d_i^2 + 7d_i d_j + d_j^2)}{(r_{ij} + d_i + d_j)^7} - \frac{r_{ij}^2 + 7r_{ij}(d_i - d_j) + 6(d_i^2 - 7d_i d_j + d_j^2)}{(r_{ij} + d_i - d_j)^7} - \frac{r_{ij}^2 - 7r_{ij}(d_i - d_j) + 6(d_i^2 - 7d_i d_j + d_j^2)}{(r_{ij} - d_i + d_j)^7} \right],$$

where σ represents the Lennard-Jones (LJ) constant sigma. Hamaker's constant A in LJ units is given by $A = 4\pi\epsilon_{LJ} (\rho\sigma^3)^2$, with ρ representing reduced density.

7.12.9.5 Charge

```

charge      = {
  active     -> boolean,
  nbonded    -> integer,
  mix        -> option,
  shift      -> boolean,
  n          -> integer,
  data       ->
  {
    {
      i0      -> id,
      i1      -> id,
      a       -> real,
      kappa   -> real,
      d1      -> real,
      d2      -> real,
      cutoff  -> real
    },
    ...
  }
};

```

Directive	Parameters	Description
active	boolean	Interaction activator; either <code>true</code> or <code>false</code> .
nbonded	integer	Number of bonded sites to exclude from nonbonded interactions; the given number is expected to be larger or equal to zero.
mix	option	Sets mixing rule; options are <code>none</code> , <code>berthelot</code> , <code>arithmetic</code> , <code>geometric</code> , and <code>sixth</code> .
shift	boolean	Sets shifting of potential at cutoff; options are <code>true</code> and <code>false</code> .
n	integer	Number of data entries; can be omitted when <code>type = {merge -> true}</code> .
data	struct	Summary of interaction constants
i0, i1	id	Site id, referring to <code>mass</code> paragraph in <code>types</code>
a	real	Model constant
kappa	real	Electrostatic screening factor
d1	real	Sets the diameter of particle <code>i0</code>
d2	real	Sets the diameter of particle <code>i1</code>
cutoff	real	Sets the local pairwise cut off

The energetic functional form of the charge contributions to the total potential, which is based on a Yukawa potential. It is described by

$$E_{charge} = A \ln(1 + e^{-\kappa H}) (4 - 2H/d_1 - 2H/d_2),$$

where d_1 and d_2 represent particle diameters, $H = r_{ij} - \frac{1}{2}(d_1 + d_2)$, and A the electrostatic constant, which can be expressed in terms of surface potentials ψ_1 and ψ_2 ,

$$A = \pi \epsilon_0 \epsilon_r (\psi_1 + \psi_2)^2 d_1 d_2 / (d_1 + d_2) / 8,$$

assuming, that both surface potentials are approximately the same.

7.12.9.6 Bond

```

bond          = {
  active      -> boolean
  n           -> integer,
  data        ->
  {
    {
      i0       -> id,
      i1       -> id,
      k        -> real,
      l        -> real
    },
    ...
  }
};

```

Directive	Parameters	Description
active	boolean	Interaction activator; either <code>true</code> or <code>false</code> .
n	integer	Number of data entries; can be omitted when <code>type = {merge -> true}</code> .
data	struct	Summary of interaction constants
i0, i1	id	Site id, referring to types as defined by the <code>mass</code> paragraph in <code>types</code>
k	real	Bond spring constants
l	real	Equilibrium bond length

The energetic functional form of the bond contributions to the total potential is described by

$$E_{bond} = 1/2 k_{bond}(l - l_0)^2,$$

where k_{bond} represents a spring constant, l the bond length of bond $\{i_0, i_1\}$, and l_0 the equilibrium bond length.

7.12.9.7 Angle

```

angle          = {
  active       -> boolean
  n            -> integer,
  data         ->
  {
    {
      i0        -> id,
      i1        -> id,
      i2        -> id,
      k         -> real,
      theta     -> real
    },
    ...
  }
};

```

Directive	Parameters	Description
active	boolean	Interaction activator; either <code>true</code> or <code>false</code> .
n	integer	Number of data entries; can be omitted when <code>type = {merge -> true}</code> .
data	struct	Summary of interaction constants
i0, i1, i2	id	Site id, referring to types as defined by the <code>mass</code> paragraph in <code>types</code>
k	real	Angle spring constants
theta	real	Equilibrium bond angle

The energetic functional form of the angle contributions to the total potential is described by

$$E_{angle} = 1/2 k_{angle}(\theta - \theta_0)^2,$$

where k_{angle} represents a spring constant, θ the angle between bonds $\{i_0, i_1\}$ and $\{i_1, i_2\}$, and θ_0 the equilibrium angle.

7.12.10 Coulomb

7.12.10.1 Syntax

```
coulomb      = {
  charge      -> struct,
  pair        -> struct
};
```

Directive	Parameters	Description
charge	struct	Diffuse coarse-grained charge interaction descriptors
pair	struct	Pair interaction descriptors

The coulomb force field is a collection of electrostatic interactions. Its functional form is a summation of nonbonded interactions given by

$$E_{coulomb} = E_{pair} + E_{charge}$$

for charge and pair contributions respectively. The following paragraphs describe each contribution in detail.

7.12.10.2 Examples

```
variables    = {
};
```

```
simulation   = {
  types      -> {
    merge     -> true,
    coulomb   -> {
      charge  -> {
        active -> true,
        data   -> {
        }
      }
    }
  }
};
```

```
simulation   = {
  types      -> {
    merge     -> true,
    coulomb   -> {
      pair     -> {
        active -> true,
        data   -> {
        }
      }
    }
  }
};
```

```
    }  
  }  
};
```

7.12.10.3 References

1. R. D. Groot, "*Electrostatic interactions in dissipative particle dynamics - simulation of polyelectrolytes and anionic surfactants*", *J. Chem. Phys.* **2003**, 118, 11265-11277.
2. M. Gonzalez-Melchor, E. Mayoral, M. E. Valsquez, J. Alejandre, "*Electrostatic interactions in dissipative particle dynamics using the Ewald sums*", *J. Chem. Phys.* **2006**, 125, 224107.

7.12.10.4 Pair

```

bond          = {
  active      -> boolean,
  nbonded     -> integer,
  cutoff      -> real
};

```

Directive	Parameters	Description
active	boolean	Interaction activator; either <code>true</code> or <code>false</code> .
nbonded	integer	Number of bonded sites to exclude from nonbonded interactions; the given number is expected to be larger or equal to zero.
n	integer	Number of data entries; can be omitted when <code>type = {merge -> true}</code> .
data	struct	Summary of interaction constants
i0, i1	id	Site id, referring to <code>mass</code> paragraph in <code>types</code>

The energetic functional form of the bond contributions to the total potential is described by

$$E_{pair} = \frac{q_i q_j}{4\pi\epsilon_0\epsilon_r r_{ij}},$$

where q_i and q_j represent the point charges of sites i and j respectively, ϵ_0 represents the dielectric permittivity, ϵ_r the dielectric constant, and r_{ij} the distance between site i and site j . Charges are set per site.

7.12.10.5 Charge

```
charge          = {
  active        -> boolean,
  nbonded       -> integer,
  k             -> real,
  cutoff        -> real
};
```

Directive	Parameters	Description
active	boolean	Interaction activator; either <code>true</code> or <code>false</code> .
n	integer	Number of data entries; can be omitted when <code>type = {merge -> true}</code> .
data	struct	Summary of interaction constants
i0, i1	id	Site id, referring to <code>mass</code> paragraph in <code>types</code>

The distribution of charges in a coarse-grained system cannot be described by simple point charges only. Instead, a spatial distribution following from the Poisson-Boltzmann equation (PBE) should be used, as described by Groot [Groot 2003]. A first approximation of a two-body approach for the solution to the PBE was given by Gonzalez-Melchor et al. [Gonzalez-Melchor 2006] as a function of exponentials. This solution, however, proved to inadequately describe the two-body solution. We propose a better approximation, which is based on an error function and is given by

$$E_{charge} = \frac{\text{erf}[kr_{ij}]}{r_{ij}},$$

where k represents a force constant and r_{ij} represents the distance between site i and site j . Charges are set per site. The advantage of this function is its easy integration in already existing standard point-charge descriptions, thus allowing for representation of long-range contributions by Ewald summations without change of already existing implementations.

7.12.11 DPD

7.12.11.1 Syntax

```
dpd          = {
  bond       -> struct,
  angle      -> struct,
  torsion    -> struct,
  improper    -> struct,
  pair       -> struct
};
```

Directive	Parameters	Description
bond	struct	Bond interaction descriptors
angle	struct	Angle interaction descriptors
torsion	struct	Torsion interaction descriptors
improper	struct	Improper interaction descriptors
pair	struct	Pair interaction descriptors

The Dissipative Particle Dynamics (DPD) force field - developed by Groot and Warren - is a compounded force field, that finds its application in mesoscale simulations.¹ Its functional form is a summation of bonded and nonbonded interactions given by

$$E_{dpd} = E_{pair} + E_{bond} + E_{angle} + E_{torsion} + E_{improper}$$

for bond, angle, torsion, improper, and pair contributions respectively. The following paragraphs describe each contribution in detail. See `./examples/dpd/` for applications.

7.12.11.2 Example

```
simulation    = {
  types       -> {
    merge     -> true,
    dpd -> {
      bond    -> {
        active -> true,
        data   -> {
          {i0  -> a, i1 -> b, k -> 20, l -> 1}
        }
      },
      angle   -> {
        active -> true,
        data   -> {
          {i0  -> a, i1 -> b, i2 -> c, k -> 5, theta -> 180}
        }
      },
      torsion -> {
        active -> true,
```

```

data    -> {
  {i0    -> a, i1 -> b, i2 -> c, i3 -> d, coefficients -> {
    {k -> 1.5, n -> 1, delta -> 0},
    {k -> 2.5, n -> 2, delta -> 0}
  }
}
},
pair    -> {
  active -> true,
  nbonded -> 3,
  mode   -> global,
  cutoff -> 1,
  data   -> {
    {i0 -> a, i1 -> a, a -> 25.0, gamma -> 4.5, cutoff -> 1},
    {i0 -> a, i1 -> b, a -> 26.2, gamma -> 4.5, cutoff -> 1},
    {i0 -> a, i1 -> c, a -> 22.1, gamma -> 4.5, cutoff -> 1},
    {i0 -> a, i1 -> d, a -> 20.0, gamma -> 4.5, cutoff -> 1},
    {i0 -> b, i1 -> b, a -> 25.0, gamma -> 4.5, cutoff -> 1},
    {i0 -> b, i1 -> c, a -> 28.0, gamma -> 4.5, cutoff -> 1},
    {i0 -> b, i1 -> d, a -> 29.0, gamma -> 4.5, cutoff -> 1},
    {i0 -> c, i1 -> c, a -> 25.0, gamma -> 4.5, cutoff -> 1},
    {i0 -> c, i1 -> d, a -> 21.3, gamma -> 4.5, cutoff -> 1},
    {i0 -> d, i1 -> d, a -> 25.0, gamma -> 4.5, cutoff -> 1}
  }
}
};

```

7.12.11.3 References

1. R.D. Groot and P.B. Warren "*Dissipative particle dynamics: Bridging the gap between atomistic and mesoscopic simulation*", *J. Chem. Phys.* **1997**, *107*, 4423-4435.

7.12.11.4 Pair

```

pair          = {
  active      -> boolean,
  mix         -> option,
  nbonded     -> integer,
  mode        -> option,
  cutoff      -> real,
  n           -> integer,
  data        ->
  {
    {
      i0      -> id,
      i1      -> id,
      a       -> real,
      gamma   -> real,
      cutoff  -> real
    },
    ...
  }
};

```

Directive	Parameters	Description
active	boolean	Interaction activator; either <code>true</code> or <code>false</code> .
mix	option	Sets mixing rule; options are <code>none</code> , <code>berthelot</code> , <code>arithmetic</code> , <code>geometric</code> , and <code>sixth</code> .
nbonded	integer	Number of bonded sites to exclude from nonbonded interactions; the given number is expected to be larger or equal to zero.
mode	option	Sets cut off mode; options are <code>global</code> using the value of <code>cutoff</code> for all contributions, <code>individual</code> defining cut offs per contribution, and <code>repulsive</code> defining the well location for each separate contribution as the cut off distance.
cutoff	real	Sets global cut off for <code>global</code> mode
n	integer	Number of data entries; can be omitted when <code>type = {merge -> true}</code> .
data	struct	Summary of interaction constants
i0, i1	id	Site id, referring to types as defined by the <code>mass</code> paragraph.
a	real	Force interaction constant.
gamma	real	Force damping constant.
cutoff	real	Site size; the point where the potential equals zero.

The energetic functional form of pair contributions to the total soft potential is described by a repulsive Hertzian spring,

$$E_{pair} = 1/2 \, a \, (1 - r_{ij}/r_{ij,cutoff})^2, \quad r_{ij} < r_{ij,cutoff}$$

where a represents the pairwise interaction paramater, r_{ij} the distance between site i and site j , and $r_{ij,cutoff}$ the cutoff distance between these two sites.

7.12.11.5 Bond

```

bond          = {
  active      -> boolean
  n           -> integer,
  data        ->
  {
    {
      i0       -> id,
      i1       -> id,
      k        -> real,
      l        -> real
    },
    ...
  }
};

```

Directive	Parameters	Description
active	boolean	Interaction activator; either <code>true</code> or <code>false</code> .
n	integer	Number of data entries; can be omitted when <code>type = {merge -> true}</code> .
data	struct	Summary of interaction constants
i0, i1	id	Site id, referring to types as defined by the <code>mass</code> paragraph in <code>types</code>
k	real	Bond spring constants
l	real	Equilibrium bond length

The energetic functional form of the bond contributions to the total potential is described by

$$E_{bond} = 1/2 k_{bond}(l - l_0)^2,$$

where k_{bond} represents a spring constant, l the bond length of bond $\{i_0, i_1\}$, and l_0 the equilibrium bond length.

7.12.11.6 Angle

```

angle          = {
  active       -> boolean
  n            -> integer,
  data         ->
  {
    {
      i0        -> id,
      i1        -> id,
      i2        -> id,
      k         -> real,
      theta     -> real
    },
    ...
  }
};

```

Directive	Parameters	Description
active	boolean	Interaction activator; either <code>true</code> or <code>false</code> .
n	integer	Number of data entries; can be omitted when <code>type = {merge -> true}</code> .
data	struct	Summary of interaction constants
i0, i1, i2	id	Site id, referring to types as defined by the <code>mass</code> paragraph in <code>types</code>
k	real	Angle spring constants
theta	real	Equilibrium bond angle

The energetic functional form of the angle contributions to the total potential is described by

$$E_{angle} = 1/2 k_{angle} (\cos(\theta) - \cos(\theta_0))^2,$$

where k_{angle} represents a spring constant, θ the angle between bonds $\{i_0, i_1\}$ and $\{i_1, i_2\}$, and θ_0 the equilibrium angle.

7.12.11.7 Torsion

```

torsion      = {
  active      -> boolean
  n           -> integer,
  data        ->
  {
    {
      i0       -> id,
      i1       -> id,
      i2       -> id,
      i3       -> id,
      coefficients ->
      {
        {
          k      -> real,
          n      -> integer,
          delta  -> real
        }
      }
    }
    ...
  }
  },
  ...
}
};

```

Directive	Parameters	Description
active	boolean	Interaction activator; either <code>true</code> or <code>false</code> .
n	integer	Number of data entries; can be omitted when <code>type = {merge -> true}</code> .
data	struct	Summary of interaction constants
i0, i1, i2, i3	id	Site id, referring to types as defined by the <code>mass</code> paragraph in <code>types</code>
k	real	Force constant
n	integer	Torsional angle prefactor; values range between 1 and 6
delta	real	Offset angle, mainly used to change the sign of the cosine function

The energetic functional form of the torsion contributions to the total potential is described by

$$E_{torsion} = \sum_{i=1}^m k_i (1 + \cos(n_i \phi - \delta_i)),$$

where k_i represents a set of torsion constants, δ_i a torsion offset, n_i the torsion pre-factor, and ϕ the bond torsion constructed by bonds $\{i_0, i_1\}$, $\{i_1, i_2\}$, and $\{i_2, i_3\}$. The maximum

allowed value of n_i is 6. Note, that the functional form presented here differs from the original OPLS definition.

7.12.11.8 Improper

```
improper      = {
  active      -> boolean
  n           -> integer,
  data        ->
  {
    {
      i0       -> id,
      i1       -> id,
      i2       -> id,
      i3       -> id,
      k        -> real,
      psi      -> real
    },
    ...
  }
};
```

Directive	Parameters	Description
active	boolean	Interaction activator; either <code>true</code> or <code>false</code> .
n	integer	Number of data entries; can be omitted when <code>type = {merge -> true}</code> .
data	struct	Summary of interaction constants
i0, i1, i2, i3	id	Site id, referring to types as defined by the <code>mass</code> paragraph in <code>types</code>
k	real	Force constant
psi	real	Improper equilibrium angle

The energetic functional form of the torsion contributions to the total potential is described by

$$E_{improper} = k_{improper}(\psi - \psi_0)^2,$$

where ψ represents the improper angle, which is given by the average of torsion angles formed by planes out of permutations of any of the following two bonds: $\{i_1, i_0\}$, $\{i_2, i_0\}$, and $\{i_3, i_0\}$. Note, that site i_0 is the central site, as opposed to some representations, where i_1 is the central site.

7.12.12 GROMACS

7.12.12.1 Syntax

```
gromacs      = {
  bond        -> struct,
  angle        -> struct,
  torsion      -> struct,
  improper      -> struct,
  pair14       -> struct,
  pair         -> struct
};
```

Directive	Parameters	Description
bond	struct	Bond interaction descriptors
angle	struct	Angle interaction descriptors
torsion	struct	Torsion interaction descriptors
improper	struct	Improper interaction descriptors
pair14	struct	1-4 intra-molecular pair interaction activator
pair	struct	Pair interaction descriptors

The gromacs force field is a compounded force field based on the GROMACS force field.¹ Its functional form is a summation of bonded and nonbonded interactions given by

$$E_{gromacs} = E_{pair} + E_{bond} + E_{angle} + E_{torsion} + E_{improper}$$

for bond, angle, torsion, and pair contributions respectively. The following paragraphs describe each contribution in detail. See `./examples/nanoparticle/coarse/` for an application.

7.12.12.2 Example

```
simulation    = {
  types       -> {
    merge      -> true,
    gromacs    -> {
      bond      -> {
        active  -> true,
        data    -> {
          {i0    -> a, i1 -> b, k -> 100, l -> 2}
        }
      },
      angle    -> {
        active  -> true,
        data    -> {
          {i0    -> a, i1 -> b, i2 -> c, k -> 100, theta -> 110}
        }
      },
      torsion  -> {
```

```

    active  -> true,
    data    -> {
      {i0    -> a, i1 -> b, i2 -> c, i3 -> d, coefficients -> {
        {k -> 1.5, n -> 1, delta -> 0},
        {k -> 2.5, n -> 2, delta -> 0}
      }
    }
  },
  improper  -> {
    active  -> true,
    data    -> {
      {i0    -> a, i1 -> b, i2 -> c, i3 -> d, k -> 100, psi -> 0}
    }
  },
  pair14    -> {
    active  -> true
  },
  pair      -> {
    active  -> true,
    nbonded -> 3,
    mode    -> global,
    cutoff  -> 10,
    mix     -> berthelot,
    data    -> {
      {i0    -> a, i1 -> a, sigma -> 4.0, epsilon -> 0.4},
      {i0    -> b, i1 -> b, sigma -> 3.7, epsilon -> 0.5},
      {i0    -> c, i1 -> c, sigma -> 4.1, epsilon -> 0.22},
      {i0    -> d, i1 -> d, sigma -> 2.4, epsilon -> 0.09},
    }
  }
}
};

```

7.12.12.3 References

1. W. L. Jorgensen and J. Tirado-Rives, "*The OPLS Potential Functions for Proteins. Energy Minimizations for Crystals of Cyclic Peptides and Crambin*", *J. Am. Chem. Soc.* **1988**, *110*, 1657-1666.

7.12.12.4 Pair

```
pair          = {
  active      -> boolean,
  nbonded     -> integer,
  mix         -> option,
  shift       -> boolean,
  coulomb     -> option,
  cutoff      -> real,
  mode        -> option,
  n           -> integer,
  data        ->
  {
    {
      i0      -> id,
      i1      -> id,
      epsilon -> real,
      sigma   -> real,
      cutoff  -> real
    },
    ...
  }
};
```

Directive	Parameters	Description
active	boolean	Interaction activator; either true or false .
nbonded	integer	Number of bonded sites to exclude from nonbonded interactions; the given number is expected to be larger or equal to zero.
mix	option	Sets mixing rule; options are none , berthelot , arithmetic , geometric , and sixth .
shift	boolean	Sets shifting of potential at cutoff; options are true and false .
coulomb	option	Set type of coulomb treatment; options are none , cut , and long .
cutoff	real	Sets global cut off for global mode
mode	option	Sets cut off mode; options are global using the value of cutoff for all contributions, individual defining cut offs per contribution, and repulsive defining the well location for each separate contribution as the cut off distance.
n	integer	Number of data entries; can be omitted when type = {merge -> true} .
data	struct	Summary of interaction constants
i0, i1	id	Site id, referring to types as defined by the mass paragraph.

epsilon	real	Force interaction constant; identical to the well depth for a Lennard-Jones 6-12 potential.
sigma	real	Site size; the point where the potential equals zero.
cutoff	real	Cut off; distance at which the pair-wise contribution is set to zero; can be omitted.

The energetic functional form of the pair contributions to the total potential is described by a 6-12 Lennard-Jones potential,

$$E_{pair} = 4\epsilon \left(\frac{\sigma}{r_{ij}} \right)^6 \left[\left(\frac{\sigma}{r_{ij}} \right)^6 - 1 \right],$$

where r_{ij} represents the distance between site i and site j .

7.12.12.5 Bond

```

bond          = {
  active      -> boolean
  n           -> integer,
  data        ->
  {
    {
      i0       -> id,
      i1       -> id,
      k        -> real,
      l        -> real
    },
    ...
  }
};

```

Directive	Parameters	Description
active	boolean	Interaction activator; either <code>true</code> or <code>false</code> .
n	integer	Number of data entries; can be omitted when <code>type = {merge -> true}</code> .
data	struct	Summary of interaction constants
i0, i1	id	Site id, referring to types as defined by the <code>mass</code> paragraph in <code>types</code>
k	real	Bond spring constants
l	real	Equilibrium bond length

The energetic functional form of the bond contributions to the total potential is described by

$$E_{bond} = 1/2 k_{bond}(l - l_0)^2,$$

where k_{bond} represents a spring constant, l the bond length of bond $\{i_0, i_1\}$, and l_0 the equilibrium bond length.

7.12.12.6 Angle

```

angle          = {
  active       -> boolean
  n            -> integer,
  data         ->
  {
    {
      i0        -> id,
      i1        -> id,
      i2        -> id,
      k         -> real,
      theta     -> real
    },
    ...
  }
};

```

Directive	Parameters	Description
active	boolean	Interaction activator; either <code>true</code> or <code>false</code> .
n	integer	Number of data entries; can be omitted when <code>type = {merge -> true}</code> .
data	struct	Summary of interaction constants
i0, i1, i2	id	Site id, referring to types as defined by the <code>mass</code> paragraph in <code>types</code>
k	real	Angle spring constants
theta	real	Equilibrium bond angle

The energetic functional form of the angle contributions to the total potential is described by

$$E_{angle} = 1/2 k_{angle}(\theta - \theta_0)^2,$$

where k_{angle} represents a spring constant, θ the angle between bonds $\{i_0, i_1\}$ and $\{i_1, i_2\}$, and θ_0 the equilibrium angle.

7.12.12.7 Torsion

```

torsion      = {
  active      -> boolean
  n           -> integer,
  data        ->
  {
    {
      i0       -> id,
      i1       -> id,
      i2       -> id,
      i3       -> id,
      coefficients ->
      {
        {
          k      -> real,
          n      -> integer,
          delta  -> real
        }
        ...
      }
    },
    ...
  }
};

```

Directive	Parameters	Description
active	boolean	Interaction activator; either <code>true</code> or <code>false</code> .
n	integer	Number of data entries; can be omitted when <code>type = {merge -> true}</code> .
data	struct	Summary of interaction constants
i0, i1, i2, i3	id	Site id, referring to types as defined by the <code>mass</code> paragraph in <code>types</code>
k	real	Force constant
n	integer	Torsional angle prefactor; values range between 1 and 4
delta	real	Offset angle, mainly used to change the sign of the cosine function

The energetic functional form of the torsion contributions to the total potential is described by

$$E_{torsion} = \sum_{i=1}^m k_i (1 + \text{sign}(n_i) \cos(n_i \phi - \delta_i)),$$

where ϕ represents the bond torsion constructed by bonds $\{i_0, i_1\}$, $\{i_1, i_2\}$, and $\{i_2, i_3\}$. Function $\text{sign}(n_i)$ is +1 for odd and -1 for even values of n_i .

7.12.12.8 Pair14

```
pair14      = {  
  active    -> boolean,  
};
```

Directive	Parameters	Description
<code>active</code>	<code>boolean</code>	Interaction activator; either <code>true</code> or <code>false</code> .

Representation of the 1-4 intra-molecular pair-wise contributions, which have the same functional form as the pair contributions in the following paragraph. The pre-factor, however is 0.5, as described in Jorgensen et al. [Jorgensen 1988].

7.12.12.9 Improper

```

improper      = {
  active      -> boolean
  n           -> integer,
  data        ->
  {
    {
      i0       -> id,
      i1       -> id,
      i2       -> id,
      i3       -> id,
      k        -> real,
      psi      -> real
    },
    ...
  }
};

```

Directive	Parameters	Description
active	boolean	Interaction activator; either <code>true</code> or <code>false</code> .
n	integer	Number of data entries; can be omitted when <code>type = {merge -> true}</code> .
data	struct	Summary of interaction constants
i0, i1, i2, i3	id	Site id, referring to types as defined by the <code>mass</code> paragraph in <code>types</code>
k	real	Force constant
psi	real	Improper equilibrium angle

The energetic functional form of the torsion contributions to the total potential is described by

$$E_{improper} = k_{improper}(\psi - \psi_0)^2,$$

where ψ represents the improper angle, which is given by the average of torsion angles formed by planes out of permutations of any of the following two bonds: $\{i_1, i_0\}$, $\{i_2, i_0\}$, and $\{i_3, i_0\}$. Note, that site i_0 is the central site, as opposed to some representations, where i_1 is the central site.

7.12.13 MARTINI

7.12.13.1 Syntax

```
martini      = {
  bond        -> struct,
  angle       -> struct,
  torsion     -> struct,
  improper     -> struct,
  pair14      -> struct,
  pair        -> struct
};
```

Directive	Parameters	Description
bond	struct	Bond interaction descriptors
angle	struct	Angle interaction descriptors
torsion	struct	Torsion interaction descriptors
improper	struct	Improper interaction descriptors
pair14	struct	1-4 intra-molecular pair interaction activator
pair	struct	Pair interaction descriptors

The martini force field is a compounded force field based on the MARTINI force field.¹ Its functional form is a summation of bonded and nonbonded interactions given by

$$E_{martini} = E_{pair} + E_{bond} + E_{angle} + E_{torsion} + E_{improper}$$

for bond, angle, torsion, improper, and pair contributions respectively. The following paragraphs describe each contribution in detail. See `./examples/field/martini/` for an application.

7.12.13.2 Example

```
simulation    = {
  types       -> {
    merge     -> true,
    martini   -> {
      bond    -> {
        active -> true,
        data   -> {
          {i0  -> a, i1 -> b, k -> 100, l -> 2}
        }
      },
      angle   -> {
        active -> true,
        data   -> {
          {i0  -> a, i1 -> b, i2 -> c, k -> 100, theta -> 110}
        }
      },
    },
  },
};
```

```

torsion    -> {
  active    -> true,
  data      -> {
    {i0      -> a, i1 -> b, i2 -> c, i3 -> d, coefficients -> {
      {k -> 1.5, n -> 1, delta -> 0},
      {k -> 2.5, n -> 2, delta -> 0}
    }
  }
},
improper    -> {
  active    -> true,
  data      -> {
    {i0      -> a, i1 -> b, i2 -> c, i3 -> d, k -> 100, psi -> 0}
  }
},
pair14      -> {
  active    -> true
},
pair        -> {
  active    -> true,
  nbonded   -> 3,
  mode      -> global,
  cutoff     -> 10,
  mix       -> berthelot,
  data      -> {
    {i0      -> a, i1 -> a, sigma -> 4.0, epsilon -> 0.4},
    {i0      -> b, i1 -> b, sigma -> 3.7, epsilon -> 0.5},
    {i0      -> c, i1 -> c, sigma -> 4.1, epsilon -> 0.22},
    {i0      -> d, i1 -> d, sigma -> 2.4, epsilon -> 0.09},
  }
}
};

```

7.12.13.3 References

1. W. L. Jorgensen and J. Tirado-Rives, "*The OPLS Potential Functions for Proteins. Energy Minimizations for Crystals of Cyclic Peptides and Crambin*", *J. Am. Chem. Soc.* **1988**, *110*, 1657-1666.

7.12.13.4 Pair

```
pair          = {
  active      -> boolean,
  nbonded     -> integer,
  mix         -> option,
  shift       -> boolean,
  coulomb     -> option,
  cutoff      -> real,
  mode        -> option,
  n           -> integer,
  data        ->
  {
    {
      i0      -> id,
      i1      -> id,
      epsilon -> real,
      sigma   -> real,
      cutoff  -> real
    },
    ...
  }
};
```

Directive	Parameters	Description
active	boolean	Interaction activator; either true or false .
nbonded	integer	Number of bonded sites to exclude from nonbonded interactions; the given number is expected to be larger or equal to zero.
mix	option	Sets mixing rule; options are none , berthelot , arithmetic , geometric , and sixth .
shift	boolean	Sets shifting of potential at cutoff; options are true and false .
coulomb	option	Set type of coulomb treatment; options are none , cut , and long .
cutoff	real	Sets global cut off for global mode
mode	option	Sets cut off mode; options are global using the value of cutoff for all contributions, individual defining cut offs per contribution, and repulsive defining the well location for each separate contribution as the cut off distance.
n	integer	Number of data entries; can be omitted when type = {merge -> true} .
data	struct	Summary of interaction constants
i0, i1	id	Site id, referring to types as defined by the mass paragraph.

epsilon	real	Force interaction constant; identical to the well depth for a Lennard-Jones 6-12 potential.
sigma	real	Site size; the point where the potential equals zero.
cutoff	real	Cut off; distance at which the pair-wise contribution is set to zero; can be omitted.

The energetic functional form of the pair contributions to the total potential is described by a 6-12 Lennard-Jones potential,

$$E_{pair} = 4\epsilon \left(\frac{\sigma}{r_{ij}} \right)^6 \left[\left(\frac{\sigma}{r_{ij}} \right)^6 - 1 \right],$$

where r_{ij} represents the distance between site i and site j .

7.12.13.5 Bond

```

bond          = {
  active      -> boolean
  n           -> integer,
  data        ->
  {
    {
      i0       -> id,
      i1       -> id,
      k        -> real,
      l        -> real
    },
    ...
  }
};

```

Directive	Parameters	Description
active	boolean	Interaction activator; either <code>true</code> or <code>false</code> .
n	integer	Number of data entries; can be omitted when <code>type = {merge -> true}</code> .
data	struct	Summary of interaction constants
i0, i1	id	Site id, referring to types as defined by the <code>mass</code> paragraph in <code>types</code>
k	real	Bond spring constants
l	real	Equilibrium bond length

The energetic functional form of the bond contributions to the total potential is described by

$$E_{bond} = 1/2 k_{bond}(l - l_0)^2,$$

where k_{bond} represents a spring constant, l the bond length of bond $\{i_0, i_1\}$, and l_0 the equilibrium bond length.

7.12.13.6 Angle

```

angle          = {
  active       -> boolean
  n            -> integer,
  data         ->
  {
    {
      i0        -> id,
      i1        -> id,
      i2        -> id,
      k         -> real,
      theta     -> real
    },
    ...
  }
};

```

Directive	Parameters	Description
active	boolean	Interaction activator; either <code>true</code> or <code>false</code> .
n	integer	Number of data entries; can be omitted when <code>type = {merge -> true}</code> .
data	struct	Summary of interaction constants
i0, i1, i2	id	Site id, referring to types as defined by the <code>mass</code> paragraph in <code>types</code>
k	real	Angle spring constants
theta	real	Equilibrium bond angle

The energetic functional form of the angle contributions to the total potential is described by

$$E_{angle} = 1/2 k_{angle}(\theta - \theta_0)^2,$$

where k_{angle} represents a spring constant, θ the angle between bonds $\{i_0, i_1\}$ and $\{i_1, i_2\}$, and θ_0 the equilibrium angle.

7.12.13.7 Torsion

```

torsion      = {
  active      -> boolean
  n           -> integer,
  data        ->
  {
    {
      i0       -> id,
      i1       -> id,
      i2       -> id,
      i3       -> id,
      coefficients ->
      {
        {
          k      -> real,
          n      -> integer,
          delta  -> real
        }
        ...
      }
    },
    ...
  }
};

```

Directive	Parameters	Description
active	boolean	Interaction activator; either <code>true</code> or <code>false</code> .
n	integer	Number of data entries; can be omitted when <code>type = {merge -> true}</code> .
data	struct	Summary of interaction constants
i0, i1, i2, i3	id	Site id, referring to types as defined by the <code>mass</code> paragraph in <code>types</code>
k	real	Force constant
n	integer	Torsional angle prefactor; values range between 1 and 4
delta	real	Offset angle, mainly used to change the sign of the cosine function

The energetic functional form of the torsion contributions to the total potential is described by

$$E_{torsion} = \sum_{i=1}^m k_i (1 + \text{sign}(n_i) \cos(n_i \phi - \delta_i)),$$

where ϕ represents the bond torsion constructed by bonds $\{i_0, i_1\}$, $\{i_1, i_2\}$, and $\{i_2, i_3\}$. Function $\text{sign}(n_i)$ is +1 for odd and -1 for even values of n_i .

7.12.13.8 Improper

```

improper      = {
  active      -> boolean
  n           -> integer,
  data        ->
  {
    {
      i0       -> id,
      i1       -> id,
      i2       -> id,
      i3       -> id,
      k        -> real,
      psi      -> real
    },
    ...
  }
};

```

Directive	Parameters	Description
active	boolean	Interaction activator; either <code>true</code> or <code>false</code> .
n	integer	Number of data entries; can be omitted when <code>type = {merge -> true}</code> .
data	struct	Summary of interaction constants
i0, i1, i2,	id	Site id, referring to types as defined by the <code>mass</code> paragraph in <code>types</code>
i3		
k	real	Force constant
psi	real	Improper equilibrium angle

The energetic functional form of the torsion contributions to the total potential is described by

$$E_{improper} = k_{improper}(\psi - \psi_0)^2,$$

where ψ represents the improper angle, which is given by the average of torsion angles formed by planes out of permutations of any of the following two bonds: $\{i_1, i_0\}$, $\{i_2, i_0\}$, and $\{i_3, i_0\}$. Note, that site i_0 is the central site, as opposed to some representations, where i_1 is the central site.

7.12.14 OPLS

7.12.14.1 Syntax

```
opls          = {
  bond        -> struct,
  angle       -> struct,
  torsion     -> struct,
  improper     -> struct,
  pair14      -> struct,
  pair        -> struct
};
```

Directive	Parameters	Description
bond	struct	Bond interaction descriptors
angle	struct	Angle interaction descriptors
torsion	struct	Torsion interaction descriptors
improper	struct	Improper interaction descriptors
pair14	struct	1-4 intra-molecular pair interaction activator
pair	struct	Pair interaction descriptors

The OPLS force field is a compounded force field based on the OPLS force field.¹ Its functional form is a summation of bonded and nonbonded interactions given by

$$E_{opls} = E_{pair} + E_{bond} + E_{angle} + E_{torsion} + E_{improper}$$

for bond, angle, torsion, and pair contributions respectively. The following paragraphs describe each contribution in detail. See `./examples/field/opls/` for applications.

7.12.14.2 Example

```
simulation    = {
  types       -> {
    merge     -> true,
    opls      -> {
      bond    -> {
        active -> true,
        data   -> {
          {i0  -> a, i1 -> b, k -> 100, l -> 2}
        }
      },
      angle   -> {
        active -> true,
        data   -> {
          {i0  -> a, i1 -> b, i2 -> c, k -> 100, theta -> 110}
        }
      },
      torsion -> {
```

```

    active  -> true,
    data    -> {
      {i0    -> a, i1 -> b, i2 -> c, i3 -> d, coefficients -> {
        {k -> 1.5, n -> 1, delta -> 0},
        {k -> 2.5, n -> 2, delta -> 0}
      }
    }
  },
  improper  -> {
    active  -> true,
    data    -> {
      {i0    -> a, i1 -> b, i2 -> c, i3 -> d, k -> 100, psi -> 0}
    }
  },
  pair14    -> {
    active  -> true
  },
  pair      -> {
    active  -> true,
    nbonded -> 3,
    mode    -> global,
    cutoff  -> 10,
    mix     -> berthelot,
    data    -> {
      {i0    -> a, i1 -> a, sigma -> 4.0, epsilon -> 0.4},
      {i0    -> b, i1 -> b, sigma -> 3.7, epsilon -> 0.5},
      {i0    -> c, i1 -> c, sigma -> 4.1, epsilon -> 0.22},
      {i0    -> d, i1 -> d, sigma -> 2.4, epsilon -> 0.09},
    }
  }
}
};

```

7.12.14.3 References

1. W. L. Jorgensen and J. Tirado-Rives, "*The OPLS Potential Functions for Proteins. Energy Minimizations for Crystals of Cyclic Peptides and Crambin*", *J. Am. Chem. Soc.* **1988**, *110*, 1657-1666.

7.12.14.4 Pair

```
pair          = {
  active      -> boolean,
  nbonded     -> integer,
  mix         -> option,
  shift       -> boolean,
  coulomb     -> option,
  cutoff      -> real,
  mode        -> option,
  n           -> integer,
  data        ->
  {
    {
      i0      -> id,
      i1      -> id,
      epsilon -> real,
      sigma   -> real,
      cutoff  -> real
    },
    ...
  }
};
```

Directive	Parameters	Description
active	boolean	Interaction activator; either true or false .
nbonded	integer	Number of bonded sites to exclude from nonbonded interactions; the given number is expected to be larger or equal to zero.
mix	option	Sets mixing rule; options are none , berthelot , arithmetic , geometric , and sixth .
shift	boolean	Sets shifting of potential at cutoff; options are true and false .
coulomb	option	Set type of coulomb treatment; options are none , cut , and long .
cutoff	real	Sets global cut off for global mode
mode	option	Sets cut off mode; options are global using the value of cutoff for all contributions, individual defining cut offs per contribution, and repulsive defining the well location for each separate contribution as the cut off distance.
n	integer	Number of data entries; can be omitted when type = {merge -> true} .
data	struct	Summary of interaction constants
i0, i1	id	Site id, referring to types as defined by the mass paragraph.

epsilon	real	Force interaction constant; identical to the well depth for a Lennard-Jones 6-12 potential.
sigma	real	Site size; the point where the potential equals zero.
cutoff	real	Cut off; distance at which the pair-wise contribution is set to zero; can be omitted.

The energetic functional form of the pair contributions to the total potential is described by a 6-12 Lennard-Jones potential,

$$E_{pair} = 4\epsilon \left(\frac{\sigma}{r_{ij}} \right)^6 \left[\left(\frac{\sigma}{r_{ij}} \right)^6 - 1 \right],$$

where r_{ij} represents the distance between site i and site j .

7.12.14.5 Bond

```

bond          = {
  active      -> boolean
  n           -> integer,
  data        ->
  {
    {
      i0       -> id,
      i1       -> id,
      k        -> real,
      l        -> real
    },
    ...
  }
};

```

Directive	Parameters	Description
active	boolean	Interaction activator; either <code>true</code> or <code>false</code> .
n	integer	Number of data entries; can be omitted when <code>type = {merge -> true}</code> .
data	struct	Summary of interaction constants
i0, i1	id	Site id, referring to types as defined by the <code>mass</code> paragraph in <code>types</code>
k	real	Bond spring constants
l	real	Equilibrium bond length

The energetic functional form of the bond contributions to the total potential is described by

$$E_{bond} = k_{bond}(l - l_0)^2,$$

where k_{bond} represents a spring constant, l the bond length of bond $\{i_0, i_1\}$, and l_0 the equilibrium bond length.

7.12.14.6 Angle

```

angle          = {
  active       -> boolean
  n            -> integer,
  data         ->
  {
    {
      i0        -> id,
      i1        -> id,
      i2        -> id,
      k         -> real,
      theta     -> real
    },
    ...
  }
};

```

Directive	Parameters	Description
active	boolean	Interaction activator; either <code>true</code> or <code>false</code> .
n	integer	Number of data entries; can be omitted when <code>type = {merge -> true}</code> .
data	struct	Summary of interaction constants
i0, i1, i2	id	Site id, referring to types as defined by the <code>mass</code> paragraph in <code>types</code>
k	real	Angle spring constants
theta	real	Equilibrium bond angle

The energetic functional form of the angle contributions to the total potential is described by

$$E_{angle} = k_{angle}(\theta - \theta_0)^2,$$

where k_{angle} represents a spring constant, θ the angle between bonds $\{i_0, i_1\}$ and $\{i_1, i_2\}$, and θ_0 the equilibrium angle.

7.12.14.7 Torsion

```

torsion      = {
  active      -> boolean
  n           -> integer,
  data        ->
  {
    {
      i0       -> id,
      i1       -> id,
      i2       -> id,
      i3       -> id,
      coefficients ->
      {
        {
          k      -> real,
          n      -> integer,
          delta  -> real
        }
        ...
      }
    },
    ...
  }
};

```

Directive	Parameters	Description
active	boolean	Interaction activator; either <code>true</code> or <code>false</code> .
n	integer	Number of data entries; can be omitted when <code>type = {merge -> true}</code> .
data	struct	Summary of interaction constants
i0, i1, i2, i3	id	Site id, referring to types as defined by the <code>mass</code> paragraph in <code>types</code>
k	real	Force constant
n	integer	Torsional angle prefactor; values range between 1 and 4
delta	real	Offset angle, mainly used to change the sign of the cosine function

The energetic functional form of the torsion contributions to the total potential is described by

$$E_{torsion} = \sum_{i=1}^m 1/2 k_i (1 + \text{sign}(n_i) \cos(n_i \phi - \delta_i)),$$

where ϕ represents the bond torsion constructed by bonds $\{i_0, i_1\}$, $\{i_1, i_2\}$, and $\{i_2, i_3\}$. Function $\text{sign}(n_i)$ is +1 for odd and -1 for even values of n_i .

7.12.14.8 Pair14

```
pair14      = {  
  active    -> boolean,  
};
```

Directive	Parameters	Description
<code>active</code>	<code>boolean</code>	Interaction activator; either <code>true</code> or <code>false</code> .

Representation of the 1-4 intra-molecular pair-wise contributions, which have the same functional form as the pair contributions in the following paragraph. The pre-factor, however is 0.5, as described in Jorgensen et al. [Jorgensen 1988].

7.12.14.9 Improper

```
improper      = {
  active      -> boolean
  n           -> integer,
  data        ->
  {
    {
      i0       -> id,
      i1       -> id,
      i2       -> id,
      i3       -> id,
      k        -> real,
      psi      -> real
    },
    ...
  }
};
```

Directive	Parameters	Description
active	boolean	Interaction activator; either <code>true</code> or <code>false</code> .
n	integer	Number of data entries; can be omitted when <code>type = {merge -> true}</code> .
data	struct	Summary of interaction constants
i0, i1, i2, i3	id	Site id, referring to types as defined by the <code>mass</code> paragraph in <code>types</code>
k	real	Force constant
psi	real	Improper equilibrium angle

The energetic functional form of the torsion contributions to the total potential is described by

$$E_{improper} = k_{improper}(\psi - \psi_0)^2,$$

where ψ represents the improper angle, which is given by the average of torsion angles formed by planes out of permutations of any of the following two bonds: $\{i_1, i_0\}$, $\{i_2, i_0\}$, and $\{i_3, i_0\}$. Note, that site i_0 is the central site, as opposed to some representations, where i_1 is the central site.

7.12.15 SDK

7.12.15.1 Syntax

```

sdk          = {
  bond       -> struct,
  angle      -> struct,
  torsion    -> struct,
  improper    -> struct,
  pair14     -> struct,
  pair       -> struct
};

```

Directive	Parameters	Description
bond	struct	Bond interaction descriptors
angle	struct	Angle interaction descriptors
torsion	struct	Torsion interaction descriptors
improper	struct	Improper interaction descriptors
pair14	struct	1-4 intra-molecular pair interaction activator
pair	struct	Pair interaction descriptors

The SDK force field is a compounded force field based on the SDK force field.¹ Its functional form is a summation of bonded and nonbonded interactions given by

$$E_{sdk} = E_{pair} + E_{bond} + E_{angle} + E_{torsion} + E_{improper}$$

for bond, angle, torsion, and pair contributions respectively. The following paragraphs describe each contribution in detail. See `./examples/field/sdk/` for applications.

7.12.15.2 Example

```

simulation    = {
  types       -> {
    merge     -> true,
    sdk -> {
      bond    -> {
        active -> true,
        data   -> {
          {i0   -> a, i1 -> b, k -> 100, l -> 2}
        }
      },
      angle   -> {
        active -> true,
        data   -> {
          {i0   -> a, i1 -> b, i2 -> c, k -> 100, theta -> 110}
        }
      },
      torsion  -> {

```

```

    active  -> true,
    data    -> {
      {i0    -> a, i1 -> b, i2 -> c, i3 -> d, coefficients -> {
        {k -> 1.5, n -> 1, delta -> 0},
        {k -> 2.5, n -> 2, delta -> 0}
      }
    }
  },
  improper  -> {
    active  -> true,
    data    -> {
      {i0    -> a, i1 -> b, i2 -> c, i3 -> d, k -> 100, psi -> 0}
    }
  },
  pair14    -> {
    active  -> true
  },
  pair      -> {
    active  -> true,
    nbonded -> 3,
    mode    -> global,
    cutoff  -> 10,
    mix     -> berthelot,
    data    -> {
      {i0    -> a, i1 -> a, sigma -> 4.0, epsilon -> 0.4},
      {i0    -> b, i1 -> b, sigma -> 3.7, epsilon -> 0.5},
      {i0    -> c, i1 -> c, sigma -> 4.1, epsilon -> 0.22},
      {i0    -> d, i1 -> d, sigma -> 2.4, epsilon -> 0.09},
    }
  }
}
};

```

7.12.15.3 References

1. Shinoda, Devane, Klein papers
2. W. L. Jorgensen and J. Tirado-Rives, "*The SDK Potential Functions for Proteins. Energy Minimizations for Crystals of Cyclic Peptides and Crambin*", *J. Am. Chem. Soc.* **1988**, *110*, 1657-1666.

7.12.15.4 Pair

```
pair          = {
  active      -> boolean,
  nbonded     -> integer,
  mix         -> option,
  shift       -> boolean,
  coulomb     -> option,
  cutoff      -> real,
  mode        -> option,
  n           -> integer,
  data        ->
  {
    {
      i0      -> id,
      i1      -> id,
      epsilon -> real,
      sigma   -> real,
      cutoff  -> real
    },
    ...
  }
};
```

Directive	Parameters	Description
active	boolean	Interaction activator; either true or false .
nbonded	integer	Number of bonded sites to exclude from nonbonded interactions; the given number is expected to be larger or equal to zero.
mix	option	Sets mixing rule; options are none , berthelot , arithmetic , geometric , and sixth .
shift	boolean	Sets shifting of potential at cutoff; options are true and false .
coulomb	option	Set type of coulomb treatment; options are none , cut , and long .
cutoff	real	Sets global cut off for global mode
mode	option	Sets cut off mode; options are global using the value of cutoff for all contributions, individual defining cut offs per contribution, and repulsive defining the well location for each separate contribution as the cut off distance.
n	integer	Number of data entries; can be omitted when type = {merge -> true} .
data	struct	Summary of interaction constants
i0, i1	id	Site id, referring to types as defined by the mass paragraph.

epsilon	real	Force interaction constant; identical to the well depth for a Lennard-Jones 6-12 potential.
sigma	real	Site size; the point where the potential equals zero.
cutoff	real	Cut off; distance at which the pair-wise contribution is set to zero; can be omitted.

The energetic functional form of the pair contributions to the total potential is described by a 6-12 Lennard-Jones potential,

$$E_{pair} = 4\epsilon \left(\frac{\sigma}{r_{ij}} \right)^6 \left[\left(\frac{\sigma}{r_{ij}} \right)^6 - 1 \right],$$

where r_{ij} represents the distance between site i and site j .

7.12.15.5 Bond

```

bond          = {
  active      -> boolean
  n           -> integer,
  data        ->
  {
    {
      i0       -> id,
      i1       -> id,
      k        -> real,
      l        -> real
    },
    ...
  }
};

```

Directive	Parameters	Description
active	boolean	Interaction activator; either <code>true</code> or <code>false</code> .
n	integer	Number of data entries; can be omitted when <code>type = {merge -> true}</code> .
data	struct	Summary of interaction constants
i0, i1	id	Site id, referring to types as defined by the <code>mass</code> paragraph in <code>types</code>
k	real	Bond spring constants
l	real	Equilibrium bond length

The energetic functional form of the bond contributions to the total potential is described by

$$E_{bond} = k_{bond}(l - l_0)^2,$$

where k_{bond} represents a spring constant, l the bond length of bond $\{i_0, i_1\}$, and l_0 the equilibrium bond length.

7.12.15.6 Angle

```

angle          = {
  active       -> boolean
  n            -> integer,
  data         ->
  {
    {
      i0        -> id,
      i1        -> id,
      i2        -> id,
      k         -> real,
      theta     -> real
    },
    ...
  }
};

```

Directive	Parameters	Description
active	boolean	Interaction activator; either <code>true</code> or <code>false</code> .
n	integer	Number of data entries; can be omitted when <code>type = {merge -> true}</code> .
data	struct	Summary of interaction constants
i0, i1, i2	id	Site id, referring to types as defined by the <code>mass</code> paragraph in <code>types</code>
k	real	Angle spring constants
theta	real	Equilibrium bond angle

The energetic functional form of the angle contributions to the total potential is described by

$$E_{angle} = k_{angle}(\theta - \theta_0)^2,$$

where k_{angle} represents a spring constant, θ the angle between bonds $\{i_0, i_1\}$ and $\{i_1, i_2\}$, and θ_0 the equilibrium angle.

7.12.15.7 Torsion

```

torsion      = {
  active      -> boolean
  n           -> integer,
  data        ->
  {
    {
      i0       -> id,
      i1       -> id,
      i2       -> id,
      i3       -> id,
      coefficients ->
      {
        {
          k      -> real,
          n      -> integer,
          delta  -> real
        }
        ...
      }
    },
    ...
  }
};

```

Directive	Parameters	Description
active	boolean	Interaction activator; either <code>true</code> or <code>false</code> .
n	integer	Number of data entries; can be omitted when <code>type = {merge -> true}</code> .
data	struct	Summary of interaction constants
i0, i1, i2, i3	id	Site id, referring to types as defined by the <code>mass</code> paragraph in <code>types</code>
k	real	Force constant
n	integer	Torsional angle prefactor; values range between 1 and 4
delta	real	Offset angle, mainly used to change the sign of the cosine function

The energetic functional form of the torsion contributions to the total potential is described by

$$E_{torsion} = \sum_{i=1}^m 1/2 k_i (1 + \text{sign}(n_i) \cos(n_i \phi - \delta_i)),$$

where ϕ represents the bond torsion constructed by bonds $\{i_0, i_1\}$, $\{i_1, i_2\}$, and $\{i_2, i_3\}$. Function $\text{sign}(n_i)$ is +1 for odd and -1 for even values of n_i .

7.12.15.8 Pair14

```
pair14      = {  
  active    -> boolean,  
};
```

Directive	Parameters	Description
active	boolean	Interaction activator; either true or false .

Representation of the 1-4 intra-molecular pair-wise contributions, which have the same functional form as the pair contributions in the following paragraph. The pre-factor, however is 0.5, as described in Jorgensen et al. [Jorgensen 1988].

7.12.15.9 Improper

```
improper      = {
  active      -> boolean
  n           -> integer,
  data        ->
  {
    {
      i0       -> id,
      i1       -> id,
      i2       -> id,
      i3       -> id,
      k        -> real,
      psi      -> real
    },
    ...
  }
};
```

Directive	Parameters	Description
active	boolean	Interaction activator; either <code>true</code> or <code>false</code> .
n	integer	Number of data entries; can be omitted when <code>type = {merge -> true}</code> .
data	struct	Summary of interaction constants
i0, i1, i2, i3	id	Site id, referring to types as defined by the <code>mass</code> paragraph in <code>types</code>
k	real	Force constant
psi	real	Improper equilibrium angle

The energetic functional form of the torsion contributions to the total potential is described by

$$E_{improper} = k_{improper}(\psi - \psi_0)^2,$$

where ψ represents the improper angle, which is given by the average of torsion angles formed by planes out of permutations of any of the following two bonds: $\{i_1, i_0\}$, $\{i_2, i_0\}$, and $\{i_3, i_0\}$. Note, that site i_0 is the central site, as opposed to some representations, where i_1 is the central site.

7.12.16 Spline

7.12.16.1 Syntax

```
spline      = {
  bond      -> struct,
  angle     -> struct,
  torsion   -> struct,
  pair      -> struct
};
```

Directive	Parameters	Description
bond	struct	Bond interaction descriptors
angle	struct	Angle interaction descriptors
torsion	struct	Torsion interaction descriptors
pair	struct	Pair interaction descriptors

The spline force field uses a set of splined functions for nonbonded and bonded interactions (see Section 7.9 [Splines], page 185). Its functional form is a summation of bonded and nonbonded interactions given by

$$E_{\text{spline}} = E_{\text{pair}} + E_{\text{bond}} + E_{\text{angle}} + E_{\text{torsion}}$$

for bond, angle, torsion, and pair contributions respectively. The following paragraphs describe each contribution in detail. See `./examples/field/spline/` for an application.

7.12.16.2 Bond

```
bond      = {
  active   -> boolean
  n        -> integer,
  data     ->
  {
    {
      i0     -> id,
      i1     -> id,
      spline -> struct
    },
    ...
  }
};
```

Directive	Parameters	Description
active	boolean	Interaction activator; either <code>true</code> or <code>false</code> .
n	integer	Number of data entries; can be omitted when <code>type = {merge -> true}</code> .
data	struct	Summary of interaction constants

i0, i1	id	Site id, referring to mass paragraph in types
spline	struct	Spline function structure.

The energetic functional form of the bond contributions to the total potential is described by

$$E_{bond} =,$$

where

7.12.16.3 Angle

```
angle      = {
  active    -> boolean
  n         -> integer,
  data      ->
  {
    {
      i0      -> id,
      i1      -> id,
      i2      -> id,
      spline  -> struct
    },
    ...
  }
};
```

Directive	Parameters	Description
active	boolean	Interaction activator; either true or false .
n	integer	Number of data entries; can be omitted when type = {merge -> true} .
data	struct	Summary of interaction constants
i0, i1, i2	id	Site id, referring to mass paragraph in types
spline	struct	Spline function structure.

The energetic functional form of the angle contributions to the total potential is described by

$$E_{angle} =,$$

where θ represents the bond angle.

7.12.16.4 Torsion

```
torsion    = {
  active    -> boolean
  n         -> integer,
```

```

data      ->
{
  {
    i0      -> id,
    i1      -> id,
    i2      -> id,
    i3      -> id,
    spline  -> struct
  },
  ...
}
};

```

Directive	Parameters	Description
active	boolean	Interaction activator; either <code>true</code> or <code>false</code> .
n	integer	Number of data entries; can be omitted when <code>type = {merge -> true}</code> .
data	struct	Summary of interaction constants
i0, i1, i2, i3	id	Site id, referring to <code>mass</code> paragraph in <code>types</code>
spline	struct	Spline function structure.

The energetic functional form of the torsion contributions to the total potential is described by

$$E_{torsion} =,$$

where ϕ represents the bond torsion.

7.12.16.5 Pair

```

bond      = {
  active   -> boolean,
  nbonded  -> integer,
  n        -> integer,
  data     ->
  {
    {
      i0      -> id,
      i1      -> id,
      spline  -> struct
    },
    ...
  }
};

```


Directive	Parameters	Description
<code>active</code>	<code>boolean</code>	Interaction activator; either <code>true</code> or <code>false</code> .
<code>nbonded</code>	<code>integer</code>	Number of bonded sites to exclude from nonbonded interactions; the given number is expected to be larger or equal to zero.
<code>n</code>	<code>integer</code>	Number of data entries; can be omitted when <code>type = {merge -> true}</code> .
<code>data</code>	<code>struct</code>	Summary of interaction constants
<code>i0, i1</code>	<code>id</code>	Site id, referring to <code>mass</code> paragraph in <code>types</code>
<code>spline</code>	<code>struct</code>	Spline function structure.

The energetic functional form of the pair contributions to the total potential is formed by a linear interpolation or a cubic spline through the provided data.

7.12.16.6 Example

```
variables      = {
};

simulation     = {
  types        -> {
    merge      -> true,
    boltzmann  -> {
      bond     -> {
        active  -> true,
        data    -> {
        }
      }
    },
    angle      -> {
      active    -> true,
      data      -> {
      }
    },
    pair       -> {
      active    -> true,
      nbonded   -> 3,
      data      -> {
      }
    }
  }
}
};
```

7.12.16.7 References

1. Spyriouni et al., *Macromolecules* 2007, 40, 3876

7.12.17 Standard

7.12.17.1 Syntax

```
standard      = {
  pair        -> struct,
  bond        -> struct,
  angle       -> struct,
  torsion     -> struct,
  improper    -> struct
};
```

Directive	Parameters	Description
pair	struct	Pair interaction descriptors
bond	struct	Bond interaction descriptors
angle	struct	Angle interaction descriptors
torsion	struct	Torsion interaction descriptors
improper	struct	Improper interaction descriptors

The standard force field is a compounded force field, that is OPLS-like in its appearance. Its functional form is a summation of nonbonded and bonded interactions given by

$$E_{\text{standard}} = E_{\text{pair}} + E_{\text{bond}} + E_{\text{angle}} + E_{\text{torsion}} + E_{\text{improper}}$$

for pair, bond, angle, torsion, and improper contributions respectively. The following paragraphs describe each contribution in detail. See `./examples/lj/` for an application. Note, that no standardized typing is provided for the standard force field.

7.12.17.2 Example

```
simulation      = {
  types        -> {
    merge      -> true,
    standard   -> {
      bond     -> {
        active  -> true,
        data    -> {
          {i0   -> a, i1 -> b, k -> 100, l -> 2}
        }
      },
      angle    -> {
        active  -> true,
        data    -> {
          {i0   -> a, i1 -> b, i2 -> c, k -> 100, theta -> 110}
        }
      },
      torsion   -> {
        active  -> true,
```

```

    data    -> {
      {i0    -> a, i1 -> b, i2 -> c, i3 -> d, coefficients -> {
        {k -> 1.5, n -> 1, delta -> 0},
        {k -> 2.5, n -> 2, delta -> 0}
      }
    }
  },
  improper -> {
    active -> true,
    data    -> {
      {i0    -> a, i1 -> b, i2 -> c, i3 -> d, k -> 100, psi -> 0}
    }
  },
  pair     -> {
    active -> true,
    nbonded -> 3,
    mode    -> global,
    cutoff  -> 10,
    mix     -> berthelot,
    data    -> {
      {i0    -> a, i1 -> a, sigma -> 4.0, epsilon -> 0.4},
      {i0    -> b, i1 -> b, sigma -> 3.7, epsilon -> 0.5},
      {i0    -> c, i1 -> c, sigma -> 4.1, epsilon -> 0.22},
      {i0    -> d, i1 -> d, sigma -> 2.4, epsilon -> 0.09}
    }
  }
}
};

```

7.12.17.3 References

1. W. L. Jorgensen and J. Tirado-Rives, "*The OPLS Potential Functions for Proteins. Energy Minimizations for Crystals of Cyclic Peptides and Crambin*", *J. Am. Chem. Soc.* **1988**, *110*, 1657-1666.

7.12.17.4 Pair

```

pair          = {
  active      -> boolean,
  nbonded     -> integer,
  mix         -> option,
  shift       -> boolean,
  coulomb     -> option,
  cutoff      -> real,
  mode        -> option,
  n           -> integer,
  data        ->
  {
    {
      i0      -> id,
      i1      -> id,
      epsilon -> real,
      sigma   -> real,
      cutoff  -> real
    },
    ...
  }
};

```

Directive	Parameters	Description
active	boolean	Interaction activator; either true or false .
nbonded	integer	Number of bonded sites to exclude from nonbonded interactions; the given number is expected to be larger or equal to zero.
mix	option	Sets mixing rule; options are none , berthelot , arithmetic , geometric , and sixth .
shift	boolean	Sets shifting of potential at cutoff; options are true and false .
coulomb	option	Set type of coulomb treatment; options are none , cut , and long .
cutoff	real	Sets global cut off for global mode
mode	option	Sets cut off mode; options are global using the value of cutoff for all contributions, individual defining cut offs per contribution, and repulsive defining the well location for each separate contribution as the cut off distance.
n	integer	Number of data entries; can be omitted when type = {merge -> true} .
data	struct	Summary of interaction constants
i0, i1	id	Site id, referring to types as defined by the mass paragraph.

epsilon	real	Force interaction constant; identical to the well depth for a Lennard-Jones 6-12 potential
sigma	real	Site size; the point where the potential equals zero

The energetic functional form of the pair contributions to the total potential is described by a 6-12 Lennard-Jones potential,

$$E_{pair} = 4\epsilon \left(\frac{\sigma}{r_{ij}} \right)^6 \left[\left(\frac{\sigma}{r_{ij}} \right)^6 - 1 \right],$$

where r_{ij} represents the distance between site i and site j .

7.12.17.5 Bond

```

bond          = {
  active      -> boolean
  n           -> integer,
  data        ->
  {
    {
      i0       -> id,
      i1       -> id,
      k        -> real,
      l        -> real
    },
    ...
  }
};

```

Directive	Parameters	Description
active	boolean	Interaction activator; either <code>true</code> or <code>false</code> .
n	integer	Number of data entries; can be omitted when <code>type = {merge -> true}</code> .
data	struct	Summary of interaction constants
i0, i1	id	Site id, referring to types as defined by the <code>mass</code> paragraph in <code>types</code>
k	real	Bond spring constants
l	real	Equilibrium bond length

The energetic functional form of the bond contributions to the total potential is described by

$$E_{bond} = 1/2 k_{bond}(l - l_0)^2,$$

where k_{bond} represents a spring constant, l the bond length of bond $\{i_0, i_1\}$, and l_0 the equilibrium bond length.

7.12.17.6 Angle

```

angle          = {
  active       -> boolean
  n            -> integer,
  data         ->
  {
    {
      i0        -> id,
      i1        -> id,
      i2        -> id,
      k         -> real,
      theta     -> real
    },
    ...
  }
};

```

Directive	Parameters	Description
active	boolean	Interaction activator; either <code>true</code> or <code>false</code> .
n	integer	Number of data entries; can be omitted when <code>type = {merge -> true}</code> .
data	struct	Summary of interaction constants
i0, i1, i2	id	Site id, referring to types as defined by the <code>mass</code> paragraph in <code>types</code>
k	real	Angle spring constants
theta	real	Equilibrium bond angle

The energetic functional form of the angle contributions to the total potential is described by

$$E_{angle} = 1/2 k_{angle}(\theta - \theta_0)^2,$$

where k_{angle} represents a spring constant, θ the angle between bonds $\{i_0, i_1\}$ and $\{i_1, i_2\}$, and θ_0 the equilibrium angle.

7.12.17.7 Torsion

```

torsion      = {
  active      -> boolean
  n           -> integer,
  data        ->
  {
    {
      i0       -> id,
      i1       -> id,
      i2       -> id,
      i3       -> id,
      coefficients ->
      {
        {
          k      -> real,
          n      -> integer,
          delta  -> real
        }
        ...
      }
    },
    ...
  }
};

```

Directive	Parameters	Description
active	boolean	Interaction activator; either <code>true</code> or <code>false</code> .
n	integer	Number of data entries; can be omitted when <code>type = {merge -> true}</code> .
data	struct	Summary of interaction constants
i0, i1, i2, i3	id	Site id, referring to types as defined by the <code>mass</code> paragraph in <code>types</code>
k	real	Force constant
n	integer	Torsional angle prefactor; values range between 1 and 6
delta	real	Offset angle, mainly used to change the sign of the cosine function

The energetic functional form of the torsion contributions to the total potential is described by

$$E_{torsion} = \sum_{i=1}^m k_i (1 + \cos(n_i \phi - \delta_i)),$$

where k_i represents a set of torsion constants, δ_i a torsion offset, n_i the torsion pre-factor, and ϕ the bond torsion constructed by bonds $\{i_0, i_1\}$, $\{i_1, i_2\}$, and $\{i_2, i_3\}$. The maximum

allowed value of n_i is 6. Note, that the functional form presented here differs from the original OPLS definition.

7.12.17.8 Improper

```

improper      = {
  active      -> boolean
  n           -> integer,
  data        ->
  {
    {
      i0       -> id,
      i1       -> id,
      i2       -> id,
      i3       -> id,
      k        -> real,
      psi      -> real
    },
    ...
  }
};

```

Directive	Parameters	Description
active	boolean	Interaction activator; either <code>true</code> or <code>false</code> .
n	integer	Number of data entries; can be omitted when <code>type = {merge -> true}</code> .
data	struct	Summary of interaction constants
i0, i1, i2,	id	Site id, referring to types as defined by the <code>mass</code> paragraph in <code>types</code>
i3		
k	real	Force constant
psi	real	Improper equilibrium angle

The energetic functional form of the torsion contributions to the total potential is described by

$$E_{improper} = k_{improper}(\psi - \psi_0)^2,$$

where ψ represents the improper angle, which is given by the average of torsion angles formed by planes out of permutations of any of the following two bonds: $\{i_1, i_0\}$, $\{i_2, i_0\}$, and $\{i_3, i_0\}$. Note, that site i_0 is the central site, as opposed to some representations, where i_1 is the central site.

7.12.18 Table

7.12.18.1 Syntax

```
table          = {
    bond        -> struct,
    angle       -> struct,
    torsion     -> struct,
    pair        -> struct
};
```

Directive	Parameters	Description
bond	struct	Bond interaction descriptors
angle	struct	Angle interaction descriptors
torsion	struct	Torsion interaction descriptors
pair	struct	Pair interaction descriptors

The tabular force field uses tabulated functions for nonbonded and bonded interactions. Its functional form is a summation of bonded and nonbonded interactions given by

$$E_{table} = E_{pair} + E_{bond} + E_{angle} + E_{torsion}$$

for bond, angle, torsion, and pair contributions respectively. The default interpolation order is set to `cubic` (see Section 7.9 [Splines], page 185). The following paragraphs describe each contribution in detail. See `./examples/field/table/` for an application.

7.12.18.2 Table Syntax

```
data          = {
    {
        x          -> real,
        energy     -> real
    },
    ...
};
```

Directive	Parameters	Description
x	real	Location; either interparticle distance for pair, bond distance for bond, bond angle of angle, or bond torsion for torsion interactions.
energy	real	Energy at the specified location.

Table entries are given in pairs, where the x coordinate represents the function variable and the y coordinate the resulting energy. The x interval between each entry must be equidistant. Non-equidistant entries result in poor interpolation.

7.12.18.3 Example

```

simulation      = {
  types         -> {
    merge       -> true,
    boltzmann   -> {
      bond      -> {
        active   -> true,
        order    -> cubic,
        data     -> {
          {i0    -> a, i1 -> a, name -> "bond_aa.dat"},
          {i0    -> a, i1 -> b, name -> "bond_ab.dat"},
          {i0    -> b, i1 -> b, name -> "bond_bb.dat"}
        }
      },
      angle     -> {
        active   -> true,
        order    -> cubic,
        data     -> {
          {i0    -> a, i1 -> a, i2 -> a, name -> "angle_aaa.dat"},
          {i0    -> a, i1 -> a, i2 -> b, name -> "angle_aab.dat"},
          {i0    -> a, i1 -> b, i2 -> b, name -> "angle_abb.dat"},
          {i0    -> b, i1 -> b, i2 -> b, name -> "angle_bbb.dat"}
        }
      },
      pair      -> {
        active   -> true,
        order    -> cubic,
        nbonded  -> 2,
        data     -> {
          {i0    -> a, i1 -> a, name -> "pair_aa.dat"},
          {i0    -> a, i1 -> b, name -> "pair_ab.dat"},
          {i0    -> b, i1 -> b, name -> "pair_bb.dat"}
        }
      }
    }
  }
};

```

7.12.18.4 Pair

```

bond          = {
  active       -> boolean,
  order        -> option,
  nbonded      -> integer,
  n            -> integer,
  data         ->
  {
    {
      i0        -> id,
      i1        -> id,
      name      -> string
    },
    ...
  }
};

```

Directive	Parameters	Description
active	boolean	Interaction activator; either <code>true</code> or <code>false</code> .
order	option	Spline interpolation order; either <code>linear</code> or <code>cubic</code> .
nbonded	integer	Number of bonded sites to exclude from nonbonded interactions; the given number is expected to be larger or equal to zero.
n	integer	Number of data entries; can be omitted when <code>type = {merge -> true}</code> .
data	struct	Summary of interaction constants
i0, i1	id	Site id, referring to <code>mass</code> paragraph in <code>types</code>
name	string	Table file name; for table format see [Table Syntax] below.

The energetic functional form of the pair contributions to the total potential is formed by a linear interpolation or a cubic spline through the provided data.

7.12.18.5 Bond

```

bond          = {
  active      -> boolean,
  order       -> option,
  n           -> integer,
  data        ->
  {
    {
      i0       -> id,
      i1       -> id,
      name     -> string
    },
    ...
  }
};

```

Directive	Parameters	Description
active	boolean	Interaction activator; either <code>true</code> or <code>false</code> .
order	option	Spline interpolation order; either <code>linear</code> or <code>cubic</code> .
n	integer	Number of data entries; can be omitted when <code>type = {merge -> true}</code> .
data	struct	Summary of interaction constants
i0, i1	id	Site id, referring to <code>mass</code> paragraph in <code>types</code>
name	string	Table file name; for table format see [Table Syntax] below.

The energetic functional form of the bond contributions to the total potential is described by

$$E_{bond} = Interpolation(l),$$

where l represents the bond distance. Either a linear or a cubic spline through the provided data defines the interpolation function.

7.12.18.6 Angle

```

angle          = {
  active       -> boolean,
  order        -> option,
  n            -> integer,
  data         ->
  {
    {
      i0        -> id,
      i1        -> id,
      i2        -> id,
      name      -> string
    },
    ...
  }
};

```

Directive	Parameters	Description
active	boolean	Interaction activator; either <code>true</code> or <code>false</code> .
order	option	Spline interpolation order; either <code>linear</code> or <code>cubic</code> .
n	integer	Number of data entries; can be omitted when <code>type = {merge -> true}</code> .
data	struct	Summary of interaction constants
i0, i1, i2	id	Site id, referring to <code>mass</code> paragraph in <code>types</code>
name	string	Table file name; for table format see [Table Syntax] below.

The energetic functional form of the angle contributions to the total potential is described by

$$E_{angle} = Interpolation(\theta),$$

where θ represents the bond angle. Either a linear or a cubic spline through the provided data defines the interpolation function.

7.12.18.7 Torsion

```

torsion      = {
  active      -> boolean,
  order       -> option,
  n           -> integer,
  data        ->
  {
    {
      i0       -> id,
      i1       -> id,
      i2       -> id,
      i3       -> id,
      name     -> string
    },
    ...
  }
};

```

Directive	Parameters	Description
active	boolean	Interaction activator; either <code>true</code> or <code>false</code> .
order	option	Spline interpolation order; either <code>linear</code> or <code>cubic</code> .
n	integer	Number of data entries; can be omitted when <code>type = {merge -> true}</code> .
data	struct	Summary of interaction constants
i0, i1, i2, i3	id	Site id, referring to <code>mass</code> paragraph in <code>types</code>
name	string	Table file name; for table format see [Table Syntax] below.

The energetic functional form of the torsion contributions to the total potential is described by

$$E_{torsion} = Interpolation(\phi),$$

where ϕ represents the bond torsion. Either a linear or a cubic spline through the provided data defines the interpolation function.

7.12.19 TraPPE

7.12.19.1 Syntax

```
trappe      = {
  bond       -> struct,
  angle      -> struct,
  torsion    -> struct,
  improper    -> struct,
  pair14     -> struct,
  pair       -> struct
};
```

Directive	Parameters	Description
bond	struct	Bond interaction descriptors
angle	struct	Angle interaction descriptors
torsion	struct	Torsion interaction descriptors
improper	struct	Improper interaction descriptors
pair14	struct	1-4 intra-molecular pair interaction activator
pair	struct	Pair interaction descriptors

The TraPPE force field is a compounded force field based on the TraPPE force field.¹ Its functional form is a summation of bonded and nonbonded interactions given by

$$E_{trappe} = E_{bond} + E_{angle} + E_{torsion} + E_{improper} + E_{pair}$$

for bond, angle, torsion, and pair contributions respectively. The following paragraphs describe each contribution in detail. See `./examples/field/trappe/` for applications.

7.12.19.2 Example

```
simulation    = {
  types       -> {
    merge     -> true,
    trappe    -> {
      bond     -> {
        active -> true,
        data   -> {
          {i0   -> a, i1 -> b, k -> 100, l -> 2}
        }
      },
      angle    -> {
        active -> true,
        data   -> {
          {i0   -> a, i1 -> b, i2 -> c, k -> 100, theta -> 110}
        }
      },
      torsion  -> {
```

```

    active  -> true,
    data    -> {
      {i0    -> a, i1 -> b, i2 -> c, i3 -> d, coefficients -> {
        {k -> 1.5, n -> 1, delta -> 0},
        {k -> 2.5, n -> 2, delta -> 0}
      }
    }
  },
  improper  -> {
    active  -> true,
    data    -> {
      {i0    -> a, i1 -> b, i2 -> c, i3 -> d, k -> 100, psi -> 0}
    }
  },
  pair14    -> {
    active  -> true
  },
  pair      -> {
    active  -> true,
    nbonded -> 3,
    mode    -> global,
    cutoff  -> 10,
    mix     -> berthelot,
    data    -> {
      {i0    -> a, i1 -> a, sigma -> 4.0, epsilon -> 0.4},
      {i0    -> b, i1 -> b, sigma -> 3.7, epsilon -> 0.5},
      {i0    -> c, i1 -> c, sigma -> 4.1, epsilon -> 0.22},
      {i0    -> d, i1 -> d, sigma -> 2.4, epsilon -> 0.09},
    }
  }
}
};

```

7.12.19.3 References

1. Siepmann papers
2. W. L. Jorgensen and J. Tirado-Rives, "*The TraPPE Potential Functions for Proteins. Energy Minimizations for Crystals of Cyclic Peptides and Crambin*", *J. Am. Chem. Soc.* **1988**, *110*, 1657-1666.

7.12.19.4 Pair

```
pair          = {
  active      -> boolean,
  nbonded     -> integer,
  mix         -> option,
  shift       -> boolean,
  coulomb     -> option,
  cutoff      -> real,
  mode        -> option,
  n           -> integer,
  data        ->
  {
    {
      i0      -> id,
      i1      -> id,
      epsilon -> real,
      sigma   -> real,
      cutoff  -> real
    },
    ...
  }
};
```

Directive	Parameters	Description
active	boolean	Interaction activator; either true or false .
nbonded	integer	Number of bonded sites to exclude from nonbonded interactions; the given number is expected to be larger or equal to zero.
mix	option	Sets mixing rule; options are none , berthelot , arithmetic , geometric , and sixth .
shift	boolean	Sets shifting of potential at cutoff; options are true and false .
coulomb	option	Set type of coulomb treatment; options are none , cut , and long .
cutoff	real	Sets global cut off for global mode
mode	option	Sets cut off mode; options are global using the value of cutoff for all contributions, individual defining cut offs per contribution, and repulsive defining the well location for each separate contribution as the cut off distance.
n	integer	Number of data entries; can be omitted when type = {merge -> true} .
data	struct	Summary of interaction constants
i0, i1	id	Site id, referring to types as defined by the mass paragraph.

epsilon	real	Force interaction constant; identical to the well depth for a Lennard-Jones 6-12 potential.
sigma	real	Site size; the point where the potential equals zero.
cutoff	real	Cut off; distance at which the pair-wise contribution is set to zero; can be omitted.

The energetic functional form of the pair contributions to the total potential is described by a 6-12 Lennard-Jones potential,

$$E_{pair} = 4\epsilon \left(\frac{\sigma}{r_{ij}} \right)^6 \left[\left(\frac{\sigma}{r_{ij}} \right)^6 - 1 \right],$$

where r_{ij} represents the distance between site i and site j .

7.12.19.5 Bond

```

bond          = {
  active      -> boolean
  n           -> integer,
  data        ->
  {
    {
      i0       -> id,
      i1       -> id,
      k        -> real,
      l        -> real
    },
    ...
  }
};

```

Directive	Parameters	Description
active	boolean	Interaction activator; either <code>true</code> or <code>false</code> .
n	integer	Number of data entries; can be omitted when <code>type = {merge -> true}</code> .
data	struct	Summary of interaction constants
i0, i1	id	Site id, referring to types as defined by the <code>mass</code> paragraph in <code>types</code>
k	real	Bond spring constants
l	real	Equilibrium bond length

The energetic functional form of the bond contributions to the total potential is described by

$$E_{bond} = 1/2 k_{bond} (l - l_0)^2,$$

where k_{bond} represents a spring constant, l the bond length of bond $\{i_0, i_1\}$, and l_0 the equilibrium bond length.

7.12.19.6 Angle

```

angle          = {
  active       -> boolean
  n            -> integer,
  data         ->
  {
    {
      i0        -> id,
      i1        -> id,
      i2        -> id,
      k         -> real,
      theta     -> real
    },
    ...
  }
};

```

Directive	Parameters	Description
active	boolean	Interaction activator; either <code>true</code> or <code>false</code> .
n	integer	Number of data entries; can be omitted when <code>type = {merge -> true}</code> .
data	struct	Summary of interaction constants
i0, i1, i2	id	Site id, referring to types as defined by the <code>mass</code> paragraph in <code>types</code>
k	real	Angle spring constants
theta	real	Equilibrium bond angle

The energetic functional form of the angle contributions to the total potential is described by

$$E_{angle} = 1/2 k_{angle} (\theta - \theta_0)^2,$$

where k_{angle} represents a spring constant, θ the angle between bonds $\{i_0, i_1\}$ and $\{i_1, i_2\}$, and θ_0 the equilibrium angle.

7.12.19.7 Torsion

```

torsion      = {
  active      -> boolean
  n           -> integer,
  data        ->
  {
    {
      i0       -> id,
      i1       -> id,
      i2       -> id,
      i3       -> id,
      coefficients ->
      {
        {
          k      -> real,
          n      -> integer,
          delta  -> real
        }
        ...
      }
    },
    ...
  }
};

```

Directive	Parameters	Description
active	boolean	Interaction activator; either <code>true</code> or <code>false</code> .
n	integer	Number of data entries; can be omitted when <code>type = {merge -> true}</code> .
data	struct	Summary of interaction constants
i0, i1, i2, i3	id	Site id, referring to types as defined by the <code>mass</code> paragraph in <code>types</code>
k	real	Force constant
n	integer	Torsional angle prefactor; values range between 1 and 4
delta	real	Offset angle, mainly used to change the sign of the cosine function

The energetic functional form of the torsion contributions to the total potential is described by

$$E_{torsion} = \sum_{i=1}^m k_i (1 + \text{sign}(n_i) \cos(n_i \phi - \delta_i)),$$

where ϕ represents the bond torsion constructed by bonds $\{i_0, i_1\}$, $\{i_1, i_2\}$, and $\{i_2, i_3\}$. Function $\text{sign}(n_i)$ is +1 for odd and -1 for even values of n_i .

7.12.19.8 Improper

```

improper      = {
  active      -> boolean
  n           -> integer,
  data        ->
  {
    {
      i0       -> id,
      i1       -> id,
      i2       -> id,
      i3       -> id,
      k        -> real,
      psi      -> real
    },
    ...
  }
};

```

Directive	Parameters	Description
active	boolean	Interaction activator; either <code>true</code> or <code>false</code> .
n	integer	Number of data entries; can be omitted when <code>type = {merge -> true}</code> .
data	struct	Summary of interaction constants
i0, i1, i2, i3	id	Site id, referring to types as defined by the <code>mass</code> paragraph in <code>types</code>
k	real	Force constant
psi	real	Improper equilibrium angle

The energetic functional form of the torsion contributions to the total potential is described by

$$E_{improper} = k_{improper}(\psi - \psi_0)^2,$$

where ψ represents the improper angle, which is given by the average of torsion angles formed by planes out of permutations of any of the following two bonds: $\{i_1, i_0\}$, $\{i_2, i_0\}$, and $\{i_3, i_0\}$. Note, that site i_0 is the central site, as opposed to some representations, where i_1 is the central site.

7.13 Units

7.13.1 Syntax

```
units          = {
  type         -> constant,
  mass         -> real,
  length       -> real,
  angstrom     -> real,
  angle        -> real,
  energy       -> real,
  kb           -> real,
  nav          -> real,
  charge       -> real,
  permittivity -> real,
  seed         -> real,
  reduced      -> boolean
};
```

Directive	Parameters	Description
type	constant	Defines the type of units used (needed upon first initialization); options are reduced or si .
mass	real	Defines the global mass scaling constant.
length	real	Defines the global length scaling constant.
angstrom	real	Defines the length scaling for ported formats (e.g. PDB or LAMMPS).
angle	real	Defines the global angle scaling constant.
energy	real	Defines the global energetic scaling constant.
kb	real	Defines Boltzmann's constant with respect to the chosen scaling.
nav	real	Defines Avogadro's constant with respect to the chosen scaling.
charge	real	Defines the global scaling of a unit charge.
permittivity	real	Defines the global permittivity scaling.
seed	real	Defines the random seed.
reduced	boolean	Reflects the output state (normally false); options are true or false .

7.13.2 Usage

This variable style describes units. Units are used to internally scale all variables with respect to their units. The units can also be used to define an appropriate relation between force field energies, angles, bead sizes, etc. Upon initialization, a suitable scaling will be deduced from force field or internal natural constant definitions when the contributor is zero (see Section 7.12 [Types], page 190).

7.13.3 Default

Unless otherwise stated, the default is given by

```
units          = {  
  type         -> si  
};
```

By default, all other contributors are set to zero and deduced upon initialization.

7.14 Vector

7.14.1 Syntax

```
vector          = {x -> real, y -> real, z -> real};
```

Directive	Parameters	Description
x, y, z	real	Sets vector components.

7.14.2 Usage

This variable style describes a vector. Vectors always consist of three components, which can be given in real numbers. Interpretation can either be **real** or **integer**, dependent on the command in which **vector** occurs.

7.14.3 Default

Unless otherwise stated, the default is given by

```
vector          = {x -> 0, y -> 0, z -> 0};
```

7.15 Voigt

7.15.1 Syntax

```
voigt          = {xx -> real, yy -> real, zz -> real,
                  zy -> real, zx -> real, yx -> real};
```

Directive	Parameters	Description
xx, yy, zz, zy, zx, yx	real	Sets voigt components.

7.15.2 Usage

This variable style describes a voigt. Voigts always consist of six components, which can be given in real numbers. Interpretation can either be **real** or **integer**, dependent on the command in which **voigt** occurs.

7.15.3 Default

Unless otherwise stated, the default is given by

```
voigt          = {xx -> 0, yy -> 0, zz -> 0, zy -> 0, zx -> 0, yx -> 0};
```

Index

B

boltzmann.....	193
bond.....	152
born.....	198
build.....	82

C

cancel.....	86
carve.....	87
cavity.....	155
CFF.....	206
CHARMM.....	222
Chemistry File.....	35
chemistry options.....	15
chemistry.esh.....	35
clusters.....	89
Clusters.....	6
coarse.....	233
colloid.....	249
Configurational Moves.....	7
constants.....	174
coulomb.....	257
crystal.....	93
cut.....	95

D

deform.....	96
displace.....	179
Distribution content.....	1
Distributions.....	7
DPD.....	261
duplicate.....	97

E

Environment File.....	32
environment options.....	12
environment.esh.....	32
export.....	98
Extensions.....	10

F

field.....	101
Field File.....	45
field.define.....	45
flag.....	106
focus.....	176
focus command.....	108
force.....	109
Force Fields.....	3
Forces.....	7
former.....	110

G

General Introduction.....	1
get.....	111
gr.....	160
GROMACS.....	270
groups.....	112
Groups.....	6
gyration.....	165

I

insight.....	116
interaction.....	167
Introduction.....	1

L

lammps.....	118
Lists.....	3

M

MARTINI.....	279
Measurements.....	7
memory.....	121
Methodology.....	3
Molecular Interactions.....	7
Molecular Representation.....	6
moves.....	122, 177

O

opls.....	287
-----------	-----

P

Parameter File..... 50
parameters.csv..... 50
pdb..... 123
Port..... 180
profiles..... 181
Program Structure..... 6
put..... 126

R

References File..... 49
references.csv..... 49
region..... 182
remove..... 128
rename..... 127
reset..... 130
restart..... 131
retype..... 132
run..... 134

S

sample..... 135
Sampling Tools..... 151
Scripting Commands..... 81
sdk..... 296
Setup Usage..... 9
setup.esh..... 32
shell..... 136
simulation..... 138
Simulation Setup..... 8
sites..... 137
Sites..... 6
SMILES..... 183

spline..... 305
splines..... 185
split..... 140
standard..... 309
System Flags..... 175, 187
systems..... 188
Systems..... 7

T

table..... 318
table format..... 318
terminate..... 141
timing..... 142
traject..... 143
translate..... 145
trappe..... 324
Types..... 7
types..... 146, 190

U

units..... 332

V

Variable Descriptions..... 172
variables..... 148
vector..... 334
voigt..... 335

X

xyz..... 149