THE DL_POLY_4 USER MANUAL

I.T. Todorov & W. Smith

STFC Daresbury Laboratory
Daresbury, Warrington WA4 4AD
Cheshire, England, United Kingdom

Version 5.0.0 – February 2021

ABOUT DL_POLY_4

DL_POLY_4 is a general purpose parallel molecular dynamics simulation package developed at Daresbury Laboratory by I.T. Todorov & W. Smith. The DL_POLY project was developed under the auspices of the Engineering and Physical Sciences Research Council (EPSRC) for the EPSRC's Collaborative Computational Project for the Computer Simulation of Condensed Phases (CCP5), the Computational Chemistry & Application Performance Engineering Groups (formerly Molecular Simulation & Advanced Research Computing Groups) at Daresbury Laboratory and the Natural Environment Research Council (NERC) for the NERC's *e*Science project *Computational Chemistry in the Environment* (*e*Minerals), directed by M.T. Dove.

DL_POLY_4 is developed worlwide and distributed under GNU Lesser General Public License version 3.0.

Latest version of DL_POLY_4, issues and comments can be made at https://gitlab.com/ccp5/dl-poly

DISCLAIMER

ACKNOWLEDGEMENTS

DL_POLY_4 was developed at Daresbury Laboratory (DL - https://stfc.ukri.org/about-us/where-we-work/daresbury-laboratory/), the Science & Technology Facilities Council (STFC - http://www.stfc.ac.uk/), UK, with support from the Engineering and Physical Sciences Research Council (EPSRC - http://www.epsrc.ac.uk/) and the Natural Environment Research Council (NERC - http://www.nerc.ac.uk/).

Advice, assistance and encouragement in the development of DL_POLY_4 has been given by many people. We gratefully acknowledge the following:

T.R. Forester, I.J. Bush, M. Leslie, M.F. Guest, R.J. Allan, D. Tildesley, M. Pinches, D. Rapaport, the UK's "Materials Chemistry Consortium" under C.R.A. Catlow and the *e*Minerals project under M.T. Dove.

Manual contributions: Laurence Elisson, Andrey Brukhno, Alin M Elena, Jim Madge, Aidan Chalk, Ivan Scivetti, Aaron Diver, Oliver Dicks, Alex Buccheri, Yaser Afshar and Jacob Wilkins. `This document is produced with LaTeX & hdvipdfm`

Manual Notation

In the DL_POLY manuals specific fonts are used to convey specific meanings:

1. *directories* - indicates UNIX file directories
2. ROUTINES - indicates subroutines, functions and programs
3. *macros* - indicates a macro (file of UNIX commands)
4. **directive** - indicates directives or keywords
5. `variables` - indicates named variables and parameters
6. FILE - indicates filenames.

Contents

List of Tables

List of Figures

Chapter 0

Quick Word / INSTALL & RUN

For the experienced and quick minded this is a very brief resume of how to INSTALL & RUN DL_POLY_4 (which is no excuse for skipping the Introduction, Chapter 1!). For the rest of us it sketches out how to start running DL_POLY_4 jobs and where one should look to obtain more detailed information if need be.

If you have followed the procedure for obtaining and downloading the DL_POLY_4 package (see Obtaining the Source Code, Section 11), have successfully unpacked it and are ready to compile the source code, then jump to the INSTALL Notes in the INSTALL file, both in the main distribution directory as well as in the Appendix E.

If you have compiled successfully then a freshly date-stamped file, named DLPOLY.Z should appear in the listing of the *execute* subdirectory (the '`ls -haltr`' command issued on a Linux/Unix-like shell within *execute* will place the executable in the last row of the list). If **unsuccessful** then you should read the Compiling and Running DL_POLY_4 (Section 8.2).

To run the code you first need to place the necessary input files within *execute*. TEST cases containing suitable input files, as well as examples of output files, can be obtained at the DL_POLY_4 FTP site, ftp://ftp.dl.ac.uk/ccp5/DL_POLY/. Examine the contents of *data*/`README.txt` and *bench*/`README.txt` for more information. To run the serial version you simply issue the command `DLPOLY.Z` within the *execute* subdirectory. If you have compiled a parallel version and are running it on a parallel machine, then naturally you will need to familiarise yourself with the local procedures of how to run jobs on that machine. In general though, running a parallel job will usually require that you issue a necessary run command (e.g. `mpirun -n 8 DLPOLY.Z`) or submit a job script from within *execute*.

If you need to know more then search the manual and examine sections of interests. You may also wish to visit DL_POLY project web-page http://www.ccp5.ac.uk/DL_POLY/ and examine the useful links within (FAQ, User Forum, etc.).

If you are looking to gain more in depth experience, then regular training workshops are available. To find about upcoming workshops, subscribe to our mail list by following instructions in Section 1.7.

If you need one-to-one training, wish to collaborate scientifically and/or would like to become a contributor/developer then get in touch with me, Dr. I.T. Todorov, by emailing to ilian.todorov@stfc.ac.uk.

Best of luck!

Chapter 1

Introduction

Scope of Chapter

This chapter describes the concept, design and directory structure of DL_POLY_4 and how to obtain a copy of the source code.

1.1 The DL_POLY Package

DL_POLY [1] is a package of subroutines, programs and data files, designed to facilitate molecular dynamics simulations of macromolecules, polymers, ionic systems and solutions on a distributed memory parallel computer. It is available in two forms: DL_POLY_Classic (written by Bill Smith & Tim Forester, http://www.ccp5.ac.uk/DL_POLY_CLASSIC/) and DL_POLY_4 (written by Ilian Todorov & Bill Smith) [2, 3]. Both versions were originally written on behalf of CCP5, the UK's Collaborative Computational Project on Molecular Simulation, which has been in existence since 1980 ([4], http://www.ccp5.ac.uk/DL_POLY/).

The two forms of DL_POLY differ primarily in their method of exploiting parallelism. DL_POLY_Classic uses a Replicated Data (RD) strategy [5, 6, 7, 8] which works well simulations of up to 30,000 atoms on up to 100 processors. DL_POLY_4 is based on the Domain Decomposition (DD) strategy [2, 3, 9, 10, 5, 6], and is best suited for large molecular simulations from 10^3 to 10^9 atoms on large processor counts. The two packages are reasonably compatible, so that it is possible to scale up from a DL_POLY_Classic to a DL_POLY_4 simulation with little effort. It should be apparent from these comments that DL_POLY_4 is not intended as a replacement for DL_POLY_Classic.

Users are reminded that we are interested in hearing what other features could be usefully incorporated. We obviously have ideas of our own and CCP5 strongly influences developments, but other input would be welcome nevertheless. We also request that our users respect the integrity of DL_POLY_4 source and not pass it on to third parties. We require that all users of the package register with us, not least because we need to keep everyone abreast of new developments and discovered bugs. We have developed various forms of licence, which we hope will ward off litigation (from both sides), without denying access to genuine scientific users.

Further information on the DL_POLY packages may be obtained from the DL_POLY project website - http://www.ccp5.ac.uk/DL_POLY/ .

1.2 Functionality

The following is a list of the features DL_POLY_4 supports.

1.2.1 Molecular Systems

DL_POLY_4 will simulate the following molecular species:

- Simple atomic systems and mixtures, e.g. Ne, Ar, Kr, etc.
- Simple unpolarisable point ions, e.g. NaCl, KCl, etc.
- Polarisable point ions and molecules, e.g. MgO, H_2O, etc.
- Simple rigid molecules e.g. CCl_4, SF_6, Benzene, etc.
- Rigid molecular ions with point charges e.g. KNO_3, $(NH_4)_2SO_4$, etc.
- Polymers with rigid bonds, e.g. C_nH_{2n+2}
- Polymers with flexible and rigid bonds and point charges, e.g. proteins, macromolecules etc.
- Silicate glasses and zeolites
- Simple metals and metal alloys, e.g. Al, Ni, Cu, Cu_3Au, etc.
- Covalent systems as hydro-carbons and transition elements, e.g. C, Si, Ge, SiC, SiGe, ets.

1.2.2 Force Field

The DL_POLY_4 force field includes the following features:

1. All common forms of non-bonded atom-atom (van der Waals) potentials
2. Atom-atom (and site-site) coulombic potentials
3. Metal-metal (local density dependent) potentials [11, 12, 13, 14, 15, 16]
4. Tersoff (local density dependent) potentials (for hydro-carbons) [17]
5. Three-body valence angle and hydrogen bond potentials
6. Four-body inversion potentials
7. Ion core-shell polarasation
8. Tether potentials
9. Chemical bond potentials
10. Valence angle potentials
11. Dihedral angle (and improper dihedral angle) potentials
12. Inversion angle potentials
13. External field potentials.

The parameters describing these potentials may be obtained, for example, from the GROMOS [18], Dreiding [19] or AMBER [20] forcefield, which share functional forms. It is relatively easy to adapt DL_POLY_4 to user specific force fields.

1.2.3 Boundary Conditions

DL_POLY_4 will accommodate the following boundary conditions:

1. None, e.g. isolated molecules *in vacuo*
2. Cubic periodic boundaries
3. Orthorhombic periodic boundaries
4. Parallelepiped periodic boundaries
5. Slab (x,y periodic, z non-periodic).

These are described in detail in Appendix B. Note that periodic boundary conditions (PBC) 1 and 5 above require careful consideration to enable efficient load balancing on a parallel computer.

1.2.4 Java Graphical User Interface

The DL_POLY_4 Graphical User Interface (GUI) is the same one that also comes with DL_POLY_Classic, which is written in the Java®programming language from Sun®Microsystems. A major advantage of this is the free availability of the Java programming environment from Sun®, and also its portability across platforms. The compiled GUI may be run without recompiling on any Java®supported machine. The GUI is an integral component of the DL_POLY suites and is available on the same terms (see the GUI manual [21]).

1.2.5 Algorithms

1.2.5.1 Parallel Algorithms

DL_POLY_4 exclusively employs the Domain Decomposition parallelisation strategy [9, 10, 5, 6] (see Section 11.1.1).

1.2.5.2 Molecular Dynamics Algorithms

DL_POLY_4 offers a selection of MD integration algorithms based on Velocity Verlet (VV) [22]. These generate NVE, NVE$_{kin}$, NVT, NPT and N$\underline{\underline{\sigma}}$T ensembles with a selection of thermostats and barostats. Parallel versions of the RATTLE [23] and SHAKE [8] algorithms are used for solving bond constraints. The rotational motion of rigid bodies (RBs) is handled with the "NOSQUISH" algorithm of Miller *et al* [24].

The following MD algorithms are available:

1. Constant E algorithm
2. Evans constant E$_{kin}$ algorithm [25]
3. Langevin constant T algorithm [26]
4. Andersen constant T algorithm [27]
5. Berendsen constant T algorithm [28]
6. Nosé-Hoover constant T algorithm [29]
7. Langevin constant T,P algorithm [30]
8. Berendsen constant T,P algorithm [28]
9. Nosé-Hoover constant T,P algorithm [29]
10. Martyna, Tuckerman and Klein (MTK) constant T,P algorithm [31]
11. Langevin constant T,$\underline{\underline{\sigma}}$ algorithm [30]
12. Berendsen constant T,$\underline{\underline{\sigma}}$ algorithm [28]
13. Nosé-Hoover constant T,$\underline{\underline{\sigma}}$ algorithm [29]
14. Martyna, Tuckerman and Klein (MTK) constant T,$\underline{\underline{\sigma}}$ algorithm [31].

1.2.6 DL_POLY_Classic features incompatible or unavalable in DL_POLY_4

- Force field
  - Rigid bodies connected with constraint links are **not available**
  - Shell models specification is **solely** determined by the presence of mass on the shells
  - Dihedral potentials with more than three *original* parameters (see OPLS) have two artificially added parameters, defining the 1-4 electrostatic and van der Waals scaling factors, which **must** be placed at fourth and fifth position respectively, extending the original parameter list split by them
- Boundary conditions
  - Truncated octahedral periodic boundaries (`imcon` = 4) are **not available**

  - Rhombic dodecahedral periodic boundaries (`imcon` = 5) are **not available**
  - Hexagonal prism periodic boundaries (`imcon` = 7) are **not available**

- Electrostatics
  - Standard Ewald Summation is **not available**, but is **substituted** by Smoothed Particle Mesh Ewald (SPME) summation
  - Hautman-Klein Ewald Summation for 3D non-periodic but 2D periodic systems is **not available**

- Non-standard functionality
  - Temperature Accelerated Dynamics
  - Hyperdynamics
  - Solvation Energies

1.3 Programming Style

The programming style of DL_POLY_4 is intended to be as uniform as possible. The following stylistic rules apply throughout. Potential contributors of code are requested to note the stylistic convention.

1.3.1 Programming Language

DL_POLY_4 is written in free format FORTRAN90. In DL_POLY_4 we have adopted the convention of *explicit type declaration* i.e. we have used

```
Implicit None
```

in all subroutines. Thus all variables must be given an explicit type: `Integer, Real( Kind = wp)`, etc.

1.3.2 Modularisation and Intent

DL_POLY_4 exploits the full potential of the modularisation concept in FORTRAN90. Variables having in common description of certain feature or method in DL_POLY_4 are grouped in modules. This simplifies subroutines' calling sequences and decreases error-proneness in programming as subroutines must define what they use and from which module. To decrease error-proneness further, arguments that are passed in calling sequences of functions or subroutines have defined intent, i.e. whether they are to be:

- passed in only (`Intent (In)`) - the argument is not allowed to be changed by the routine
- passed out only (`Intent (Out)`) - the "coming in" value of the argument is unimportant
- passed in both directions in and out (`Intent (InOut)`) - the "coming in" value of the argument is important and the argument is allowed to be changed.

1.3.3 Memory Management

DL_POLY_4 exploits the dynamic array allocation features of FORTRAN90 to assign the necessary array dimensions.

1.3.4 Target Platforms

DL_POLY_4 is intended for distributed memory parallel computers.

Compilation of DL_POLY_4 in parallel mode requires **only** a FORTRAN90 compiler and Message Passing Interface (MPI) to handle communications. Compilation of DL_POLY_4 in serial mode is also possible and requires **only** a FORTRAN90 compiler.

1.3.5 Internal Documentation

All subroutines are supplied with a header block of FORTRAN90 comment (!) records giving:

1. The name of the author and/or modifying author
2. The version number or date of production
3. A brief description of the function of the subroutine
4. A copyright statement.

Elsewhere FORTRAN90 comment cards (!) are used liberally.

1.3.6 FORTRAN90 Parameters and Arithmetic Precision

All global parameters defined by the FORTRAN90 parameter statements are specified in the module file: SETUP_MODULE, which is included at compilation time in all subroutines requiring the parameters. All parameters specified in SETUP_MODULE are described by one or more comment cards.

One super-global parameter is defined at compilation time in the KINDS_F90 module file specifying the working precision (`wp`) by kind for real and complex variables and parameters. The default is 64-bit (double) precision, i.e. `Real(wp)`. Users wishing to compile the code with quadruple precision must ensure that their architecture and FORTRAN90 compiler can allow that and then change the default in KINDS_F90. Changing the precision to anything else that is allowed by the FORTRAN90 compiler and the machine architecture must also be compliant with the MPI working precision `mpi_wp` as defined in COMMS_MODULE (in such cases users must correct for that in there).

1.3.7 Units

Internally all DL_POLY_4 subroutines and functions assume the use of the following defined *molecular units*:

- The unit of time (t_o) is 1×10^{-12} seconds (i.e. picoseconds)
- The unit of length (ℓ_o) is 1×10^{-10} metres (i.e. Ångstroms)
- The unit of mass (m_o) is $1.6605402 \times 10^{-27}$ kilograms (i.e. Daltons - atomic mass units)
- The unit of charge (q_o) is $1.60217733 \times 10^{-19}$ Coulombs (i.e. electrons - units of proton charge)
- The unit of energy ($E_o = m_o(\ell_o/t_o)^2$) is $1.6605402 \times 10^{-23}$ Joules (10 J mol^{-1})
- The unit of pressure ($\mathcal{P}_o = E_o\ell_o^{-3}$) is 1.6605402×10^{7} Pascals (163.882576 atmospheres)
- Planck's constant ($\hbar$) which is $6.350780668 \times E_o t_o$.

In addition, the following conversion factors are used:

- The coulombic conversion factor (γ_o) is:

$$\gamma_o = \frac{1}{E_o}\left[\frac{q_o^2}{4\pi\epsilon_o\ell_o}\right] = 138935.4835 \quad ,$$

 such that:

$$U_{\texttt{MKS}} = E_o\gamma_o U_{\texttt{Internal}} \quad ,$$

 where U represents the configuration energy.

- The Boltzmann factor (k_B) is 0.831451115 E_o K^{-1}, such that:

$$T = E_{kin}/k_B$$

 represents the conversion from kinetic energy (in internal units) to temperature (in Kelvin).

Note: In the DL_POLY_4 OUTPUT file, the print out of pressure is in units of katms (kilo-atmospheres) at all times. The unit of energy is either DL_POLY units specified above, or in other units specified by the user at run time (see Section 10.1.3). The default is the DL_POLY unit.

Externally, DL_POLY_4 accepts information in its own specific formatting as described in Section 10.1. Irrespective of formatting rules, all values provided to define input entities are read in DL_POLY units (except otherwise specified as in the case of energy units) or their composite mixture representing the corresponding entity physically, i.e. velocities' components are in Ångstroms/picosecond.

Exception: It should be noted that when DL_POLY_4 is used in a DPD mode (see Section 3.4.7 and Appendix A) then the meaning of the molecular units is somewhat lost and it is only the interrelationship between units that is important (which can be exploited by the modeller)! The fundamental units for a DPD simulation are related those of mass $[M]$, length $[L]$ and energy $[E]$ - all irrespectively of the actually chosen energy units by the **UNITS** directive in the FIELD file. Therefore, the DPD unit of time is equivalent to $[L]\sqrt{[M]/[E]}$ while temperature (in the form k_BT) is defined as two-thirds of the kinetic energy of the system's particles. Similarly, volume is in units of $[L]^3$ and pressure in $[E]/[L]^3$.

1.3.8 Error Messages

All errors detected by DL_POLY_4 during run time initiate a call to the subroutine ERROR, which prints an error message in the standard output file and terminates the program. All terminations of the program are global (i.e. every node of the parallel computer will be informed of the termination condition and stop executing).

In addition to terminal error messages, DL_POLY_4 will sometimes print warning messages. These indicate that the code has detected something that is unusual or inconsistent. The detection is non-fatal, but the user should make sure that the warning does represent a harmless condition.

1.4 Directory Structure

The entire DL_POLY_4 package is stored in a UNIX directory structure. The topmost directory is named *dl_poly_4.nn*, where *nn* is a generation number. Beneath this directory are several sub-directories named: *manual*, *source*, *build*, *cmake*, *utils*, *execute*, *data*, *bench*, *java*, and *utility*.

Briefly, the content of each sub-directory is as follows:

sub-directory	contents
manual	DL_POLY_4 main user manual and DL_POLY_4 Java GUI manual

source	primary subroutines for the DL_POLY_4 package
build	makefiles to assemble and compile DL_POLY_4 source
cmake	contains files needed for DL_POLY_4*cmake* build system
utils	contains a series of scripts needed for testing
execute	the DL_POLY_4 run-time directory
data	example input and output files for DL_POLY_4
bench	large test cases suitable for benchmarking
java	directory of Java and FORTRAN routines for the Java GUI
utility	directory of routines donated by DL_POLY_4 users.

A more detailed description of each sub-directory follows.

1.4.1 The *source* Sub-directory

In this sub-directory all the essential source code for DL_POLY_4, excluding the utility software is stored. In keeping with the 'package' concept of DL_POLY_4, it does not contain any complete programs; these are assembled at compile time using an appropriate makefile. The subroutines in this sub-directory are documented in Chapter 11.

1.4.2 The *build* Sub-directory

This sub-directory contains legacy makefiles for the creation (i.e. compilation and linking) of the DL_POLY_4 simulation program. The makefiles supplied select the appropriate subroutines from the *source* sub-directory and deposit the executable program in the *execute* directory. Building DL_POLY_4 by using these legacy makefiles is described in Section **??**.

1.4.3 The *cmake* Sub-directory

This sub-directory contains necessary scripts and information needed for the DL_POLY_4 CMake system. Building DL_POLY_4 with *cmake* is described in Section **??**.

1.4.4 The *utils* Sub-directory

This sub-directory contains a framework of scripts needed by DL_POLY_4 developers for testing purposes. The general user is welcome to look and learn from it. The scripts are the documentation themselves.

1.4.5 The *execute* Sub-directory

In the supplied version of DL_POLY_4, this sub-directory contains only a few macros for copying and storing data from and to the *data* sub-directory and for submitting programs for execution (see Appendix C). However, if the DL_POLY_4 program is assembled by using a legacy makefile, the executable will be placed in this sub-directory and could be used from here. Then output files from a job run in here will also appear here, so users may find it convenient to use this sub-directory as originally intended. (The experienced user is not at all required to use DL_POLY_4 this way however.)

1.4.6 The *data* Sub-directory

This sub-directory contains examples of input and output files for testing the released version of DL_POLY_4. The examples of input data are copied into the *execute* sub-directory when a program is being tested. The test cases are documented in Chapter 12. Note that these are no longer within the distribution of any DL_POLY version but are made available on-line at the DL_POLY FTP - ftp://ftp.dl.ac.uk/ccp5/DL_POLY/ .

1.4.7 The *bench* Sub-directory

This directory contains examples of input and output data for DL_POLY_4 that are suitable for benchmarking DL_POLY_4 on large scale computers. These are described in Chapter 12. Note that these are no longer within the distribution of any DL_POLY version but are made available on-line at the DL_POLY FTP - ftp://ftp.dl.ac.uk/ccp5/DL_POLY/ .

1.4.8 The *java* Sub-directory

The DL_POLY_4 Java Graphical User Interface (GUI) is based on the Java language developed by Sun. The Java source code for this GUI is to be found in this sub-directory. The source is complete and sufficient to create a working GUI, provided the user has installed the Java Development Kit, (1.7 or above) which is available free from Sun at http://java.sun.com/. The GUI, once compiled, may be executed on any machine where Java is installed [21].

1.4.9 The *utility* Sub-directory

This sub-directory contains assorted routines donated by DL_POLY users. Potential users should note that these routines are **unsupported** and come **without any guarantee or liability whatsoever**. They should be regarded as potentially useful resources to be hacked into shape as needed by the user. Some of the various routines in this sub-directory are documented in the DL_POLY_Classic User Manual. Users who devise their own utilities are advised to store them in the *utility* sub-directory.

1.5 Obtaining the Source Code

To obtain a copy of DL_POLY_4 it is necessary to have internet connection. Log on to the DL_POLY website - http://www.ccp5.ac.uk/DL_POLY/ , and follow the links to the DL_POLY_4 registration page, where you will firstly be shown the DL_POLY_4 academic software licence (see Appendix **??**), which details the terms and conditions under which the code will be supplied. **By proceeding further with the registration and download process you are signalling your acceptance of the terms of this licence.** Click the 'Registration' button to find the registration page, where you will be invited to enter your name, address and e-mail address. The code is supplied free of charge to **academic** users, but **commercial** users will be required to purchase a software licence.

Once the online registration has been completed, information on downloading the DL_POLY_4 source code will be sent by e-mail, so **it is therefore essential to supply a correct e-mail address**.

The *data* and *bench* subdirectories of DL_POLY_4 are not issued in the standard package, but can be downloaded directly from the FTP site (in the ccp5/DL_POLY/DL_POLY_4.0/ directory).

Note: Daresbury Laboratory is the **sole centre** for the distribution of DL_POLY_4 and copies obtained from elsewhere will be regarded as illegal and will not be supported.

1.6 OS and Hardware Specific Ports

Note that no support is offered for these highly specific developments!

1.7 Other Information

The DL_POLY website - http://www.ccp5.ac.uk/DL_POLY/ , provides additional information in the form of

1. Access to all documentation (including licences)
2. Frequently asked questions
3. Bug reports
4. Access to the DL_Software portal.

Daresbury Laboratory also maintains a DL_POLY_4 associated electronic mailing list, *dl_poly_4_news*, to which all registered DL_POLY_4 users are automatically subscribed. It is via this list that error reports and announcements of new versions are made. If you are a DL_POLY_4 user, but not on this list you may request to be added by sending a mail message to majordomo@dl.ac.uk with the one-line message: *subscribe dl_poly_4_news*.

The DL_Software **Portal** is a web based centre for all DL_POLY users to exchange comments and queries. You may access the forum through the DL_POLY website. A registration (and vetting) process is required before you can use the forum, but it is open, in principle, to everyone.

Chapter 2

Force Field Interactions

Scope of Chapter

This chapter describes the variety of interaction potentials available in DL_POLY_4.

2.1 Introduction to the DL_POLY_4 Force Field

The force field is the set of functions needed to define the interactions in a molecular system. These may have a wide variety of analytical forms, with some basis in chemical physics, which must be parameterised to give the correct energy and forces. A huge variety of forms is possible and for this reason the DL_POLY_4 force field is designed to be agnostic and adaptable. While it is not supplied with its own force field parameters, many of the functions familiar to GROMOS [18], Dreiding [19] and AMBER [20] users have been coded in the package, as well as less familiar forms. In addition DL_POLY_4 retains the possibility of the user defining additional potentials.

In DL_POLY_4 the total configuration energy of a molecular system may be written as:

$$
\begin{aligned}
U(\underline{r}_1,\underline{r}_2,\ldots,\underline{r}_N) &= \sum_{i_{shel}=1}^{N_{shel}} U_{shel}(i_{shel},\underline{r}_{core},\underline{r}_{shell}) \\
&+ \sum_{i_{teth}=1}^{N_{teth}} U_{teth}(i_{teth},\underline{r}_i^{\mathbf{t}=t},\underline{r}_i^{\mathbf{t}=0}) \\
&+ \sum_{i_{bond}=1}^{N_{bond}} U_{bond}(i_{bond},\underline{r}_a,\underline{r}_b) \\
&+ \sum_{i_{angl}=1}^{N_{angl}} U_{angl}(i_{angl},\underline{r}_a,\underline{r}_b,\underline{r}_c) \\
&+ \sum_{i_{dihd}=1}^{N_{dihd}} U_{dihd}(i_{dihd},\underline{r}_a,\underline{r}_b,\underline{r}_c,\underline{r}_d) \\
&+ \sum_{i_{inv}=1}^{N_{inv}} U_{inv}(i_{inv},\underline{r}_a,\underline{r}_b,\underline{r}_c,\underline{r}_d) \\
&+ \sum_{i=1}^{N-1}\sum_{j>i}^{N} U_{2\text{-}body}^{(metal,vdw,electostatics)}(i,j,|\underline{r}_i-\underline{r}_j|) \qquad (2.1) \\
&+ \sum_{i=1}^{N}\sum_{j\neq i}^{N}\sum_{k\neq j}^{N} U_{tersoff}(i,j,k,\underline{r}_i,\underline{r}_j,\underline{r}_k) \\
&+ \sum_{i=1}^{N-2}\sum_{j>i}^{N-1}\sum_{k>j}^{N} U_{3\text{-}body}(i,j,k,\underline{r}_i,\underline{r}_j,\underline{r}_k) \\
&+ \sum_{i=1}^{N-3}\sum_{j>i}^{N-2}\sum_{k>j}^{N-1}\sum_{n>k}^{N} U_{4\text{-}body}(i,j,k,n,\underline{r}_i,\underline{r}_j,\underline{r}_k,\underline{r}_n) \\
&+ \sum_{i=1}^{N} U_{extn}(i,\underline{r}_i,\underline{v}_i) \quad ,
\end{aligned}
$$

where U_{shel}, U_{teth}, U_{bond}, U_{angl}, U_{dihd}, U_{inv}, $U_{2\text{-}body}^{(metal)}$, $U_{tersoff}$, $U_{3\text{-}body}$ and $U_{4\text{-}body}$ are empirical interaction functions representing ion core-shell polarisation, tethered particles, chemical bonds, valence angles, dihedral (and improper dihedral angles), inversion angles, two-body, Tersoff, three-body and four-body forces respectively. The first six are regarded by DL_POLY_4 as *intra*-molecular interactions and the next four as *inter*-molecular interactions. The final term U_{extn} represents an *external field* potential. The position vectors $\underline{r}_a, \underline{r}_b, \underline{r}_c$ and $\underline{r}_d$ refer to the positions of the atoms specifically involved in a given interaction. (Almost universally, it is the *differences* in position that determine the interaction.) The numbers N_{shel}, N_{teth}, N_{bond}, N_{angl}, N_{dihd} and N_{inv} refer to the total numbers of these respective interactions present in the simulated system, and the indices i_{shel}, i_{teth}, i_{bond}, i_{angl}, i_{dihd} and i_{inv} uniquely specify an individual interaction of each type. It is important to note that there is no global specification of the intramolecular interactions in DL_POLY_4 - all core-shell units, tethered particles, chemical bonds, valence angles, dihedral angles and inversion angles must be individually cited. The same applies for bond constraints and PMF constraints.

The indices i, j (and k, n) appearing in the intermolecular interactions' (non-bonded) terms indicate the atoms involved in the interaction. There is normally a very large number of these and they are therefore specified globally according to the atom *types* involved rather than indices. In DL_POLY_4 it is assumed that the "pure" two-body terms arise from short-ranged interactions such as van der Waals interactions (or alternatively DPD soft interactions, coarse-grained interactions, hard-wall nuclear interactions) and electrostatic interactions (coulombic, also regarded as long-ranged). Long-ranged forces require special techniques to evaluate accurately (see Section 2.4). The metal terms are many-body interactions which are functionally presented in an expansion of many two-body contributions augmented by a function of the local density, which again is derived from the two-body spatial distribution (and these are, therefore, evaluated in the two-body routines). In DL_POLY_4 the three-body terms are restricted to valence angle and H-bond forms.

Throughout this chapter the description of the force field assumes the simulated system is described as an assembly of atoms. This is for convenience only, and readers should understand that DL_POLY_4 does recognize molecular entities, defined through constraint bonds and rigid bodies. In the case of rigid bodies, the atomic forces are resolved into molecular forces and torques. These matters are discussed in greater detail in Sections 3.2 and 3.6.

2.2 The Intramolecular Potential Functions

In this section we catalogue and describe the forms of potential function available in DL_POLY_4. The **keywords** required to select potential forms are given in brackets () before each definition. The derivations of the atomic forces, virial and stress tensor are also outlined.

2.2.1 Bond Potentials

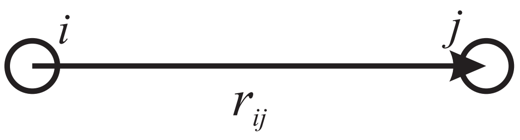


Figure 2.1: The interatomic bond vector

The bond potentials describe *explicit* chemical bonds between specified atoms. They are all functions of the interatomic distance. Only the coulomb potential makes an exception as it depends on the charges of the specified atoms. The potential functions available are as follows:

1. Harmonic bond: (**harm**)
$$U(r_{ij}) = \frac{1}{2}k(r_{ij} - r_o)^2 \tag{2.2}$$

2. Morse potential: (**mors**)
$$U(r_{ij}) = E_o[\{1 - \exp(-k(r_{ij} - r_o))\}^2 - 1] \tag{2.3}$$

3. 12-6 potential bond: (**12-6**)

$$U(r_{ij}) = \left(\frac{A}{r_{ij}^{12}}\right) - \left(\frac{B}{r_{ij}^{6}}\right) \tag{2.4}$$

4. Lennard-Jones potential: (**lj**)

$$U(r_{ij}) = 4\epsilon\left[\left(\frac{\sigma}{r_{ij}}\right)^{12} - \left(\frac{\sigma}{r_{ij}}\right)^{6}\right] \tag{2.5}$$

5. Restrained harmonic: (**rhrm**)

$$U(r_{ij}) = \begin{cases} \frac{1}{2}k(r_{ij}-r_o)^2 & : \quad |r_{ij}-r_o| \le r_c \\ \frac{1}{2}kr_c^2 + kr_c(|r_{ij}-r_o|-r_c) & : \quad |r_{ij}-r_o| > r_c \end{cases} \tag{2.6}$$

6. Quartic potential: (**quar**)

$$U(r_{ij}) = \frac{k}{2}(r_{ij}-r_o)^2 + \frac{k'}{3}(r_{ij}-r_o)^3 + \frac{k''}{4}(r_{ij}-r_o)^4 \tag{2.7}$$

7. Buckingham potential: (**buck**)

$$U(r_{ij}) = A\ \exp\left(-\frac{r_{ij}}{\rho}\right) - \frac{C}{r_{ij}^{6}} \tag{2.8}$$

8. Coulomb potential: (**coul**)

$$U(r_{ij}) = k \cdot U^{Electrostatics}(r_{ij})\ \left(= \frac{k}{4\pi\epsilon_0\epsilon}\frac{q_i q_j}{r_{ij}}\right) \quad , \tag{2.9}$$

where q_ℓ is the charge on an atom labelled ℓ. It is worth noting that the Coulomb potential switches to the particular model of Electrostatics opted in CONTROL.

9. Shifted finitely extendible non-linear elastic (FENE) potential [32, 33, 34]: (**fene**)

$$U(r_{ij}) = \begin{cases} -0.5\ k\ R_o^2\ ln\left[1 - \left(\frac{r_{ij}-\Delta}{R_o}\right)^2\right] & : \quad |r_{ij}-\Delta| < R_o \\ \infty & : \quad |r_{ij}-\Delta| \ge R_o \end{cases} \tag{2.10}$$

The FENE potential is used to maintain the distance between connected beads and to prevent chains from crossing each other. It is used in combination with the WCA, equation (2.97), potential to create a potential well for the flexible bonds of a molecule, that maintains the topology of the molecule. This implementation allows for a radius shift of up to half a R_o ($|\Delta| \le 0.5\ R_o$) with a default of zero ($\Delta_{default} = 0$).

10. MM3 bond stretch potential [35]: (**mmst**)

$$U(r_{ij}) = k\ (r_{ij}-r_o)^2\left[1 - 2.55\ (r_{ij}-r_o) + (7/12)\ 2.55^2\ (r_{ij}-r_o)^2\right] \tag{2.11}$$

11. Tabulated potential: (**tab**). The potential is defined numerically in TABBND (see Section 4.3 and Section 10.1.9).

In these formulae r_{ij} is the distance between atoms labelled i and j:

$$r_{ij} = |\underline{r}_j - \underline{r}_i|^{*} \quad , \tag{2.12}$$

where $\underline{r}_\ell$ is the position vector of an atom labelled ℓ.

* **Note**: some DL_POLY_4 routines may use the convention that $\underline{r_{ij}} = \underline{r}_i - \underline{r}_j$.

The force on the atom j arising from a bond potential is obtained using the general formula:

$$\underline{f}_j = -\frac{1}{r_{ij}} \left[\frac{\partial}{\partial r_{ij}} U(r_{ij}) \right] \underline{r}_{ij} \quad . \tag{2.13}$$

The force $\underline{f}_i$ acting on atom i is the negative of this.
The contribution to be added to the atomic virial is given by

$$\mathcal{W} = -\underline{r}_{ij} \cdot \underline{f}_j \quad , \tag{2.14}$$

with only *one* such contribution from each bond.
The contribution to be added to the atomic stress tensor is given by

$$\sigma^{\alpha\beta} = r_{ij}^{\alpha} f_j^{\beta} \quad , \tag{2.15}$$

where α and β indicate the x, y, z components. The atomic stress tensor derived in this way is symmetric.
In DL_POLY_4 bond forces are handled by the routine BONDS_FORCES (and INTRA_COUL called within).

2.2.2 Distance Restraints

In DL_POLY_4 distance restraints, in which the separation between two atoms, is maintained around some preset value r_0 is handled as a special case of bond potentials. As a consequence, distance restraints may be applied only between atoms in the same molecule. Unlike with application of the "pure" bond potentials, the electrostatic and van der Waals interactions between the pair of atoms are still evaluated when distance restraints are applied. All the potential forms of the previous section are available as distance restraints, although they have different key words:

1. Harmonic potential: (**-hrm**)
2. Morse potential: (**-mrs**)
3. 12-6 potential bond: (**-126**)
4. Lennard-Jones potential: (**-lj**)
5. Restrained harmonic: (**-rhm**)
6. Quartic potential: (**-qur**)
7. Buckingham potential: (**-bck**)
8. Coulomb potential: (**-cul**)
9. FENE potential: (**-fne**)
10. MM3 bond stretch potential [35]: (**-m3s**)
11. Tabulated potential: (**-tab**). The potential is defined numerically in TABBND (see Section 4.3 and Section 10.1.9).

In DL_POLY_4 distance restraints are handled by the routine BONDS_FORCES (and INTRA_COUL called within).

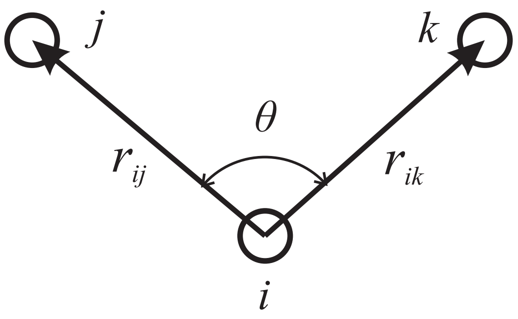


Figure 2.2: The valence angle and associated vectors

2.2.3 Valence Angle Potentials

The valence angle potentials describe the bond bending terms between the specified atoms. They should not be confused with the three-body potentials described later, which are defined by atom types rather than indices.

1. Harmonic: (**harm**)
$$U(\theta_{jik}) = \frac{k}{2}(\theta_{jik} - \theta_0)^2 \tag{2.16}$$

2. Quartic: (**quar**)
$$U(\theta_{jik}) = \frac{k}{2}(\theta_{jik} - \theta_0)^2 + \frac{k'}{3}(\theta_{jik} - \theta_0)^3 + \frac{k''}{4}(\theta_{jik} - \theta_0)^4 \tag{2.17}$$

3. Truncated harmonic: (**thrm**)
$$U(\theta_{jik}) = \frac{k}{2}(\theta_{jik} - \theta_0)^2 \exp[-(r_{ij}^8 + r_{ik}^8)/\rho^8] \tag{2.18}$$

4. Screened harmonic: (**shrm**)
$$U(\theta_{jik}) = \frac{k}{2}(\theta_{jik} - \theta_0)^2 \exp[-(r_{ij}/\rho_1 + r_{ik}/\rho_2)] \tag{2.19}$$

5. Screened Vessal [36]: (**bvs1**)
$$\begin{aligned} U(\theta_{jik}) &= \frac{k}{8(\theta_0 - \pi)^2}\left[(\theta_0 - \pi)^2 - (\theta_{jik} - \pi)^2\right]^2 \times \\ & \exp[-(r_{ij}/\rho_1 + r_{ik}/\rho_2)] \end{aligned} \tag{2.20}$$

6. Truncated Vessal [37]: (**bvs2**)
$$\begin{aligned} U(\theta_{jik}) &= k\ (\theta_{jik} - \theta_0)^2\ \Big[\ \theta_{jik}^a(\theta_{jik} + \theta_0 - 2\pi)^2 + \\ & \frac{a}{2}\pi^{a-1}(\theta_0 - \pi)^3\ \Big]\quad \exp[-(r_{ij}^8 + r_{ik}^8)/\rho^8] \end{aligned} \tag{2.21}$$

7. Harmonic cosine: (**hcos**)
$$U(\theta_{jik}) = \frac{k}{2}(\cos(\theta_{jik}) - \cos(\theta_0))^2 \tag{2.22}$$

8. Cosine: (**cos**)
$$U(\theta_{jik}) = A\ [1 + \cos(m\ \theta_{jik} - \delta)] \tag{2.23}$$

9. MM3 stretch-bend [35]: (**mmsb**)

$$U(\theta_{jik}) = A\ (\theta_{jik} - \theta_0)\ (r_{ij} - r_{ij}^o)\ (r_{ik} - r_{ik}^o) \tag{2.24}$$

10. Compass stretch-stretch [38]: (**stst**)

$$U(\theta_{jik}) = A\ (r_{ij} - r_{ij}^o)\ (r_{ik} - r_{ik}^o) \tag{2.25}$$

11. Compass stretch-bend [38]: (**stbe**)

$$U(\theta_{jik}) = A\ (\theta_{jik} - \theta_0)\ (r_{ij} - r_{ij}^o) \tag{2.26}$$

12. Compass all terms [38]: (**cmps**)

$$U(\theta_{jik}) = A\ (r_{ij} - r_{ij}^o)\ (r_{ik} - r_{ik}^o) + (\theta_{jik} - \theta_0)\ [B\ (r_{ij} - r_{ij}^o) + C\ (r_{ik} - r_{ik}^o)] \tag{2.27}$$

13. MM3 angle bend term [35]: (**m3ab**)

$$\begin{aligned} U(\theta_{jik}) = k\ (\theta_{jik} - \theta_0)^2 [1 &- 1.4 \cdot 10^{-2}(\theta_{jik} - \theta_0)\ + 5.6 \cdot 10^{-5}(\theta_{jik} - \theta_0)^2 \\ &- 7.0 \cdot 10^{-7}(\theta_{jik} - \theta_0)^3 + 2.2 \cdot 10^{-8}(\theta_{jik} - \theta_0)^4] \end{aligned} \tag{2.28}$$

14. KKY [39]: (**kky**)

$$\begin{aligned} U(\theta_{jik}) &= 2\ f_k\ \sqrt{K_{ij} \cdot K_{ik}}\ \sin^2 [(\theta_{jik} - \theta_0)] \\ K_{ij} &= \frac{1}{\exp [g_r(r_{ij} - r_o)] + 1} \end{aligned} \tag{2.29}$$

15. Tabulated potential: (**tab**). The potential is defined numerically in TABANG (see Section 4.3 and Section 10.1.9).

In these formulae θ_{jik} is the angle between bond vectors $\underline{r}_{ij}$ and $\underline{r}_{ik}$:

$$\theta_{jik} = cos^{-1} \left\{ \frac{\underline{r}_{ij} \cdot \underline{r}_{ik}}{r_{ij} r_{ik}} \right\} \quad . \tag{2.30}$$

In DL_POLY_4 the most general form for the valence angle potentials can be written as:

$$U(\theta_{jik}, r_{ij}, r_{ik}) = A(\theta_{jik})\ S(r_{ij})\ S(r_{ik})\ S(r_{ik}) \quad , \tag{2.31}$$

where $A(\theta)$ is a purely angular function and $S(r)$ is a screening or truncation function. All the function arguments are scalars. With this reduction the force on an atom derived from the valence angle potential is given by:

$$f_\ell^\alpha = -\frac{\partial}{\partial r_\ell^\alpha} U(\theta_{jik}, r_{ij}, r_{ik}, r_{jk}) \quad , \tag{2.32}$$

with atomic label ℓ being one of i, j, k and α indicating the x, y, z component. The derivative is

$$\begin{aligned} -\frac{\partial}{\partial r_\ell^\alpha} U(\theta_{jik}, r_{ij}, r_{ik}, r_{jk}) \quad &= \quad -S(r_{ij})S(r_{ik})S(r_{jk}) \frac{\partial}{\partial r_\ell^\alpha} A(\theta_{jik}) \\ &\quad -A(\theta_{jik})S(r_{ik})S(r_{jk})(\delta_{\ell j} - \delta_{\ell i}) \frac{r_{ij}^\alpha}{r_{ij}} \frac{\partial}{\partial r_{ij}} S(r_{ij}) \\ &\quad -A(\theta_{jik})S(r_{ij})S(r_{jk})(\delta_{\ell k} - \delta_{\ell i}) \frac{r_{ik}^\alpha}{r_{ik}} \frac{\partial}{\partial r_{ik}} S(r_{ik}) \\ &\quad -A(\theta_{jik})S(r_{ij})S(r_{ik})(\delta_{\ell k} - \delta_{\ell j}) \frac{r_{jk}^\alpha}{r_{jk}} \frac{\partial}{\partial r_{jk}} S(r_{jk}) \quad , \end{aligned} \tag{2.33}$$

with $\delta_{ab} = 1$ if $a = b$ and $\delta_{ab} = 0$ if $a \neq b$. In the absence of screening terms $S(r)$, this formula reduces to:

$$-\frac{\partial}{\partial r_\ell^\alpha} U(\theta_{jik}, r_{ij}, r_{ik}, r_{jk}) = -\frac{\partial}{\partial r_\ell^\alpha} A(\theta_{jik}) \quad . \tag{2.34}$$

The derivative of the angular function is

$$-\frac{\partial}{\partial r_\ell^\alpha} A(\theta_{jik}) = \left\{ \frac{1}{\sin(\theta_{jik})} \right\} \frac{\partial}{\partial \theta_{jik}} A(\theta_{jik}) \frac{\partial}{\partial r_\ell^\alpha} \left\{ \frac{\underline{r}_{ij} \cdot \underline{r}_{ik}}{r_{ij} r_{ik}} \right\} \quad , \tag{2.35}$$

with

$$\begin{aligned} \frac{\partial}{\partial r_\ell^\alpha} \left\{ \frac{\underline{r}_{ij} \cdot \underline{r}_{ik}}{r_{ij} r_{ik}} \right\} &= (\delta_{\ell j} - \delta_{\ell i}) \frac{r_{ik}^\alpha}{r_{ij} r_{ik}} + (\delta_{\ell k} - \delta_{\ell i}) \frac{r_{ij}^\alpha}{r_{ij} r_{ik}} - \\ & \cos(\theta_{jik}) \left\{ (\delta_{\ell j} - \delta_{\ell i}) \frac{r_{ij}^\alpha}{r_{ij}^2} + (\delta_{\ell k} - \delta_{\ell i}) \frac{r_{ik}^\alpha}{r_{ik}^2} \right\} \quad . \end{aligned} \tag{2.36}$$

The atomic forces are then completely specified by the derivatives of the particular functions $A(\theta)$ and $S(r)$.

The contribution to be added to the atomic virial is given by

$$\mathcal{W} = -(\underline{r}_{ij} \cdot \underline{f}_j + \underline{r}_{ik} \cdot \underline{f}_k) \quad . \tag{2.37}$$

It is worth noting that in the absence of screening terms S(r), the virial is zero [40].

The contribution to be added to the atomic stress tensor is given by

$$\sigma^{\alpha\beta} = r_{ij}^\alpha f_j^\beta + r_{ik}^\alpha f_k^\beta \tag{2.38}$$

and the stress tensor is symmetric.

In DL_POLY_4 valence forces are handled by the routine ANGLES_FORCES.

2.2.4 Angular Restraints

In DL_POLY_4 angle restraints, in which the angle subtended by a triplet of atoms, is maintained around some preset value θ_0 is handled as a special case of angle potentials. As a consequence angle restraints may be applied only between atoms in the same molecule. Unlike with application of the "pure" angle potentials, the electrostatic and van der Waals interactions between the pair of atoms are still evaluated when distance restraints are applied. All the potential forms of the previous section are available as angular restraints, although they have different key words:

1. Harmonic: (**-hrm**)
2. Quartic: (**-qur**)
3. Truncated harmonic: (**-thm**)
4. Screened harmonic: (**-shm**)
5. Screened Vessal [36]: (**-bv1**)
6. Truncated Vessal [37]: (**-bv2**)
7. Harmonic cosine: (**-hcs**)
8. Cosine: (**-cos**)
9. MM3 stretch-bend [35]: (**-msb**)

10. Compass stretch-stretch [38]: (**-sts**)
11. Compass stretch-bend [38]: (**-stb**)
12. Compass all terms [38]: (**-cmp**)
13. MM3 angle bend [35]: (**-m3a**)
14. KKY [39]: (**-kky**)
15. Tabulated potential: (**-tab**). The potential is defined numerically in TABANG (see Section 4.3 and Section 10.1.9).

In DL_POLY_4 angular restraints are handled by the routine ANGLES_FORCES.

2.2.5 Dihedral Angle Potentials

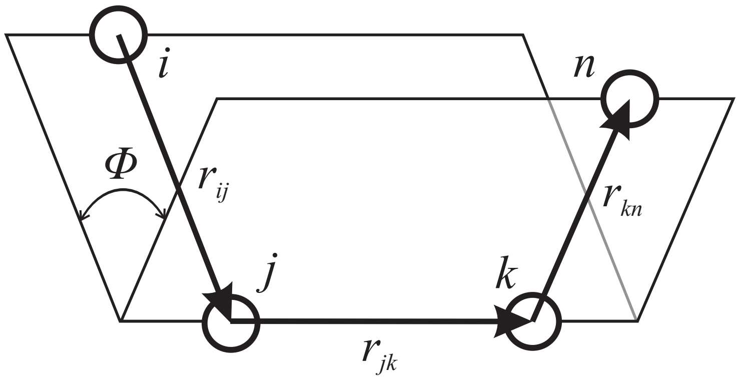


Figure 2.3: The dihedral angle and associated vectors

The dihedral angle potentials describe the interaction arising from torsional forces in molecules. (They are sometimes referred to as torsion potentials.) They require the specification of four atomic positions. The potential functions available in DL_POLY_4 are as follows:

1. Cosine potential: (**cos**)
$$U(\phi_{ijkn}) = A\ \left[1 + \cos(m\phi_{ijkn} - \delta)\right] \tag{2.39}$$
2. Harmonic: (**harm**)
$$U(\phi_{ijkn}) = \frac{k}{2}\ (\phi_{ijkn} - \phi_0)^2 \tag{2.40}$$
3. Harmonic cosine: (**hcos**)
$$U(\phi_{ijkn}) = \frac{k}{2}\ (\cos(\phi_{ijkn}) - \cos(\phi_0))^2 \tag{2.41}$$
4. Triple cosine: (**cos3**)
$$U(\phi) = \frac{1}{2}\ \{A_1\ (1 + \cos(\phi)) + A_2\ (1 - \cos(2\phi)) + A_3\ (1 + \cos(3\phi))\} \tag{2.42}$$
5. Ryckaert-Bellemans [41] with fixed constants a-f: (**ryck**)
$$U(\phi) = A\ \{\ a + b\ \cos(\phi) + c\ \cos^2(\phi) + d\ \cos^3(\phi) + e\ \cos^4(\phi) + f\ \cos^5(\phi)\ \} \tag{2.43}$$

6. Fluorinated Ryckaert-Bellemans [42] with fixed constants a-h: (**rbf**)

$$U(\phi) = A \left\{ a + b \cos(\phi) + c \cos^2(\phi) + d \cos^3(\phi) + e \cos^4(\phi) + f \cos^5(\phi) + g \exp(-h(\phi - \pi)^2)) \right\} \tag{2.44}$$

7. OPLS torsion potential: (**opls**)

$$U(\phi) = A_0 + \frac{1}{2} \{A_1 (1 + \cos(\phi)) + A_2 (1 - \cos(2\phi)) + A_3 (1 + \cos(3\phi))\} \tag{2.45}$$

8. Tabulated potential: (**tab**). The potential is defined numerically in TABDIH (see Section 4.3 and Section 10.1.9).

In these formulae ϕ_{ijkn} is the dihedral angle defined by

$$\phi_{ijkn} = \cos^{-1}\{B(\underline{r}_{ij}, \underline{r}_{jk}, \underline{r}_{kn})\} \quad , \tag{2.46}$$

with

$$B(\underline{r}_{ij}, \underline{r}_{jk}, \underline{r}_{kn}) = \left\{ \frac{(\underline{r}_{ij} \times \underline{r}_{jk}) \cdot (\underline{r}_{jk} \times \underline{r}_{kn})}{|\underline{r}_{ij} \times \underline{r}_{jk}||\underline{r}_{jk} \times \underline{r}_{kn}|} \right\} \quad . \tag{2.47}$$

With this definition, the sign of the dihedral angle is positive if the vector product $(\underline{r}_{ij} \times \underline{r}_{jk}) \times (\underline{r}_{jk} \times \underline{r}_{kn})$ is in the same direction as the bond vector $\underline{r}_{jk}$ and negative if in the opposite direction.

The force on an atom arising from the dihedral potential is given by

$$f_\ell^\alpha = -\frac{\partial}{\partial r_\ell^\alpha} U(\phi_{ijkn}) \quad , \tag{2.48}$$

with ℓ being one of i, j, k, n and α one of x, y, z. This may be expanded into

$$-\frac{\partial}{\partial r_\ell^\alpha} U(\phi_{ijkn}) = \left\{ \frac{1}{\sin(\phi_{ijkn})} \right\} \frac{\partial}{\partial \phi_{ijkn}} U(\phi_{ijkn}) \frac{\partial}{\partial r_\ell^\alpha} B(\underline{r}_{ij}, \underline{r}_{jk}, \underline{r}_{kn}) \quad . \tag{2.49}$$

The derivative of the function $B(\underline{r}_{ij}, \underline{r}_{jk}, \underline{r}_{kn})$ is

$$\begin{aligned} \frac{\partial}{\partial r_\ell^\alpha} B(\underline{r}_{ij}, \underline{r}_{jk}, \underline{r}_{kn}) &= \frac{1}{|\underline{r}_{ij} \times \underline{r}_{jk}||\underline{r}_{jk} \times \underline{r}_{kn}|} \frac{\partial}{\partial r_\ell^\alpha} \{(\underline{r}_{ij} \times \underline{r}_{jk}) \cdot (\underline{r}_{jk} \times \underline{r}_{kn})\} - \\ & \frac{\cos(\phi_{ijkn})}{2} \left\{ \frac{1}{|\underline{r}_{ij} \times \underline{r}_{jk}|^2} \frac{\partial}{\partial r_\ell^\alpha} |\underline{r}_{ij} \times \underline{r}_{jk}|^2 + \frac{1}{|\underline{r}_{jk} \times \underline{r}_{kn}|^2} \frac{\partial}{\partial r_\ell^\alpha} |\underline{r}_{jk} \times \underline{r}_{kn}|^2 \right\} \quad , \end{aligned} \tag{2.50}$$

with

$$\begin{aligned} \frac{\partial}{\partial r_\ell^\alpha} \{(\underline{r}_{ij} \times \underline{r}_{jk}) \cdot (\underline{r}_{jk} \times \underline{r}_{kn})\} &= r_{ij}^\alpha ([\underline{r}_{jk}\underline{r}_{jk}]_\alpha (\delta_{\ell k} - \delta_{\ell n}) + [\underline{r}_{jk}\underline{r}_{kn}]_\alpha (\delta_{\ell k} - \delta_{\ell j})) + \\ & r_{jk}^\alpha ([\underline{r}_{ij}\underline{r}_{jk}]_\alpha (\delta_{\ell n} - \delta_{\ell k}) + [\underline{r}_{jk}\underline{r}_{kn}]_\alpha (\delta_{\ell j} - \delta_{\ell i})) + \\ & r_{kn}^\alpha ([\underline{r}_{ij}\underline{r}_{jk}]_\alpha (\delta_{\ell k} - \delta_{\ell j}) + [\underline{r}_{jk}\underline{r}_{jk}]_\alpha (\delta_{\ell i} - \delta_{\ell j})) + \\ & 2 r_{jk}^\alpha [\underline{r}_{ij}\underline{r}_{kn}]_\alpha (\delta_{\ell j} - \delta_{\ell k}) \quad , \end{aligned} \tag{2.51}$$

$$\begin{aligned} \frac{\partial}{\partial r_\ell^\alpha} |\underline{r}_{ij} \times \underline{r}_{jk}|^2 &= 2 r_{ij}^\alpha ([\underline{r}_{jk}\underline{r}_{jk}]_\alpha (\delta_{\ell j} - \delta_{\ell i}) + [\underline{r}_{ij}\underline{r}_{jk}]_\alpha (\delta_{\ell j} - \delta_{\ell k})) + \\ & 2 r_{jk}^\alpha ([\underline{r}_{ij}\underline{r}_{ij}]_\alpha (\delta_{\ell k} - \delta_{\ell j}) + [\underline{r}_{ij}\underline{r}_{jk}]_\alpha (\delta_{\ell i} - \delta_{\ell j})) \quad , \end{aligned} \tag{2.52}$$

$$\begin{aligned}\frac{\partial}{\partial r_\ell^\alpha}|\underline{r}_{jk} \times \underline{r}_{kn}|^2 &= 2r_{kn}^\alpha([\underline{r}_{jk}\underline{r}_{jk}]_\alpha(\delta_{\ell n} - \delta_{\ell k}) + [\underline{r}_{jk}\underline{r}_{kn}]_\alpha(\delta_{\ell j} - \delta_{\ell k})) + \\ &\quad 2r_{jk}^\alpha([\underline{r}_{kn}\underline{r}_{kn}]_\alpha(\delta_{\ell k} - \delta_{\ell j}) + [\underline{r}_{jk}\underline{r}_{kn}]_\alpha(\delta_{\ell k} - \delta_{\ell n})) \quad .\end{aligned} \tag{2.53}$$

Where we have used the following definition:

$$[\underline{a}\ \underline{b}]_\alpha = \sum_\beta (1 - \delta_{\alpha\beta}) a^\beta b^\beta \quad . \tag{2.54}$$

Formally, the contribution to be added to the atomic virial is given by

$$\mathcal{W} = -\sum_{i=1}^{4} \underline{r}_i \cdot \underline{f}_i \quad . \tag{2.55}$$

However, it is possible to show (by tedious algebra using the above formulae, or more elegantly by thermodynamic arguments [40],) that the dihedral makes *no* contribution to the atomic virial.

The contribution to be added to the atomic stress tensor is given by

$$\begin{aligned}\sigma^{\alpha\beta} &= r_{ij}^\alpha p_i^\beta + r_{jk}^\alpha p_{jk}^\beta + r_{kn}^\alpha p_n^\beta \\ &\quad -\frac{\cos(\phi_{ijkn})}{2}\left\{r_{ij}^\alpha g_i^\beta + r_{jk}^\alpha g_k^\beta + r_{jk}^\alpha h_j^\beta + r_{kn}^\alpha h_n^\beta\right\} \quad ,\end{aligned} \tag{2.56}$$

with

$$\begin{aligned}p_i^\alpha &= (r_{jk}^\alpha[\underline{r}_{jk}\underline{r}_{kn}]_\alpha - r_{kn}^\alpha[\underline{r}_{jk}\underline{r}_{jk}]_\alpha)/(|\underline{r}_{ij} \times \underline{r}_{jk}||\underline{r}_{jk} \times \underline{r}_{kn}|) \\ p_n^\alpha &= (r_{jk}^\alpha[\underline{r}_{ij}\underline{r}_{jk}]_\alpha - r_{ij}^\alpha[\underline{r}_{jk}\underline{r}_{jk}]_\alpha)/(|\underline{r}_{ij} \times \underline{r}_{jk}||\underline{r}_{jk} \times \underline{r}_{kn}|) \\ p_{jk}^\alpha &= (r_{ij}^\alpha[\underline{r}_{jk}\underline{r}_{kn}]_\alpha + r_{kn}^\alpha[\underline{r}_{ij}\underline{r}_{jk}]_\alpha - 2r_{jk}^\alpha[\underline{r}_{ij}\underline{r}_{kn}]_\alpha)/(|\underline{r}_{ij} \times \underline{r}_{jk}||\underline{r}_{jk} \times \underline{r}_{kn}|) \\ g_i^\alpha &= 2(r_{ij}^\alpha[\underline{r}_{jk}\underline{r}_{jk}]_\alpha - r_{jk}^\alpha[\underline{r}_{ij}\underline{r}_{jk}]_\alpha)/|\underline{r}_{ij} \times \underline{r}_{jk}|^2 \\ g_k^\alpha &= 2(r_{jk}^\alpha[\underline{r}_{ij}\underline{r}_{ij}]_\alpha - r_{ij}^\alpha[\underline{r}_{ij}\underline{r}_{jk}]_\alpha)/|\underline{r}_{ij} \times \underline{r}_{jk}|^2 \\ h_j^\alpha &= 2(r_{jk}^\alpha[\underline{r}_{kn}\underline{r}_{kn}]_\alpha - r_{kn}^\alpha[\underline{r}_{jk}\underline{r}_{kn}]_\alpha)/|\underline{r}_{jk} \times \underline{r}_{kn}|^2 \\ h_n^\alpha &= 2(r_{kn}^\alpha[\underline{r}_{kn}\underline{r}_{kn}]_\alpha - r_{jk}^\alpha[\underline{r}_{jk}\underline{r}_{kn}]_\alpha)/|\underline{r}_{jk} \times \underline{r}_{kn}|^2 \quad .\end{aligned} \tag{2.57}$$

The sum of the diagonal elements of the stress tensor is zero (since the virial is zero) and the matrix is symmetric.

Lastly, it should be noted that the above description does not take into account the possible inclusion of distance-dependent 1-4 interactions, as permitted by some force fields. Such interactions are permissible in DL_POLY_4 and are described in the section on pair potentials below. DL_POLY_4 also permits scaling of the 1-4 van der Waals and Coulomb interactions by a numerical factor (see Table 10.7). **Note** that scaling is abandoned when the 1-4 members are also 1-3 members in a valence angle intercation (1-4 checks are performed in DIHEDRALS_14_CHECK routine). 1-4 interactions do, of course, contribute to the atomic virial.

In DL_POLY_4 dihedral forces are handled by the routine DIHEDRALS_FORCES (and INTRA_COUL and DIHEDRALS_14_VDW called within).

2.2.6 Improper Dihedral Angle Potentials

Improper dihedrals are used to restrict the geometry of molecules and as such need not have a simple relation to conventional chemical bonding. DL_POLY_4 makes no distinction between dihedral and improper dihedral angle functions (both are calculated by the same subroutines) and all the comments made in the preceding section apply.

An important example of the use of the improper dihedral is to conserve the structure of chiral centres in molecules modelled by united-atom centres. For example α-amino acids such as alanine ($CH_3CH(NH_2)COOH$), in which it is common to represent the CH_3 and CH groups as single centres. Conservation of the chirality of the α carbon is achieved by defining a harmonic improper dihedral angle potential with an equilibrium angle of 35.264^o. The angle is defined by vectors $\underline{r}_{12}$, $\underline{r}_{23}$ and $\underline{r}_{34}$, where the atoms 1,2,3 and 4 are shown in the following figure. The figure defines the D and L enantiomers consistent with the international (IUPAC) convention. When defining the dihedral, the atom indices are entered in DL_POLY_4 in the order 1-2-3-4.

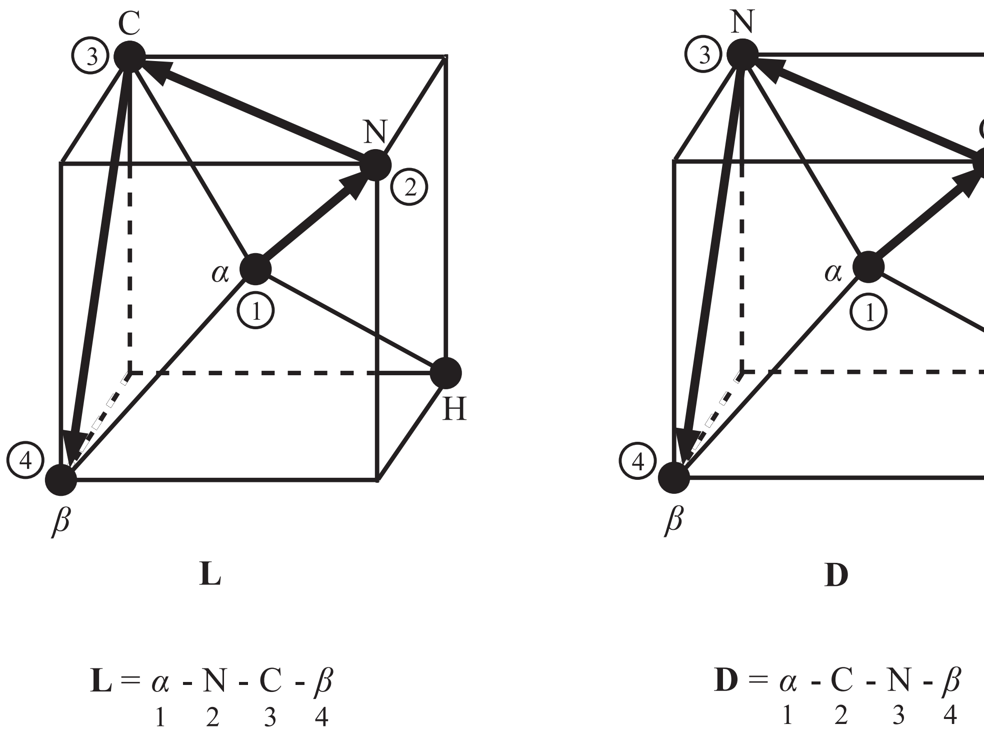


Figure 2.4: The L and D enantiomers and defining vectors

In DL_POLY_4 improper dihedral forces are handled by the routine DIHEDRALS_FORCES.

2.2.7 Torsional Restraints

In DL_POLY_4 the torsional restraints, in which the dihedral angle as defined by a quadruplet of atoms, is maintained around some preset value ϕ_0 is handled as a special case of dihedral potential. As a consequence angle restraints may be applied only between atoms in the same molecule. Unlike with application of the "pure" dihedral potentials, the electrostatic and van der Waals interactions between the pair of atoms are still evaluated when distance restraints are applied. All the potential forms of the previous section are available as torsional restraints, although they have different key words:

1. Cosine potential: (**-cos**)
2. Harmonic: (**-hrm**)
3. Harmonic cosine: (**-hcs**)
4. Triple cosine: (**-cs3**)
5. Ryckaert-Bellemans [41] with fixed constants a-f: (**-rck**)
6. Fluorinated Ryckaert-Bellemans [42] with fixed constants a-h: (**-rbf**)
7. OPLS torsion potential: (**-opl**)

8. Tabulated potential: (**-tab**). The potential is defined numerically in TABDIH (see Section 4.3 and Section 10.1.9).

In DL_POLY_4 torsional restraints are handled by the routine DIHEDRALS_FORCES.

2.2.8 Inversion Angle Potentials

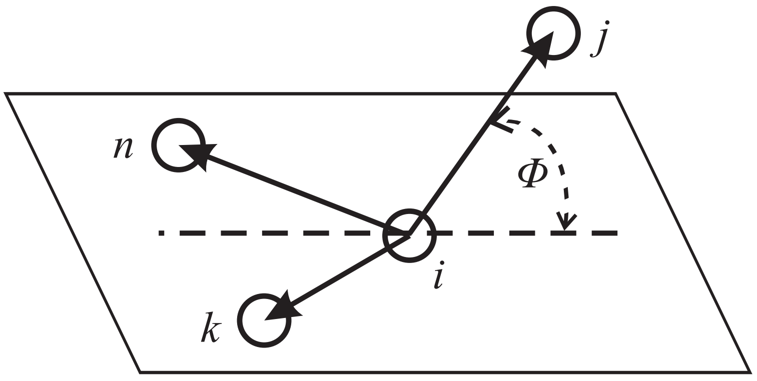


Figure 2.5: The inversion angle and associated vectors

The inversion angle potentials describe the interaction arising from a particular geometry of three atoms around a central atom. The best known example of this is the arrangement of hydrogen atoms around nitrogen in ammonia to form a trigonal pyramid. The hydrogens can 'flip' like an inverting umbrella to an alternative structure, which in this case is identical, but in principle causes a change in chirality. The force restraining the ammonia to one structure can be described as an inversion potential (though it is usually augmented by valence angle potentials also). The inversion angle is defined in the figure above - **note that the inversion angle potential is a sum of the three possible inversion angle terms**. It resembles a dihedral potential in that it requires the specification of four atomic positions.

The potential functions available in DL_POLY_4 are as follows:

1. Harmonic: (**harm**)
$$U(\phi_{ijkn}) = \frac{k}{2}\ (\phi_{ijkn} - \phi_0)^2 \tag{2.58}$$

2. Harmonic cosine: (**hcos**)
$$U(\phi_{ijkn}) = \frac{k}{2}\ (\cos(\phi_{ijkn}) - \cos(\phi_0))^2 \tag{2.59}$$

3. Planar potential: (**plan**)
$$U(\phi_{ijkn}) = A\ \ [1 - \cos(\phi_{ijkn})] \tag{2.60}$$

4. Extended planar potential: (**xpln**)
$$U(\phi_{ijkn}) = \frac{k}{2}\ \ [1 - \cos(m\ \phi_{ijkn} - \phi_0)] \tag{2.61}$$

5. Tabulated potential: (**tab**). The potential is defined numerically in TABINV (see Section 4.3 and Section 10.1.9).

In these formulae ϕ_{ijkn} is the inversion angle defined by

$$\phi_{ijkn} = \cos^{-1}\left\{\frac{\underline{r}_{ij}\cdot\underline{w}_{kn}}{r_{ij}w_{kn}}\right\} \quad , \tag{2.62}$$

with

$$\underline{w}_{kn} = (\underline{r}_{ij}\cdot\hat{\underline{u}}_{kn})\hat{\underline{u}}_{kn} + (\underline{r}_{ij}\cdot\hat{\underline{v}}_{kn})\hat{\underline{v}}_{kn} \tag{2.63}$$

and the unit vectors

$$\begin{aligned} \hat{\underline{u}}_{kn} &= (\hat{\underline{r}}_{ik}+\hat{\underline{r}}_{in})/|\hat{\underline{r}}_{ik}+\hat{\underline{r}}_{in}| \\ \hat{\underline{v}}_{kn} &= (\hat{\underline{r}}_{ik}-\hat{\underline{r}}_{in})/|\hat{\underline{r}}_{ik}-\hat{\underline{r}}_{in}| \quad . \end{aligned} \tag{2.64}$$

As usual, $\underline{r}_{ij} = \underline{r}_j - \underline{r}_i$ *etc.* and the hat $\hat{\underline{r}}$ indicates a *unit* vector in the direction of $\underline{r}$. The total inversion potential requires the calculation of three such angles, the formula being derived from the above using the cyclic permutation of the indices $j \to k \to n \to j$ *etc.*

Equivalently, the angle ϕ_{ijkn} may be written as

$$\phi_{ijkn} = \cos^{-1}\left\{\frac{[(\underline{r}_{ij}\cdot\hat{\underline{u}}_{kn})^2 + (\underline{r}_{ij}\cdot\hat{\underline{v}}_{kn})^2]^{1/2}}{r_{ij}}\right\} \quad . \tag{2.65}$$

Formally, the force on an atom arising from the inversion potential is given by

$$f_\ell^\alpha = -\frac{\partial}{\partial r_\ell^\alpha}U(\phi_{ijkn}) \quad , \tag{2.66}$$

with ℓ being one of i, j, k, n and α one of x, y, z. This may be expanded into

$$\begin{aligned} -\frac{\partial}{\partial r_\ell^\alpha}U(\phi_{ijkn}) &= \left\{\frac{1}{\sin(\phi_{ijkn})}\right\}\frac{\partial}{\partial\phi_{ijkn}}U(\phi_{ijkn}) \times \\ & \quad \frac{\partial}{\partial r_\ell^\alpha}\left\{\frac{[(\underline{r}_{ij}\cdot\hat{\underline{u}}_{kn})^2 + (\underline{r}_{ij}\cdot\hat{\underline{v}}_{kn})^2]^{1/2}}{r_{ij}}\right\} \quad . \end{aligned} \tag{2.67}$$

Following through, the (extremely tedious!) differentiation gives the result:

$$\begin{aligned} f_\ell^\alpha &= \left\{\frac{1}{\sin(\phi_{ijkn})}\right\}\frac{\partial}{\partial\phi_{ijkn}}U(\phi_{ijkn}) \times \\ & \left\{-(\delta_{\ell j}-\delta_{\ell i})\frac{\cos(\phi_{ijkn})}{r_{ij}^2}r_{ij}^\alpha + \frac{1}{r_{ij}w_{kn}}\left[(\delta_{\ell j}-\delta_{\ell i})\{(\underline{r}_{ij}\cdot\hat{\underline{u}}_{kn})\hat{u}_{kn}^\alpha + (\underline{r}_{ij}\cdot\hat{\underline{v}}_{kn})\hat{v}_{kn}^\alpha\}\right.\right. \\ & +(\delta_{\ell k}-\delta_{\ell i})\frac{\underline{r}_{ij}\cdot\hat{\underline{u}}_{kn}}{u_{kn}r_{ik}}\left\{r_{ij}^\alpha - (\underline{r}_{ij}\cdot\hat{\underline{u}}_{kn})\hat{u}_{kn}^\alpha - (\underline{r}_{ij}\cdot\underline{r}_{ik} - (\underline{r}_{ij}\cdot\hat{\underline{u}}_{kn})(\underline{r}_{ik}\cdot\hat{\underline{u}}_{kn}))\frac{r_{ik}^\alpha}{r_{ik}^2}\right\} \\ & +(\delta_{\ell k}-\delta_{\ell i})\frac{\underline{r}_{ij}\cdot\hat{\underline{v}}_{kn}}{v_{kn}r_{ik}}\left\{r_{ij}^\alpha - (\underline{r}_{ij}\cdot\hat{\underline{v}}_{kn})\hat{v}_{kn}^\alpha - (\underline{r}_{ij}\cdot\underline{r}_{ik} - (\underline{r}_{ij}\cdot\hat{\underline{v}}_{kn})(\underline{r}_{ik}\cdot\hat{\underline{v}}_{kn}))\frac{r_{ik}^\alpha}{r_{ik}^2}\right\} \\ & +(\delta_{\ell n}-\delta_{\ell i})\frac{\underline{r}_{ij}\cdot\hat{\underline{u}}_{kn}}{u_{kn}r_{in}}\left\{r_{ij}^\alpha - (\underline{r}_{ij}\cdot\hat{\underline{u}}_{kn})\hat{u}_{kn}^\alpha - (\underline{r}_{ij}\cdot\underline{r}_{in} - (\underline{r}_{ij}\cdot\hat{\underline{u}}_{kn})(\underline{r}_{in}\cdot\hat{\underline{u}}_{kn}))\frac{r_{in}^\alpha}{r_{in}^2}\right\} \\ & \left.\left. -(\delta_{\ell n}-\delta_{\ell i})\frac{\underline{r}_{ij}\cdot\hat{\underline{v}}_{kn}}{v_{kn}r_{in}}\left\{r_{ij}^\alpha - (\underline{r}_{ij}\cdot\hat{\underline{v}}_{kn})\hat{v}_{kn}^\alpha - (\underline{r}_{ij}\cdot\underline{r}_{in} - (\underline{r}_{ij}\cdot\hat{\underline{v}}_{kn})(\underline{r}_{in}\cdot\hat{\underline{v}}_{kn}))\frac{r_{in}^\alpha}{r_{in}^2}\right\}\right]\right\} \quad . \end{aligned} \tag{2.68}$$

This general formula applies to all atoms $\ell = i, j, k, n$. It must be remembered however, that these formulae apply to just one of the three contributing terms (i.e. one angle ϕ) of the full inversion potential: specifically the inversion angle pertaining to the out-of-plane vector $\underline{r}_{ij}$. The contributions arising from the other vectors $\underline{r}_{ik}$ and $\underline{r}_{in}$ are obtained by the cyclic permutation of the indices in the manner described above. All these force contributions must be added to the final atomic forces.

Formally, the contribution to be added to the atomic virial is given by

$$\mathcal{W} = -\sum_{i=1}^{4} \underline{r}_i \cdot \underline{f}_i \quad . \tag{2.69}$$

However, it is possible to show by thermodynamic arguments (*cf* [40],) or simply from the fact that the sum of forces on atoms j,k and n is equal and opposite to the force on atom i, that the inversion potential makes *no* contribution to the atomic virial.

If the force components f_ℓ^α for atoms $\ell = i, j, k, n$ are calculated using the above formulae, it is easily seen that the contribution to be added to the atomic stress tensor is given by

$$\sigma^{\alpha\beta} = r_{ij}^\alpha f_j^\beta + r_{ik}^\alpha f_k^\beta + r_{in}^\alpha f_n^\beta \quad . \tag{2.70}$$

The sum of the diagonal elements of the stress tensor is zero (since the virial is zero) and the matrix is symmetric.

In DL_POLY_4 inversion forces are handled by the routine INVERSIONS_FORCES.

2.2.9 The Calcite Four-Body Potential

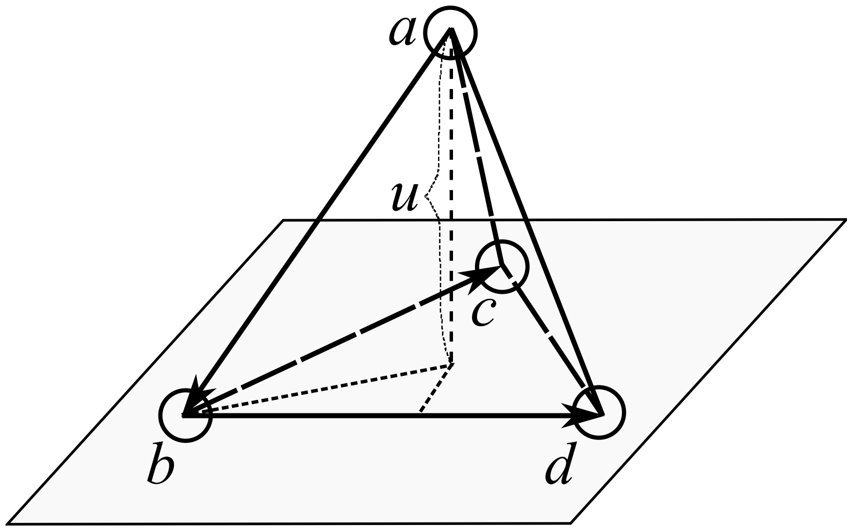


Figure 2.6: The vectors of the calcite potential

This potential [43, 44] is designed to help maintain the planar structure of the carbonate anion $[CO_3]^{2-}$ in a similar manner to the planar inversion potential described above. However, it is *not* an angular potential. It is dependent on the perpendicular displacement (u) of an atom a from a plane defined by three other atoms b, c, and d (see Figure 2.6) and has the form:

$$U_{abcd}(u) = Au^2 + Bu^4 \quad , \tag{2.71}$$

where the displacement u is given by

$$u = \frac{\underline{r}_{ab} \cdot \underline{r}_{bc} \times \underline{r}_{bd}}{|\underline{r}_{bc} \times \underline{r}_{bd}|} \quad . \tag{2.72}$$

Vectors $\underline{r}_{ab}$,$\underline{r}_{ac}$ and $\underline{r}_{ad}$ define bonds between the central atom a and the peripheral atoms b, c and d. Vectors $\underline{r}_{bc}$ and $\underline{r}_{bd}$ define the plane and are related to the bond vectors by

$$\begin{aligned} \underline{r}_{bc} &= \underline{r}_{ac} - \underline{r}_{ab} \\ \underline{r}_{bd} &= \underline{r}_{ad} - \underline{r}_{ab} \quad . \end{aligned} \tag{2.73}$$

In what follows it is convenient to define the vector product appearing in both the numerator and denominator of equation (2.72) as the vector $\underline{w}_{cd}$ *vis.*

$$\underline{w}_{cd} = \underline{r}_{bc} \times \underline{r}_{bd} \quad . \tag{2.74}$$

We also define the quantity $\gamma(u)$ as

$$\gamma(u) = -(2Au + 4Bu^3) \quad . \tag{2.75}$$

The forces on the individual atoms due to the calcite potential are then given by

$$\begin{aligned} \underline{f}_a &= -\gamma(u)\ \underline{\hat{w}}_{cd} \\ \underline{f}_c &= \quad \underline{r}_{bd} \times (\underline{r}_{ab} - u\underline{\hat{w}}_{cd})\ \gamma(u)/w_{cd} \\ \underline{f}_d &= -\underline{r}_{bc} \times (\underline{r}_{ab} - u\underline{\hat{w}}_{cd})\ \gamma(u)/w_{cd} \\ \underline{f}_b &= -(\underline{f}_a + \underline{f}_c + \underline{f}_d) \quad , \end{aligned} \tag{2.76}$$

where $w_{cd} = |\underline{w}_{cd}|$ and $\underline{\hat{w}}_{cd} = \underline{w}_{cd}/w_{cd}$. The virial contribution $\psi_{abcd}(u)$ is given by

$$\psi_{abcd}(u) = 2Au^2 + 4Bu^4 \tag{2.77}$$

and the stress tensor contribution $\sigma^{\alpha\beta}_{abcd}(u)$ by

$$\sigma^{\alpha\beta}_{abcd}(u) = \frac{u\ \gamma(u)}{w^2_{cd}}\ w^{\alpha}_{cd}\ w^{\beta}_{cd} \quad . \tag{2.78}$$

In DL_POLY_4 the calcite forces are handled by the routine INVERSIONS_FORCES, which is a convenient *intramolecular* four-body force routine. However, it is manifestly *not* an inversion potential as such.

2.2.10 Inversional Restraints

In DL_POLY_4 the inversional restraints, in which the inversion angle, as defined by a quadruplet of atoms, is maintained around some preset value ϕ_0, is handled as a special case of inversion potential. As a consequence angle restraints may be applied only between atoms in the same molecule. Unlike with application of the "pure" dihedral potentials, the electrostatic and van der Waals interactions between the pair of atoms are still evaluated when distance restraints are applied. All the potential forms of the previous section are available as torsional restraints, although they have different key words:

1. Harmonic: (**-hrm**)
2. Harmonic cosine: (**-hcs**)
3. Planar potential: (**-pln**)
4. Extended planar potential: (**-xpl**)
5. Tabulated potential: (**-tab**). The potential is defined numerically in TABINV (see Section 4.3 and Section 10.1.9).

In DL_POLY_4 inversional restraints are handled by the routine INVERSIONS_FORCES.

2.2.11 Tethering Forces

DL_POLY_4 also allows atomic sites to be tethered to a fixed point in space, $\vec{r_0}$, taken as their position at the beginning of the simulation (t = 0). This is also known as position restraining. The specification, which comes as part of the molecular description, requires a tether potential type and the associated interaction parameters.

Note, firstly, that application of tethering potentials means that the momentum will no longer be a conserved quantity of the simulation. Secondly, in constant pressure simulations, where the MD cell changes size or shape, the tethers' reference positions are scaled with the cell vectors.

The tethering potential functions available in DL_POLY_4 are as follows:

1. Harmonic: (**harm**)
$$U(r_{ij}) = \frac{1}{2}k(r_{i0})^2 \tag{2.79}$$

2. Restrained harmonic: (**rhrm**)
$$U(r_{ij}) = \left\{ \begin{array}{lcl} \frac{1}{2}k(r_{i0})^2 & : & |r_{i0}| \le r_c \\ \frac{1}{2}kr_c^2 + kr_c(r_{i0} - r_c) & : & |r_{i0}| > r_c \end{array} \right. \tag{2.80}$$

3. Quartic potential: (**quar**)
$$U(r_{ij}) = \frac{k}{2}(r_{i0})^2 + \frac{k'}{3}(r_{i0})^3 + \frac{k''}{4}(r_{i0})^4 \tag{2.81}$$

as in each case r_{io} is the distance between the atom positions at moment $t = t1$ and $t = 0$.

The force on the atom i arising from a tether potential potential is obtained using the general formula:

$$\underline{f}_i = -\frac{1}{r_{i0}} \left[\frac{\partial}{\partial r_{i0}} U(r_{i0}) \right] \underline{r}_{i0} \quad . \tag{2.82}$$

The contribution to be added to the atomic virial is given by

$$\mathcal{W} = \underline{r}_{i0} \cdot \underline{f}_i \quad . \tag{2.83}$$

The contribution to be added to the atomic stress tensor is given by

$$\sigma^{\alpha\beta} = -r_{i0}^{\alpha} f_i^{\beta} \quad , \tag{2.84}$$

where α and β indicate the x, y, z components. The atomic stress tensor derived in this way is symmetric.

In DL_POLY_4 tether forces are handled by the routine TETHERS_FORCES.

2.3 The Intermolecular Potential Functions

In this section we outline the two-body, metal, Tersoff, three-body and four-body potential functions in DL_POLY_4. An important distinction between these and intramolecular (bond) forces in DL_POLY_4 is that they are specified by *atom types* rather than atom indices.

2.3.1 Short Ranged (van der Waals) Potentials

The short ranged pair forces available in DL_POLY_4 are as follows:

1. 12-6 potential: (**12-6**)
$$U(r_{ij}) = \left(\frac{A}{r_{ij}^{12}}\right) - \left(\frac{B}{r_{ij}^{6}}\right) \tag{2.85}$$

2. Lennard-Jones potential: (**lj**)
$$U(r_{ij}) = 4\epsilon\left[\left(\frac{\sigma}{r_{ij}}\right)^{12} - \left(\frac{\sigma}{r_{ij}}\right)^{6}\right] \tag{2.86}$$

3. Lennard-Jones cohesive potential [45]: (**ljc**)
$$U(r_{ij}) = 4\epsilon\left[\left(\frac{\sigma}{r_{ij}}\right)^{12} - c_{ij}\left(\frac{\sigma}{r_{ij}}\right)^{6}\right] \tag{2.87}$$
This potential has an extra constant to tune the attractive part of the potential serving to describe the different cohesiveness between different fluids and surfaces in engineering flows models.

4. Lennard Jones, generalised by Frenkel et al. [46]: (**ljf**)
$$U(r_{ij}) = \begin{cases} \varepsilon_{AB}\alpha_{AB}\left[\left(\frac{\sigma_{AB}}{r_{ij}}\right)^2 - 1\right]\left[\left(\frac{r^c_{AB}}{r_{ij}}\right)^2 - 1\right]^2 & r \le r^c_{AB} \\ 0 & r > r^c_{AB} \end{cases} \tag{2.88}$$
with
$$\alpha_{AB} = \frac{1}{4}\left(\frac{r^c_{AB}}{\sigma_{AB}}\right)^2\left(\frac{3}{\left(\frac{r^c_{AB}}{\sigma_{AB}}\right)^2 - 1}\right)^3 \tag{2.89}$$

5. n-m potential (aka Mie) [47, 48]: (**nm**)
$$U(r_{ij}) = \frac{E_o}{(n-m)}\left[m\left(\frac{r_o}{r_{ij}}\right)^n - n\left(\frac{r_o}{r_{ij}}\right)^m\right] \tag{2.90}$$

6. Buckingham potential: (**buck**)
$$U(r_{ij}) = A\ \exp\left(-\frac{r_{ij}}{\rho}\right) - \frac{C}{r_{ij}^6} \tag{2.91}$$

7. Born-Huggins-Meyer potential: (**bhm**)
$$U(r_{ij}) = A\ \exp[B(\sigma - r_{ij})] - \frac{C}{r_{ij}^6} - \frac{D}{r_{ij}^8} \tag{2.92}$$

8. Hydrogen-bond (12-10) potential: (**hbnd**)
$$U(r_{ij}) = \left(\frac{A}{r_{ij}^{12}}\right) - \left(\frac{B}{r_{ij}^{10}}\right) \tag{2.93}$$

9. Shifted force n-m potential (aka Mie) [47, 48]: (**snm**)
$$\begin{aligned} U(r_{ij}) &= \frac{\alpha E_o}{(n-m)}\left[m\beta^n\left\{\left(\frac{r_o}{r_{ij}}\right)^n - \left(\frac{1}{\gamma}\right)^n\right\} - n\beta^m\left\{\left(\frac{r_o}{r_{ij}}\right)^m - \left(\frac{1}{\gamma}\right)^m\right\}\right] + \\ & \frac{nm\alpha E_o}{(n-m)}\left(\frac{r_{ij} - \gamma r_o}{\gamma r_o}\right)\left\{\left(\frac{\beta}{\gamma}\right)^n - \left(\frac{\beta}{\gamma}\right)^m\right\} \quad , \end{aligned} \tag{2.94}$$

with

$$\begin{aligned}\alpha &= \frac{(n-m)}{[n\beta^m(1+(m/\gamma-m-1)/\gamma^m)-m\beta^n(1+(n/\gamma-n-1)/\gamma^n)]}\\ \beta &= \gamma\left(\frac{\gamma^{m+1}-1}{\gamma^{n+1}-1}\right)^{\frac{1}{n-m}}\\ \gamma &= \frac{r_{\mathrm{cut}}}{r_o}\ .\end{aligned}\tag{2.95}$$

This peculiar form has the advantage over the standard shifted n-m potential in that both E_o and r_0 (well depth and location of minimum) retain their original values after the shifting process.

10. Morse potential: (**mors**)

$$U(r_{ij}) = E_o\ [\{1-\exp(-k(r_{ij}-r_o))\}^2-1]\tag{2.96}$$

11. Shifted Weeks-Chandler-Andersen (WCA) potential [49]: (**wca**)

$$U(r_{ij}) = \begin{cases}4\epsilon\left[\left(\frac{\sigma}{r_{ij}-\Delta}\right)^{12}-\left(\frac{\sigma}{r_{ij}-\Delta}\right)^{6}\right]+\epsilon & : \quad r_{ij}<2^{\frac{1}{6}}\ \sigma+\Delta\\ 0 & : \quad r_{ij}\ge 2^{\frac{1}{6}}\ \sigma+\Delta\end{cases}\tag{2.97}$$

The WCA potential is the Lennard-Jones potential truncated at the position of the minimum and shifted to eliminate discontinuity (includes the effect of excluded volume). It is usually used in combination with the FENE, equation (2.10), bond potential. This implementation allows for a radius shift of up to half a σ ($|\Delta| \le 0.5\ \sigma$) with a default of zero ($\Delta_{default}=0$).

12. Standard DPD potential: (**dpd**)

$$U(r_{ij}) = \begin{cases}\frac{A}{2}\ r_c\ \left(1-\frac{r_{ij}}{r_c}\right)^2 & : \quad r_{ij}<r_c\\ 0 & : \quad r_{ij}\ge r_c\end{cases}\tag{2.98}$$

It takes the Groot-Warren [50] form giving a soft and purely repulsive interaction.

13. 14-7 pair potential [51]: (**14-7**)

$$U(r_{ij}) = \epsilon\left(\frac{1.07}{(r_{ij}/r_o)+0.07}\right)^7\left(\frac{1.12}{(r_{ij}/r_o)^7+0.12}-2\right)\tag{2.99}$$

14. Morse modified [52]: (**mstw**)

$$U\left(r_{ij}\right) = E_0\ \{[1-\exp\left(-k\left(r_{ij}-r_0\right)\right)]^2-1\}+\frac{c}{r_{ij}^{12}}\tag{2.100}$$

15. Rydberg: (**ryd**)

$$U\left(r_{ij}\right) = \left(a+br_{ij}\right)\exp\left(-r_{ij}/\rho\right)\tag{2.101}$$

16. Ziegler-Biersack-Littmark (ZBL): [53] (**zbl**)

$$U(r_{ij}) = \frac{Z_1Z_2e^2}{4\pi\varepsilon_0\varepsilon_r}\sum_{i=1}^{4}b_i\exp\left(-c_ir/a\right)\quad ,\tag{2.102}$$

where

$$\begin{aligned}a &= \frac{0.88534\cdot a_B}{Z_1^{0.23}+Z_2^{0.23}}\\ b &= [0.18175, 0.50986, 0.28022, 0.02817]\\ c &= [3.1998, 0.94229, 0.40290, 0.20162]\\ a_B &= 0.52917721067\ \text{Å}\quad .\end{aligned}$$

17. ZBL mixed with Morse, [54]: (**zbls**)

$$U\left(r_{ij}\right) = f\left(r_{ij}\right) U_{ZBL}\left(r_{ij}\right) + \left(1 - f\left(r_{ij}\right)\right) U_{morse}\left(r_{ij}\right) \quad , \tag{2.103}$$

with $f(r)$ defined by

$$f(r) = \begin{cases} 1 - e^{-(r_m - r)/\xi}/2 & : \quad r < r_m \\ e^{-(r-r_m)/\xi}/2 & : \quad r \geq r_m \quad . \end{cases}$$

18. ZBL mixed with Buckingham, [54]: (**zblb**)

$$U\left(r_{ij}\right) = f\left(r_{ij}\right) U_{ZBL}\left(r_{ij}\right) + \left(1 - f\left(r_{ij}\right)\right) U_{buckingham}\left(r_{ij}\right) \quad , \tag{2.104}$$

with $f(r)$ defined by

$$f(r) = \begin{cases} 1 - e^{-(r_m - r)/\xi}/2 & : \quad r < r_m \\ e^{-(r-r_m)/\xi}/2 & : \quad r \geq r_m \quad . \end{cases}$$

19. Lennard-Jones tapered with Mei-Davenport-Fernando taper (MDF), [55]: (**mlj**)

$$U\left(r_{ij}\right) = f\left(r_{ij}\right) U_{LJ}\left(r_{ij}\right) \quad , \tag{2.105}$$

where

$$f(r) = \begin{cases} 1 & : \quad r < r_i \\ \frac{(r_c - r)^3\left(10r_i^2 - 5r_c r_i - 15 r r_i + r_c^2 + 3 r r_c + 6r^2\right)}{(r_c - r_i)^5} & : \quad r_i \leq r \leq r_c \\ 0 & : \quad r > r_c \quad . \end{cases} \tag{2.106}$$

r_c is set to r_{vdw} and controled by **rvdw**.

20. Buckingham tapered with MDF: (**mbuc**)

$$U\left(r_{ij}\right) = f\left(r_{ij}\right) U_{Buckingham}\left(r_{ij}\right) \tag{2.107}$$

See **mlj** for more details.

21. 12-6 Lennard-Jones tapered with Mei-Davenport-Fernando taper (MDF): (**m126**)

$$U\left(r_{ij}\right) = f\left(r_{ij}\right) U_{12-6}\left(r_{ij}\right) \tag{2.108}$$

See **mlj** for more details.

22. Tabulation: (**tab**). The potential is defined numerically only.

The parameters defining these potentials are supplied to DL_POLY_4 at run time (see the description of the FIELD file in Section 10.1.3). Each atom type in the system is specified by a unique eight-character label defined by the user. The pair potential is then defined internally by the combination of two atom labels.

It is worth noting that some potentials are implemented in an extended form from their original reference specification. Often this is done by replacing the r argument by $r - r_o$ to define a surface softness/hardness width/radius.

As well as the numerical parameters defining the potentials, DL_POLY_4 should also be provided with a cutoff radius, $r_{\rm vdw}$, which sets a range limit on the computation of the interactions. It is worth noting that some interaction come with a hard-wired cutoff in their parameter sets! Thus any provided cutoff radius, $r_{\rm vdw}$, will be reset if it is not equal or larger that the largest of these all. Together with the parameters, the cutoff is used by the subroutine VDW_GENERATE to construct an interpolation array `vvdw` for the potential function over the range 0 to $r_{\rm vdw}$. A second array `gvdw` is also calculated, which is related to the potential via the formula:

$$G(r_{ij}) = -r_{ij}\frac{\partial}{\partial r_{ij}}U(r_{ij}) \quad , \tag{2.109}$$

and is used in the calculation of the forces. Both arrays are tabulated in units of energy. The use of interpolation arrays, rather than the explicit formulae, makes the routines for calculating the potential energy and atomic forces very general, and enables the use of user defined pair potential functions. DL_POLY_4 also allows the user to read in the interpolation arrays directly from a file (implemented in the VDW_TABLE_READ routine) and the TABLE file (Section 10.1.7). This is particularly useful if the pair potential function has no simple analytical description (e.g. spline potentials).

The force on an atom j derived from one of these potentials is formally calculated with the standard formula:

$$\underline{f}_j = -\frac{1}{r_{ij}}\left[\frac{\partial}{\partial r_{ij}}U(r_{ij})\right]\underline{r}_{ij} \quad , \tag{2.110}$$

where $\underline{r}_{ij} = \underline{r}_j - \underline{r}_i$. The force on atom i is the negative of this.

The contribution to be added to the atomic virial (for each pair interaction) is

$$\mathcal{W} = -\underline{r}_{ij}\cdot\underline{f}_j \quad . \tag{2.111}$$

The contribution to be added to the atomic stress tensor is given by

$$\sigma^{\alpha\beta} = r_{ij}^{\alpha} f_j^{\beta} \quad , \tag{2.112}$$

where α and β indicate the x, y, z components. The atomic stress tensor derived from the pair forces is symmetric.

Since the calculation of pair potentials assumes a spherical cutoff ($r_{\rm vdw}$) it is necessary to apply a *long-ranged correction* to the system potential energy and virial. Explicit formulae are needed for each case and are derived as follows. For two atom types a and b, the correction for the potential energy is calculated via the integral

$$U_{corr}^{ab} = 2\pi\frac{N_a N_b}{V}\int_{r_{\rm vdw}}^{\infty} g_{ab}(r)U_{ab}(r)r^2 dr \quad , \tag{2.113}$$

where N_a, N_b are the numbers of atoms of types a and b in the system, V is the system volume and $g_{ab}(r)$ and $U_{ab}(r)$ are the appropriate pair correlation function and pair potential respectively. It is usual to assume $g_{ab}(r) = 1$ for $r > r_{\rm vdw}$. DL_POLY_4 sometimes makes the additional assumption that the repulsive part of the short ranged potential is negligible beyond $r_{\rm vdw}$.

The correction for the system virial is

$$\mathcal{W}_{corr}^{ab} = -2\pi\frac{N_a N_b}{V}\int_{r_{\rm vdw}}^{\infty} g_{ab}(r)\frac{\partial}{\partial r}U_{ab}(r)r^3 dr \quad , \tag{2.114}$$

where the same approximations are applied.

Note that these formulae are based on the assumption that the system is reasonably isotropic beyond the cutoff. It is worth noting that the 14-7 pair potential's corrections to system energy and virial are solved numerically.

In DL_POLY_4 the short ranged forces are calculated by the subroutine VDW_FORCES. The long-ranged corrections are calculated by routine VDW_LRC. The calculation makes use of the Verlet neighbour list (see above).

Notes on mixing rules for short-ranged interactions

DL_POLY_4 allows a short cut for mixing some of the explicitly specified pair interactions for single species of the same type so that cross-species interactions are generated if unspecified. This is only possible for the **12-6, lj, dpd, 14-7, wca & ljc** types. The mixing is derived from the Lennard-Jones style characteristic paramteres for energy (ϵ) and distance (σ or r_0) terms. The available types of mixing within DL_POLY_4 are borrowed from [56]. The rules' names and formulae are as follows:

1. Lorentz-Berthelot
$$\epsilon_{ij} = \sqrt{\epsilon_i \ \epsilon_j} \ \ ; \ \ \sigma_{ij} = \frac{\sigma_i + \sigma_j}{2} \tag{2.115}$$

2. Fender-Halsey
$$\epsilon_{ij} = 2\frac{\epsilon_i \ \epsilon_j}{\epsilon_i + \epsilon_j} \ \ ; \ \ \sigma_{ij} = \frac{\sigma_i + \sigma_j}{2} \tag{2.116}$$

3. Hogervorst (good hope)
$$\epsilon_{ij} = \sqrt{\epsilon_i \ \epsilon_j} \ \ ; \ \ \sigma_{ij} = \sqrt{\sigma_i \ \sigma_j} \tag{2.117}$$

4. Halgren HHG
$$\epsilon_{ij} = 4\frac{\epsilon_i \ \epsilon_j}{\left(\epsilon_i^{1/2} + \epsilon_j^{1/2}\right)^2} \ \ ; \ \ \sigma_{ij} = \frac{\sigma_i^3 + \sigma_j^3}{\sigma_i^2 + \sigma_j^2} \tag{2.118}$$

5. Waldman-Hagler
$$\epsilon_{ij} = 2\sqrt{\epsilon_i \ \epsilon_j}\frac{(\sigma_i \ \sigma_j)^3}{\sigma_i^6 + \sigma_j^6} \ \ ; \ \ \sigma_{ij} = \left(\frac{\sigma_i^6 + \sigma_j^6}{2}\right)^{\frac{1}{6}} \tag{2.119}$$

6. Tang-Toennies
$$\epsilon_{ij}\sigma_{ij}^6 = \sqrt{\epsilon_i\sigma_i^6 \ \epsilon_j\sigma_j^6} \ \ ; \ \ \epsilon_{ij}\sigma_{ij}^{12} = \left[\frac{\left(\epsilon_i\sigma_i^{12}\right)^{13} + \left(\epsilon_j\sigma_j^{12}\right)^{13}}{2}\right]^{13} \tag{2.120}$$

7. Functional
$$\epsilon_{ij} = \frac{3 \ \sqrt{\epsilon_i \ \epsilon_j} \ (\sigma_i \ \sigma_j)^3}{\sum_{L=0}^{2}\left[\frac{\left(\sigma_i^3+\sigma_j^3\right)^2}{4 \ (\sigma_i \ \sigma_i)^L}\right]^{\frac{6}{6-2L}}} \ \ ; \ \ \sigma_{ij} = \frac{1}{3}\sum_{L=0}^{2}\left[\frac{\left(\sigma_i^3 + \sigma_j^3\right)^2}{4 \ (\sigma_i \ \sigma_i)^L}\right]^{\frac{1}{6-2L}} \tag{2.121}$$

It is woth noting that the i and j symbols in the equations for mixing denote atom types (species) and the indices for the same species interaction parameters are contracted to a single species index for simplicity.

2.3.2 Metal Potentials

The metal potentials in DL_POLY_4 follow two similar but distinct formalisms. The first of these is the embedded atom model (EAM) [11, 12] and the second is the Finnis-Sinclair model (FS) [13]. Both are density dependent potentials derived from density functional theory (DFT) and describe the bonding of a metal atom ultimately in terms of the local electronic density. They are suitable for calculating the properties of metals and metal alloys. The extended EAM (EEAM) [57, 58] is a generalisation of the EAM formalism which can include both EAM and FS type of mixing rules (see below).

It is worth noting that the same formalism applies to the many-body perturbation component of the actinide oxide potentials as in [59]. Thus their many-body component description is included in this Section.

For single component metals the two main approaches, FS and EAM, are the same. **However**, they are subtly different in the way they are extended to handle alloys (see below). It follows that EAM and FS class potentials cannot be mixed in a single simulation. Furthermore, even for FS class potentials possessing different analytical forms there is no agreed procedure for mixing the parameters. Mixing EAM and EEAM potentials is only possible if the EAM ones are generalised to EEAM form (see below). The user is, therefore, strongly advised to be consistent in the choice of potential when modelling alloys.

The general form of the EAM and FS types of potentials is [60]

$$U_{metal} = \frac{1}{2}\sum_{i=1}^{N}\sum_{j\neq i}^{N} V_{ij}(r_{ij}) + \sum_{i=1}^{N} F(\rho_i) \quad , \tag{2.122}$$

where $F(\rho_i)$ is a functional describing the energy of embedding an atom in the bulk density, ρ_i, which is defined as

$$\rho_i = \sum_{j=1,j\neq i}^{N} \rho_{ij}(r_{ij}) \quad . \tag{2.123}$$

It should be noted that the density is determined by the coordination number of the atom defined by *pairs* of atoms. This makes the metal potential dependent on the local density (environmental). $V_{ij}(r_{ij})$ is a pair potential incorporating repulsive electrostatic and overlap interactions. N is the number of interacting particles in the MD box.

In DL_POLY_4 EAM and thus EEAM can be further generalised to include two-band (2B) densities [61, 62], for s- and d-bands,

$$F(\rho_i) = F^s(\rho_i^s) + F^d(\rho_i^d) \quad , \tag{2.124}$$

where

$$\rho_i^q = \sum_{j=1,j\neq i}^{N} \rho_{ij}^q(r_{ij}) \ , \quad q = s, d \quad , \tag{2.125}$$

instead of just the one, s, as in equations (2.122) and (2.123). These will be referred in the following text as 2BEAM and 2BEEAM. Mixing 2BEAM and EAM and alternatively 2BEEAM and EEAM potentials is only possible if the single band ones are generalised to 2B forms. The user is, again, reminded to be consistent in the choice of potential when modelling alloys.

The types of metal potentials available in DL_POLY_4 are as follows:

1. EAM potential: (**eam**) There are no explicit mathematical expressions for EAM potentials, so this potential type is read exclusively in the form of interpolation arrays from the TABEAM table file (as implemented in the METAL_TABLE_READ routine - Section 10.1.8.) The rules for combining the potentials from different metals to handle alloys are different from the FS class of potentials (see below).

2. EEAM potential (**eeam**) Similar to EAM above, it is given in the form of interpolation arrays from the TABEAM file, but the rules for combining the potentials from different metals are different from both EAM and FS classes (see below).

3. 2BEAM potential (**2beam**) Similar to EEAM for the s density terms and to EAM for the d ones. It is and given in the form of interpolation arrays from the TABEAM file, but the rules for combining the potentials from different metals are different from both EAM, EEAM and FS classes (see below).

4. 2BEEAM potential (**2beeam**) Similar to EEAM for both s and d density terms. It is and given in the form of interpolation arrays from the TABEAM file, but the rules for combining the potentials from different metals are different from both EAM, EEAM, 2BEAM and FS classes (see below).

5. Finnis-Sinclair potential [13]: (**fnsc**) Finnis-Sinclair potential is explicitly analytical. It has the following form:

$$\begin{aligned} V_{ij}(r_{ij}) &= \begin{cases} (r_{ij}-c)^2(c_0+c_1 r_{ij}+c_2 r_{ij}^2) & : \; r_{ij} < c \\ 0 & : \; r_{ij} > c \end{cases} \\ \rho_{ij}(r_{ij}) &= \begin{cases} (r_{ij}-d)^2+\beta\dfrac{(r_{ij}-d)^3}{d} & : \; r_{ij} < d \\ 0 & : \; r_{ij} > d \end{cases} \\ F(\rho_i) &= -A\sqrt{\rho_i} \quad , \end{aligned} \tag{2.126}$$

with parameters: c_0, c_1, c_2, c, A, d, β, both c and d are cutoffs. Since first being proposed a number of alternative analytical forms have been proposed, some of which are described below. The rules for combining different metal potentials to model alloys are different from the EAM potentials (see below).

6. Extended Finnis-Sinclair potential [63]: (**exfs**) It has the following form:

$$\begin{aligned} V_{ij}(r_{ij}) &= \begin{cases} (r_{ij}-c)^2(c_0+c_1 r_{ij}+c_2 r_{ij}^2+c_3 r_{ij}^3+c_4 r_{ij}^4) & : \; r_{ij} < c \\ 0 & : \; r_{ij} > c \end{cases} \\ \rho_{ij}(r_{ij}) &= \begin{cases} (r_{ij}-d)^2+B^2(r_{ij}-d)^4 & : \; r_{ij} < d \\ 0 & : \; r_{ij} > d \end{cases} \\ F(\rho_i) &= -A\sqrt{\rho_i} \quad , \end{aligned} \tag{2.127}$$

with parameters: c_0, c_1, c_2, c_3, c_4, c, A, d, B, both c and d are cutoffs.

7. Sutton-Chen potential [14, 15, 16]: (**stch**) The Sutton Chen potential is an analytical potential in the FS class. It has the form:

$$\begin{aligned} V_{ij}(r_{ij}) &= \epsilon\left(\frac{a}{r_{ij}}\right)^n \\ \rho_{ij}(r_{ij}) &= \left(\frac{a}{r_{ij}}\right)^m \\ F(\rho_i) &= -c\epsilon\sqrt{\rho_i} \quad , \end{aligned} \tag{2.128}$$

with parameters: ϵ, a, n, m, c. **Note** that the parameter c for the mixed potential in multi-component allys is irrelevant as outlined in [15]!

8. Gupta potential [64]: (**gupt**) The Gupta potential is another analytical potential in the FS class. It has the form:

$$\begin{aligned} V_{ij}(r_{ij}) &= 2A\exp\left(-p\frac{r_{ij}-r_0}{r_0}\right) \\ \rho_{ij}(r_{ij}) &= \exp\left(-2q_{ij}\frac{r_{ij}-r_0}{r_0}\right) \\ F(\rho_i) &= -B\sqrt{\rho_i} \quad , \end{aligned} \tag{2.129}$$

with parameters: A, r_0, p, B, q_{ij}.

9. Many body perturbation component potential [59]: (**mbpc**) This component is another analytical potential in the FS class which two body part may be defined by a matching van der Waals potential in the vdw section of the FIELD file. It has the form:

$$\begin{aligned} V_{ij}(r_{ij}) &= 0 \\ \rho_{ij}(r_{ij}) &= \left(\frac{a}{r_{ij}^m}\right)\frac{1}{2}\left[1+\mathrm{erf}\left(\alpha(r_{ij}-r_o)\right)\right] \\ F(\rho_i) &= -\epsilon\sqrt{\rho_i} \quad , \end{aligned} \tag{2.130}$$

with parameters: ϵ, a, m, α and r_o.

Note that the parameters α and r_o must be the same for all defined potentials of this type. DL_POLY_4 will set $\alpha = \mathrm{Max}(0, \alpha_{pq})$ and $r_\mathrm{o} = \mathrm{Max}(0, r_{\mathrm{o}\_pq})$ for all defined interactions of this type between species p and q. If after this any is left undefined, i.e. zero, the undefined entities will be set to their defaults: $\alpha = 20$ and $r_\mathrm{o} = \mathrm{Min}(1.5, 0.2\ r_\mathrm{cut})$.

All of these metal potentials can be decomposed into pair contributions and thus fit within the general tabulation scheme of DL_POLY_4, where they are treated as pair interactions (though note that the metal cutoff, r_met has nothing to do with short ranged cutoff, r_vdw). DL_POLY_4 calculates this potential in two stages: the first calculates the local density, ρ_i, for each atom; and the second calculates the potential energy and forces. Interpolation arrays, `vmet`, `gmet` and `fmet` (METAL_GENERATE, METAL_TABLE_READ) are used in both these stages in the same spirit as in the van der Waals interaction calculations.

The total force $\underline{f}_k^{tot}$ on an atom k derived from this potential is calculated in the standard way:

$$\underline{f}_k^{tot} = -\underline{\nabla}_k U_{metal} \quad . \tag{2.131}$$

We rewrite the EAM/FS potential, equation (2.122), as

$$\begin{aligned} U_{metal} &= U_1 + U_2 \\ U_1 &= \frac{1}{2}\sum_{i=1}^{N}\sum_{j\neq i}^{N} V_{ij}(r_{ij}) \\ U_2 &= \sum_{i=1}^{N} F(\rho_i) \quad , \end{aligned} \tag{2.132}$$

where $\underline{r}_{ij} = \underline{r}_j - \underline{r}_i$. The force on atom k is the sum of the derivatives of U_1 and U_2 with respect to $\underline{r}_k$, which is recognisable as a sum of pair forces:

$$\begin{aligned} -\frac{\partial U_1}{\partial \underline{r}_k} &= -\frac{1}{2}\sum_{i=1}^{N}\sum_{j\neq i}^{N} \frac{\partial V_{ij}(r_{ij})}{\partial r_{ij}} \frac{\partial r_{ij}}{\partial \underline{r}_k} = \sum_{j=1,j\neq k}^{N} \frac{\partial V_{kj}(r_{kj})}{\partial r_{kj}} \frac{\underline{\underline{r}}_{kj}}{r_{kj}} \\ -\frac{\partial U_2}{\partial \underline{r}_k} &= -\sum_{i=1}^{N} \frac{\partial F}{\partial \rho_i} \sum_{j\neq i}^{N} \frac{\partial \rho_{ij}(r_{ij})}{\partial r_{ij}} \frac{\partial r_{ij}}{\partial \underline{r}_k} \\ &= -\sum_{i=1,i\neq k}^{N} \frac{\partial F}{\partial \rho_i} \frac{\partial \rho_{ik}(r_{ik})}{\partial r_{ik}} \frac{\partial r_{ik}}{\partial \underline{r}_k} - \sum_{j=1,j\neq k}^{N} \frac{\partial F}{\partial \rho_k} \frac{\partial \rho_{kj}(r_{kj})}{\partial r_{kj}} \frac{\partial r_{kj}}{\partial \underline{r}_k} \\ &= \sum_{j=1,j\neq k}^{N} \left(\frac{\partial F}{\partial \rho_k} + \frac{\partial F}{\partial \rho_j} \right) \frac{\partial \rho_{kj}(r_{kj})}{\partial r_{kj}} \frac{\underline{\underline{r}}_{kj}}{r_{kj}} \quad . \end{aligned} \tag{2.133}$$

1. EAM force
   The same as shown above. However, it is worth noting that the generation of the force arrays from tabulated data (implemented in the METAL_TABLE_DERIVATIVES routine) is done using a five point interpolation procedure.

2. EEAM force
   Information the same as that for EAM.

3. 2BEAM force
   Information the same as that for EAM. However, as there is a second embedding contribution from the extra band complexity: $U_2 = U_2^s + U_2^d$!

4. 2BEEAM force
   Information the same as that for EAM. However, as there is a second embedding contribution from the extra band complexity: $U_2 = U_2^s + U_2^d$!

5. Finnis-Sinclair force

$$\begin{aligned}
-\frac{\partial U_1}{\partial \underline{r_k}} &= \sum_{j=1,j\neq k}^{N} \left\{2(r_{kj}-c)(c_0+c_1 r_{kj}+c_2 r_{kj}^2)+(r_{kj}-c)^2(c_1+2c_2 r_{kj})\right\}\frac{\underline{\underline{r_{kj}}}}{r_{kj}} \\
-\frac{\partial U_2}{\partial \underline{r_k}} &= -\sum_{j=1,j\neq k}^{N} \frac{A}{2}\left(\frac{1}{\sqrt{\rho_k}}+\frac{1}{\sqrt{\rho_j}}\right)\left\{2(r_{kj}-d)+3\beta\frac{(r_{kj}-d)^2}{d}\right\}\frac{\underline{\underline{r_{kj}}}}{r_{kj}} \quad . \qquad (2.134)
\end{aligned}$$

6. Extended Finnis-Sinclair force

$$\begin{aligned}
-\frac{\partial U_1}{\partial \underline{r_k}} &= \sum_{j=1,j\neq k}^{N} \left\{2(r_{kj}-c)(c_0+c_1 r_{kj}+c_2 r_{kj}^2+c_3 r_{kj}^3+c_4 r_{kj}^4)+\right. \\
&\qquad \left.(r_{kj}-c)^2(c_1+2c_2 r_{kj}+3c_3 r_{kj}^2+4c_4 r_{kj}^3)\right\}\frac{\underline{\underline{r_{kj}}}}{r_{kj}} \qquad (2.135) \\
-\frac{\partial U_2}{\partial \underline{r_k}} &= -\sum_{j=1,j\neq k}^{N} \frac{A}{2}\left(\frac{1}{\sqrt{\rho_k}}+\frac{1}{\sqrt{\rho_j}}\right)\left\{2(r_{kj}-d)+4B^2(r_{kj}-d)^3\right\}\frac{\underline{\underline{r_{kj}}}}{r_{kj}} \quad .
\end{aligned}$$

7. Sutton-Chen force

$$\begin{aligned}
-\frac{\partial U_1}{\partial \underline{r_k}} &= -\sum_{j=1,j\neq k}^{N} n\epsilon\left(\frac{a}{r_{kj}}\right)^n\frac{\underline{\underline{r_{kj}}}}{r_{kj}} \\
-\frac{\partial U_2}{\partial \underline{r_k}} &= \sum_{j=1,j\neq k}^{N} \frac{mc\epsilon}{2}\left(\frac{1}{\sqrt{\rho_k}}+\frac{1}{\sqrt{\rho_j}}\right)\left(\frac{a}{r_{kj}}\right)^m\frac{\underline{\underline{r_{kj}}}}{r_{kj}} \quad . \qquad (2.136)
\end{aligned}$$

8. Gupta force

$$\begin{aligned}
-\frac{\partial U_1}{\partial \underline{r_k}} &= -\sum_{j=1,j\neq k}^{N} \frac{2Ap}{r_0}\exp\left(-p\frac{r_{kj}-r_0}{r_0}\right)\frac{\underline{\underline{r_{kj}}}}{r_{kj}} \\
-\frac{\partial U_2}{\partial \underline{r_k}} &= \sum_{j=1,j\neq k}^{N} \frac{Bq_{kj}}{r_0}\left(\frac{1}{\sqrt{\rho_k}}+\frac{1}{\sqrt{\rho_j}}\right)\exp\left(-2q_{kj}\frac{r_{kj}-r_0}{r_0}\right)\frac{\underline{\underline{r_{kj}}}}{r_{kj}} \quad . \qquad (2.137)
\end{aligned}$$

9. Many body perturbation component potential force

$$\begin{aligned}
-\frac{\partial U_1}{\partial \underline{r_k}} &= 0 \\
-\frac{\partial U_2}{\partial \underline{r_k}} &= \sum_{j=1,j\neq k}^{N} \frac{m\epsilon}{2}\left(\frac{1}{\sqrt{\rho_k}}+\frac{1}{\sqrt{\rho_j}}\right)\frac{a}{r_{kj}^m}\frac{\underline{\underline{r_{kj}}}}{r_{kj}} \quad . \qquad (2.138)
\end{aligned}$$

With the metal forces thus defined the contribution to be added to the atomic virial *from each atom pair* is then

$$\mathcal{W} = -\underline{r}_{ij}\cdot\underline{f}_j \quad , \qquad (2.139)$$

which equates to:

$$\begin{aligned}
\Psi &= 3V\frac{\partial U}{\partial V} \\
\Psi &= \frac{3}{2}V\sum_{i=1}^{N}\sum_{j\neq i}^{N}\frac{\partial V_{ij}(r_{ij})}{\partial r_{ij}}\frac{\partial r_{ij}}{\partial V}+3V\sum_{i=1}^{N}\frac{\partial F(\rho_i)}{\partial \rho_i}\frac{\partial \rho_i}{\partial V}=\Psi_1+\Psi_2 \\
&\frac{\partial r_{ij}}{\partial V}=\frac{\partial V^{1/3}s_{ij}}{\partial V}=\frac{1}{3}V^{-2/3}s_{ij}=\frac{r_{ij}}{3V} \\
\Psi_1 &= \frac{1}{2}\sum_{i=1}^{N}\sum_{j\neq i}^{N}\frac{\partial V_{ij}(r_{ij})}{\partial r_{ij}}r_{ij} \qquad (2.140)\\
&\frac{\partial \rho_i}{\partial V}=\frac{\partial}{\partial V}\sum_{j=1,j\neq i}^{N}\rho_{ij}(r_{ij})=\sum_{j=1,j\neq i}^{N}\frac{\partial \rho_{ij}(r_{ij})}{\partial r_{ij}}\frac{\partial r_{ij}}{\partial V}=\frac{1}{3V}\sum_{j=1,j\neq i}^{N}\frac{\partial \rho_{ij}(r_{ij})}{\partial r_{ij}}r_{ij} \\
\Psi_2 &= \frac{1}{2}\sum_{i=1}^{N}\sum_{j\neq i}^{N}\left(\frac{\partial F(\rho_i)}{\partial \rho_i}+\frac{\partial F(\rho_j)}{\partial \rho_j}\right)\frac{\partial \rho_{ij}(r_{ij})}{\partial r_{ij}}r_{ij} \quad .
\end{aligned}$$

1. EAM virial
   The same as above.
2. EEAM virial
   The same as above.
3. 2BEAM virial
   The same as above but with a second embedding contribution from the extra band complexity: $\Psi_2 = \Psi_2^s + \Psi_2^d$!
4. 2BEEAM virial
   The same as above but with a second embedding contribution from the extra band complexity: $\Psi_2 = \Psi_2^s + \Psi_2^d$!
5. Finnis-Sinclair virial

$$\begin{aligned}
\Psi_1 &= \frac{1}{2}\sum_{i=1}^{N}\sum_{j\neq i}^{N}\left\{2(r_{ij}-c)(c_0+c_1r_{ij}+c_2r_{ij}^2)+(r_{ij}-c)^2(c_1+2c_2r_{ij})\right\}r_{ij} \\
\Psi_2 &= \frac{1}{2}\sum_{i=1}^{N}\sum_{j\neq i}^{N}\frac{A}{2}\left(\frac{1}{\sqrt{\rho_k}}+\frac{1}{\sqrt{\rho_j}}\right)\left\{2(r_{ij}-d)+3\beta\frac{(r_{ij}-d)^2}{d}\right\}r_{ij}a \quad . \qquad (2.141)
\end{aligned}$$

6. Extended Finnis-Sinclair virial

$$\begin{aligned}
\Psi_1 &= \frac{1}{2}\sum_{i=1}^{N}\sum_{j\neq i}^{N}\left\{2(r_{ij}-c)(c_0+c_1r_{ij}+c_2r_{ij}^2+c_3r_{ij}^3+c_4r_{ij}^4)+\right. \\
&\qquad \left.(r_{ij}-c)^2(c_1+2c_2r_{ij}+3c_3r_{ij}^2+4c_4r_{ij}i^3)\right\}r_{ij} \qquad (2.142)\\
\Psi_2 &= \frac{1}{2}\sum_{i=1}^{N}\sum_{j\neq i}^{N}\frac{A}{2}\left(\frac{1}{\sqrt{\rho_k}}+\frac{1}{\sqrt{\rho_j}}\right)\left\{2(r_{ij}-d)+4B^2(r_{ij}-d)^3\right\}r_{ij}a \quad .
\end{aligned}$$

7. Sutton-Chen virial

$$\begin{aligned}
\Psi_1 &= -\frac{1}{2}\sum_{i=1}^{N}\sum_{j\neq i}^{N}n\epsilon\left(\frac{a}{r_{ij}}\right)^n \\
\Psi_2 &= \frac{1}{2}\sum_{i=1}^{N}\sum_{j\neq i}^{N}\frac{mc\epsilon}{2}\left(\frac{\partial F(\rho_i)}{\partial \rho_i}+\frac{\partial F(\rho_j)}{\partial \rho_j}\right)\left(\frac{a}{r_{ij}}\right)^m \quad . \qquad (2.143)
\end{aligned}$$

8. Gupta virial

$$\begin{aligned}\Psi_1 &= -\sum_{i=1}^{N}\sum_{j\neq i}^{N}\frac{Ap}{r_0}\exp\left(-p\frac{r_{ij}-r_0}{r_0}\right)r_{ij}\\ \Psi_2 &= \frac{1}{2}\sum_{i=1}^{N}\sum_{j\neq i}^{N}\frac{Bq_{ij}}{r_0}\left(\frac{1}{\sqrt{\rho_k}}+\frac{1}{\sqrt{\rho_j}}\right)\exp\left(-2q_{ij}\frac{r_{ij}-r_0}{r_0}\right)r_{ij} \quad .\end{aligned} \tag{2.144}$$

9. Many body perturbation component virial

$$\begin{aligned}\Psi_1 &= 0\\ \Psi_2 &= \frac{1}{2}\sum_{i=1}^{N}\sum_{j\neq i}^{N}\frac{m\epsilon}{2}\left(\frac{\partial F(\rho_i)}{\partial \rho_i}+\frac{\partial F(\rho_j)}{\partial \rho_j}\right)\frac{a}{r_{ij}^{m}} \quad .\end{aligned} \tag{2.145}$$

The contribution to be added to the atomic stress tensor is given by

$$\sigma^{\alpha\beta} = r_{ij}^{\alpha} f_j^{\beta} \quad , \tag{2.146}$$

where α and β indicate the x, y, z components. The atomic stress tensor is symmetric.

The long-ranged correction for the DL_POLY_4 metal potential is in two parts. Firstly, by analogy with the short ranged potentials, the correction to the local density is

$$\begin{aligned}\rho_i &= \sum_{j=1,j\neq i}^{\infty}\rho_{ij}(r_{ij})\\ \rho_i &= \sum_{j=1,j\neq i}^{r_{ij}<r_{\rm met}}\rho_{ij}(r_{ij}) + \sum_{j=1,j\neq i}^{r_{ij}\geq r_{\rm met}}\rho_{ij}(r_{ij}) = \rho_i^o + \delta\rho_i\\ \delta\rho_i &= 4\pi\bar{\rho}\int_{r_{\rm met}}^{\infty}\rho_{ij}(r)dr \quad ,\end{aligned} \tag{2.147}$$

where ρ_i^o is the uncorrected local density and $\bar{\rho}$ is the *mean particle density.* Evaluating the integral part of the above equation yields:

1. EAM density correction
   No long-ranged corrections apply beyond $r_{\rm met}$.

2. EEAM density correction
   No long-ranged corrections apply beyond $r_{\rm met}$.

3. 2BEAM density correction
   No long-ranged corrections apply beyond $r_{\rm met}$.

4. 2BEAM density correction
   No long-ranged corrections apply beyond $r_{\rm met}$.

5. Finnis-Sinclair density correction
   No long-ranged corrections apply beyond cutoffs c and d.

6. Extended Finnis-Sinclair density correction
   No long-ranged corrections apply beyond cutoffs c and d.

7. Sutton-Chen density correction

$$\delta\rho_i = \frac{4\pi\bar{\rho}a^3}{(m-3)}\left(\frac{a}{r_{\rm met}}\right)^{m-3} \quad . \tag{2.148}$$

8. Gupta density correction
$$\delta\rho_i = \frac{2\pi\bar{\rho}r_0}{q_{ij}}\left[r_{\rm met}^2 + 2r_{\rm met}\left(\frac{r_0}{q_{ij}}\right) + 2\left(\frac{r_0}{q_{ij}}\right)^2\right]\exp\left(-2q_{ij}\frac{r_{\rm met}-r_0}{r_0}\right) \quad . \tag{2.149}$$

9. Many body perturbation component density correction
$$\delta\rho_i = \frac{4\pi\bar{\rho}}{(m-3)}\frac{a}{r_{\rm met}^{m-3}} \quad . \tag{2.150}$$

The density correction is applied immediately after the local density is calculated. The pair term correction is obtained by analogy with the short ranged potentials and is

$$\begin{aligned}
U_1 &= \frac{1}{2}\sum_{i=1}^{N}\sum_{j\neq i}^{\infty} V_{ij}(r_{ij}) \\
U_1 &= \frac{1}{2}\sum_{i=1}^{N}\sum_{j\neq i}^{r_{ij}<r_{\rm met}} V_{ij}(r_{ij}) + \frac{1}{2}\sum_{i=1}^{N}\sum_{j\neq i}^{r_{ij}\geq r_{\rm met}} V_{ij}(r_{ij}) = U_1^o + \delta U_1 \\
\delta U_1 &= 2\pi N\bar{\rho}\int_{r_{\rm met}}^{\infty} V_{ij}(r)r^2 dr \\
U_2 &= \sum_{i=1}^{N} F(\rho_i^0 + \delta\rho_i) \\
U_2 &= \sum_{i=1}^{N} F(\rho_i^0) + \sum_{i=1}^{N}\frac{\partial F(\rho_i)_0}{\partial\rho_i}\delta\rho_i = U_2^0 + \delta U_2 \\
\delta U_2 &= 4\pi\bar{\rho}\sum_{i=1}^{N}\frac{\partial F(\rho_i)_0}{\partial\rho_i}\int_{r_{\rm met}}^{\infty}\rho_{ij}(r)r^2 dr \quad .
\end{aligned} \tag{2.151}$$

Note: that $\delta U2$ is not required if ρ_i has already been corrected. Evaluating the integral part of the above equations yields:

1. EAM energy correction
   No long-ranged corrections apply beyond $r_{\rm met}$.
2. EEAM energy correction
   No long-ranged corrections apply beyond $r_{\rm met}$.
3. 2BEAM energy correction
   No long-ranged corrections apply beyond $r_{\rm met}$.
4. 2BEEAM energy correction
   No long-ranged corrections apply beyond $r_{\rm met}$.
5. Finnis-Sinclair energy correction
   No long-ranged corrections apply beyond cutoffs c and d.
6. Extended Finnis-Sinclair energy correction
   No long-ranged corrections apply beyond cutoffs c and d.
7. Sutton-Chen energy correction
$$\begin{aligned}
\delta U_1 &= \frac{2\pi N\bar{\rho}\epsilon a^3}{(n-3)}\left(\frac{a}{r_{\rm met}}\right)^{n-3} \\
\delta U_2 &= -\frac{4\pi\bar{\rho}a^3}{(m-3)}\left(\frac{a}{r_{\rm met}}\right)^{m-3}\left\langle\frac{Nc\epsilon}{2\sqrt{\rho_i^0}}\right\rangle \quad .
\end{aligned} \tag{2.152}$$

8. Gupta energy correction

$$\begin{aligned}
\delta U_1 &= \frac{4\pi N\bar{\rho}Ar_0}{p}\left[r_{\rm met}^2 + 2r_{\rm met}\left(\frac{r_0}{p}\right) + 2\left(\frac{r_0}{p}\right)^2\right] \times \\
&\quad \exp\left(-p\frac{r_{\rm met}-r_0}{r_0}\right) \\
\delta U_2 &= -\frac{2\pi\bar{\rho}r_0}{q_{ij}}\left[r_{\rm met}^2 + 2r_{\rm met}\left(\frac{r_0}{q_{ij}}\right) + 2\left(\frac{r_0}{q_{ij}}\right)^2\right] \times \\
&\quad \exp\left(-2q_{ij}\frac{r_{\rm met}-r_0}{r_0}\right)\left\langle\frac{NB}{2\sqrt{\rho_i^0}}\right\rangle \quad .
\end{aligned} \tag{2.153}$$

9. Many body perturbation component energy correction

$$\begin{aligned}
\delta U_1 &= 0 \\
\delta U_2 &= -\frac{4\pi\bar{\rho}}{(m-3)}\frac{a}{r_{\rm met}^{m-3}}\left\langle\frac{N\epsilon}{2\sqrt{\rho_i^0}}\right\rangle \quad .
\end{aligned} \tag{2.154}$$

To estimate the virial correction we assume the corrected local densities are constants (i.e. independent of distance - at least beyond the range $r_{\rm met}$). This allows the virial correction to be computed by the methods used in the short ranged potentials:

$$\begin{aligned}
\Psi_1 &= \frac{1}{2}\sum_{i=1}^{N}\sum_{j\neq i}^{\infty}\frac{\partial V_{ij}(r_{ij})}{\partial r_{ij}}r_{ij} \\
\Psi_1 &= \frac{1}{2}\sum_{i=1}^{N}\sum_{j\neq i}^{r_{ij}<r_{\rm met}}\frac{\partial V_{ij}(r_{ij})}{\partial r_{ij}}r_{ij} + \frac{1}{2}\sum_{i=1}^{N}\sum_{j\neq i}^{r_{ij}\geq r_{\rm met}}\frac{\partial V_{ij}(r_{ij})}{\partial r_{ij}}r_{ij} = \Psi_1^0 + \delta\Psi_1 \\
\delta\Psi_1 &= 2\pi N\bar{\rho}\int_{r_{\rm met}}^{\infty}\frac{\partial V_{ij}(r)}{\partial r_{ij}}r^3 dr \\
\Psi_2 &= \sum_{i=1}^{N}\frac{\partial F(\rho_i)}{\partial \rho_i}\sum_{j\neq i}^{\infty}\frac{\partial \rho_{ij}(r_{ij})}{\partial r_{ij}}r_{ij} \\
\Psi_2 &= \sum_{i=1}^{N}\frac{\partial F(\rho_i)}{\partial \rho_i}\sum_{j\neq i}^{r_{ij}<r_{\rm met}}\frac{\partial \rho_{ij}(r_{ij})}{\partial r_{ij}}r_{ij} + \sum_{i=1}^{N}\frac{\partial F(\rho_i)}{\partial \rho_i}\sum_{j\neq i}^{r_{ij}\geq r_{\rm met}}\frac{\partial \rho_{ij}(r_{ij})}{\partial r_{ij}}r_{ij} = \Psi_2^0 + \delta\Psi_2 \\
\delta\Psi_2 &= 4\pi\bar{\rho}\sum_{i=1}^{N}\frac{\partial F(\rho_i)}{\partial \rho_i}\int_{r_{\rm met}}^{\infty}\frac{\partial \rho_{ij}(r)}{\partial r}r^3 dr \quad .
\end{aligned} \tag{2.155}$$

Evaluating the integral part of the above equations yields:

1. EAM virial correction
   No long-ranged corrections apply beyond $r_{\rm met}$.

2. EEAM virial correction
   No long-ranged corrections apply beyond $r_{\rm met}$.

3. 2BEAM virial correction
   No long-ranged corrections apply beyond $r_{\rm met}$.

4. 2BEEAM virial correction
   No long-ranged corrections apply beyond $r_{\rm met}$.

5. Finnis-Sinclair virial correction
   No long-ranged corrections apply beyond cutoffs c and d.

6. Extended Finnis-Sinclair virial correction
   No long-ranged corrections apply beyond cutoffs c and d.

7. Sutton-Chen virial correction

$$\begin{aligned} \delta\Psi_1 &= -n\frac{2\pi N\bar{\rho}\epsilon a^3}{(n-3)}\left(\frac{a}{r_{\rm met}}\right)^{n-3} \\ \delta\Psi_2 &= m\frac{4\pi\bar{\rho}a^3}{(m-3)}\left(\frac{a}{r_{\rm met}}\right)^{m-3}\left\langle\frac{Nc\epsilon}{2\sqrt{\rho_i^0}}\right\rangle \quad . \end{aligned} \tag{2.156}$$

8. Gupta virial correction

$$\begin{aligned} \delta\Psi_1 &= -\frac{p}{r_0}\frac{4\pi N\bar{\rho}Ar_0}{p}\left[r_{\rm met}^3+3r_{\rm met}^2\left(\frac{r_0}{p}\right)+6r_{\rm met}\left(\frac{r_0}{p}\right)^2+6\left(\frac{r_0}{p}\right)^3\right]\times \\ &\quad \exp\left(-p\frac{r_{\rm met}-r_0}{r_0}\right) \\ \delta\Psi_2 &= \frac{q_{ij}}{r_0}\frac{2\pi\bar{\rho}r_0}{q_{ij}}\left[r_{\rm met}^3+3r_{\rm met}^2\left(\frac{r_0}{q_{ij}}\right)+6r_{\rm met}\left(\frac{r_0}{q_{ij}}\right)^2+6\left(\frac{r_0}{q_{ij}}\right)^3\right]\times \\ &\quad \exp\left(-2q_{ij}\frac{r_{\rm met}-r_0}{r_0}\right)\left\langle\frac{NB}{2\sqrt{\rho_i^0}}\right\rangle \quad . \end{aligned} \tag{2.157}$$

9. Many body perturbation component virial correction

$$\begin{aligned} \delta\Psi_1 &= 0 \\ \delta\Psi_2 &= m\frac{4\pi\bar{\rho}}{(m-3)}\frac{a}{r_{\rm met}^{m-3}}\left\langle\frac{N\epsilon}{2\sqrt{\rho_i^0}}\right\rangle \quad . \end{aligned} \tag{2.158}$$

In the energy and virial corrections we have used the approximation:

$$\sum_i^N \rho_i^{-1/2} = \frac{N}{<\rho_i^{1/2}>} \quad , \tag{2.159}$$

where $<\rho_i^{1/2}>$ is regarded as a constant of the system.

In DL_POLY_4 the metal forces are handled by the routine METAL_FORCES. The local density is calculated by the routines METAL_LD_COLLECT_EAM, METAL_LD_COLLECT_FST, METAL_LD_COMPUTE, METAL_LD_SET_HALO and METAL_LD_EXPORT. The long-ranged corrections are calculated by METAL_LRC. Reading and generation of EAM table data from TABEAM is handled by METAL_TABLE_READ and METAL_TABLE_DERIVATIVES.

Notes on the Treatment of Alloys

The distinction to be made between EAM and FS potentials with regard to alloys concerns the mixing rules for unlike interactions. Starting with equations (2.122) and (2.123), it is clear that we require mixing rules for terms $V_{ij}(r_{ij})$ and $\rho_{ij}(r_{ij})$ when atoms i and j are of different kinds. Thus two different metals A and B we can distinguish 4 possible variants of each:

$$V_{ij}^{AA}(r_{ij}),\ V_{ij}^{BB}(r_{ij}),\ V_{ij}^{AB}(r_{ij}),\ V_{ij}^{BA}(r_{ij})$$

and

$$\rho_{ij}^{AA}(r_{ij}),\ \rho_{ij}^{BB}(r_{ij}),\ \rho_{ij}^{AB}(r_{ij}),\ \rho_{ij}^{BA}(r_{ij}) \ .$$

These forms recognise that the contribution of a type A atom to the potential of a type B atom may be different from the contribution of a type B atom to the potential of a type A atom. In both EAM [65] and FS [15] cases it turns out that

$$V_{ij}^{BA}(r_{ij}) = V_{ij}^{BA}(r_{ij}) \ , \tag{2.160}$$

though the mixing rules are different in each case (**beware!**). This has the following implications to densities of mixtures for different potential frameworks:

- EAM case - it is required that [65]:

$$\begin{aligned} \rho_{ij}^{AB}(r_{ij}) &= \rho_{ij}^{BB}(r_{ij}) \\ \rho_{ij}^{BA}(r_{ij}) &= \rho_{ij}^{AA}(r_{ij}) \ , \end{aligned} \tag{2.161}$$

  which means that an atom of type A contributes the same density to the environment of an atom of type B as it does to an atom of type A, and *vice versa*.

- EEAM case - all densities can be different [57, 58]:

$$\rho_{ij}^{AA}(r_{ij}) \neq \rho_{ij}^{BB}(r_{ij}) \neq \rho_{ij}^{AB}(r_{ij}) \neq \rho_{ij}^{BA}(r_{ij}) \ ! \tag{2.162}$$

- 2BEAM case - similarly to the EAM case it is required that:

$$\begin{aligned} {\rho_{ij}^{d}}^{AB}(r_{ij}) &= {\rho_{ij}^{d}}^{BB}(r_{ij}) \\ {\rho_{ij}^{d}}^{BA}(r_{ij}) &= {\rho_{ij}^{d}}^{AA}(r_{ij}) \ , \end{aligned} \tag{2.163}$$

  for the d-band densities, whereas for the s-band ones:

$${\rho_{ij}^{s}}^{BA}(r_{ij}) = {\rho_{ij}^{s}}^{AB}(r_{ij}) \ , \tag{2.164}$$

  which means that an atom of type A contributes the same s density to the environment of an atom of type B as an atom of type B to an environment of an atom of type A. However, in general:

$${\rho_{ij}^{s}}^{AA}(r_{ij}) \neq {\rho_{ij}^{s}}^{BB}(r_{ij}) \neq {\rho_{ij}^{s}}^{AB}(r_{ij}) \ . \tag{2.165}$$

- 2BEEAM case - similarly to the EEAM case all s and d densities can be different:

$$\begin{aligned} {\rho_{ij}^{s}}^{AA}(r_{ij}) \neq {\rho_{ij}^{s}}^{BB}(r_{ij}) \neq {\rho_{ij}^{s}}^{AB}(r_{ij}) \neq {\rho_{ij}^{s}}^{BA}(r_{ij}) \\ {\rho_{ij}^{d}}^{AA}(r_{ij}) \neq {\rho_{ij}^{d}}^{BB}(r_{ij}) \neq {\rho_{ij}^{d}}^{AB}(r_{ij}) \neq {\rho_{ij}^{d}}^{BA}(r_{ij}) \ . \end{aligned} \tag{2.166}$$

- FS case - here a different rule applies [15]:

$$\rho_{ij}^{AB}(r_{ij}) = (\rho_{ij}^{AA}(r_{ij})\ \rho_{ij}^{BB}(r_{ij}))^{1/2} \tag{2.167}$$

  so that atoms of type A and B contribute the same densities to each other, but not to atoms of the same type.

The above rules have the following consequences to the specifications of these potentials in the DL_POLY_4 FIELD file for an alloy composed of n different metal atom types both the EAM types and FS types of potentials require the specification of $n(n+1)/2$ pair functions $V_{ij}^{AB}(r_{ij})$. However, the its only the simple EAM type together with all the FS types that require only n density functions $\rho_{ij}^{AA}(r_{ij})$, whereas the EEAM class requires all the cross functions $\rho_{ij}^{AB}(r_{ij})$ possible or n^2 in total! In addition to the $n(n+1)/2$ pair functions and n or n^2 density functions both the EAM and EEAM potentials require further specification of

n functional forms of the density dependence (i.e. the embedding function $F(\rho_i)$ in equation (2.122)). The matter is further complicated when the 2BEAM type of potential is used with the extra specification of n embedding functions and $n(n+1)/2$ density functions for the s-band. Similarly, in the 2BEEAM an extra n embedding functions and n^2 density functions for the s-band are required.

It is worth noting that in the 2BEAM and 2BEEAM the s-band contribution is usually only for the alloy component, so that local concentrations of a single element revert to the standard EAM or EEAM! In such case, the densities functions must be zeroed in the DL_POLY_4 TABEAM file.

For EAM, EEAM, 2BEAM and 2BEEAM potentials all the functions are supplied in tabular form via the table file TABEAM (see section 10.1.8) to which DL_POLY_4 is redirected by the FIELD file data. The FS potentials are defined via the necessary parameters in the FIELD file.

2.3.3 Tersoff Potentials

The Tersoff [17] potential is is a bond-order potential, developed to be used in multi-component covalent systems by an effective coupling of two-body and higher many-body correlations into one model. The central idea is that in real systems, the strength of each bond depends on the local environment, i.e. an atom with many neighbors forms weaker bonds than an atom with few neighbors. Effectively, it is a pair potential the strength of which depends on the environment. At the present there are two versions of this potential available in DL_POLY_4: **ters** and **kihs**. In these particular implementations **ters** has 11 atomic and 2 bi-atomic parameters whereas **kihs** [66] has 16 atomic parameters. The energy is modelled as a sum of pair-like interactions, where the coefficient of the attractive term in the pair-like potential (which plays the role of a bond order) depends on the local environment giving a many-body potential.

The form of the Tersoff potential is: (**ters**)

$$U_{ij} = f_C(r_{ij})\ [f_R(r_{ij}) - \gamma_{ij}\ f_A(r_{ij})]\ \ , \tag{2.168}$$

where f_R and f_A are the repulsive and attractive pair potential respectively:

$$f_R(r_{ij}) = A_{ij}\ \exp(-a_{ij}\ r_{ij})\ \ ,\ \ f_A(r_{ij}) = B_{ij}\ \exp(-b_{ij}\ r_{ij}) \tag{2.169}$$

and f_C is a smooth cutoff function with parameters R and S so chosen that to include the first-neighbor shell:

- **ters**:

$$f_C(r_{ij}) = \begin{cases} 1 & : \ r_{ij} < R_{ij} \\ \frac{1}{2} + \frac{1}{2}\cos\left[\pi\ \frac{r_{ij}-R_{ij}}{S_{ij}-R_{ij}}\right] & : \ R_{ij} < r_{ij} < S_{ij} \\ 0 & : \ r_{ij} > S_{ij} \end{cases} \tag{2.170}$$

- **kihs** - here f_C is modified to a have continuous second-order differential:

$$f_C(r_{ij}) = \begin{cases} 1 & : \ r_{ij} < R_{ij} \\ \frac{1}{2} + \frac{9}{16}\cos\left[\pi\ \frac{r_{ij}-R_{ij}}{S_{ij}-R_{ij}}\right] - \frac{1}{16}\cos\left[3\pi\ \frac{r_{ij}-R_{ij}}{S_{ij}-R_{ij}}\right] & : \ R_{ij} < r_{ij} < S_{ij} \\ 0 & : \ r_{ij} > S_{ij}\ \ . \end{cases} \tag{2.171}$$

γ_{ij} expresses a dependence that can accentuate or diminish the attractive force relative to the repulsive force, according to the local environment, such that:

- **ters**:

$$\begin{aligned} \gamma_{ij} &= \chi_{ij}\ (1 + {\beta_i}^{\eta_i}\ \mathcal{L}_{ij}^{\eta_i})^{-\frac{1}{2\eta_i}} \\ \mathcal{L}_{ij} &= \sum_{k\neq i,j} f_C(r_{ik})\ \omega_{ik}\ g(\theta_{ijk}) \\ g(\theta_{ijk}) &= 1 + \frac{c_i^2}{d_i^2} - \frac{c_i^2}{d_i^2 + (h_i - \cos\theta_{ijk})^2} \end{aligned} \tag{2.172}$$

- **kihs**:

$$\begin{aligned}
\gamma_{ij} &= (1+\mathcal{L}_{ij}^{\eta_i})^{-\delta_i} \\
\mathcal{L}_{ij} &= \sum_{k\neq i,j} f_C(r_{ik})\ g(\theta_{ijk})\ \overbrace{\exp\left[\alpha_i(r_{ij}-r_{ik})^{\beta_i}\right]}^{\omega_{ik}} \\
g(\theta_{ijk}) &= c_{1i}+g_o(\theta_{ijk})\ g_a(\theta_{ijk}) \\
g_o(\theta_{ijk}) &= \frac{c_{2i}\ (h_i-\cos\theta_{ijk})^2}{c_{3i}+(h_i-\cos\theta_{ijk})^2} \\
g_a(\theta_{ijk}) &= 1+c_{4i}\ \exp\left[-c_{5i}\ (h_i-\cos\theta_{ijk})^2\right]\ ,
\end{aligned} \tag{2.173}$$

where the term $\mathcal{L}_{ij}$ defines the effective coordination number of atom i i.e. the number of nearest neighbors, taking into account the relative distance of the two neighbors, i and k, $r_{ij}-r_{ik}$, and the bond angle, θ_{ijk}, between them with respect to the central atom i. The function $g(\theta)$ has a minimum for $h_i=\cos(\theta_{ijk})$, the parameter d_i in **ters** and c_{3i} in **kihs** determines how sharp the dependence on angle is, whereas the rest express the strength of the angular effect. Further mixed parameters are defined as:

$$\begin{aligned}
a_{ij}=(a_i+a_j)/2 \quad &, \quad b_{ij}=(b_i+b_j)/2 \\
A_{ij}=(A_iA_j)^{1/2} \quad &, \quad B_{ij}=(B_iB_j)^{1/2} \\
R_{ij}=(R_iR_j)^{1/2} \quad &, \quad S_{ij}=(S_iS_j)^{1/2}\ .
\end{aligned} \tag{2.174}$$

Singly subscripted parameters, such as a_i and η_i, depend only on the type of atom.

For **ters** the chemistry between different atom types is locked in the two sets bi-atomic parameters χ_{ij} and ω_{ij}:

$$\begin{aligned}
\chi_{ii} = 1 \quad &, \quad \chi_{ij} = \chi_{ji} \\
\omega_{ii} = 1 \quad &, \quad \omega_{ij} = \omega_{ji}\ ,
\end{aligned} \tag{2.175}$$

which define only one independent parameter each per pair of atom types. The χ parameter is used to strengthen or weaken the heteropolar bonds, relative to the value obtained by simple interpolation. The ω parameter is used to permit greater flexibility when dealing with more drastically different types of atoms.

The force on an atom ℓ derived from this potential is formally calculated with the formula:

$$f_\ell^\alpha = -\frac{\partial}{\partial r_\ell^\alpha}E_{\texttt{tersoff}} = \frac{1}{2}\sum_i\sum_{j\neq i} -\frac{\partial}{\partial r_\ell^\alpha}U_{ij}\ , \tag{2.176}$$

with atomic label ℓ being one of i,j,k and α indicating the x,y,z component. The derivative after the summation is worked out as

$$-\frac{\partial U_{ij}}{\partial r_\ell^\alpha} = -\frac{\partial}{\partial r_\ell^\alpha}f_C(r_{ij})f_R(r_{ij}) + \gamma_{ij}\frac{\partial}{\partial r_\ell^\alpha}f_C(r_{ij})f_A(r_{ij}) + f_C(r_{ij})f_A(r_{ij})\frac{\partial}{\partial r_\ell^\alpha}\gamma_{ij}\ , \tag{2.177}$$

with the contributions from the first two terms being:

$$\begin{aligned}
-\frac{\partial}{\partial r_\ell^\alpha}f_C(r_{ij})f_R(r_{ij}) &= -\left\{f_C(r_{ij})\frac{\partial}{\partial r_{ij}}f_R(r_{ij}) + f_R(r_{ij})\frac{\partial}{\partial r_{ij}}f_C(r_{ij})\right\}\times \\
&\quad \left\{\delta_{j\ell}\frac{r_{i\ell}^\alpha}{r_{i\ell}} - \delta_{i\ell}\frac{r_{\ell j}^\alpha}{r_{\ell j}}\right\}
\end{aligned} \tag{2.178}$$

$$\begin{aligned}
\gamma_{ij}\frac{\partial}{\partial r_\ell^\alpha}f_C(r_{ij})f_A(r_{ij}) &= \gamma_{ij}\left\{f_C(r_{ij})\frac{\partial}{\partial r_{ij}}f_A(r_{ij}) + f_A(r_{ij})\frac{\partial}{\partial r_{ij}}f_C(r_{ij})\right\}\times \\
&\quad \left\{\delta_{j\ell}\frac{r_{i\ell}^\alpha}{r_{i\ell}} - \delta_{i\ell}\frac{r_{\ell j}^\alpha}{r_{\ell j}}\right\}\ ,
\end{aligned} \tag{2.179}$$

and from the third (angular) term:

- **ters**:

$$\begin{aligned} f_C(r_{ij}) f_A(r_{ij}) \frac{\partial}{\partial r_\ell^\alpha} \gamma_{ij} &= f_C(r_{ij}) f_A(r_{ij})\ \chi_{ij} \quad \times \\ & \left(-\frac{1}{2}\right) \left(1 + {\beta_i}^{\eta_i}\ \mathcal{L}_{ij}^{\eta_i}\right)^{-\frac{1}{2\eta_i}-1} {\beta_i}^{\eta_i}\ \mathcal{L}_{ij}^{\eta_i - 1} \frac{\partial}{\partial r_\ell^\alpha} \mathcal{L}_{ij} \quad , \end{aligned} \tag{2.180}$$

where

$$\frac{\partial}{\partial r_\ell^\alpha} \mathcal{L}_{ij} = \frac{\partial}{\partial r_\ell^\alpha} \sum_{k \neq i,j} \omega_{ik}\ f_C(r_{ik})\ g(\theta_{ijk}) \quad .$$

The angular term can have three different contributions depending on the index of the particle participating in the interaction:

$$\begin{aligned} \ell = i \quad &: \quad \frac{\partial}{\partial r_i^\alpha} \mathcal{L}_{ij} = \sum_{k \neq i,j} \omega_{ik} \left[g(\theta_{ijk}) \frac{\partial}{\partial r_i^\alpha} f_C(r_{ik}) + f_C(r_{ik}) \frac{\partial}{\partial r_i^\alpha} g(\theta_{ijk}) \right] \\ \ell = j \quad &: \quad \frac{\partial}{\partial r_j^\alpha} \mathcal{L}_{ij} = \sum_{k \neq i,j} \omega_{ik}\ f_C(r_{ik}) \frac{\partial}{\partial r_j^\alpha} g(\theta_{ijk}) \\ \ell \neq i,j \quad &: \quad \frac{\partial}{\partial r_\ell^\alpha} \mathcal{L}_{ij} = \omega_{i\ell} \left[g(\theta_{ij\ell}) \frac{\partial}{\partial r_\ell^\alpha} f_C(r_{i\ell}) + f_C(r_{i\ell}) \frac{\partial}{\partial r_\ell^\alpha} g(\theta_{ij\ell}) \right] \quad , \end{aligned} \tag{2.181}$$

- **kihs**:

$$\begin{aligned} f_C(r_{ij}) f_A(r_{ij}) \frac{\partial}{\partial r_\ell^\alpha} \gamma_{ij} &= f_C(r_{ij}) f_A(r_{ij}) \quad \times \\ & (-\delta_i\ \eta_i) \left(1 + \mathcal{L}_{ij}^{\eta_i}\right)^{-\delta_i - 1} \mathcal{L}_{ij}^{\eta_i - 1} \frac{\partial}{\partial r_\ell^\alpha} \mathcal{L}_{ij} \quad , \end{aligned} \tag{2.182}$$

where

$$\frac{\partial}{\partial r_\ell^\alpha} \mathcal{L}_{ij} = \frac{\partial}{\partial r_\ell^\alpha} \sum_{k \neq i,j} \omega_{ik}(r_{ij}, r_{ik})\ f_C(r_{ik})\ g(\theta_{ijk}) \quad .$$

It is worth noting that the derivative of ω_{ik}:

$$\frac{\partial}{\partial r_\ell^\alpha} \omega_{ik} = \alpha_i\ \beta_i\ (r_{ij} - r_{ik})^{\beta_i - 1}\ \omega_{ik}\ \left\{ (\delta_{\ell j} - \delta_{\ell i}) \frac{r_{ij}^\alpha}{r_{ij}} - (\delta_{\ell k} - \delta_{\ell i}) \frac{r_{ik}^\alpha}{r_{ik}} \right\} \quad , \tag{2.183}$$

now has three different contributions depending on the index of the particle participating in the interaction! Hence the angular term's derivative is more elaborate to express than the one in the **ters** case.

The derivative of $g(\theta_{ijk})$ is worked out in the following manner:

$$\frac{\partial}{\partial r_\ell^\alpha} g(\theta_{ijk}) = \frac{\partial g(\theta_{ijk})}{\partial \theta_{ijk}}\ \frac{-1}{\sin\theta_{ijk}}\ \frac{\partial}{\partial r_\ell^\alpha} \left\{ \frac{\underline{r}_{ij} \cdot \underline{r}_{ik}}{r_{ij}\ r_{ik}} \right\} \quad , \tag{2.184}$$

where

$$\frac{\partial g(\theta_{ijk})}{\partial \theta_{ijk}} = \frac{2\ c_i^2 (h_i - \cos\theta_{ijk})\ \sin\theta_{ijk}}{[d_i^2 + (h_i - \cos\theta_{ijk})^2]^2} \tag{2.185}$$

$$\begin{aligned} \frac{\partial}{\partial r_\ell^\alpha} \left\{ \frac{\underline{r}_{ij} \cdot \underline{r}_{ik}}{r_{ij} r_{ik}} \right\} &= (\delta_{\ell j} - \delta_{\ell i}) \frac{r_{ik}^\alpha}{r_{ij} r_{ik}} + (\delta_{\ell k} - \delta_{\ell i}) \frac{r_{ij}^\alpha}{r_{ij} r_{ik}} - \\ & \cos(\theta_{jik}) \left\{ (\delta_{\ell j} - \delta_{\ell i}) \frac{r_{ij}^\alpha}{r_{ij}^2} + (\delta_{\ell k} - \delta_{\ell i}) \frac{r_{ik}^\alpha}{r_{ik}^2} \right\} \quad . \end{aligned} \tag{2.186}$$

The contribution to be added to the atomic virial can be derived as

$$\begin{aligned}\mathcal{W} &= 3V\frac{\partial E_{\texttt{tersoff}}}{\partial V} = \frac{3\ V}{2}\sum_{i}\sum_{j\neq i}\frac{\partial U_{ij}}{\partial V} = \sum_{i}\underline{r_i}\cdot\underline{f_i} \\ &= \frac{1}{2}\sum_{i}\sum_{j\neq i}-\left(\underline{r_{ij}}\cdot\underline{f_{ij}}+\underline{r_{ik}}\cdot\underline{f_{ik}}\right) = \frac{1}{2}\sum_{i}\sum_{j\neq i}\left(\frac{\partial U_{ij}}{\partial r_{ij}}\cdot\underline{r_{ij}}+\frac{\partial U_{ik}}{\partial r_{ik}}\cdot\underline{r_{ik}}\right)\end{aligned} \tag{2.187}$$

- **ters**:

$$\begin{aligned}\mathcal{W} = \frac{1}{2}\sum_{i}\sum_{j\neq i}\Bigg\{&\left[\frac{\partial}{\partial r_{ij}}f_C(r_{ij})f_R(r_{ij}) - \gamma_{ij}\frac{\partial}{\partial r_{ij}}f_C(r_{ij})f_A(r_{ij})\right]r_{ij} - \\ &\left(-\frac{1}{2}\right)f_C(r_{ij})f_A(r_{ij})\ \chi_{ij}\left(1+{\beta_i}^{\eta_i}\ \mathcal{L}_{ij}^{\eta_i}\right)^{-\frac{1}{2\eta_i}-1}\ {\beta_i}^{\eta_i}\ \mathcal{L}_{ij}^{\eta_i-1}\ \times \\ &\sum_{k\neq i,j}\omega_{ik}\ g(\theta_{ijk})\left[\frac{\partial}{\partial r_{ik}}f_C(r_{ik})\right]r_{ik}\Bigg\}\quad,\end{aligned} \tag{2.188}$$

- **hiks**:

$$\begin{aligned}\mathcal{W} = \frac{1}{2}\sum_{i}\sum_{j\neq i}\Bigg\{&\left[\frac{\partial}{\partial r_{ij}}f_C(r_{ij})f_R(r_{ij}) - \gamma_{ij}\frac{\partial}{\partial r_{ij}}f_C(r_{ij})f_A(r_{ij})\right]r_{ij} - \\ &(-\delta_i\ \eta_i)\,f_C(r_{ij})f_A(r_{ij})\ \chi_{ij}\left(1+\mathcal{L}_{ij}^{\eta_i}\right)^{-\delta_i-1}\ \mathcal{L}_{ij}^{\eta_i-1}\ \times \\ &\sum_{k\neq i,j}\omega_{ik}\left[r_{ik}\ g(\theta_{ijk})\frac{\partial}{\partial r_{ik}}f_C(r_{ik}) + \alpha_i\ \beta_i\ (r_{ij}-r_{ik})^{\beta_i}f_C(r_{ik})\right]\Bigg\}\quad.\end{aligned} \tag{2.189}$$

The contribution to be added to the atomic stress tensor is given by

$$\sigma^{\alpha\beta} = -r_i^{\alpha}f_i^{\beta}\quad, \tag{2.190}$$

where α and β indicate the x, y, z components. The stress tensor is symmetric.

Interpolation arrays, `vter` and `gter` (set up in TERSOFF_GENERATE) - similar to those in van der Waals interactions (Section 2.3.1), are used in the calculation of the Tersoff forces, virial and stress.

The Tersoff potentials are very short ranged, typically of order 3 Å. This property, plus the fact that Tersoff potentials (two- and three-body contributions) scale as N^3, where N is the number of particles, makes it essential that these terms are calculated by the link-cell method [67].

DL_POLY_4 applies no long-ranged corrections to the Tersoff potentials. In DL_POLY_4 Tersoff forces are handled by the routine TERSOFF_FORCES.

2.3.4 Three-Body Potentials

The three-body potentials in DL_POLY_4 are mostly valence angle forms. (They are primarily included to permit simulation of amorphous materials e.g. silicate glasses.) However, these have been extended to include the Dreiding [19] hydrogen bond. The potential forms available are as follows:

1. Harmonic: (**harm**)

$$U(\theta_{jik}) = \frac{k}{2}(\theta_{jik}-\theta_0)^2 \tag{2.191}$$

2. Truncated harmonic: (**thrm**)

$$U(\theta_{jik}) = \frac{k}{2}(\theta_{jik}-\theta_0)^2\exp[-(r_{ij}^8+r_{ik}^8)/\rho^8] \tag{2.192}$$

3. Screened Harmonic: (**shrm**)

$$U(\theta_{jik}) = \frac{k}{2}(\theta_{jik} - \theta_0)^2 \exp[-(r_{ij}/\rho_1 + r_{ik}/\rho_2)] \tag{2.193}$$

4. Screened Vessal [36]: (**bvs1**)

$$\begin{aligned} U(\theta_{jik}) &= \frac{k}{8(\theta_0 - \pi)^2} \left\{ \left[(\theta_0 - \pi)^2 - (\theta_{jik} - \pi)^2 \right]^2 \right\} \times \\ & \exp[-(r_{ij}/\rho_1 + r_{ik}/\rho_2)] \end{aligned} \tag{2.194}$$

5. Truncated Vessal [37]: (**bvs2**)

$$\begin{aligned} U(\theta_{jik}) &= k \left[\theta_{jik}^a (\theta_{jik} - \theta_0)^2 (\theta_{jik} + \theta_0 - 2\pi)^2 - \right. \\ & \left. \frac{a}{2} \pi^{a-1} (\theta_{jik} - \theta_0)^2 (\pi - \theta_0)^3 \right] \exp[-(r_{ij}^8 + r_{ik}^8)/\rho^8] \end{aligned} \tag{2.195}$$

6. Dreiding hydrogen bond [19]: (**hbnd**)

$$U(\theta_{jik}) = D_{hb} \cos^4(\theta_{jik}) \left[5(R_{hb}/r_{jk})^{12} - 6(R_{hb}/r_{jk})^{10} \right] \tag{2.196}$$

Note that for the hydrogen bond, the hydrogen atom *must* be the central atom. Several of these functions are identical to those appearing in the *intra*-molecular valence angle descriptions above. There are significant differences in implementation however, arising from the fact that the three-body potentials are regarded as *inter*-molecular. Firstly, the atoms involved are defined by atom types, not specific indices. Secondly, there are *no* excluded atoms arising from the three-body terms. (The inclusion of other potentials, for example pair potentials, may in fact be essential to maintain the structure of the system.)

The three-body potentials are very short ranged, typically of order 3 Å. This property, plus the fact that three-body potentials scale as N^4, where N is the number of particles, makes it essential that these terms are calculated by the link-cell method [67].

The calculation of the forces, virial and stress tensor as described in the section valence angle potentials above.

DL_POLY_4 applies no long-ranged corrections to the three-body potentials. The three-body forces are calculated by the routine THREE_BODY_FORCES.

2.3.5 Four-Body Potentials

The four-body potentials in DL_POLY_4 are entirely inversion angle forms, primarily included to permit simulation of amorphous materials (particularly borate glasses). The potential forms available in DL_POLY_4 are as follows:

1. Harmonic: (**harm**)

$$U(\phi_{ijkn}) = \frac{k}{2} (\phi_{ijkn} - \phi_0)^2 \tag{2.197}$$

2. Harmonic cosine: (**hcos**)

$$U(\phi_{ijkn}) = \frac{k}{2} (\cos(\phi_{ijkn}) - \cos(\phi_0))^2 \tag{2.198}$$

3. Planar potential: (**plan**)

$$U(\phi_{ijkn}) = A \left[1 - \cos(\phi_{ijkn}) \right] \tag{2.199}$$

These functions are identical to those appearing in the *intra*-molecular inversion angle descriptions above. There are significant differences in implementation however, arising from the fact that the four-body potentials are regarded as *inter*-molecular. Firstly, the atoms involved are defined by atom types, not specific indices. Secondly, there are *no* excluded atoms arising from the four-body terms. (The inclusion of other potentials, for example pair potentials, may in fact be essential to maintain the structure of the system.)

The four-body potentials are very short ranged, typically of order 3 Å. This property, plus the fact that four-body potentials scale as N^4, where N is the number of particles, makes it essential that these terms are calculated by the link-cell method [67].

The calculation of the forces, virial and stress tensor described in the section on inversion angle potentials above.

DL_POLY_4 applies no long-ranged corrections to the four body potentials. The four-body forces are calculated by the routine FOUR_BODY_FORCES.

2.4 Long Ranged Electrostatic (coulombic) Potentials

DL_POLY_4 incorporates several techniques for dealing with long-ranged electrostatic potentials[*]. These are as follows:

1. Direct Coulomb sum
2. Force-shifted Coulomb sum
3. Coulomb sum with distance dependent dielectric
4. Reaction field
5. Smoothed Particle Mesh Ewald (SPME)

All of these can be used in conjunction with the shell model technique used to account for ions polarisation.

The SPME technique is restricted to periodic systems only. (Users must exercise care when using pseudo-periodic boundary conditions.) The other techniques can be used with either periodic or non-periodic systems safely, although in the case of the direct Coulomb sum there are likely to be problems with convergence.

DL_POLY_4 will correctly handle the electrostatics of both molecular and atomic species. However, it is assumed that the system is electrically neutral. A warning message is printed if the system is found to be charged, but otherwise the simulation proceeds as normal.

Note that DL_POLY_4 does not use the basic Ewald method, which is an option in DL_POLY_Classic, on account of it being too slow for large scale systems. The SPME method is the standard Ewald method in DL_POLY_4.

2.4.1 Default (Point Charges) Electrostatics

2.4.1.1 Direct Coulomb Sum

Use of the direct Coulomb sum is sometimes necessary for accurate simulation of isolated (non-periodic) systems. It is *not* recommended for periodic systems.

[*] Unlike the other elements of the force field, the electrostatic forces are NOT specified in the input FIELD file, but by setting appropriate directives in the CONTROL file. See Section 10.1.1.

The interaction potential for two charged ions is

$$U(r_{ij}) = \frac{1}{4\pi\epsilon_0\epsilon}\frac{q_i q_j}{r_{ij}} \quad , \tag{2.200}$$

with q_ℓ the charge on an atom labelled ℓ, and r_{ij} the magnitude of the separation vector $\underline{r}_{ij} = \underline{r}_j - \underline{r}_i$.

The force on an atom j derived from this force is

$$\underline{f}_j = \frac{1}{4\pi\epsilon_0\epsilon}\frac{q_i q_j}{r_{ij}^3}\underline{r}_{ij} \quad , \tag{2.201}$$

with the force on atom i the negative of this.

The contribution to the atomic virial is

$$\mathcal{W} = -\frac{1}{4\pi\epsilon_0\epsilon}\frac{q_i q_j}{r_{ij}} \quad , \tag{2.202}$$

which is simply the negative of the potential term.

The contribution to be added to the atomic stress tensor is

$$\sigma^{\alpha\beta} = r_{ij}^\alpha f_j^\beta \quad , \tag{2.203}$$

where α, β are x, y, z components. The atomic stress tensor is symmetric.

In DL_POLY_4 these forces are handled by the subroutine COUL_CP_FORCES.

2.4.1.2 Force-Shifted Coulomb Sum

This form of the Coulomb sum has the advantage that it drastically reduces the range of electrostatic interactions, without giving rise to a violent step in the potential energy at the cutoff. Its main use is for preliminary preparation of systems and it is not recommended for realistic models.

The form of the simple truncated and shifted potential function is

$$U(r_{ij}) = \frac{q_i q_j}{4\pi\epsilon_0\epsilon}\left\{\frac{1}{r_{ij}} - \frac{1}{r_{\rm cut}}\right\} \quad , \tag{2.204}$$

with q_ℓ the charge on an atom labelled ℓ, $r_{\rm cut}$ the cutoff radius and r_{ij} the magnitude of the separation vector $\underline{r}_{ij} = \underline{r}_j - \underline{r}_i$.

A further refinement of this approach is to truncate the $1/r$ potential at $r_{\rm cut}$ and add a linear term to the potential in order to make both the energy and the force zero at the cutoff. This removes the heating effects that arise from the discontinuity in the forces at the cutoff in the simple truncated and shifted potential (the formula above). (The physics of this potential, however, is little better. It is only recommended for very crude structure optimizations.)

The force-shifted potential is thus

$$U(r_{ij}) = \frac{q_i q_j}{4\pi\epsilon_0\epsilon}\left[\left\{\frac{1}{r_{ij}} + \frac{1}{r_{\rm cut}^2}\, r_{ij}\right\} - \left\{\frac{1}{r_{\rm cut}} + \frac{1}{r_{\rm cut}^2}\, r_{\rm cut}\right\}\right] = \frac{q_i q_j}{4\pi\epsilon_0\epsilon}\left[\frac{1}{r_{ij}} + \frac{r_{ij}}{r_{\rm cut}^2} - \frac{2}{r_{\rm cut}}\right] \quad , \tag{2.205}$$

with the force on an atom j given by

$$\underline{f}_j = \frac{q_i q_j}{4\pi\epsilon_0\epsilon}\left[\frac{1}{r_{ij}^3} - \frac{1}{r_{ij} r_{\rm cut}^2}\right]\underline{r}_{ij} \quad , \tag{2.206}$$

with the force on atom i the negative of this.

The force-shifted Coulomb potential can be elegantly extended to emulate long-range ordering by including distance depending damping function $\mathrm{erfc}(\alpha\ r_{ij})$ (identical to that seen in the real-space portion of the Ewald sum) and thus mirror the effective charge screening [68] as shown below

$$U(r_{ij}) = \frac{q_i q_j}{4\pi\epsilon_0\epsilon}\left[\left\{\frac{\mathrm{erfc}(\alpha\ r_{ij})}{r_{ij}} + \left(\frac{\mathrm{erfc}(\alpha\ r_{\mathrm{cut}})}{r_{\mathrm{cut}}^2} + \frac{2\alpha}{\sqrt{\pi}}\ \frac{\exp(-\alpha^2\ r_{\mathrm{cut}}^2)}{r_{\mathrm{cut}}}\right)\ r_{ij}\right\} - \right.$$
$$\left.\left\{\frac{\mathrm{erfc}(\alpha\ r_{\mathrm{cut}})}{r_{\mathrm{cut}}} + \left(\frac{\mathrm{erfc}(\alpha\ r_{\mathrm{cut}})}{r_{\mathrm{cut}}^2} + \frac{2\alpha}{\sqrt{\pi}}\ \frac{\exp(-\alpha^2\ r_{\mathrm{cut}}^2)}{r_{\mathrm{cut}}}\right)\ r_{\mathrm{cut}}\right\}\right] \quad , \tag{2.207}$$

with the force on an atom j given by

$$\underline{f}_j = \frac{q_i q_j}{4\pi\epsilon_0\epsilon}\left[\left(\frac{\mathrm{erfc}(\alpha\ r_{ij})}{r_{ij}^2} + \frac{2\alpha}{\sqrt{\pi}}\ \frac{\exp(-\alpha^2\ r_{ij}^2)}{r_{ij}}\right) - \right.$$
$$\left.\left(\frac{\mathrm{erfc}(\alpha\ r_{\mathrm{cut}})}{r_{\mathrm{cut}}^2} + \frac{2\alpha}{\sqrt{\pi}}\ \frac{\exp(-\alpha^2\ r_{\mathrm{cut}}^2)}{r_{\mathrm{cut}}}\right)\right]\ \frac{\underline{r}_{ij}}{r_{ij}} \quad , \tag{2.208}$$

with the force on atom i the negative of this.

It is worth noting that, as discussed in [68] and references therein, this is only an approximation of the Ewald sum and its accuracy and effectiveness become better when the cutoff is large ($>$ 10 preferably 12 Å).

The contribution to the atomic virial is

$$\mathcal{W} = -\underline{r}_{ij} \cdot \underline{f}_j \quad , \tag{2.209}$$

which is *not* the negative of the potential term in this case.

The contribution to be added to the atomic stress tensor is given by

$$\sigma^{\alpha\beta} = r_{ij}^{\alpha} f_j^{\beta} \quad , \tag{2.210}$$

where α, β are x, y, z components. The atomic stress tensor is symmetric.

In DL_POLY_4 these forces are handled by the routine COUL_FSCP_FORCES.

2.4.1.3 Coulomb Sum with Distance Dependent Dielectric

This potential attempts to address the difficulties of applying the direct Coulomb sum, without the brutal truncation of the previous case. It hinges on the assumption that the electrostatic forces are effectively 'screened' in real systems - an effect which is approximated by introducing a dielectric term that increases with distance.

The interatomic potential for two charged ions is

$$U(r_{ij}) = \frac{1}{4\pi\epsilon_0\epsilon(r_{ij})}\frac{q_i q_j}{r_{ij}} \quad , \tag{2.211}$$

with q_ℓ the charge on an atom labelled ℓ, and r_{ij} the magnitude of the separation vector $\underline{r}_{ij} = \underline{r}_j - \underline{r}_i$. $\epsilon(r)$ is the distance dependent dielectric function. In DL_POLY_4 it is assumed that this function has the form

$$\epsilon(r) = \epsilon\ r \quad , \tag{2.212}$$

where ϵ is a constant. Inclusion of this term effectively accelerates the rate of convergence of the Coulomb sum.

The force on an atom j derived from this potential is

$$\underline{f}_j = \frac{1}{2\pi\epsilon_0\epsilon}\frac{q_i q_j}{r_{ij}^4}\underline{r}_{ij} \quad , \tag{2.213}$$

with the force on atom i the negative of this.

The contribution to the atomic virial is

$$\mathcal{W} = -\underline{r}_{ij} \cdot \underline{f}_j \quad , \tag{2.214}$$

which is -2 times the potential term.

The contribution to be added to the atomic stress tensor is given by

$$\sigma^{\alpha\beta} = r_{ij}^{\alpha} f_j^{\beta} \quad , \tag{2.215}$$

where α, β are x, y, z components. The atomic stress tensor is symmetric.

In DL_POLY_4 these forces are handled by the routine COUL_DDDP_FORCES.

2.4.1.4 Reaction Field

In the reaction field method it is assumed that any given molecule is surrounded by a spherical cavity of finite radius within which the electrostatic interactions are calculated explicitly. Outside the cavity the system is treated as a dielectric continuum. The occurrence of any net dipole within the cavity induces a polarisation in the dielectric, which in turn interacts with the given molecule. The model allows the replacement of the infinite Coulomb sum by a finite sum plus the reaction field.

The reaction field model coded into DL_POLY_4 is the implementation of Neumann based on charge-charge interactions [69]. In this model, the total coulombic potential is given by

$$U_c = \frac{1}{4\pi\epsilon_0\epsilon} \sum_{j<n} q_j q_n \left[\frac{1}{r_{nj}} + \frac{B_0 r_{nj}^2}{2R_c^3} \right] \quad , \tag{2.216}$$

where the second term on the right is the reaction field correction to the explicit sum, with R_c the radius of the cavity. The constant B_0 is defined as

$$B_0 = \frac{2(\epsilon_1 - 1)}{(2\epsilon_1 + 1)} \quad , \tag{2.217}$$

with ϵ_1 the dielectric constant outside the cavity. The effective pair potential is therefore

$$U(r_{ij}) = \frac{1}{4\pi\epsilon_0\epsilon} q_i q_j \left[\frac{1}{r_{ij}} + \frac{B_0 r_{ij}^2}{2R_c^3} \right] \quad . \tag{2.218}$$

This expression unfortunately leads to large fluctuations in the system coulombic energy, due to the large 'step' in the function at the cavity boundary. In DL_POLY_4 this is countered by subtracting the value of the potential at the cavity boundary from each pair contribution. The term subtracted is

$$\frac{1}{4\pi\epsilon_0\epsilon} \frac{q_i q_j}{R_c} \left[1 + \frac{B_0}{2} \right] \quad . \tag{2.219}$$

The effective pair force on an atom j arising from another atom n within the cavity is given by

$$\underline{f}_j = \frac{q_i q_j}{4\pi\epsilon_0\epsilon} \left[\frac{1}{r_{ij}^3} - \frac{B_0}{R_c^3} \right] \underline{r}_{ij} \quad . \tag{2.220}$$

In DL_POLY_4 the reaction field is optionally extended to emulate long-range ordering in a force-shifted manner by countering the reaction term and using a distance depending damping function erfc($\alpha\ r_{ij}$)

(identical to that seen in the real-space portion of the Ewald sum) and thus mirror the effective charge screening [68]:

$$U(r_{ij}) = \frac{q_i q_j}{4\pi\epsilon_0\epsilon} \left[\left\{ \frac{\mathrm{erfc}(\alpha\ r_{ij})}{r_{ij}} + \left(\frac{\mathrm{erfc}(\alpha\ r_{\mathrm{cut}})}{r_{\mathrm{cut}}^2} + \frac{2\alpha}{\sqrt{\pi}}\ \frac{\exp(-\alpha^2\ r_{\mathrm{cut}}^2)}{r_{\mathrm{cut}}} \right)\ r_{ij} \right\} - \right.$$
$$\left. \left\{ \frac{\mathrm{erfc}(\alpha\ r_{\mathrm{cut}})}{r_{\mathrm{cut}}} + \left(\frac{\mathrm{erfc}(\alpha\ r_{\mathrm{cut}})}{r_{\mathrm{cut}}^2} + \frac{2\alpha}{\sqrt{\pi}}\ \frac{\exp(-\alpha^2\ r_{\mathrm{cut}}^2)}{r_{\mathrm{cut}}} \right)\ r_{\mathrm{cut}} \right\} + \frac{B_0(r_{ij}^2 - r_{\mathrm{cut}}^2)}{2r_{\mathrm{cut}}^3} \right] \quad , \tag{2.221}$$

with the force on an atom j given by

$$\underline{f}_j = \frac{q_i q_j}{4\pi\epsilon_0\epsilon} \left[\left(\frac{\mathrm{erfc}(\alpha\ r_{ij})}{r_{ij}^2} + \frac{2\alpha}{\sqrt{\pi}}\ \frac{\exp(-\alpha^2\ r_{ij}^2)}{r_{ij}} \right) - \right.$$
$$\left. \left(\frac{\mathrm{erfc}(\alpha\ r_{\mathrm{cut}})}{r_{\mathrm{cut}}^2} + \frac{2\alpha}{\sqrt{\pi}}\ \frac{\exp(-\alpha^2\ r_{\mathrm{cut}}^2)}{r_{\mathrm{cut}}} \right) - \frac{B_0 r_{ij}}{r_{\mathrm{cut}}^3} \right] \frac{\underline{r}_{ij}}{r_{ij}} \quad , \tag{2.222}$$

with the force on atom i the negative of this.

It is worth noting that, as discussed in [68] and references therein, this is only an approximation of the Ewald sum and its accuracy and effectiveness become better when the cutoff is large (> 10 preferably 12 Å).

The contribution of each effective pair interaction to the atomic virial is

$$\mathcal{W} = -\underline{r}_{ij} \cdot \underline{f}_j \tag{2.223}$$

and the contribution to the atomic stress tensor is

$$\sigma^{\alpha\beta} = r_{ij}^{\alpha} f_j^{\beta} \quad , \tag{2.224}$$

where α, β are x, y, z components. The atomic stress tensor is symmetric.

In DL_POLY_4 the reaction field is handled by the subroutine COUL_RFP_FORCES.

2.4.1.5 Smoothed Particle Mesh Ewald

The Ewald sum [22] is the best technique for calculating electrostatic interactions in a periodic (or pseudo-periodic) system.

The basic model for a neutral periodic system is a system of charged point ions mutually interacting via the Coulomb potential. The Ewald method makes two amendments to this simple model. Firstly, each ion is effectively neutralised (at long-ranged) by the superposition of a spherical Gaussian cloud of opposite charge centred on the ion. The combined assembly of point ions and Gaussian charges becomes the *Real Space* part of the Ewald sum, which is now short ranged and treatable by the methods described above (Chapter 2)*. The second modification is to superimpose a second set of Gaussian charges, this time with the same charges as the original point ions and again centred on the point ions (so nullifying the effect of the first set of Gaussians). The potential due to these Gaussians is obtained from Poisson's equation and is solved as a Fourier series in *Reciprocal Space*. The complete Ewald sum requires an additional correction, known as the self energy correction, which arises from a Gaussian acting on its own site, and is constant. Ewald's method, therefore, replaces a potentially infinite sum in real space by two finite sums: one in real space and one in reciprocal space; and the self energy correction.

For molecular systems, as opposed to systems comprised simply of point ions, additional modifications EWALD_EXCL_FORCES are necessary to correct for the excluded (intra-molecular) coulombic interactions. In

* Strictly speaking, the real space sum ranges over all periodic images of the simulation cell, but in the DL_POLY_4 implementation, the parameters are chosen to restrict the sum to the simulation cell and its nearest neighbours, i.e. the *minimum images* of the cell contents.

the real space sum these are simply omitted. In reciprocal space however, the effects of individual Gaussian charges cannot easily be extracted, and the correction is made in real space. It amounts to removing terms corresponding to the potential energy of an ion ℓ due to the Gaussian charge on a neighbouring charge m (or *vice versa*). This correction appears in the term noting a summation over *molecules* in the full Ewald formula below.

The same considerations and modifications EWALD_FRZN_FORCES are taken into account for frozen atoms, which mutual coulombic interaction must also be excluded. This correction appears in the term noting a summation over F^* (all frozen-frozen pairs in the MD cell) in the full Ewald formula below.

Note the distinction between the *error function* **erf** and the more usual *complementary error function* **erfc** found in the real space sums below.

The total electrostatic energy is given by the following formula:

$$\begin{aligned} U_c &= \frac{1}{2V_o\epsilon_0\epsilon}\sum_{\underline{k}\neq\underline{0}}^{\infty}\frac{\exp(-k^2/4\alpha^2)}{k^2}\left|\sum_j^N q_j\exp(-i\underline{k}\cdot\underline{r}_j)\right|^2 - \frac{1}{4\pi\epsilon_0\epsilon}\frac{\alpha}{\sqrt{\pi}}\sum_j^N q_j^2 + \\ &\frac{1}{4\pi\epsilon_0\epsilon}\sum_{n<j}^{N^*}\frac{q_jq_n}{r_{nj}}\mathrm{erfc}(\alpha r_{nj}) - \frac{1}{4\pi\epsilon_0\epsilon}\sum_{molecules}\sum_{\ell\leq m}^{M^*} q_\ell q_m\left\{\delta_{\ell m}\frac{\alpha}{\sqrt{\pi}} + \frac{\mathrm{erf}(\alpha r_{\ell m})}{r_{\ell m}^{1-\delta_{\ell m}}}\right\} - \\ &\frac{1}{4\pi\epsilon_0\epsilon}\sum_{\ell\leq m}^{F^*} q_\ell q_m\left\{\delta_{\ell m}\frac{\alpha}{\sqrt{\pi}} + \frac{\mathrm{erf}(\alpha r_{\ell m})}{r_{\ell m}^{1-\delta_{\ell m}}}\right\} - \frac{1}{4\pi\epsilon_0\epsilon}\frac{\pi}{2V_o\alpha^2}\left\{\sum_j^N q_j\right\}^2 \quad , \end{aligned} \tag{2.225}$$

where N is the number of ions in the system and N^* the same number discounting any excluded (intramolecular and frozen) interactions. M^* represents the number of excluded atoms in a given molecule. F^* represents the number of frozen atoms in the MD cell. V_o is the simulation cell volume and $\underline{k}$ is a reciprocal lattice vector defined by

$$\underline{k} = \ell\underline{u} + m\underline{v} + n\underline{w} \quad , \tag{2.226}$$

where ℓ, m, n are integers and $\underline{u}, \underline{v}, \underline{w}$ are the *reciprocal space* basis vectors. Both V_o and $\underline{u}, \underline{v}, \underline{w}$ are derived from the vectors $(\underline{a}, \underline{b}, \underline{c})$ defining the simulation cell. Thus

$$V_o = |\underline{a}\cdot\underline{b}\times\underline{c}| \tag{2.227}$$

and

$$\begin{aligned} \underline{u} &= 2\pi\frac{\underline{b}\times\underline{c}}{\underline{a}\cdot\underline{b}\times\underline{c}} \\ \underline{v} &= 2\pi\frac{\underline{c}\times\underline{a}}{\underline{a}\cdot\underline{b}\times\underline{c}} \\ \underline{w} &= 2\pi\frac{\underline{a}\times\underline{b}}{\underline{a}\cdot\underline{b}\times\underline{c}} \quad . \end{aligned} \tag{2.228}$$

With these definitions, the Ewald formula above is applicable to general periodic systems. The last term in the Ewald formula above is the Fuchs correction [70] for electrically non-neutral MD cells which prevents the build-up of a charged background and the introduction of extra pressure due to it.

In practice the convergence of the Ewald sum is controlled by three variables: the real space cutoff r_{cut}; the convergence parameter α and the largest reciprocal space vector $\underline{k}_{max}$ used in the reciprocal space sum. These are discussed more fully in Section 8.3.5. DL_POLY_4 can provide estimates if requested (see CONTROL file description 10.1.1).

As its name implies the Smoothed Particle Mesh Ewald (SPME) method is a modification of the standard Ewald method. DL_POLY_4 implements the SPME method of Essmann *et al.* [71]. Formally, this method is capable of treating van der Waals forces also, but in DL_POLY_4 it is confined to electrostatic forces only. The main difference from the standard Ewald method is in its treatment of the reciprocal space terms. By means of an interpolation procedure involving (complex) B-splines, the sum in reciprocal space is represented on a three dimensional rectangular grid. In this form the Fast Fourier Transform (FFT) may be used to perform the primary mathematical operation, which is a 3D convolution. The efficiency of these procedures greatly reduces the cost of the reciprocal space sum when the range of $\underline{k}$ vectors is large. The method (briefly) is as follows (for full details see [71]):

1. Interpolation of the $\exp(-i\ \underline{k}\cdot\underline{r}_j)$ terms (given here for one dimension):

$$\exp(2\pi i\ u_j k/L) \approx b(k) \sum_{\ell=-\infty}^{\infty} M_n(u_j-\ell)\ \exp(2\pi i\ k\ell/K) \quad , \tag{2.229}$$

in which k is the integer index of the $\underline{k}$ vector in a principal direction, K is the total number of grid points in the same direction and u_j is the fractional coordinate of ion j scaled by a factor K (i.e. $u_j = K s_j^x$) . **Note** that the definition of the B-splines implies a dependence on the integer K, which limits the formally infinite sum over ℓ. The coefficients $M_n(u)$ are B-splines of order n and the factor $b(k)$ is a constant computable from the formula:

$$b(k) = \exp(2\pi i\ (n-1)k/K) \left[\sum_{\ell=0}^{n-2} M_n(\ell+1)\ \exp(2\pi i\ k\ell/K)\right]^{-1} \quad . \tag{2.230}$$

2. Approximation of the structure factor $S(\underline{k})$:

$$S(\underline{k}) \approx b_1(k_1)\ b_2(k_2)\ b_3(k_3)\ Q^\dagger(k_1,k_2,k_3) \quad , \tag{2.231}$$

where $Q^\dagger(k_1,k_2,k_3)$ is the discrete Fourier transform of the *charge array* $Q(\ell_1,\ell_2,\ell_3)$ defined as

$$\begin{aligned} Q(\ell_1,\ell_2,\ell_3) \quad = \quad & \sum_{j=1}^{N} q_j \sum_{n_1,n_2,n_3} M_n(u_{1j}-\ell_1-n_1L_1) \ \times\ M_n(u_{2j}-\ell_2-n_2L_2)\ \times \\ & M_n(u_{3j}-\ell_3-n_3L_3) \quad , \end{aligned} \tag{2.232}$$

in which the sums over $n_{1,2,3}$ etc are required to capture contributions from all relevant periodic cell images (which in practice means the nearest images).

3. Approximating the reciprocal space energy U_{recip}:

$$U_{recip} = \frac{1}{2V_o\epsilon_0\epsilon} \sum_{k_1,k_2,k_3} G^\dagger(k_1,k_2,k_3)\ Q(k_1,k_2,k_3) \quad , \tag{2.233}$$

where $G^\dagger$ is the discrete Fourier transform of the function

$$G(k_1,k_2,k_3) = \frac{\exp(-k^2/4\alpha^2)}{k^2}\ B(k_1,k_2,k_3)\ (Q^\dagger(k_1,k_2,k_3))^* \quad , \tag{2.234}$$

in which $(Q^\dagger(k_1,k_2,k_3))^*$ is the complex conjugate of $Q^\dagger(k_1,k_2,k_3)$ and

$$B(k_1,k_2,k_3) = |b_1(k_1)|^2\ |b_2(k_2)|^2\ |b_3(k_3)|^2 \quad . \tag{2.235}$$

The function $G(k_1,k_2,k_3)$ is thus a relatively simple product of the Gaussian screening term appearing in the conventional Ewald sum, the function $B(k_1,k_2,k_3)$ and the discrete Fourier transform of $Q(k_1,k_2,k_3)$.

4. Calculating the atomic forces, which are given formally by:

$$f_j^\alpha = -\frac{\partial U_{recip}}{\partial r_j^\alpha} = -\frac{1}{V_o\epsilon_0\epsilon} \sum_{k_1,k_2,k_3} G^\dagger(k_1,k_2,k_3)\ \frac{\partial Q(k_1,k_2,k_3)}{\partial r_j^\alpha} \quad . \tag{2.236}$$

Fortunately, due to the recursive properties of the B-splines, these formulae are easily evaluated.

The virial and the stress tensor are calculated in the same manner as for the conventional Ewald sum.

The DL_POLY_4 subroutines required to calculate the SPME contributions are:

1. SPME_CONTAINER containing
   (a) BSPGEN, which calculates the B-splines
   (b) BSPCOE, which calculates B-spline coefficients
   (c) SPL_CEXP, which calculates the FFT and B-spline complex exponentials
2. PARALLEL_FFT and GPFA_MODULE (native DL_POLY_4 subroutines that respect the domain decomposition concept) which calculate the 3D complex fast Fourier transforms
3. EWALD_SPME_FORCES, which calculates the reciprocal space contributions (uncorrected)
4. EWALD_REAL_FORCES, which calculates the real space contributions (corrected)
5. EWALD_EXCL_FORCES, which calculates the reciprocal space corrections due to the coulombic exclusions in intramolecular interactions
6. EWALD_FRZN_FORCES, which calculates the reciprocal space corrections due to the exclusion interactions between frozen atoms
7. TWO_BODY_FORCES, in which all of the above subroutines are called sequentially and also the Fuchs correction [70] for electrically non-neutral MD cells is applied if needed.

2.4.2 Multipolar Electrostatics

DL_POLY_4 offers advanced potential energy calculations through multipolar electrostatics. This is an extension to the point-charge model where the charge density of chemical species are described by higher order point multipoles. The generic algorithms in DL_POLY_4 are designed to allow for arbitrary order [72] multipoles but for practical reasons the functionality is limited to hexadecapoles only.

Multipoles

Define the multipolar operator, $\hat{L}_i$ as

$$\hat{L}_i = (q_i + \mathbf{p}_i \cdot \nabla_i + \mathbf{Q}_i : \nabla_i\nabla_i + \mathbf{O}_i \vdots \nabla_i\nabla_i\nabla_i + \mathbf{H}_i :: \nabla_i\nabla_i\nabla_i\nabla_i + \dots) \;\;, \tag{2.237}$$

where q_i, $\mathbf{p}_i$, $\mathbf{Q}_i$, $\mathbf{O}_i$, and $\mathbf{H}_i$ are the point charge, dipole, quadrupole, octupole, and hexadecapole tensors, respectively of atom i, ∇_i refers to the three-dimensional gradient with respect to the position of atom i and the "dot" products stand for tensor contraction. By defining a unidimensional vector of independent (non-degenerate) multipole moments, $\mathcal{M}_i$, for atom i, the corresponding multipolar operator to an arbitrary order p can be written in a more compact form as

$$\hat{L}_i = \sum_{||\mathbf{s}||=0}^{p} \mathcal{M}_i^{\mathbf{s}} \partial_i^{\mathbf{s}} = \sum_{s_3=0}^{p} \sum_{s_2=0}^{p-s_3} \sum_{s_1=0}^{p-s_3-s_2} \mathcal{M}_j^{s_1 s_2 s_3} \partial_{z_i}^{s_3} \partial_{y_i}^{s_2} \partial_{x_i}^{s_1} \;\;. \tag{2.238}$$

Here, $\mathbf{s} = (s_1, s_2, s_3)$ is the triplet that runs over all independent multipoles, $||\mathbf{s}|| = s_1+s_2+s_3$, $\mathcal{M}_i^{\mathbf{s}} = \mathcal{M}_i^{s_1 s_2 s_3}$ and $\partial_i^{\mathbf{s}} = \partial_{z_i}^{s_3} \partial_{y_i}^{s_2} \partial_{x_i}^{s_1}$ is the multidimensional derivative with respect to the position $\langle x_i, y_i, z_i \rangle$ of atom i with orders s_1, s_2 and s_3 in the x, y and z directions respectively. Individual components of $\mathcal{M}$ contain the sum of all degenerate original multipole components. As an example, the octupole $\mathcal{M}^{111}$, is a sum of all six degenerate original octupole components formed from the permutation of the triplet $\{x, y, z\}$. If the

original octupole vector with degnerate components is labelled as O', then
$\mathcal{M}^{111} = O'_{xyz} + O'_{xzy} + O'_{yxz} + O'_{yzx} + O'_{zxy} + O'_{zyx} = 6\ O'_{xyz}$. For pair potentials it is often convenient to redefine the multipolar operator for atom j in terms of the derivatives with respect to the position of atom i to arrive at

$$\hat{L}_{j_i} = \sum_{||\mathbf{s}||=0}^{p} \mathcal{M}_j^{\mathbf{s}} \partial_j^{\mathbf{s}} = \sum_{||\mathbf{s}||=0}^{p} (-1)^{||\mathbf{s}||} \mathcal{M}_j^{\mathbf{s}} \partial_i^{\mathbf{s}} = \sum_{s_3=0}^{p} \sum_{s_2=0}^{p-s_3} \sum_{s_1=0}^{p-s_3-s_2} (-1)^{s_1+s_2+s_3} \mathcal{M}_j^{s_1 s_2 s_3} \partial_{z_i}^{s_3} \partial_{y_i}^{s_2} \partial_{x_i}^{s_1} \quad . \tag{2.239}$$

Application to Pair Potentials

In DL_POLY_4 for N point-multipoles interacting via a pair potential function ψ, the multipolar electrostatic potential at position $\mathbf{r_i}$ is computed as

$$\phi(\mathbf{r_i}) = \sum_{j\neq i}^{N} \hat{L}_{j_i} \psi(\mathbf{r_{ji}}) = \sum_{j\neq i}^{N} \sum_{\mathbf{s}=\mathbf{0}}^{p} (-1)^{||\mathbf{s}||} \mathcal{M}_j^{\mathbf{s}} \partial_i^{\mathbf{s}} \psi(r_{ij}) \quad , \tag{2.240}$$

the electrostatic field at $\mathbf{r_i}$ is

$$\mathbf{E}(\mathbf{r_{ij}}) = -\nabla_i \phi(r_{ij}) = -\sum_{j\neq i}^{N} \sum_{\mathbf{s}=\mathbf{0}}^{p} (-1)^{||\mathbf{s}||} \mathcal{M}_j^{\mathbf{s}} \begin{bmatrix} \partial_i^{\mathbf{s}+\mathbf{e}_1} \\ \partial_i^{\mathbf{s}+\mathbf{e}_2} \\ \partial_i^{\mathbf{s}+\mathbf{e}_3} \end{bmatrix} \psi(r_{ij}) \quad , \tag{2.241}$$

where $\mathbf{e}_1 = \langle 1,0,0 \rangle$, $\mathbf{e}_2 = \langle 0,1,0 \rangle$, and $\mathbf{e}_3 = \langle 0,0,1 \rangle$ and the torque [73] on particle i in the α-direction, $\tau_{i,\alpha}$, is obtained as

$$\tau_{i,\alpha} = \sum_{\mathbf{s}=\mathbf{0}}^{p} \mathcal{M}_{i,\alpha}^{\mathbf{s}} \partial_i^{\mathbf{s}} \phi(\mathbf{r_{ij}}) = \sum_{\mathbf{s}=\mathbf{0}}^{p} \mathcal{M}_{i,\alpha}^{\mathbf{s}} \sum_{j\neq i}^{N} \sum_{\mathbf{k}=\mathbf{0}}^{p} (-1)^{||\mathbf{k}||} \mathcal{M}_j^{\mathbf{k}} \partial_i^{\mathbf{s}+\mathbf{k}} \psi(r_{ij}) \quad , \tag{2.242}$$

where $\mathcal{M}_{i,\alpha}$ is the infinitesimal counter-clockwise rotation of multipole vector $\mathcal{M}_i$ about the α-axis. The total electrostatic potential energy is given by

$$U = \sum_{i<j}^{N} \hat{L}_i \hat{L}_{j_i} \psi(r_{ij}) = \sum_{i<j}^{N} \sum_{\mathbf{s}=\mathbf{0}}^{p} (-1)^{||\mathbf{s}||} \mathcal{M}_j^{\mathbf{s}} \sum_{\mathbf{k}=\mathbf{0}}^{p} \mathcal{M}_i^{\mathbf{k}} \partial_i^{\mathbf{s}+\mathbf{k}} \psi(r_{ij}) \quad , \tag{2.243}$$

where $\mathbf{s}+\mathbf{k} = (s_1+k_1, s_2+k_2, s_3+k_3)$ and the force on atom i is

$$\mathbf{f}_i = -\nabla_i \sum_{j\neq i}^{N} \hat{L}_i \hat{L}_{j_i} \psi(r_{ij}) = -\sum_{j\neq i}^{N} \sum_{\mathbf{s}=\mathbf{0}}^{p} (-1)^{||\mathbf{s}||} \mathcal{M}_j^{\mathbf{s}} \sum_{\mathbf{k}=\mathbf{0}}^{p} \mathcal{M}_i^{\mathbf{k}} \begin{bmatrix} \partial_i^{\mathbf{s}+\mathbf{k}+\mathbf{e}_1} \\ \partial_i^{\mathbf{s}+\mathbf{k}+\mathbf{e}_2} \\ \partial_i^{\mathbf{s}+\mathbf{k}+\mathbf{e}_3} \end{bmatrix} \psi(r_{ij}) \quad . \tag{2.244}$$

To implement equations (2.240)-(2.244) for the variety of potentials in DL_POLY_4 a number of recurrence relations are used to compute the multi-dimensional derivatives of the kernels corresponding to the potentials. These kernels are

$$\theta(|\mathbf{x}|) = \frac{1}{|\mathbf{x}|^{\nu}}, \quad \Omega(|\mathbf{x}|) = \frac{1}{2}\exp(-\alpha^2|\mathbf{x}|^2), \quad \psi(|\mathbf{x}|) = \frac{\sqrt{\pi}}{2}\frac{\mathrm{erfc}(\alpha|\mathbf{x}|)}{|\mathbf{x}|}, \quad \text{and} \quad \Gamma(|\bar{\mathbf{x}}|) = \frac{\sqrt{\pi}}{2}\frac{\mathrm{erf}(\alpha|\mathbf{x}|)}{|\mathbf{x}|} \quad ; \tag{2.245}$$

with

$$a_{\mathbf{s}}(\nu) = \frac{\partial^{||\mathbf{s}||}\theta(|\mathbf{x}|)}{\partial_{x_1}^{s_1}\partial_{x_2}^{s_2}\partial_{x_3}^{s_3}}, \quad b_{\mathbf{s}} = \frac{\partial^{||\mathbf{s}||}\Omega(|\mathbf{x}|)}{\partial_{x_1}^{s_1}\partial_{x_2}^{s_2}\partial_{x_3}^{s_3}}, \quad c_{\mathbf{s}} = \frac{\partial^{||\mathbf{s}||}\psi(|\mathbf{x}|)}{\partial_{x_1}^{s_1}\partial_{x_2}^{s_2}\partial_{x_3}^{s_3}}, \quad \text{and} \quad d_{\mathbf{s}} = \frac{\partial^{||\mathbf{s}||}\Gamma(|\bar{\mathbf{x}}|)}{\partial_{x_1}^{s_1}\partial_{x_2}^{s_2}\partial_{x_3}^{s_3}} \quad . \tag{2.246}$$

The recurrence relations used in DL_POLY_4 are

$$a_{\mathbf{s}}(\nu) = \frac{1}{|\mathbf{x}|^2}\left\{\left(\frac{2-\nu}{||\mathbf{s}||}-2\right)\sum_{i=1}^{3} s_i x_i a_{\mathbf{s}-\mathbf{e}_i} + \left(\frac{2-\nu}{||\mathbf{s}||}-1\right)\sum_{i=1}^{3} s_i(s_i-1) a_{\mathbf{s}-2\mathbf{e}_i}\right\} \quad , \tag{2.247}$$

$$b_{\mathbf{s}} = \frac{-2\alpha^2}{||\mathbf{s}||} \sum_{i=1}^{3} \left[s_i x_i b_{\mathbf{s}-\mathbf{e}_i} + s_i(s_i-1) b_{\mathbf{s}-2\mathbf{e}_i} \right] \quad , \tag{2.248}$$

$$c_{\mathbf{s}} = \frac{1}{|\mathbf{x}|^2} \left\{ \left(\frac{1}{||\mathbf{s}||} - 2 \right) \sum_{i=1}^{3} s_i x_i c_{\mathbf{s}-\mathbf{e}_i} + \left(\frac{1}{||\mathbf{s}||} - 1 \right) \sum_{i=1}^{3} s_i(s_i-1) c_{\mathbf{s}-2\mathbf{e}_i} + \frac{1}{\alpha} b_{\mathbf{s}} \right\} \quad , \tag{2.249}$$

and

$$d_{\mathbf{s}} = \frac{1}{|\mathbf{x}|^2} \left\{ \left(\frac{1}{||\mathbf{s}||} - 2 \right) \sum_{i=1}^{3} s_i x_i d_{\mathbf{s}-\mathbf{e}_i} + \left(\frac{1}{||\mathbf{s}||} - 1 \right) \sum_{i=1}^{3} s_i(s_i-1) d_{\mathbf{s}-2\mathbf{e}_i} - \frac{1}{\alpha} b_{\mathbf{s}} \right\} \quad . \tag{2.250}$$

2.4.2.1 Direct Coulomb Sum

For two interacting ions i and j, the potential energy is given as

$$U(r_{ij}) = \frac{1}{4\pi\epsilon_0\epsilon} \hat{L}_i \hat{L}_{j_i} \left[\frac{1}{r_{ij}} \right] \quad , \tag{2.251}$$

and the relevant kernel is $\psi(r_{ij}) = \frac{1}{r_{ij}}$. The derivatives for this kernel are obtained by using equation (2.247) with $\nu = 1$. Thus,

$$\partial_i^{\mathbf{s}} \psi(r_{ij}) = a_{\mathbf{s}}(1) \quad . \tag{2.252}$$

In DL_POLY_4 the multipolar direct Coulomb sum is handled by the routine COUL_CP_MFORCES.

2.4.2.2 Force-Shifted Coulomb Sum

DL_POLY_4 employs two forms of the force-shifted Coulomb sum. In the first form, the potential energy due to two interacting ions i and j is

$$U(r_{ij}) = \frac{1}{4\pi\epsilon_0\epsilon} \hat{L}_i \hat{L}_{j_i} \left[\frac{1}{r_{ij}} + \frac{r_{ij}}{r_{\text{cut}}^2} - \frac{2}{r_{\text{cut}}} \right] \quad , \tag{2.253}$$

where r_{cut} is the cutoff radius. The kernel is $\psi(r_{ij}) = \frac{1}{r_{ij}} + \frac{r_{ij}}{r_{\text{cut}}^2} - \frac{2}{r_{\text{cut}}}$. The last term, $\frac{2}{r_{\text{cut}}}$, is a constant which has a zero derivative, hence the derivatives of the kernel are obtained as a sum of the derivatives of the first term and second terms. Thus,

$$\partial_i^{\mathbf{s}} \psi(r_{ij}) = a_{\mathbf{s}}(1) + \frac{a_{\mathbf{s}}(-1)}{r_{\text{cut}}^2} \quad . \tag{2.254}$$

The potential energy due to two point-multipoles i and j interacting via the second form of the force-shifted Coulomb sum is

$$\begin{aligned} U(r_{ij}) \; = \; & \frac{1}{4\pi\epsilon_0\epsilon} \hat{L}_i \hat{L}_{j_i} \left[\left\{ \frac{\text{erfc}(\alpha \cdot r_{ij})}{r_{ij}} + \left(\frac{\text{erfc}(\alpha \cdot r_{\text{cut}})}{r_{\text{cut}}^2} + \frac{2\alpha}{\sqrt{\pi}} \frac{\exp(-\alpha^2 r_{\text{cut}}^2)}{r_{\text{cut}}} \right) r_{ij} \right\} - \right. \\ & \left. \left\{ \frac{\text{erfc}(\alpha \cdot r_{\text{cut}})}{r_{\text{cut}}} + \left(\frac{\text{erfc}(\alpha \cdot r_{\text{cut}})}{r_{\text{cut}}^2} + \frac{2\alpha}{\sqrt{\pi}} \frac{\exp(-\alpha^2 r_{\text{cut}}^2)}{r_{\text{cut}}} \right) r_{\text{cut}} \right\} \right] \quad . \end{aligned} \tag{2.255}$$

The kernel, $\psi(r_{ij})$ is the terms in the square bracket but the only terms which contribute to the derivatives are the first and second terms which are functions of r_{ij} . The derivative of the first term is obtained from equations (2.249) and the derivative for r_{ij} in the second term is given by $d_{\mathbf{s}}(-1)$. Thus,

$$D_i^{\mathbf{s}} \psi(r_{ij}) = \frac{2}{\sqrt{\pi}} c_{\mathbf{s}} + \left(\frac{\text{erfc}(\alpha \cdot r_{\text{cut}})}{r_{\text{cut}}^2} + \frac{2\alpha}{\sqrt{\pi}} \frac{\exp(-\alpha^2 r_{\text{cut}}^2)}{r_{\text{cut}}} \right) \cdot a_{\mathbf{s}}(-1) \quad . \tag{2.256}$$

In DL_POLY_4 the multipolar force-shifted Coulomb sum is handled by the routine COUL_FSCP_MFORCES

2.4.2.3 Coulomb Sum with Distance Dependent Dielectric

The potential energy between two interacting ions i and j is

$$U(r_{ij}) = \frac{1}{4\pi\epsilon_0\epsilon}\hat{L}_i\hat{L}_{j_i}\left[\frac{1}{r_{ij}^2}\right] \quad , \tag{2.257}$$

and the kernel is $\psi(r_{ij}) = \frac{1}{r_{ij}^2}$. The derivatives for this kernel are obtained by using equation (2.247) with $\nu = 2$. Hence,

$$\partial_i^{\mathbf{s}}\psi(r_{ij}) = a_{\mathbf{s}}(2) \quad . \tag{2.258}$$

In DL_POLY_4 the multipolar Coulomb sum with distance dependent dielectric is handled by the routine COUL_DDDP_MFORCES.

2.4.2.4 Reaction Field

DL_POLY_4 provides two forms of a multipolar reaction field potential. In the first form, the effective pair potential energy due to two interacting point multipoles i and j is given as

$$U(r_{ij}) = \frac{1}{4\pi\epsilon_0\epsilon}\hat{L}_i\hat{L}_{j_i}\left[\frac{1}{r_{ij}} + \frac{B_0 r_{ij}^2}{2R_c^3} - 1 - \frac{B_0}{2}\right] \quad , \tag{2.259}$$

where

$$B_0 = \frac{2(\epsilon_1 - 1)}{(2\epsilon_1 + 1)} \quad , \tag{2.260}$$

R_c is the radius of the spherical cavity and ϵ_1 is the dielectric constant outside the cavity. Again the kernel $\psi(r_{ij})$ is the terms in the square bracket and only the first and second terms contribute to its derivatives. The derivatives of the first and second terms are given by equation (2.247) with $\nu = 1$ and $\nu = -2$ respectively. Thus,

$$\partial_i^{\mathbf{s}}\psi(r_{ij}) = a_{\mathbf{s}}(1) + \frac{B_0}{2R_c^3}\cdot a_{\mathbf{s}}(-2) \quad . \tag{2.261}$$

The second form of the reaction field method is similar to that of the force-shifted Coulomb sum. The potential energy due to interacting ions i and j is

$$\begin{aligned} U(r_{ij}) \;=\; & \frac{1}{4\pi\epsilon_0\epsilon}\hat{L}_i\hat{L}_{j_i}\left[\left\{\frac{\mathrm{erfc}(\alpha\cdot r_{ij})}{r_{ij}} + \left(\frac{\mathrm{erfc}(\alpha\cdot r_{\mathrm{cut}})}{r_{\mathrm{cut}}^2} + \frac{2\alpha}{\sqrt{\pi}}\frac{\exp(-\alpha^2 r_{\mathrm{cut}}^2)}{r_{\mathrm{cut}}}\right) r_{ij}\right\} - \right. \\ & \left.\left\{\frac{\mathrm{erfc}(\alpha\cdot r_{\mathrm{cut}})}{r_{\mathrm{cut}}} + \left(\frac{\mathrm{erfc}(\alpha\cdot r_{\mathrm{cut}})}{r_{\mathrm{cut}}^2} + \frac{2\alpha}{\sqrt{\pi}}\frac{\exp(-\alpha^2 r_{\mathrm{cut}}^2)}{r_{\mathrm{cut}}}\right) r_{\mathrm{cut}}\right\} - \frac{B_0 r_{\mathrm{cut}}^2}{2r_{\mathrm{cut}}^3} + \frac{B_0 r_{ij}^2}{2r_{\mathrm{cut}}^3}\right] \quad . \end{aligned} \tag{2.262}$$

The kernel, $\psi(r_{ij})$ is the terms in the square bracket and the only terms which contribute to the derivatives are the first, second and last terms which are functions of r_{ij} . The derivative of the first term is obtained from equation (2.249) and the derivative for r_{ij} in the second term is given by $a_{\mathbf{s}}(-1)$ and the derivative for r_{ij}^2 in the last term is given by $d_{\mathbf{s}}(-2)$. Thus,

$$D_i^{\mathbf{s}}\psi(r_{ij}) = \frac{2}{\sqrt{\pi}}c_{\mathbf{s}} + \left(\frac{\mathrm{erfc}(\alpha\cdot r_{\mathrm{cut}})}{r_{\mathrm{cut}}^2} + \frac{2\alpha}{\sqrt{\pi}}\frac{\exp(-\alpha^2 r_{\mathrm{cut}}^2)}{r_{\mathrm{cut}}}\right)\cdot a_{\mathbf{s}}(-1) + \frac{B_0}{2r_{\mathrm{cut}}^3}\cdot a_{\mathbf{s}}(-2) \quad . \tag{2.263}$$

In DL_POLY_4 the multipolar reaction field is handled by the routine COUL_RFP_MFORCES.

2.4.2.5 Smoothed Particle Mesh Ewald

DL_POLY_4 provides two different smooth particle Mesh Ewald implementations for multipolar electrostatics. The first implementation is for systems with charges, dipoles and quadrupoles and does not use recurrence relations. The second implementation, which uses recurrence relations, is more general and allows for specification of an arbitrary order up to hexadecapoles.

When the multipolar form of SPME is employed, the total electrostatic energy for a system on N point ions is given as

$$U_c = U_{\rm dir} + U_{\rm rec} - U_{\rm excl} - U_{\rm frzn} - U_{\rm self} \quad , \tag{2.264}$$

where

$$U_{\rm dir} = \sum_{i<j}^{N^*} \sum_{\mathbf{n}}^{\prime} \hat{L}_i \hat{L}_{j_i} \frac{{\rm erfc}(\alpha \cdot |\mathbf{r_{ij}} + \mathbf{n}|)}{4\pi\epsilon_0\epsilon |\mathbf{r_{ij}} + \mathbf{n}|} \quad , \tag{2.265}$$

$$U_{\rm excl} = \frac{1}{4\pi\epsilon_0\epsilon} \sum_{(i,j)\in M^*} \hat{L}_i \hat{L}_{j_i} \frac{{\rm erf}(\alpha \cdot r_{ij})}{r_{ij}} \quad , \tag{2.266}$$

$$U_{\rm frzn} = \frac{1}{4\pi\epsilon_0\epsilon} \sum_{(i,j)\in F^*} \hat{L}_i \hat{L}_{j_i} \frac{{\rm erf}(\alpha \cdot r_{ij})}{r_{ij}} \quad , \tag{2.267}$$

$$U_{\rm self} = \frac{1}{8\pi\epsilon_0\epsilon} \lim_{|\mathbf{r_i}|\to 0} \sum_{i=1}^{N} \hat{L}_i \hat{L}_i \frac{{\rm erf}(\alpha \cdot |\mathbf{r_i}|)}{|\mathbf{r_i}|} \quad , \tag{2.268}$$

and

$$U_{\rm rec} = \frac{1}{2V_o\epsilon_0\epsilon} \sum_{\mathbf{k}\neq 0} \frac{\exp(-k^2/4\alpha^2)}{k^2} |S(\mathbf{k})|^2 \quad , \tag{2.269}$$

with

$$S(\mathbf{k}) = \sum_{i=1}^{N} \hat{L}_i \exp(\imath \mathbf{k} \cdot \mathbf{r_i}) \quad . \tag{2.270}$$

In the expressions above, M^* is the set of all excluded interactions due to intramolecular bonds in the simulation cell, F^* the set of frozen-frozen interactions in the simulation cell, $N^* = N - M^* - F^*$, V_o is the volume of the simulation cell and $S(\mathbf{k})$ is the structure factor.

Real Space Sum

The relevant kernel for the real space from equation (2.265) is $\psi(r_{ij}) = \frac{{\rm erfc}(\alpha|\mathbf{r_{ij}} + \mathbf{n}|)}{|\mathbf{r_{ij}} + \mathbf{n}|}$. DL_POLY_4 uses the recurrence giving in equation (2.249) to generate the multidimensional derivatives of the kernel. Thus, the derivatives of the kernel are computed as

$$\mathbf{D}_i^{\mathbf{s}} \psi(r_{ij}) = \frac{2}{\sqrt{\pi}} c_{\mathbf{s}} \quad . \tag{2.271}$$

In DL_POLY_4 the routine EWALD_REAL_MFORCES_D computes the real space interactions explicitly for simulations with multipoles of order 2 without using the recurrence relation. The routine EWALD_REAL_MFORCES handles the general version of up to order 4 using recurrence relations.

Excluded Sum

The relevant kernel for the real space from equation (2.266) is $\psi(r_{ij}) = \dfrac{\mathrm{erf}(\alpha \cdot r_{ij})}{r_{ij}}$. DL_POLY_4 uses the recurrence giving in equation (2.250) to generate the multidimensional derivatives of the kernel. Thus, the derivatives of the kernel are computed as

$$\mathbf{D}_i^{\mathbf{s}}\psi(r_{ij}) = \frac{2}{\sqrt{\pi}} d_{\mathbf{s}} \quad . \tag{2.272}$$

In DL_POLY_4 the routine EWALD_EXCL_MFORCES_D computes the reciprocal space corrections due to the exclusions between intramolecularly related atoms explicitly for simulations with multipoles of order 2 without using the recurrence relation. The routine EWALD_EXCL_MFORCES handles the general version of up to order 4 using recurrence relations.

Frozen Sum

The relevant kernel for the real space from equation (2.267) is $\psi(r_{ij}) = \dfrac{\mathrm{erf}(\alpha \cdot r_{ij})}{r_{ij}}$. DL_POLY_4 uses the recurrence giving in equation (2.250) to generate the multidimensional derivatives of the kernel. Thus, the derivatives of the kernel are computed as

$$\mathbf{D}_i^{\mathbf{s}}\psi(r_{ij}) = \frac{2}{\sqrt{\pi}} d_{\mathbf{s}} \quad . \tag{2.273}$$

In DL_POLY_4 the routine EWALD_FRZN_MFORCES computes computes the reciprocal space corrections due to the exclusions between frozen atoms generically for simulations with multipoles up to order 4 using recurrence relations.

Self-Interaction

DL_POLY_4 computes U_{self} directly for interactions involving multipoles up to order 4 using the series representation of the kernel $\psi(r_{ij}) = \dfrac{\mathrm{erf}(\alpha \cdot r_i)}{r_i}$. The self interaction is computed in EWALD_REAL_MFORCES_D for simulations with multipoles of maximum order 2. For simulations of arbitrary order, the self-interaction is computed in the reciprocal space.

Reciprocal Space Sum

The key idea of SPME is in approximating the structure factor, in a uniform grid, with $K_1 \times K_2 \times K_3$ dimensions, that fills the simulation cell. Define the fractional coordinates of an ion i as
$\langle s_{i_1}, s_{i_2}, s_{i_3} \rangle = \langle \mathbf{a}_1^* \cdot \mathbf{r_i}, \mathbf{a}_2^* \cdot \mathbf{r_i}, \mathbf{a}_3^* \cdot \mathbf{r_i} \rangle$, $u_{\alpha_i} = K_\alpha \cdot s_i^\alpha$ and M_n is a B-spline of order n then the approximation of the structure factor is given as

$$S(\mathbf{k}) \approx b_1(k_1) b_2(k_2) b_3(k_3) Q^{\mathcal{F}}(k_1, k_2, k_3) \quad , \tag{2.274}$$

where $\mathbf{k} = \langle k_1, k_2, k_3 \rangle$ is a reciprocal space vector,

$$b_i(k_i) = \exp(2\pi\imath(n-1)k_i/K_i) \left[\sum_{l=0}^{n-2} M_n(l+1) \exp(2\pi\imath k l / K_i) \right]^{-1} \quad , \tag{2.275}$$

Q is the multipolar array defined on the uniform grid and $Q^{\mathcal{F}}$ its discrete Fourier transform. At position (l_1, l_2, l_3) on the grid, the multipolar array is defined by

$$Q(l_1, l_2, l_3) = \sum_{i=1}^{N} \hat{L}_i \sum_{n_1, n_2, n_3} M_n(u_{1_i} - l_1 - n_1 K_1) \times M_n(u_{2_i} - l_2 - n_2 K_2) \times M_n(u_{3_i} - l_3 - n_3 K_3) \quad , \tag{2.276}$$

where, $u_{\alpha_i} - l_\alpha - n_\alpha K_\alpha$ are evaluation points of the B-spline on the grid that spans the fundamental cell and the periodic images. Then from equation (2.238) and considering only the fundamental cell, the multipolar array can be written explicitly as

$$Q(l_1, l_2, l_3) = \sum_{i=1}^{N} \sum_{s_3=0}^{p} \sum_{s_2=0}^{p-s_3} \sum_{s_1=0}^{p-s_3-s_2} \mathcal{M}_i^{s_1 s_2 s_3} \partial_{z_i}^{s_3} \partial_{y_i}^{s_2} \partial_{x_i}^{s_1} \left\{ M_n(u_{1_i} - l_1) M_n(u_{2_i} - l_2) M_n(u_{3_i} - l_3) \right\} \quad . \tag{2.277}$$

To compute the arbitrary order multidimensional derivatives of the product of three b-splines in equation (2.277), DL_POLY_4 uses the closed form formula:

$$\begin{aligned} &\partial_{z_i}^{s_3} \partial_{y_i}^{s_2} \partial_{x_i}^{s_1} \left\{ M_n(u_{1_i} - l_1) M_n(u_{2_i} - l_2) M_n(u_{3_i} - l_3) \right\} = \\ &\sum_{k_3=0}^{s_3} (K_1 a_{13}^*)^{k_3} \binom{s_3}{k_3} \sum_{k_2=0}^{s_2} (K_1 a_{12}^*)^{k_2} \binom{s_2}{k_2} \sum_{k_1=0}^{s_1} (K_1 a_{11}^*)^{k_1} \binom{s_1}{k_1} \partial_{u_{1_i}}^{||\mathbf{k}||} M_n(u_{1_i} - l_1) \times \\ &\sum_{j_3=0}^{s_3-k_3} (K_2 a_{23}^*)^{j_3} (K_3 a_{33}^*)^{s_3-k_3-j_3} \binom{s_3-k_3}{j_3} \sum_{j_2=0}^{s_2-k_2} (K_2 a_{22}^*)^{j_2} (K_3 a_{32}^*)^{s_2-k_2-j_2} \binom{s_2-k_2}{j_2} \times \\ &\sum_{j_1=0}^{s_1-k_1} (K_2 a_{21}^*)^{j_1} (K_3 a_{31}^*)^{s_1-k_1-j_1} \binom{s_1-k_1}{j_1} \partial_{u_{2_i}}^{||\mathbf{j}||} M_n(u_{2_i} - l_2) \partial_{u_{3_i}}^{||\mathbf{s-k-j}||} M_n(u_{3_i} - l_3) \quad , \end{aligned} \tag{2.278}$$

where $\mathbf{a}_1^* = \langle a_{11}^*, a_{12}^*, a_{13}^* \rangle$, $\mathbf{a}_2^* = \langle a_{21}^*, a_{22}^*, a_{23}^* \rangle$, and $\mathbf{a}_3^* = \langle a_{31}^*, a_{32}^*, a_{33}^* \rangle$ are the reciprocal space basis vectors and K_1, K_2, and K_3, the maximum number of grid points in the fundamental cell in the x, y, and z directions respectively. For an orthogonal box, where

$$a_{12}^* = a_{13}^* = a_{21}^* = a_{23}^* = a_{31}^* = a_{32}^* = 0 \quad , \tag{2.279}$$

DL_POLY_4 uses the simplification of equation (2.279) to

$$\begin{aligned} &\partial_{z_i}^{s_3} \partial_{y_i}^{s_2} \partial_{x_i}^{s_1} \left\{ M_n(u_{1_i} - l_1) M_n(u_{2_i} - l_2) M_n(u_{3_i} - l_3) \right\} = \\ &(K_1 a_{11}^*)^{s_1} (K_2 a_{22}^*)^{s_2} (K_3 a_{33}^*)^{s_3} \partial_{u_{1_i}}^{s_1} M_n(u_{1_i} - l_1) \partial_{u_{2_i}}^{s_2} M_n(u_{2_i} - l_2) \partial_{u_{3_i}}^{s_3} M_n(u_{3_i} - l_3) \quad . \end{aligned} \tag{2.280}$$

The formulas in equations (2.279) and (2.280) require derivatives of a b-spline. To compute an arbitrary $p_{\rm th}$ order derivative of a b-spline of order n, M_n, at an arbitrary grid point j, DL_POLY_4 uses the closed form formula

$$\frac{d^p}{du^p} M_n(u_j) = \sum_{t=\max\{0,j-k\}}^{\min\{j-1,p\}} \binom{p}{t} (-1)^t M_k(u_j - t) \quad . \tag{2.281}$$

In DL_POLY_4 the stress tensor due to the reciprocal space, for an arbitrary $p_{\rm th}$ order multipolar electrostatic interaction is computed by the formula

$$V \sigma_{\alpha\beta}^{\rm rec} = \frac{1}{2 V_o \epsilon_0 \epsilon} \sum_{\mathbf{k} \neq 0} \frac{\exp(-k^2/4\eta^2)}{k^2} \left\{ |S(\mathbf{k})|^2 \left[\delta_{\alpha\beta} - 2 \left(\frac{k^2/4\eta^2 + 1}{k^2} \right) k_\alpha k_\beta \right] + 2 S(\mathbf{k}) S_i^\beta(-\mathbf{k}) \frac{k_\alpha}{k_\beta} \right\} \quad , \tag{2.282}$$

where

$$\mathcal{J}_i^\ell(\mathbf{k}) = \mathcal{M}_i^\ell \partial_i^\ell e^{\imath \mathbf{k} \cdot \mathbf{r_i}} \quad , \tag{2.283}$$

$$S_i^\beta(-\mathbf{k}) = \sum_{\ell=\mathbf{0}}^{p} \ell_\beta \sum_{i=1}^{N} \mathcal{J}_i^\ell(-\mathbf{k}) \quad , \tag{2.284}$$

and $\ell = (\ell_1, \ell_2, \ell_3)$.

In DL_POLY_4 the routine EWALD_SPME_MFORCES_D computes the reciprocal space interactions explicitly for simulations with multipoles of maximum order 2. The routine EWALD_SPME_MFORCES handles the general version with multipoles up to order 4.

The DL_POLY_4 subroutines required to calculate the contributions from the reciprocal space, in addition to the routines used for the point charges, are:

1. BSPGEN_MPOLES, in SPME_CONTAINER evaluates equation (2.279) or (2.280) to compute the B-splines.
2. LIMIT_ERFR_DERIV in MPOLES_CONTAINER which computes the limit of the derivatives of the kernel for the self-interaction term. LIMIT_ERFR_DERIV is called in EWALD_SPME_MFORCES.

2.5 Polarisation Shell Models

An atom or ion is polarisable if it develops a dipole moment when placed in an electric field. It is commonly expressed by the equation

$$\frac{1}{4\pi\epsilon_0\epsilon}\underline{\mu} = \alpha\underline{E} \quad , \tag{2.285}$$

where $\underline{\mu}$ is the induced dipole and $\underline{E}$ is the electric field. The constant α is the polarisability.

In the *static* shell model, also called core-shell model or known as the Drude model or Druder oscillator [74, 75], a polarisable atom is represented by a massive core (often called a nucleus) and a “massless” shell (also known as a Druder particle), connected by a harmonic spring, hereafter called the core-shell unit. The core and shell carry different electric charges, the sum of which equals the charge on the original *rigid-ion* atom. There is no electrostatic interaction (i.e. self-interaction) between the core and shell of the same atom. Non-coulombic interactions arise from the shell alone.

The core-shell interaction is described by a harmonic spring potential of the form:

$$U_{spring}(r_{ij}) = \frac{1}{2}k_2 r_{ij}^2 \quad , \tag{2.286}$$

However, sometimes an anharmonic spring is used, described by a quartic form:

$$U_{spring}(r_{ij}) = \frac{1}{2}k_2 r_{ij}^2 + \frac{1}{4!}k_4 r_{ij}^4 . \tag{2.287}$$

Normally, in practice, k_2 is much larger than k_4.

The effect of an external electric field, $\underline{E}$ is to separate the core and shell by a distance

$$\underline{d} = q_s\underline{E}/k_2 \quad , \tag{2.288}$$

giving rise to a *polarisation* dipole

$$\underline{\mu} = q_s\underline{d} \quad . \tag{2.289}$$

The condition of static equilibrium then gives the polarisability as:

$$\alpha = \frac{1}{4\pi\epsilon_0\epsilon} q_s^2/k_2 \quad , \tag{2.290}$$

where q_s is the shell charge and k_2 is the force constant of the harmonic spring.

The calculation of the forces, virial and stress tensor in this model is based on that for a diatomic molecule with charged atoms. The part coming from the spring potential is similar in spirit as for chemical bonds, equations (2.13-2.15), while the electrostatics is as described in the above section. The relationship between the kinetic energy and the temperature is different however, as the core-shell unit is permitted only three translational degrees of freedom, and the degrees of freedom corresponding to rotation and vibration of the unit are discounted as if the kinetic energy of these is regarded as zero (equation 3.7).

2.5.1 CHARMM Shell Model Self-Induction

The CHARMM model for self-induced polarisablility relies on the Druder formalism as described above. However, the CHARMM core-shell model interactions, conventions and controls have further specificity [76] that is worth outlining in DL_POLY_4 terms.

To enable the CHARMM core-shell model in DL_POLY_4, the user, *at the very least*, needs to specify atomic polarisabilities (and, optionally, the respective Thole dumping factors) for all cores in the MPOLES file (see Section 10.1.4), make sure that reading MPOLES is triggered by using the **multi**polar order *n* directive in the FIELD file (see Section 10.1.3), and use the **polar**isability *CHARMM* **thole** *f* directive in CONTROL (see Section 10.1.1). **Note that** if no Thole dumping factors are specified in MPOLES as well as no global Thole dumping factor is (optional) provided with the above directive in CONTROL, then a default one of 1.3 is assumed for all inducible particles! Also, if no Thole dumping factors are specified in MPOLES and a **zero** global Thole dumping factor is (optional) provided in CONTROL that will also invalidate the use of CHARMM scaled electrostatics in the simulation although the option will help with checking, verifying and setting CHARMM related defaults for shell charges, core-shell spring force constants and atomic polarisations!

Equation (2.290) governs the relation between the force constant, k_2 (positive), the shell charge q_s, and the atomic polarisability, α (positive). Thus if one is missing, undefined or zero, it can be recovered from the rest in DL_POLY_4. CHARMM only allows for q_s to be recovered. **Note** that in DL_POLY_4 if any q_s is recovered, it has an opposite sign to that of its corresponding q_c! In the special case when all Druders' force constants, k_2, are undefined or zero (in FIELD), DL_POLY_4 will resort to using the value of 1000 kcal mol^{-1}Å^{-2} for all core-shell units, as per CHARMM recommendations in [76]. In all other cases if two (or more) of these three quantities are undefined or zero DL_POLY_4 will terminate execution in a controlled manner, indicating the exact nature of the problem.

CHARMM postulates a scaled intra-molecular self-induction between 1-2 and 1-3 intra-molecular neighbours. Thus cross core-shell coulombic interactions within *conventional* bonded interactions are not fully excluded (dipole-dipole only, no charge-dipole). They are scaled (see below) for all possible 1-2 (chemical bonds or constraints) and 1-2-3 (chemical bond angles) qualifying neighbours. For example, let 1-2-3-4 define a torsion angle, then all 1-2-3 and 2-3-4 core-shell cross-interaction are considered. **Note** that this is not the case for the 1-4 one, which is excluded in this scenario but it could still be scaled via torsion 1-4 coulombic scaling!. In DL_POLY_4's CHARMM context, also, then all possible coulombic cross-interactions are considered in the case of an inversion angle interaction. It is worth **stressing** that in DL_POLY_4 will further exclude (disregard as non-contributing) any core-core bonded interactions if they are frozen or mapped onto a RB!

Let's demonstrate this for a conventional bond angle unit, 1-2-3, with only members 1 and 2 being polarisable (having a core and a shell). So, the only CHARMM scaled intra-molecular coulombic interactions for this scenario will be the four pairs 1_{core}-2_{core}, 1_{core}-2_{shell}, 1_{shell}-2_{core} and 1_{shell}-2_{shell} (dipole-dipole interactions only, the charge-dipole interactions are not considered!). In the case when members 1 and 3 are frozen or in a RB configuration then the pair 1_{core}-2_{core} will be disregarded from that list.

The CHARMM, self-induced intra-molecular (dipole-dipole) interactions are scaled by a factor of

$$S(r_{ij}) = 1 - \left[1 + \frac{1}{2}\frac{a_i + a_j}{(\alpha_i\alpha_j)^{1/6}} r_{ij}\right] \exp\left(-\frac{a_i + a_j}{(\alpha_i\alpha_j)^{1/6}} r_{ij}\right) \quad , \tag{2.291}$$

where r_{ij} is the distance between atoms i and j, α_i and α_j are the respective atomic polarisabilities, and a_i and a_j are the respective atomic (Thole) damping constants. This equation is equivalent to a smeared charge distribution described by

$$\rho = \frac{a^3}{8\pi} \exp\left(-\frac{a}{(\alpha_i\alpha_j)^{1/6}} r_{ij}\right) \quad , \tag{2.292}$$

with $a = a_i + a_j$, as originally proposed by Thole [77].

2.5.2 Dynamical (Adiabatic Shells) Shell Model

The dynamical shell model is a method of incorporating polarisability into a molecular dynamics simulation. The method used in DL_POLY_4 is that devised by Fincham *et al* [78] and is known as the adiabatic shell model.

In the adiabatic method, a fraction of the atomic mass is assigned to the shell to permit a dynamical description. The fraction of mass, x, is chosen to ensure that the natural frequency of vibration $\nu_{\texttt{core-shell}}$ of the harmonic spring (which depends on the reduced mass, i.e.

$$\nu_{\texttt{core-shell}} = \frac{1}{2\pi}\left[\frac{k_2}{x(1-x)m}\right]^{1/2} \quad , \tag{2.293}$$

with m the rigid ion atomic mass) is well above the frequency of vibration of the whole atom in the bulk system. Dynamically, the core-shell unit resembles a diatomic molecule with a harmonic bond, however, the high vibrational frequency of the bond prevents effective exchange of kinetic energy between the core-shell unit and the remaining system. Therefore, from an initial condition in which the core-shell units have negligible internal vibrational energy, the units will remain close to this condition throughout the simulation. This is essential if the core-shell unit is to maintain a net polarisation. (In practice, there is a slow leakage of kinetic energy into the core-shell units, but this should should not amount to more than a few percent of the total kinetic energy. To determine safe shell masses in practice, first a rigid ion simulation is performed in order to gather the velocity autocorrelation functions, VAF, of the ions of interest to polarise. Each VAF is then Fast Fourier transformed to find their highest frequency of interaction, $\nu_{\texttt{rigid-ion}}$. It is, then, a safe choice to assign a shell mass, $x\ m$, so that $\nu_{\texttt{core-shell}} \geq 3\ \nu_{\texttt{rigid-ion}}$. The user must make sure to assign the correct mass, $(1-x)\ m$, to the core!)

2.5.3 Relaxed (Massless Shells) Model

The relaxed shell model is presented in [79], where shells have no mass and as such their motion is not governed by the usual Newtonian equation, whereas their cores' motion is. Because of that, shells respond instantaneously to the motion of the cores: for any set of core positions, the positions of the shells are such that the force on every shell is zero. The energy is thus a minimum with respect to the shell positions. This represents the physical fact that the system is always in the ground state with respect to the electronic degrees of freedom.

Relaxation of the shells is carried out at each time step and involves a search in the multidimensional space of shell configurations. The search in DL_POLY_4 is based on the powerful conjugate-gradients technique [80] in an adaptation as shown in [79]. Each time step a few iterations (10÷30) are needed to achieve convergence to zero net force.

2.5.4 Breathing Shell Model Extension

While for low-symmetry structures, the conventional, dipolar rigid shell model (RSM) is sufficient to absorb most of the effects of partial covalency/ionic polarisation, for some high-symmetry systems, a breathing shell model (BSM) [81] is used as a refinement to represent the contribution of higher-order charge deformations (of oxide species). This is done by the inclusion of non-central ion interaction to account for a finite ion shell radius, r_i^o, which is allowed to deform isotropically under its environment. However, all short-range repulsion potentials, i.e. **vdw**, a BSM ion interacts by with its environment, must act upon the radius of the ion, $U = U(r_i - r_i^o)$, rather than the nuclear position, $U = U(r_i)$! A further constraining potential is then added to represent the self-energy of the ion's breathing shell. This most commonly uses the same shape as the one of the harmonic bond, see Section 2.2.1:

$$U(r_{ij}) = \frac{1}{2}k(r_{ij}^{BSM} - r_{ij}^o)^2 \quad , \tag{2.294}$$

where i and j are the intramolecular index of the ion's core and shell respectively. Hence, to employ the BSM, the user needs to specify an extra bonded interaction for each BSM'ed core-shell pair in the relevant **bonds** sections in the FIELD file for all molecules that contain BSM ions. It is worth noting that the BSM energy and virial are thus part of the bonds' energy and virial and their calculation as part of the bonds forces routine BONDS_FORCES.

The most significant consequence of the introduction of the BSM is that, by coupling the repulsive interactions via common shell radii, it creates a many-body effect that is able to reproduce the Cauchy violation ($C_{44} \neq C_{12}$) for rock salt structured materials.

2.5.5 Further Notes

In DL_POLY_4 the core-shell forces of the rigid shell model are handled by the routine CORE_SHELL_FORCES. In case of the adiabatic shell model the kinetic energy is calculated by CORE_SHELL_KINETIC and temperature scaling applied by routine CORE_SHELL_QUENCH. In case of the relaxed shell model shell are relaxed to zero force by CORE_SHELL_RELAXED.

Note that DL_POLY_4 determines which shell model to use by scanning shell weights provided the FIELD file (see Section 10.1.3). If all shells have zero weight the DL_POLY_4 will choose the relaxed shell model. If no shell has zero weight then DL_POLY_4 will choose the dynamical one. In case when some shells are massless and some are not DL_POLY_4 will terminate execution controllably and provide information about the error and possible possible choices of action in the OUTPUT file (see Section 10.2.6).

Note that all DL_POLY_4's shell models can be used in conjunction with the methods for long-ranged forces described above. This also includes uses in the context of multipolar electrostatics where self-induced polarisation is often a crucial part of polarisable force-field models such as CHARMM, AMBER, AMOEBA. Currently, there are the following restrictions within DL_POLY_4:

- shell particles are restricted to only bear a charge.
- shell particles cannot be frozen, part of a rigid body formation, constraint bonded, PMF constrained or tethered.
- shell particles cannot have shells.
- a core and shell unit cannot be in a relation via an angle, dihedral or inversion type of interaction. However, they can be in a bond type of interactions (see the BSM section above).

It is worth noting that bonded interactions such as chemical bonds, angles, dihedrals and inversions can be defined over any possible mixture of core and shell members of different core-shell units! For example, in the solid state materials community it is common to define bonds and angles over the shells as the shells represent the electronic clouds between which the bonding occurs. However, this is right the opposite in the liquid, organic and bio-chemical communities, where all bonding is between the nuclei (cores) and the shells are only there to account purely for the polarisability. As DL_POLY_4 allows for too much flexibility in the space of possible model definitions, it is strongly advised that the modeller must exercise great care to define the correct force-field model representation within DL_POLY_4 input files!

2.6 External Fields

In addition to the molecular force field, DL_POLY_4 allows the use of an *external* force field. Examples of fields available include:

1. Electric field: (**elec**)
$$\underline{F_i} = \underline{F_i} + q_i \cdot \underline{E} \tag{2.295}$$
2. Oscillating shear: (**oshr**)
$$\underline{F}_x = A\cos(2n\pi \cdot z/L_z) \tag{2.296}$$
3. Continuous shear: (**shrx**)
$$\underline{v}_x = \frac{1}{2}A\frac{|z|}{z} \qquad : |z| > z_0 \tag{2.297}$$

4. Gravitational field: (**grav**)
$$\underline{F_i} = \underline{F_i} + m_i \cdot \underline{G} \tag{2.298}$$

5. Magnetic field: (**magn**)
$$\underline{F_i} = \underline{F_i} + q_i \cdot (\underline{v_i} \times \underline{H}) \tag{2.299}$$

6. Containing sphere: (**sphr**)
$$\underline{F} = A\ (R_0 - r)^{-n} \qquad : r > R_{\text{cut}} \tag{2.300}$$

7. Repulsive wall: (**zbnd**)
$$\underline{F}_z = A\ (z_o - z) \qquad : f \cdot z > f \cdot z_o \ \ , \tag{2.301}$$
where $f = +/-1$ with default of 1.

8. X-Piston: (**xpis**)
$$\underline{F}_x = \frac{m_k}{\sum_{k=i}^{j} m_k} P \cdot \underline{\text{Area}}(\perp \text{X-}\mathit{direction}) \qquad : \forall\ k = i, .., j \ \ . \tag{2.302}$$

9. Harmonic restraint zone in z-direction: (**zres**)
$$\underline{F}_z = \begin{cases} A\ (z_{com} - z_{max}) & : \quad z_{com} > z_{max} \\ A\ (z_{min} - z_{com}) & : \quad z_{com} < z_{min} \ \ , \end{cases} \tag{2.303}$$
where z_{com} is the chosen molecule centre of mass.

10. Harmonic restraint zone in z-direction − (push out): (**zrs−**)
$$\underline{F}_z = \begin{cases} A\ (z - z_{max}) & : \quad z \geq (z_{max} + z_{min})/2 \\ A\ (z_{min} - z) & : \quad z < (z_{max} + z_{min})/2 \end{cases} \tag{2.304}$$

11. Harmonic restraint zone in z-direction + (pull in): (**zrs+**)
$$\underline{F}_z = \begin{cases} A\ (z - z_{max}) & : \quad z > z_{max} \\ A\ (z_{min} - z) & : \quad z < z_{min} \end{cases} \tag{2.305}$$

12. Oscillating electric field: (**osel**)
$$\underline{F_i} = \underline{F_i} + q_i \cdot \underline{E} \cdot \sin(2\pi\omega t)\ , \tag{2.306}$$
where t is the simulated time.

13. Umbrella sampling (harmonic restraint) [82, 83]: (**ushr**)
$$U_{AB} = \frac{k}{2}(R_{AB} - R_0)^2\ , \tag{2.307}$$
is an umbrella sampling harmonic restraint between the centres of masses of two molecules (non-overlapping clusters of particles), A and B, with a force constant, k, and an equilibrium distance, R_0.

It is recommended that the use of an external field should be accompanied by a thermostat (this does not apply to examples 6 and 7, since these are conservative fields). The "Oscillating shear" and "X-piston" fields may only be used with orthorhombic cell geometry (imcon=1,2) and "Continuous shear" field with slab cell geometry (imcon=6).

In the case of the "X-piston" field it is strongly advised that the number of piston particles is chosen to be very small in comparison with the rest of the system (< 5%) and that the piston contains its own set of whole molecules (i.e. there are no molecules partially mapped on the piston), which do not include any core-shell, CB, PMF or RB units! The field releases the system's centre of mass to move unconstrained and gain momentum. This makes any temperature control options control the full kinetic energy of the system and thus the only ensemble valid under this conditions and possible within DL_POLY_4 at the present is the micro-canonical (NVE)!

The user is advised to be careful with the parameters' units! For more insight, do examine Table 10.14 and the example at equation (10.9) in Section 10.1.3.

In DL_POLY_4 external field forces are handled by the routines EXTERNAL_FIELD_APPLY and EXTERNAL_FIELD_CORRECT.

2.7 Treatment of Frozen Atoms, Rigid Body and Core-Shell Units

Frozen atoms, core-shell units and rigid body units are treated in a manner similar to that of the **intra**-molecular interactions due to their "by site" definition.

DL_POLY_4 allows for atoms to be completely immobilized (*i.e.* "frozen" at a fixed point in the MD cell). This is achieved by setting all forces and velocities associated with that atom to zero during each MD timestep. Frozen atoms are signalled by assigning an atom a non-zero value for the freeze parameter in the FIELD file. DL_POLY_4 does not calculate contributions to the virial or the stress tensor arising from the constraints required to freeze atomic positions. Neither does it calculate contributions from *intra*- and *inter*-molecular interactions between frozen atoms. As with the tethering potential, the reference position of a frozen site is scaled with the cell vectors in constant pressure simulations. In the case of frozen rigid bodies, their "centre of mass" is scaled with the cell vectors in constant pressure simulations and the positions of their constituent sites are then moved accordingly.

In DL_POLY_4 the frozen atom option is handled by the subroutine FREEZE_ATOMS.

The rigid body dynamics (see Section 3.6) is resolved by solving the Eulerian equations of rotational motion. However, their statics includes calculation of the individual contributions of each RB's centre of mass stress and virial due to the action of the resolved forces on sites/atoms constituting it. These contribute towards the total system stress and pressure.

As seen in Section 2.5 core-shell units are dealt with (i) kinetically by the adiabatic shell model or (ii) statically by the dynamic shell model. Both contribute to the total system stress (pressure) but in different manner. The former does it via the kinetic stress (energy) and atomic stress (potential energy) due to the core-shell spring. The latter via atomic stress (potential energy) due to the shells move to minimised configuration.

2.8 Tabulation and interpolation in the treatment of intermolecular interactions

By default DL_POLY_4 tabulates in memory most of the intermolecular interactions keeping values of the potential and the negative of its first derivative times the distance (or virial) over an equidistant grid. This is done for reasons of speed as due to the large variety of potential forms, some could be quite expensive to evaluate if run unoptimised. The memory tabulation could be overridden for non-tabulated interactions upon user specified options such as **metal direct** for metal interactions and **vdw direct** for van der Waals interactions.

When energy and force are calculated for tabulated interactions a 3-point interpolation scheme of our own is used to interpolate the value for the requested distance.

A 5-point interpolation is used for finding the numerical derivatives of (2B)(E)EAM type potentials which are supplied in TABEAM by the user. For this a Lagrange formula is used, which can be found in any textbook on numerical methods.

2.9 Free Energy Capabilities via the PLUMED plugin

DL_POLY_4 supports a native integration with PLUMED - http://www.plumed.org/. PLUMED is an open source library for free energy calculations in molecular systems which works together with some of the most popular molecular dynamics engines. Free energy calculations can be performed as a function of many order parameters with a particular focus on biological problems, using state of the art methods such as metadynamics, umbrella sampling [82, 83] and Jarzynski-equation based steered MD. The software, written in C++, can be easily interfaced with both FORTRAN and C/C++ codes.

Using PLUMED can be as simple as adding the keyword **plumed** in your CONTROL file. By default the input file for PLUMED is called PLUMED and shall be placed in the same place as your other DL_POLY_4 input files. Once DL_POLY_4 runs by default OUTPUT.PLUMED will be generated in addition to the normal PLUMED and DL_POLY_4 output files. The default names of the files can be changed by using **input** and **log** parameters with the **plumed** keyword (see Section 10.1.1).

Note that this feature should be considered as ***young*** rather than mature, so please report any bugs and provide feedback regarding its improvements.

Note that to use the PLUMED functionality within DL_POLY_4 one must further ensure that DL_POLY_4 is cross-compiled with it.

2.10 Open Knowledgebase of Interatomic Models - OpenKIM

DL_POLY_4 force-field allows for model interactions specification by using an OpenKIM model - https://openkim.org/. A KIM model contains all necessary (non-bonded) interactions and their parameters for a specific model within a designated container. Thus a KIM model can be used as a force-field container when made available to DL_POLY_4 at run time (see the description of the FIELD file in Section 10.1.3) upon a specification within FIELD together with a matching molecular description for the model system, also specified in FIELD. Due to the history of the OpenKIM initiative and the constraints of the logic of the DL_POLY_4 FIELD file the designated place for a KIM model specification is in the non-bonded interactions section **??**.

Employing OpenKIM interatomic models (IMs) provides DL_POLY_4 users with multiple benefits, including:

Reliability

- All content archived in OpenKIM is reviewed by the KIM Editor for quality.
- IMs in OpenKIM are archived with full provenance control. Each is associated with a maintainer responsible for the integrity of the content. All changes are tracked and recorded.
- IMs in OpenKIM are exhaustively tested using KIM Tests that compute a host of material properties, and KIM Verification Checks that provide the user with information on various aspects of the IM behavior and coding correctness. This information is displayed on the IMs page accessible through the OpenKIM browse interface.

Reproducibility

- Each IM in OpenKIM is issued a unique identifier (KIM ID), which includes a version number (last three digits). Any changes that can result in different numerical values lead to a version increment in the KIM ID. This makes it possible to reproduce simulations since the specific version of a specific IM used can be retrieved using its KIM ID.

- OpenKIM is a member organization of DataCite and issues digital object identifiers (DOIs) to all IMs archived in OpenKIM. This makes it possible to cite the IM code used in a simulation in a publications to give credit to the developers and further facilitate reproducibility.

Currently, DL_POLY_4 supports one type of IM archived in OpenKIM, which is called a KIM Portable Model (PM). A KIM PM is an independent computer implementation of an IM written in one of the languages supported by KIM (C, C++, Fortran) that conforms to the KIM Application Programming Interface (KIM API) Portable Model Interface (PMI) standard. A KIM PM will work seamlessly with any simulation code that supports the KIM API/PMI standard (including DL_POLY_4; see complete list of supported codes).

OpenKIM IMs are uniquely identified by a KIM ID. The extended KIM ID consists of a human-readable prefix identifying the type of IM, authors, publication year, and supported species, separated by two underscores from the KIM ID itself, which begins with an IM code (MO for a KIM Portable Model) followed by a unique 12-digit code and a 3-digit version identifier
(e.g. **SNAP_ChenDengTran_2017_Mo__MO_698578166685_000**).

Each OpenKIM IM has a dedicated "Model Page" on OpenKIM providing all the information on the IM including a title, description, authorship and citation information, test and verification check results, visualizations of results, a wiki with documentation and user comments, and access to raw files, and other information. The URL for the Model Page is constructed from the extended KIM ID of the IM
as **https://openkim.org/id/extended_KIM_ID**.

For example, for the spectral neighbor analysis potential (SNAP) listed above the Model Page is located at: https://openkim.org/id/SNAP_ChenDengTran_2017_Mo__MO_698578166685_000.

See the current list of KIM PMs archived in OpenKIM. This list is sorted by species and can be filtered to display only IMs for certain species combinations. You can also see Obtaining KIM Models to learn how to install a pre-build binary of the OpenKIM repository of models.

Note that it is also possible to locally install IMs not archived in OpenKIM, in which case their names do not have to conform to the KIM ID format.

To download, build, and install the KIM library on your system, see the detailed instructions .

Note that to use OpenKIM functionality within DL_POLY_4 one must further ensure that DL_POLY_4 is compiled with OpenKIM support (see building.md).

Citation of OpenKIM IMs

When publishing results obtained using OpenKIM IMs researchers are requested to cite the OpenKIM project (Tadmor), KIM API (Elliott), and the specific IM codes used in the simulations, in addition to the relevant scientific references for the IM. The citation format for an IM is displayed on its page on OpenKIM along with the corresponding BibTex file, and is automatically added to the DL_POLY_4 *log.cite* file.

Citing the IM software (KIM infrastructure and specific PM or SM codes) used in the simulation gives credit to the researchers who developed them and enables open source efforts like OpenKIM to function.

Chapter 3

Integration Algorithms

Scope of Chapter

This chapter describes the integration algorithms coded into DL_POLY_4.

3.1 Introduction

As a default the DL_POLY_4 integration algorithms are based on the Velocity Verlet (VV) scheme, which is both simple and time reversible [22]. It generates trajectories in the microcanonical (NVE) ensemble in which the total energy (kinetic plus potential) is conserved. If this property drifts or fluctuates excessively in the course of a simulation it indicates that the timestep is too large or the potential cutoffs too small (relative r.m.s. fluctuations in the total energy of 10^{-5} are typical with this algorithm).

The VV algorithm has two stages (VV1 and VV2). At the first stage it requires values of position ($\underline{r}$), velocity ($\underline{v}$) and force ($\underline{f}$) at time t. The first stage is to advance the velocities to $t+(1/2)\Delta t$ by integration of the force and then to advance the positions to a full step $t+\Delta t$ using the new half-step velocities:

1. VV1:
$$\underline{v}(t+\frac{1}{2}\Delta t) \leftarrow \underline{v}(t) + \frac{\Delta t}{2}\,\frac{\underline{f}(t)}{m} \quad , \tag{3.1}$$
where m is the mass of a site and Δt is the timestep
$$\underline{r}(t+\Delta t) \leftarrow \underline{r}(t) + \Delta t\;\underline{v}(t+\frac{1}{2}\Delta t) \tag{3.2}$$

2. FF:
Between the first and the second stage a recalculation of the force at time $t+\Delta t$ is required since the positions have changed
$$\underline{f}(t+\Delta t) \leftarrow \underline{f}(t) \tag{3.3}$$

3. VV2:
In the second stage the half-step velocities are advanced to to a full step using the new force
$$\underline{v}(t+\Delta t) \leftarrow \underline{v}(t+\frac{1}{2}\Delta t) + \frac{\Delta t}{2}\,\frac{\underline{f}(t+\Delta t)}{m} \tag{3.4}$$

The instantaneous kinetic energy, for example, can then be obtained from the atomic velocities as
$$E_{kin}(t) = \frac{1}{2}\sum_{1}^{\mathcal{N}} m_i v_i^2(t) \quad , \tag{3.5}$$
and assuming the system has no net momentum the instantaneous temperature is
$$\mathcal{T}(t) = \frac{2}{k_B f} E_{kin}(t) \quad , \tag{3.6}$$
where i labels particles (that can be free atoms or rigid bodies), $\mathcal{N}$ the number of particles (free atoms and rigid bodies) in the system, k_B the Boltzmann's constant and f the number of degrees of freedom in the system.
$$f = 3\mathcal{N} - 3\mathcal{N}_{frozen} - 3\mathcal{N}_{shells} - \mathcal{N}_{constraints} - 3 - p \quad . \tag{3.7}$$
Here $\mathcal{N}_{frozen}$ indicates the number of frozen atoms in the system, $\mathcal{N}_{shells}$ number of core-shell units and $\mathcal{N}_{constraints}$ number of bond and PMF constraints. Three degrees of freedom are subtracted for the centre of mass zero net momentum (which we impose) and p is zero for periodic or three for non-periodic systems, where it accounts for fixing angular momentum about origin (which we impose).

In the case of rigid bodies (see Section 3.6) the first part of equation (3.7)
$$f\prime = 3\mathcal{N} - 3\mathcal{N}_{frozen} \tag{3.8}$$
splits into
$$f\prime = \left(3\mathcal{N}^{FP} - 3\mathcal{N}^{FP}_{frozen}\right) + \left(3\mathcal{N}^{RB(\mathbf{tra})} - 3\mathcal{N}^{RB(\mathbf{tra})}_{frozen}\right) + \left(3\mathcal{N}^{RB(\mathbf{rot})} - 3\mathcal{N}^{RB(\mathbf{rot})}_{frozen}\right) \tag{3.9}$$

or

$$f\prime = f^{FP} + f^{RB(\mathbf{tra})} + f^{RB(\mathbf{rot})} \quad . \tag{3.10}$$

Here FP stands for a free particle, i.e. a particle not participating in the constitution of a rigid body, and RB for a rigid body. In general a rigid body has 3 translational (**tra**) degrees of freedom, corresponding to its centre of mass being allowed to move in the 3 general direction of space, and 3 rotational (**rot**), corresponding to the RB being allowed to rotate around the 3 general axis in space. It is not far removed to see that for a not fully frozen rigid body one must assign 0 translational degrees of freedom but depending on the "frozenness" of the RB one may assign 1 rotational degrees of freedom when all the frozen sites are in line (i.e. rotation around one axis only) or 3 when just one site is frozen.

The routine NVE_0_VV implement the Verlet algorithm in velocity verlet for free particles and calculate the instantaneous temperature. Whereas the routines NVE_1_VV implements the same for systems also containing rigid bodies. The conserved quantity is the total energy of the system

$$\mathcal{H}_{\mathrm{NVE}} = U + E_{kin} \quad , \tag{3.11}$$

where U is the potential energy of the system and E_{kin} the kinetic energy at time t.

The full selection of integration algorithms within DL_POLY_4 is as follows:

NVE_0_VV	Constant E algorithm
NVE_1_VV	The same as the above but also incorporating RB integration
DPD_THERMOSTAT	Constant T algorithm (DPD [84])
NVT_E0_VV	Constant E_{kin} algorithm (Evans [25])
NVT_E1_VV	The same as the above but also incorporating RB integration
NVT_L0_VV	Constant T algorithm (Langevin [26])
NVT_L1_VV	The same as the above but also incorporating RB integration
NVT_L2_VV	Constant T algorithmensemble!Inhomogeneous Langevin NVT) (inhomogeneous Langevin [85
NVT_A0_VV	Constant T algorithm (Andersen [27])
NVT_A1_VV	The same as the above but also incorporating RB integration
NVT_B0_VV	Constant T algorithm (Berendsen [28])
NVT_B1_VV	The same as the above but also incorporating RB integration
NVT_H0_VV	Constant T algorithm (Hoover [29])
NVT_H1_VV	The same as the above but also incorporating RB integration
NVT_G0_VV	Constant T algorithm (GST [86])
NVT_G1_VV	The same as the above but also incorporating RB integration
NPT_L0_VV	Constant T,P algorithm (Langevin [30])
NPT_L1_VV	The same as the above but also incorporating RB integration
NPT_B0_VV	Constant T,P algorithm (Berendsen [28])
NPT_B1_VV	The same as the above but also incorporating RB integration
NPT_H0_VV	Constant T,P algorithm (Hoover [29])
NPT_H1_VV	The same as the above but also incorporating RB integration
NPT_M0_VV	Constant T,P algorithm (Martyna-Tuckerman-Klein [31])
NPT_M1_VV	The same as the above but also incorporating RB integration
NPT_L0_VV	Constant T,$\underline{\underline{\sigma}}$ algorithm (Langevin [30])
NPT_L1_VV	The same as the above but also incorporating RB integration
NST_B0_VV	Constant T,$\underline{\underline{\sigma}}$ algorithm (Berendsen [28])
NST_B1_VV	The same as the above but also incorporating RB integration
NST_H0_VV	Constant T,$\underline{\underline{\sigma}}$ algorithm (Hoover [29])
NST_H1_VV	The same as the above but also incorporating RB integration
NST_M0_VV	Constant T,$\underline{\underline{\sigma}}$ algorithm (Martyna-Tuckerman-Klein [31])
NST_M0_VV	The same as the above but also incorporating RB integration

It is worth noting that the last four ensembles are also optionally available in an extended from to constant normal pressure and constant surface area, NP$_n$AT, or constant surface tension, NP$_n\gamma$T [87].

3.2 Bond Constraints

The SHAKE algorithm for bond constraints was devised by Ryckaert *et al.* [88] and is widely used in molecular simulation. It is a two stage algorithm based on the leapfrog Verlet integration scheme [22]. In the first stage the LFV algorithm calculates the motion of the atoms in the system assuming a complete absence of the rigid bond forces. The positions of the atoms at the end of this stage do not conserve the distance constraint required by the rigid bond and a correction is necessary. In the second stage the deviation in the length of a given rigid bond is used retrospectively to compute the constraint force needed to conserve the bondlength. It is relatively simple to show that the constraint force has the form:

$$\underline{G}_{ij} \approx \frac{1}{2}\frac{\mu_{ij}}{\Delta t^2}\frac{(d_{ij}^2 - d_{ij}'^2)}{\underline{d}_{ij}^o \cdot \underline{d}_{ij}'}\,\underline{d}_{ij}^o \quad , \tag{3.12}$$

where: μ_{ij} is the reduced mass of the two atoms connected by the bond; $\underline{d}_{ij}^o$ and $\underline{d}_{ij}'$ are the original and intermediate bond vectors; d_{ij} is the constrained bondlength; and Δt is the Verlet integration time step. It should be noted that this formula is an approximation only.

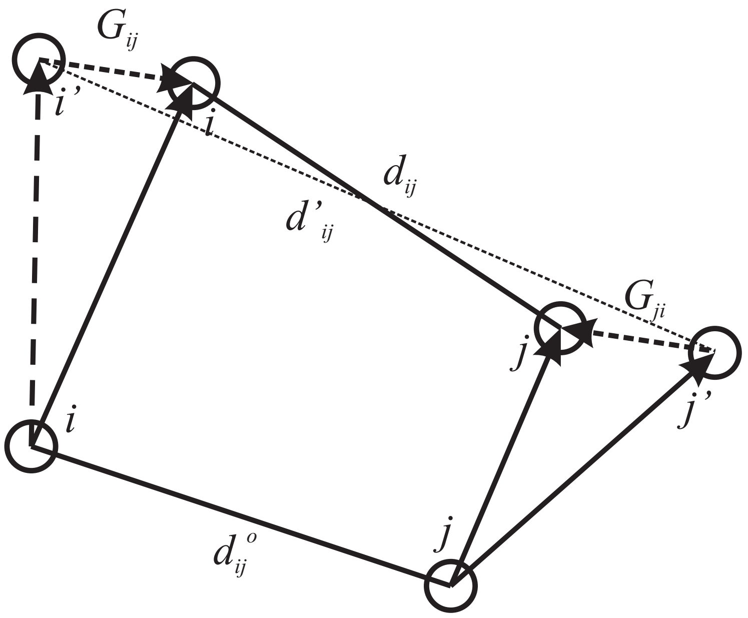


Figure 3.1: The SHAKE (RATTLE_VV1) schematics and associated vectors. The algorithm calculates the constraint force $\underline{G}_{ij} = -\underline{G}_{ji}$ that conserves the bondlength d_{ij} between atoms i and j, following the initial movement to positions $i\prime$ and $j\prime$ under the unconstrained forces $\underline{F}_i$ and $\underline{F}_j$ and velocities $\underline{v}_i$ and $\underline{v}_j$.

The RATTLE algorithm was devised by Andersen [23] and it is the SHAKE algorithm used with Velocity Verlet integration scheme. It consists of two parts RATTLE_VV1 and RATTLE_VV2 applied respectively in stages one and two of Velocity Verlet algorithm. RATTLE_VV1 is similar to the SHAKE algorithm as described above and handles the bond length constraint. However, due to the difference in the velocity update between VV (VV1) and LFV schemes, the constraint force generated to conserve the bondlength in RATTLE_VV1 has the form as in (3.12) but missing the factor of a half:

$$\underline{G}_{ij} \approx \frac{\mu_{ij}}{\Delta t^2}\frac{(d_{ij}^2 - d_{ij}'^2)}{\underline{d}_{ij}^o \cdot \underline{d}_{ij}'}\,\underline{d}_{ij}^o \quad . \tag{3.13}$$

The constraint force in RATTLE_VV2 imposes a new condition of rigidity on constraint bonded atom velocities. RATTLE_VV2 is also a two stage algorithm. In the first stage, the VV2 algorithm calculates the velocities of the atoms in the system assuming a complete absence of the rigid bond forces (since forces have just been recalculated afresh after VV1). The relative velocity of atom i with respect to atom j (or vice versa) constituting the rigid bond ij may not be perpendicular to the bond - i.e. may have a non-zero component along the bond. However, by the stricter definition of rigidity this is is required to be zero as it will otherwise lead to a change in the rigid bond length during the consequent timestepping. In the second stage the deviation from zero of the scalar product $\underline{d}_{ij} \cdot (\underline{v}_j - \underline{v}_i)$ is used retrospectively to compute the constraint force needed to keep the bond rigid over the length of the timestep Δt. It is relatively simple to show that the constraint force has the form:

$$\underline{B}_{ij} \approx \frac{\mu_{ij}}{\Delta t} \frac{\underline{d}_{ij} \cdot (\underline{v}_j - \underline{v}_i)}{d_{ij}^2} \underline{d}_{ij} \quad . \tag{3.14}$$

The velocity corrections can therefore be written as

$$\underline{v}_i^{corr} = \Delta t \frac{\underline{B}_{ij}}{m_i} = \frac{\mu_{ij}}{m_i} \frac{\underline{d}_{ij} \cdot (\underline{v}_j - \underline{v}_i)}{d_{ij}^2} \underline{d}_{ij} \quad . \tag{3.15}$$

For a system of simple diatomic molecules, computation of the constraint force will, in principle, allow the correct atomic positions to be calculated in one pass. However, in the general polyatomic case this correction is merely an interim adjustment, not only because the above formula is approximate, but the successive correction of other bonds in a molecule has the effect of perturbing previously corrected bonds. Either part of the RATTLE algorithm is therefore iterative, with the correction cycle being repeated for all bonds until: each has converged to the correct length, within a given tolerance for RATTLE_VV1 (SHAKE) and the relative bond velocities are perpendicular to their respective bonds within a given tolerance for RATTLE_VV2 (RATTLE). The tolerance may be of the order 10^{-4} Å to 10^{-8} Å depending on the precision desired.

The SHAKE procedure may be summarised as follows:

1. All atoms in the system are moved using the LFV algorithm, assuming an absence of rigid bonds (constraint forces). (This is stage 1 of the SHAKE algorithm.)

2. The deviation in each bondlength is used to calculate the corresponding constraint force, equation (3.12), that (retrospectively) 'corrects' the bond length.

3. After the correction, equation (3.12), has been applied to all bonds, every bondlength is checked. If the largest deviation found exceeds the desired tolerance, the correction calculation is repeated.

4. Steps 2 and 3 are repeated until all bondlengths satisfy the convergence criterion (this iteration constitutes stage 2 of the SHAKE algorithm).

The RATTLE procedures may be summarised as follows:

1. RATTLE stage 1:

   (a) All atoms in the system are moved using the VV algorithm, assuming an absence of rigid bonds (constraint forces). (This is stage 1 of the RATTLE_VV1 algorithm.)

   (b) The deviation in each bondlength is used to calculate the corresponding constraint force, equation (3.13), that (retrospectively) 'corrects' the bond length.

   (c) After the correction, equation (3.13), has been applied to all bonds, every bondlength is checked. If the largest deviation found exceeds the desired tolerance, the correction calculation is repeated.

   (d) Steps (b) and (c) are repeated until all bondlengths satisfy the convergence criterion (this iteration constitutes stage 2 of the RATTLE_VV1 algorithm).

2. Forces calculated afresh.
3. RATTLE stage 2:
   (a) All atom velocities are updated to a full step, assuming an absence of rigid bonds. (This is stage 1 of the RATTLE_VV2 algorithm.)
   (b) The deviation of $\underline{d}_{ij} \cdot (\underline{v}_j - \underline{d}_i)$ in each bond is used to calculate the corresponding constraint force that (retrospectively) 'corrects' the bond velocities.
   (c) After the correction, equation (3.14), has been applied to all bonds, every bond velocity is checked against the above condition. If the largest deviation found exceeds the desired tolerance, the correction calculation is repeated.
   (d) Steps (b) and (c) are repeated until all bonds satisfy the convergence criterion (this iteration constitutes stage 2 of the RATTLE_VV2 algorithm).

The parallel version of the RATTLE algorithm, as implemented in DL_POLY_4, is derived from the RD_SHAKE algorithm [8] although its implementation in the Domain Decomposition framework requires no *global merging* operations and is consequently significantly more efficient. The routine CONSTRAINTS_SHAKE is called to apply corrections to the atomic positions and the routine CONSTRAINTS_RATTLE to apply corrections to the atomic velocities of constrained particles.

It should be noted that the fully converged constraint forces G_{ij} make a contribution to the system virial and the stress tensor.

The contribution to be added to the atomic virial (for each constrained bond) is

$$\mathcal{W} = -\underline{d}_{ij} \cdot \underline{G}_{ij} \quad . \tag{3.16}$$

The contribution to be added to the atomic stress tensor (for each constrained bond) is given by

$$\sigma^{\alpha\beta} = d_{ij}^{\alpha} G_{ij}^{\beta} \quad , \tag{3.17}$$

where α and β indicate the x, y, z components. The atomic stress tensor derived from the pair forces is symmetric.

3.3 Potential of Mean Force (PMF) Constraints and the Evaluation of Free Energy

A generalization of bond constraints can be made to constrain a system to some point along a reaction coordinate. A simple example of such a reaction coordinate would be the distance between two ions in solution. If a number of simulations are conducted with the system constrained to different points along the reaction coordinate then the mean constraint force may be plotted as a function of reaction coordinate and the function integrated to obtain the free energy for the overall process [89]. The PMF constraint force, virial and contributions to the stress tensor are obtained in a manner analogous to that for a bond constraint (see previous section). The only difference is that the constraint is now applied between the centres of two groups which need not be atoms alone. DL_POLY_4 reports the PMF constraint virial, $\mathcal{W}_{PMF}$, for each simulation. Users can convert this to the PMF constraint force from

$$G_{PMF} = \frac{\mathcal{W}_{PMF}}{d_{PMF}} \quad , \tag{3.18}$$

where is d_{PMF} the constraint distance between the two groups used to define the reaction coordinate.

The routines PMF_SHAKE and PMF_RATTLE are called to apply corrections to the atomic positions and respectively the atomic velocities of all particles constituting PMF units.

In presence of both bond constraints and PMF constraints. The constraint procedures, i.e. SHAKE or RATTLE, for both types of constraints are applied iteratively in order bonds-PMFs until convergence of $\mathcal{W}_{PMF}$ reached. The number of iteration cycles is limited by the same limit as for the bond constraints' procedures (SHAKE/RATTLE).

3.4 Thermostats

The system may be coupled to a heat bath to ensure that the average system temperature is maintained close to the requested temperature, $T_{\rm ext}$. When this is done the equations of motion are modified and the system no longer samples the microcanonical ensemble. Instead trajectories in the canonical (NVT) ensemble, or something close to it are generated. DL_POLY_4 comes with seven different thermostats: Evans (Gaussian constraints) [25], Langevin (both standard [26, 90] and inhomogeneous [85] variants), Andersen [27], Berendsen [28], Nosé-Hoover (N-H) [29] and the gentle stochastic thermostat (GST) [86, 91] as well as Dissipative Particle Dynamics (DPD) method [92, 93, 50, 84]. Of these, only the Langevin, N-H, GST and DPD algorithms generate *true* trajectories in the canonical (NVT) ensemble. The rest will produce properties that typically differ from canonical averages by $\mathcal{O}(1/\mathcal{N})$ [22] (where $\mathcal{N}$ is the number of particles in the system), as the Evans algorithm generates trajectories in the (NVE$_{kin}$) ensemble.

3.4.1 Evans Thermostat (Gaussian Constraints)

Kinetic temperature can be made a constant of the equations of motion by imposing an additional constraint on the system. If one writes the equations of motion as:

$$\begin{aligned} \frac{d\underline{r}(t)}{dt} &= \underline{v}(t) \\ \frac{d\underline{v}(t)}{dt} &= \frac{\underline{f}(t)}{m} - \chi(t)\;\underline{v}(t) \quad , \end{aligned} \tag{3.19}$$

the kinetic temperature constraint χ can be found as follows:

$$\begin{aligned} \frac{d}{dt}\mathcal{T} &\propto \frac{d}{dt}\left(\frac{1}{2}\sum_i m_i \underline{v}_i^2\right) = \sum_i m_i \underline{v}_i \cdot \frac{d}{dt}\underline{v}_i = 0 \\ & \sum_i m_i \underline{v}_i(t) \cdot \left\{\frac{\underline{f}_i(t)}{m_i} - \chi(t)\;\underline{v}_i(t)\right\} = 0 \\ \chi(t) &= \frac{\sum_i \underline{v}_i(t)\cdot \underline{f}_i(t)}{\sum_i m_i \underline{v}_i^2(t)} \quad , \end{aligned} \tag{3.20}$$

where $\mathcal{T}$ is the instantaneous temperature defined in equation (3.6).

The VV implementation of the Evans algorithm is straight forward. The conventional VV1 and VV2 steps are carried out as before the start of VV1 and after the end of VV2 there is an application of thermal constraining. This involves the calculation of $\chi(t)$ before the VV1 stage and $\chi(t+\Delta t)$ after the VV2 stage with consecutive thermalisation on the unthermostated velocities for half a timestep at each stage in the following manner:

1. Thermostat VV1

$$\begin{aligned} \chi(t) &\leftarrow \frac{\sum_i \underline{v}_i(t)\cdot \underline{f}_i(t)}{2\;E_{kin}(t)} \\ \underline{v}(t) &\leftarrow \underline{v}(t)\;\exp\left(-\chi(t)\frac{\Delta t}{2}\right) \quad . \end{aligned} \tag{3.21}$$

2. VV1:

$$\begin{aligned} \underline{v}(t+\frac{1}{2}\Delta t) &\leftarrow \underline{v}(t)+\frac{\Delta t}{2}\,\frac{\underline{f}(t)}{m} \\ \underline{r}(t+\Delta t) &\leftarrow \underline{r}(t)+\Delta t\;\underline{v}(t+\frac{1}{2}\Delta t) \end{aligned} \tag{3.22}$$

3. RATTLE_VV1

4. FF:

$$\underline{f}(t+\Delta t) \leftarrow \underline{f}(t) \tag{3.23}$$

5. VV2:

$$\underline{v}(t+\Delta t) \leftarrow \underline{v}(t+\frac{1}{2}\Delta t)+\frac{\Delta t}{2}\left[\frac{\underline{f}(t+\Delta t)}{m}\right] \tag{3.24}$$

6. RATTLE_VV2

7. Thermostat VV2

$$\begin{aligned} \chi(t+\Delta t) &\leftarrow \frac{\sum_i \underline{v}_i(t+\Delta t)\cdot \underline{f}_i(t+\Delta t)}{2\;E_{kin}(t+\Delta t)} \\ \underline{v}(t+\Delta t) &\leftarrow \underline{v}(t+\Delta t)\;\exp\left(-\chi(t+\Delta t)\frac{\Delta t}{2}\right)\quad . \end{aligned} \tag{3.25}$$

The algorithm is self-consistent and requires no iterations.

The conserved quantity by these algorithms is the system kinetic energy.

The VV flavour of the Gaussian constraints algorithm is implemented in the DL_POLY_4 routines NVT_E0_VV. The routine NVT_E1_VV implement the same but also incorporate RB dynamics.

3.4.2 Langevin Thermostat

The Langevin thermostat works by coupling every particle to a viscous background and a stochastic heath bath (Brownian dynamics) such that

$$\begin{aligned} \frac{d\underline{r}_i(t)}{dt} &= \underline{v}_i(t) \\ \frac{d\underline{v}_i(t)}{dt} &= \frac{\underline{f}_i(t)+\underline{R}_i(t)}{m_i}-\chi\;\underline{v}_i(t)\quad , \end{aligned} \tag{3.26}$$

where χ is the user defined *constant* (positive, in units of ps^{-1}) specifying the thermostat friction parameter and $R(t)$ is stochastic force with zero mean that satisfies the fluctuation- dissipation theorem:

$$\left\langle R_i^{\alpha}(t)\;R_j^{\beta}(t')\right\rangle = 2\;\chi\;m_i\;k_B T\;\delta_{ij}\;\delta_{\alpha\beta}\;\delta(t-t')\quad , \tag{3.27}$$

where superscripts denote Cartesian indices, subscripts particle indices, k_B is the Boltzmann constant, T the target temperature and m_i the particle's mass. The Stokes-Einstein relation for the diffusion coefficient can then be used to show that the average value of $R_i(t)$ over a time step (in thermal equilibrium) should be a random deviate drawn from a Gaussian distribution of zero mean and unit variance, `Gauss`$(0,1)$, scaled by $\sqrt{\frac{2\;\chi\;m_i\;k_B T}{\Delta t}}$.

The effect of this algorithm is thermostat the system on a local scale. Particles that are too "cold" are given more energy by the noise term and particles that are too "hot" are slowed down by the friction. Numerical instabilities, which usually arise from inaccurate calculation of a local collision-like process, are thus efficiently kept under control and cannot propagate.

The generation of random forces is implemented in the routine LANGEVIN_FORCES.

An inhomogeneous variant of the Langevin thermostat [85] allows χ to vary according to atomic velocity: it can be increased from χ_{ep} to $\chi_{ep} + \chi_{es}$ when the atomic speed is higher than a cut-off value. When using this thermostat as part of the two-temperature model (TTM), χ_{ep} represents the required friction parameter for electron-phonon coupling and χ_{es} gives the increase due to electronic stopping. TTM simulations also require the random deviate to be scaled by $\sqrt{\frac{2\ \chi_{ep}\ m_i\ k_B T_e}{\Delta t}}$, where T_e is the local electronic temperature for the atom.

The VV implementation of the algorithm is tailored in a Langevin Impulse (LI) manner [90]:

1. VV1:

$$\begin{aligned}
\underline{v}(t+\epsilon) &\leftarrow \underline{v}(t) + \frac{\Delta t}{2}\,\frac{\underline{f}(t)}{m} \\
\underline{v}(t+\frac{1}{2}\Delta t-\epsilon) &\leftarrow \exp(-\chi\ \Delta t)\ \underline{v}(t+\epsilon) + \frac{\sqrt{2\ \chi\ m\ k_B T}}{m}\ \underline{Z}_1(\chi,\Delta t) \\
\underline{r}(t+\Delta t) &\leftarrow \underline{r}(t) + \frac{1-\exp(-\chi\ \Delta t)}{\chi}\ \underline{v}(t+\epsilon) + \frac{\sqrt{2\ \chi\ m\ k_B T}}{\chi\ m}\ \underline{Z}_2(\chi,\Delta t)\quad ,
\end{aligned} \tag{3.28}$$

   where $\underline{Z}_1(\chi,\Delta t)$ and $\underline{Z}_2(\chi,\Delta t)$ are joint Gaussian random variables of zero mean, sampling from a *bivariate* Gaussian distribution [90]:

$$\begin{bmatrix} \underline{Z}_1 \\ \underline{Z}_2 \end{bmatrix} = \begin{bmatrix} \sigma_2^{1/2} & 0 \\ (\sigma_1-\sigma_2)\sigma_2^{-1/2} & (\Delta t - \sigma_1^2\sigma_2^{-1})^{1/2} \end{bmatrix} \begin{bmatrix} \underline{R}_1 \\ \underline{R}_2 \end{bmatrix} \tag{3.29}$$

   with

$$\sigma_k = \frac{1-\exp(-k\ \chi\ \Delta t)}{k\ \chi}\quad ,\quad k=1,2 \tag{3.30}$$

   and $\underline{R}_k$ vectors of independent standard Gaussian random numbers of zero mean and unit variance, `Gauss`$(0,1)$, - easily related to the Langevin random forces as defined in equation (3.27).

2. RATTLE_VV1

3. FF:

$$\underline{f}(t+\Delta t) \leftarrow \underline{f}(t) \tag{3.31}$$

4. VV2:

$$\underline{v}(t+\Delta t) \leftarrow \underline{v}(t+\frac{1}{2}\Delta t-\epsilon) + \frac{\Delta t}{2}\,\frac{\underline{f}(t+\Delta t)}{m} \tag{3.32}$$

5. RATTLE_VV2 .

The algorithm is self-consistent and requires no iterations. It is worth noting that the integration is conditional upon the Cholesky factorisation which is impossible when $\Delta t < \sigma_1^2/\sigma_2$. That is why a safety check is put in place which when failed iteratively increases the integration timestep to a safe value with respect to the above condition.

Note that by the nature of the ensemble the centre of mass will not be stationary although the ensemble average warrants its proximity to the its original position, i.e. the COM momentum accumulation ensemble average will tend towards zero. By default this accumulation is removed and thus the correct application of stochastic dynamics the user is advised to use in the **no vom** option in the CONTROL file (see Section 10.1.1). If the option is not applied then the dynamics will lead to peculiar thermalisation of different atomic species to mass- and system size-dependent temperatures.

The VV flavour of the Langevin thermostat is implemented in the DL_POLY_4 routines NVT_L0_VV. The routines NVT_L1_VV implements the same but also incorporate RB dynamics. The inhomogeneous Langevin thermostat is implemented in the DL_POLY_4 routine NVT_L2_VV no RB dynamics are currently available for this form of Langevin thermostat.

3.4.3 Andersen Thermostat

This thermostat assumes the idea that the system, or some subset of the system, has an instantaneous interaction with some fictional particles and exchanges energy. Practically, this interaction amounts to replacing the momentum of some atoms with a new momentum drawn from the correct Boltzmann distribution at the desired temperature. The strength of the thermostat can be adjusted by setting the average time interval over which the interactions occur, and by setting the magnitude of the interaction. The collisions are best described as a random (Poisson) process so that the probability that a collision occurs in a time step Δt is

$$P_{\texttt{collision}}(t) = 1 - \exp\left(-\frac{\Delta t}{\tau_T}\right) \quad , \tag{3.33}$$

where τ_T is the thermostat relaxation time. The hardest collision is to completely reset the momentum of the Poisson selected atoms in the system, with a new one selected from the Boltzmann distribution

$$F(\underline{v}_i) = \sqrt{\left(\frac{m_i}{2\pi k_B T_{\texttt{ext}}}\right)^3} \exp\left(-\frac{m_i\ \underline{v}_i^2}{2k_B T_{\texttt{ext}}}\right) = \sqrt{\frac{k_B T_{\texttt{ext}}}{2m_i}}\ \texttt{Gauss}(0,1) \quad . \tag{3.34}$$

where subscripts denote particle indices, k_B is the Boltzmann constant, $T_{\texttt{ext}}$ the target temperature and m_i the particle's mass. The thermostat can be made softer by mixing the new momentum $\underline{v}_i^{\texttt{new}}$ drawn from $F(\underline{v}_i)$ with the old momentum $\underline{v}_i^{\texttt{old}}$

$$\underline{v}_i = \alpha\ \underline{v}_i^{\texttt{old}} + \sqrt{1-\alpha^2}\ \underline{v}_i^{\texttt{new}} \quad , \tag{3.35}$$

where α $(0 \le \alpha \le 1)$ is the softness of the thermostat. In practice, a uniform distribution random number, $\texttt{uni}(i)$, is generated for each particle in the system, which is compared to the collision probability. If $\texttt{uni}(i) \le 1 - \exp\left(-\frac{\Delta t}{\tau_T}\right)$ the particle momentum is changed as described above.

The VV implementation of the Andersen algorithm is as follows:

1. VV1:
$$\begin{aligned} \underline{v}(t+\frac{1}{2}\Delta t) &\leftarrow \underline{v}(t) + \frac{\Delta t}{2}\frac{\underline{f}(t)}{m} \\ \underline{r}(t+\Delta t) &\leftarrow \underline{r}(t) + \Delta t\ \underline{v}(t+\frac{1}{2}\Delta t) \end{aligned} \tag{3.36}$$

2. RATTLE_VV1

3. FF:
$$\underline{f}(t+\Delta t) \leftarrow \underline{f}(t) \tag{3.37}$$

4. VV2:
$$\underline{v}(t+\Delta t) \leftarrow \underline{v}(t+\frac{1}{2}\Delta t) + \frac{\Delta t}{2}\left[\frac{\underline{f}(t+\Delta t)}{m}\right] \tag{3.38}$$

5. RATTLE_VV2

6. Thermostat: Note that the MD cell centre of mass momentum must not change!
$$\begin{aligned} \texttt{If} \quad & \left(\texttt{uni}(i) \le 1 - \exp\left(-\frac{\Delta t}{\tau_T}\right)\right) \texttt{ Then} \\ \underline{v}_i^{\texttt{new}}(t+\Delta t) &\leftarrow \sqrt{\frac{k_B T}{2m_i}}\ \texttt{Gauss}(0,1) \\ \underline{v}_i(t+\Delta t) &\leftarrow \alpha\ \underline{v}_i(t+\Delta t) + \sqrt{1-\alpha^2}\ \underline{v}_i^{\texttt{new}}(t+\Delta t) \\ \texttt{End} \quad & \texttt{If} \ . \end{aligned} \tag{3.39}$$

The algorithm is self-consistent and requires no iterations.

The VV flavour of the Andersen thermostat is implemented in the DL_POLY_4 routine NVT_A0_VV. The routine NVT_A1_VV implements the same but also incorporate RB dynamics.

3.4.4 Berendsen Thermostat

In the Berendsen algorithm the instantaneous temperature is pushed towards the desired temperature $T_{\rm ext}$ by scaling the velocities at each step by

$$\chi(t) = \left[1 + \frac{\Delta t}{\tau_T}\left(\frac{\sigma}{E_{kin}(t)} - 1\right)\right]^{1/2} \quad , \tag{3.40}$$

where

$$\sigma = \frac{f}{2}\; k_B \; T_{\rm ext} \tag{3.41}$$

is the target thermostat energy (depending on the external temperature and the system total degrees of freedom, f - equation (3.7)) and τ_T a specified time constant for temperature fluctuations (normally in the range [0.5, 2] ps).

The VV implementation of the Berendsen algorithm is straight forward. A conventional VV1 and VV2 (thermally unconstrained) steps are carried out. At the end of VV2 velocities are scaled by a factor of χ in the following manner

1. VV1:

$$\begin{aligned} \underline{v}(t + \frac{1}{2}\Delta t) &\leftarrow \underline{v}(t) + \frac{\Delta t}{2}\,\frac{\underline{f}(t)}{m} \\ \underline{r}(t + \Delta t) &\leftarrow \underline{r}(t) + \Delta t\; \underline{v}(t + \frac{1}{2}\Delta t) \end{aligned} \tag{3.42}$$

2. RATTLE_VV1
3. FF:

$$\underline{f}(t + \Delta t) \leftarrow \underline{f}(t) \tag{3.43}$$

4. VV2:

$$\underline{v}(t + \Delta t) \leftarrow \underline{v}(t + \frac{1}{2}\Delta t) + \frac{\Delta t}{2}\,\frac{\underline{f}(t + \Delta t)}{m} \tag{3.44}$$

5. RATTLE_VV2
6. Thermostat:

$$\begin{aligned} \chi(t + \Delta t) &\leftarrow \left[1 + \frac{\Delta t}{\tau_T}\left(\frac{\sigma}{E_{kin}(t + \Delta t)} - 1\right)\right]^{1/2} \\ \underline{v}(t + \Delta t) &\leftarrow \underline{v}(t + \Delta t)\;\chi \quad . \end{aligned} \tag{3.45}$$

Note that the MD cell's centre of mass momentum is removed at the end of the integration algorithms.

The Berendsen algorithms conserve total momentum but not energy.

The VV flavour of the Berendsen thermostat is implemented in the DL_POLY_4 routine NVT_B0_VV. The routine NVT_B1_VV implements the same but also incorporate RB dynamics.

3.4.5 Nosé-Hoover Thermostat

In the Nosé-Hoover algorithm [29] Newton's equations of motion are modified to read:

$$\begin{aligned} \frac{d\underline{r}(t)}{dt} &= \underline{v}(t) \\ \frac{d\underline{v}(t)}{dt} &= \frac{\underline{f}(t)}{m} - \chi(t)\;\underline{v}(t) \end{aligned} \tag{3.46}$$

The friction coefficient, χ, is controlled by the first order differential equation

$$\frac{d\chi(t)}{dt} = \frac{2E_{kin}(t) - 2\sigma}{q_{mass}} \tag{3.47}$$

where σ is the target thermostat energy, equation (3.41), and

$$q_{mass} = 2\ \sigma\ \tau_T^2 \tag{3.48}$$

is the thermostat mass, which depends on a specified time constant τ_T (for temperature fluctuations normally in the range [0.5, 2] ps).

The VV implementation of the Nosé-Hoover algorithm takes place in a symplectic manner as follows:

1. Thermostat: Note $E_{kin}(t)$ changes inside

$$\begin{aligned} \chi(t + \frac{1}{4}\Delta t) &\leftarrow \chi(t) + \frac{\Delta t}{4}\ \frac{2E_{kin}(t) - 2\sigma}{q_{mass}} \\ \underline{v}(t) &\leftarrow \underline{v}(t)\ \exp\left(-\chi(t + \frac{1}{4}\Delta t)\ \frac{\Delta t}{2}\right) \qquad (3.49) \\ \chi(t + \frac{1}{2}\Delta t) &\leftarrow \chi(t + \frac{1}{4}\Delta t) + \frac{\Delta t}{4}\ \frac{2E_{kin}(t) - 2\sigma}{q_{mass}} \end{aligned} \tag{3.50}$$

2. VV1:

$$\begin{aligned} \underline{v}(t + \frac{1}{2}\Delta t) &\leftarrow \underline{v}(t) + \frac{\Delta t}{2}\ \frac{\underline{f}(t)}{m} \\ \underline{r}(t + \Delta t) &\leftarrow \underline{r}(t) + \Delta t\ \underline{v}(t + \frac{1}{2}\Delta t) \end{aligned} \tag{3.51}$$

3. RATTLE_VV1

4. FF:

$$\underline{f}(t + \Delta t) \leftarrow \underline{f}(t) \tag{3.52}$$

5. VV2:

$$\underline{v}(t + \Delta t) \leftarrow \underline{v}(t + \frac{1}{2}\Delta t) + \frac{\Delta t}{2}\ \frac{\underline{f}(t + \Delta t)}{m} \tag{3.53}$$

6. RATTLE_VV2

7. Thermostat: Note $E_{kin}(t + \Delta t)$ changes inside

$$\begin{aligned} \chi(t + \frac{3}{4}\Delta t) &\leftarrow \chi(t + \frac{1}{2}\Delta t) + \frac{\Delta t}{4}\ \frac{2E_{kin}(t + \Delta t) - 2\sigma}{q_{mass}} \\ \underline{v}(t + \Delta t) &\leftarrow \underline{v}(t + \Delta t)\ \exp\left(-\chi(t + \frac{3}{4}\Delta t)\ \frac{\Delta t}{2}\right) \\ \chi(t + \Delta t) &\leftarrow \chi(t + \frac{3}{4}\Delta t) + \frac{\Delta t}{4}\ \frac{2E_{kin}(t + \Delta t) - 2\sigma}{q_{mass}} \quad . \end{aligned} \tag{3.54}$$

The algorithm is self-consistent and requires no iterations.

The conserved quantity is derived from the extended Hamiltonian for the system which, to within a constant, is the Helmholtz free energy:

$$\mathcal{H}_{\rm NVT} = \mathcal{H}_{\rm NVE} + \frac{q_{mass}\ \chi(t)^2}{2} + f\ k_B\ T_{\rm ext} \int_o^t \chi(s) ds \quad , \tag{3.55}$$

where f is the system's degrees of freedom - equation (3.7).

The VV flavour of the Nosé-Hoover thermostat is implemented in the DL_POLY_4 routine NVT_H0_VV. The routine NVT_H1_VV implements the same but also incorporate RB dynamics.

3.4.6 Gentle Stochastic Thermostat

The Gentle Stochastic Thermostat [86, 91] is an extension of the Nosé-Hoover algorithm [29]

$$
\begin{aligned}
\frac{d\underline{r}(t)}{dt} &= \underline{v}(t) \\
\frac{d\underline{v}(t)}{dt} &= \frac{\underline{f}(t)}{m} - \chi(t)\ \underline{v}(t)
\end{aligned} \tag{3.56}
$$

in which the thermostat friction, χ, has its own Brownian dynamics:

$$
\frac{d\chi(t)}{dt} = \frac{2E_{kin}(t) - 2\sigma}{q_{mass}} - \gamma\ \chi(t) + \frac{\sqrt{2\ \gamma\ k_B\ T_{\text{ext}}\ q_{mass}}}{q_{mass}}\ \frac{d\omega(t)}{dt} \quad , \tag{3.57}
$$

governed by the Langevin friction γ (positive, in units of ps^{-1}), where $\omega(t)$ is the standard Brownian motion (Wiener process - `Gauss`(0,1)), σ is the target thermostat energy, as in equation (3.41).

$$
q_{mass} = 2\ \sigma\ \tau_T^2 \tag{3.58}
$$

is the thermostat mass, which depends on a specified time constant τ_T (for temperature fluctuations normally in the range [0.5, 2] ps).

It is worth noting that equation (3.57) similar to the Ornstein-Uhlenbeck equation:

$$
\frac{d\chi}{dt} = -\frac{\alpha\sigma^2}{2}\chi + \sigma\frac{d\omega}{dt} \quad , \tag{3.59}
$$

which for a given realization of the Wiener process $\omega(t)$ has an exact solution:

$$
\chi_{n+1} = e^{-\epsilon t}\left(\chi_n + \sigma\sqrt{\frac{e^{2\ \epsilon t} - 1}{2\epsilon}}\Delta\omega\right) \quad , \tag{3.60}
$$

where $\epsilon = \alpha\sigma^2/2$ and $\Delta\omega \sim \mathcal{N}(0,1)$. The VV implementation of the Gentle Stochastic Thermostat algorithm takes place in a symplectic manner as follows:

1. Thermostat: Note $E_{kin}(t)$ changes inside and $R_{g_{1,2}}(t)$, drawn from `Gauss`$(0,1)$, are independent

$$
\begin{aligned}
\chi(t + \frac{1}{4}\Delta t) &\leftarrow \chi(t)\ \exp\left[-\gamma\ \frac{\Delta t}{4}\right] + \sqrt{\frac{k_B\ T_{\text{ext}}}{q_{mass}}\left(1 - \exp^2\left[-\gamma\ \frac{\Delta t}{4}\right]\right)}\ R_{g_1}(t) \\
&\qquad + \frac{\Delta t}{4}\ \frac{2E_{kin}(t) - 2\sigma}{q_{mass}} \\
\underline{v}(t) &\leftarrow \underline{v}(t)\ \exp\left(-\chi(t + \frac{1}{4}\Delta t)\ \frac{\Delta t}{2}\right) \\
\chi(t + \frac{1}{2}\Delta t) &\leftarrow \chi(t + \frac{1}{4}\Delta t)\ \exp\left[-\gamma\ \frac{\Delta t}{4}\right] + \sqrt{\frac{k_B\ T_{\text{ext}}}{q_{mass}}\left(1 - \exp^2\left[-\gamma\ \frac{\Delta t}{4}\right]\right)}\ R_{g_2}(t + \frac{1}{4}\Delta t) \\
&\qquad + \frac{\Delta t}{4}\ \frac{2E_{kin}(t) - 2\sigma}{q_{mass}}
\end{aligned} \tag{3.61}
$$

2. VV1:

$$
\begin{aligned}
\underline{v}(t + \frac{1}{2}\Delta t) &\leftarrow \underline{v}(t) + \frac{\Delta t}{2}\ \frac{\underline{f}(t)}{m} \\
\underline{r}(t + \Delta t) &\leftarrow \underline{r}(t) + \Delta t\ \underline{v}(t + \frac{1}{2}\Delta t)
\end{aligned} \tag{3.62}
$$

3. RATTLE_VV1

4. FF:

$$\underline{f}(t + \Delta t) \leftarrow \underline{f}(t) \tag{3.63}$$

5. VV2:

$$\underline{v}(t + \Delta t) \leftarrow \underline{v}(t + \frac{1}{2}\Delta t) + \frac{\Delta t}{2} \, \frac{\underline{f}(t + \Delta t)}{m} \tag{3.64}$$

6. RATTLE_VV2

7. Thermostat: Note $E_{kin}(t + \Delta t)$ changes inside and $R_{g_{3,4}}(t)$, drawn from `Gauss`$(0, 1)$, are independent

$$\begin{aligned}
\chi(t + \frac{3}{4}\Delta t) \quad &\leftarrow \quad \chi(t + \frac{1}{2}\Delta t) \; \exp\left[-\gamma \; \frac{\Delta t}{4}\right] + \sqrt{\frac{k_B \; T_{\rm ext}}{q_{mass}} \left(1 - \exp^2\left[-\gamma \; \frac{\Delta t}{4}\right]\right)} \; R_{g_3}(t + \frac{2}{4}\Delta t) \\
&\qquad + \frac{\Delta t}{4} \; \frac{2E_{kin}(t) - 2\sigma}{q_{mass}} \\
\underline{v}(t + \Delta t) \quad &\leftarrow \quad \underline{v}(t + \Delta t) \; \exp\left(-\chi(t + \frac{3}{4}\Delta t) \; \frac{\Delta t}{2}\right) \\
\chi(t + \Delta t) \quad &\leftarrow \quad \chi(t + \frac{3}{4}\Delta t) \; \exp\left[-\gamma \; \frac{\Delta t}{4}\right] + \sqrt{\frac{k_B \; T_{\rm ext}}{q_{mass}} \left(1 - \exp^2\left[-\gamma \; \frac{\Delta t}{4}\right]\right)} \; R_{g_4}(t + \frac{3}{4}\Delta t) \\
&\qquad + \frac{\Delta t}{4} \; \frac{2E_{kin}(t) - 2\sigma}{q_{mass}}
\end{aligned} \tag{3.65}$$

The algorithm is self-consistent and requires no iterations.

The conserved quantity is derived from the extended Hamiltonian for the system which, to within a constant, is the Helmholtz free energy:

$$\mathcal{H}_{\rm NVT} = \mathcal{H}_{\rm NVE} + \frac{q_{mass} \; \chi(t)^2}{2} + f \; k_B \; T_{\rm ext} \int_o^t \chi(s) ds \quad , \tag{3.66}$$

where f is the system's degrees of freedom - equation (3.7).

The VV flavour of the Gentle Stochastic Thermostat is implemented in the DL_POLY_4 routine NVT_G0_VV. The routine NVT_G1_VV implements the same but also incorporate RB dynamics.

3.4.7 Dissipative Particle Dynamics Thermostat

An elegant way to integrate the DPD equations of motions, as shown in Appendix A, is introduced by Shardlow [84]. By applying ideas commonly used in solving differential equations to the case of integrating the equations of motion in DPD, the integration process is factorised by splitting the conservative forces calculation from that of the dissipative and random terms. In this way the conservative part can be solved using traditional molecular dynamics methods, while the fluctuation-dissipation part is solved separately as a stochastic differential (Langevin) equation. There are two Shardlow integrators, called S1 (**dpds1**) and S2 (**dpds2**), based on splitting the equations of motion up to first and second order, respectively, using Suzuki-Trotter(Strang) expansion of the Liouville evolution operator and thus warranting the integrators' symplectic.

To describe the two integrators we define the algorithmic sequence

$$\mathrm{S}(\Delta t) \;=\; \begin{cases} \text{For all pairs of particles for which } r_{ij} < r_c : \\ (1): \quad \underline{v}_i \;\leftarrow\; \underline{v}_i - \frac{1}{2m_i}\left\{\gamma_{ij} w^2_{(r_{ij})}(\underline{v}_{ij}\cdot\underline{e}_{ij})\underline{e}_{ij}\Delta t + \sqrt{2\gamma_{ij}k_BT}w_{(r_{ij})}\zeta_{ij}\underline{e}_{ij}\sqrt{\Delta t}\right\} \\ (2): \quad \underline{v}_j \;\leftarrow\; \underline{v}_j + \frac{1}{2m_i}\left\{\gamma_{ij} w^2_{(r_{ij})}(\underline{v}_{ij}\cdot\underline{e}_{ij})\underline{e}_{ij}\Delta t - \sqrt{2\gamma_{ij}k_BT}w_{(r_{ij})}\zeta_{ij}\underline{e}_{ij}\sqrt{\Delta t}\right\} \\ (3): \quad \underline{v}_i \;\leftarrow\; \underline{v}_i + \frac{1}{2m_i}\left\{\sqrt{2\gamma_{ij}k_BT}w_{(r_{ij})}\zeta_{ij}\underline{e}_{ij}\sqrt{\Delta t} - \frac{\gamma_{ij}w^2(r_{ij})\Delta t}{1+\gamma_{ij}w^2(r_{ij})\Delta t}\times \right. \\ \qquad\qquad \left. \left[(\underline{v}_{ij}\cdot\underline{e}_{ij})\underline{e}_{ij} + \sqrt{2\gamma_{ij}k_BT}w_{(r_{ij})}\zeta_{ij}\underline{e}_{ij}\sqrt{\Delta t}\right]\right\} \\ (4): \quad \underline{v}_j \;\leftarrow\; \underline{v}_j - \frac{1}{2m_i}\left\{\sqrt{2\gamma_{ij}k_BT}w_{(r_{ij})}\zeta_{ij}\underline{e}_{ij}\sqrt{\Delta t} + \frac{\gamma_{ij}w^2(r_{ij})\Delta t}{1+\gamma_{ij}w^2(r_{ij})\Delta t}\times \right. \\ \qquad\qquad \left. \left[(\underline{v}_{ij}\cdot\underline{e}_{ij})\underline{e}_{ij} + \sqrt{2\gamma_{ij}k_BT}w_{(r_{ij})}\zeta_{ij}\underline{e}_{ij}\sqrt{\Delta t}\right]\right\} \end{cases} \tag{3.67}$$

as the Shardlow operator, where ζ_{ij} is the random number with zero mean and unit variance, unique for every unique pair $\{ij\}$ in the system, and $w_{(r_{ij})} = w^C(r_{ij})$ is the DPD conservative force switching function in equation (A.4), γ_{ij} is the drag coefficient between the types of particles i and j, $\underline{v}_{ij} = \underline{v}_j - \underline{v}_i$ is the inter-particle relative velocity and $\underline{e}_{ij} = \underline{r}_{ij}/r_{ij}$ is the inter-particle unit vector. k_B and T are the Boltzmann constant and the system target temperature.

If we define the velocity Verlet micro-canonical (NVE) ensemble sequence as NVE$(t+\Delta t)$ then Shardlow's first ($\mathcal{S}1$) and second ($\mathcal{S}2$) order splittings can be written algorithmically as the following sequential operators applications:

$$\begin{aligned} \mathcal{S}1 &:: \mathrm{S}(\;\Delta\mathrm{t}\;) \rightarrow \mathrm{NVE}(t+\Delta t) \\ \mathcal{S}2 &:: \mathrm{S}(\Delta t/2) \rightarrow \mathrm{NVE}(t+\Delta t) \rightarrow \mathrm{S}(\Delta t/2)\;\;. \end{aligned} \tag{3.68}$$

The application of these DPD thermostats are implemented in the DL_POLY_4 routine DPD_THERMOSTAT and which only applies as a perturbation around the NVE integrator incorporating both particle and RB dynamics.

3.5 Barostats

The size and shape of the simulation cell may be dynamically adjusted by coupling the system to a barostat in order to obtain a desired average pressure ($P_{\rm ext}$) and/or isotropic stress tensor ($\underline{\underline{\sigma}}$). DL_POLY_4 has four such algorithms: the Langevin type barostat [30], the Berendsen barostat [28], the Nosé-Hoover type barostat [29] and the Martyna-Tuckerman-Klein (MTK) barsotat [31]. Only the Berendsen barostat does not have defined conserved quantity.

Note that the MD cell's centre of mass momentum is removed at the end of the integration algorithms with barostats.

3.5.1 Instantaneous pressure and stress

The instantaneous pressure in a system,

$$\mathcal{P}(t) = \frac{[2E_{kin}(t) - \mathcal{W}_{\rm atomic}(t) - \mathcal{W}_{\rm constrain}(t-\Delta t) - \mathcal{W}_{\rm PMF}(t-\Delta t)]}{3V(t)} \quad , \tag{3.69}$$

is a function of the system volume, kinetic energy and virial, $\mathcal{W}$.

Note that when bond constraints or/and PMF constraints are present in the system $\mathcal{P}$ will not converge to the exact value of $P_{\rm ext}$ during equilibration in NPT and NσT simulations. This is due to iterative nature of the constrained motion, in which the virials $\mathcal{W}_{\rm constrain}$ and $\mathcal{W}_{\rm PMF}$ are calculated retrospectively to the forcefield virial $\mathcal{W}_{\rm atomic}$.

The instantaneous stress tensor in a system,

$$\underline{\underline{\sigma}}(t) = \underline{\underline{\sigma}}_{kin}(t) + \underline{\underline{\sigma}}_{\rm atomic}(t) + \underline{\underline{\sigma}}_{\rm constrain}(t-\Delta t) + \underline{\underline{\sigma}}_{\rm PMF}(t-\Delta t) \quad , \tag{3.70}$$

is a sum of the forcefield, $\underline{\underline{\sigma}}_{\rm atomic}$, constrain, $\underline{\underline{\sigma}}_{\rm constrains}$, and PMF, $\underline{\underline{\sigma}}_{\rm PMF}$, stresses.

Note that when bond constraints or/and PMF constraints are present in the system, the quantity $\frac{\mathtt{Tr}[\underline{\underline{\sigma}}]}{3V}$ will not converge to the exact value of $P_{\rm ext}$ during equilibration in NPT and NσT simulations. This is due to iterative nature of the constrained motion in which the constraint and PMF stresses are calculated retrospectively to the forcefield stress.

3.5.2 Langevin Barostat

DL_POLY_4 implements a Langevin barostat [30] for isotropic and anisotropic cell fluctuations.

Cell size variations

For isotropic fluctuations the equations of motion are:

$$\begin{aligned}
\frac{d}{dt}\underline{r}(t) &= \underline{v}(t) + \eta(t)\ \underline{r}(t) \\
\frac{d}{dt}\underline{v}(t) &= \frac{\underline{f}(t)+\underline{R}(t)}{m} - \left[\chi + \left(1+\frac{3}{f}\right)\eta(t)\right]\underline{v}(t) \\
\frac{d}{dt}\eta(t) &= 3V(t)\frac{\mathcal{P}(t)-P_{\rm ext}}{p_{mass}} + 3\frac{2E_{kin}(t)}{f}\frac{1}{p_{mass}} - \chi_p\ \eta(t) + \frac{R_p}{p_{mass}} \\
p_{mass} &= \frac{(f+3)\ k_B\ T_{\rm ext}}{(2\pi\ \chi_p)^2} \\
\frac{d}{dt}\underline{\underline{\mathbf{H}}}(t) &= \eta(t)\ \underline{\underline{\mathbf{H}}}(t) \\
\frac{d}{dt}V(t) &= [3\eta(t)]\ V(t) \quad ,
\end{aligned} \tag{3.71}$$

where χ and χ_p are the user defined *constants* (positive, in units of ps^{-1}), specifying the thermostat and barostat friction parameters, $R(t)$ is the Langevin stochastic force (see equation (3.27)), $\mathcal{P}$ the instantaneous pressure (equation (3.69)) and R_p is the stochastic (Langevin) pressure variable

$$\langle R_p(t)\ R_p(t')\rangle = 2\ \chi_p\ p_{mass}\ k_B T\ \delta(t-t') \quad , \tag{3.72}$$

which is drawn from Gaussian distribution of zero mean and unit variance, $\mathtt{Gauss}(0,1)$, scaled by // $\sqrt{\frac{2\ \chi_p\ p_{mass}\ k_B T}{\Delta t}}$. k_B is the Boltzmann constant, T the target temperature and p_{mass} the barostat mass. $\underline{\underline{\mathbf{H}}}$ is the cell matrix whose columns are the three cell vectors $\underline{a}, \underline{b}, \underline{c}$.

The conserved quantity these generate is:

$$\mathcal{H}_{\rm NPT} = \mathcal{H}_{\rm NVE} + \frac{p_{mass}\ \eta(t)^2}{2} + P_{\rm ext}V(t) \quad . \tag{3.73}$$

The VV implementation of the Langevin algorithm only requires iterations if bond or PMF constraints are present (4 until satisfactory convergence of the constraint forces is achieved). These are with respect to the pressure (i.e. $\eta(t)$) in the first part, VV1+RATTLE_VV1. The second part is conventional, VV2+RATTLE_VV2, as at the end the velocities are scaled by a factor of χ.

1. Thermostat: Note $E_{kin}(t)$ changes inside

$$\underline{v}(t) \leftarrow \exp\left(-\chi\ \frac{\Delta t}{4}\right)\ \underline{v}(t) \tag{3.74}$$

2. Barostat: Note $E_{kin}(t)$ and $\mathcal{P}(t)$ have changed and change inside

$$\begin{aligned}
\eta(t) &\leftarrow \exp\left(-\chi_p \frac{\Delta t}{8}\right) \eta(t) \\
\eta(t+\frac{1}{4}\Delta t) &\leftarrow \eta(t) + \frac{\Delta t}{4}\left[3V(t)\frac{\mathcal{P}(t)-P_{\rm ext}}{p_{mass}} + \right. \\
&\qquad \left. 3\frac{2E_{kin}(t)}{f}\frac{1}{p_{mass}} + \frac{R_p(t)}{p_{mass}}\right] \\
\eta(t+\frac{1}{4}\Delta t) &\leftarrow \exp\left(-\chi_p \frac{\Delta t}{8}\right) \eta(t+\frac{1}{4}\Delta t) \\
\underline{v}(t) &\leftarrow \exp\left[-\left(1+\frac{3}{f}\right)\eta(t+\frac{1}{4}\Delta t)\,\frac{\Delta t}{2}\right] \underline{v}(t) \\
\eta(t+\frac{1}{4}\Delta t) &\leftarrow \exp\left(-\chi_p \frac{\Delta t}{8}\right) \eta(t+\frac{1}{4}\Delta t) \\
\eta(t+\frac{1}{2}\Delta t) &\leftarrow \eta(t+\frac{1}{4}\Delta t) + \frac{\Delta t}{4}\left[3V(t)\frac{\mathcal{P}(t)-P_{\rm ext}}{p_{mass}} + \right. \\
&\qquad \left. 3\frac{2E_{kin}(t)}{f}\frac{1}{p_{mass}} + \frac{R_p(t)}{p_{mass}}\right] \\
\eta(t+\frac{1}{2}\Delta t) &\leftarrow \exp\left(-\chi_p \frac{\Delta t}{8}\right) \eta(t+\frac{1}{2}\Delta t)
\end{aligned} \tag{3.75}$$

3. Thermostat: Note $E_{kin}(t)$ has changed and changes inside

$$\underline{v}(t) \leftarrow \exp\left(-\chi \frac{\Delta t}{4}\right) \underline{v}(t) \tag{3.76}$$

4. VV1:

$$\begin{aligned}
\underline{v}(t+\frac{1}{2}\Delta t) &\leftarrow \underline{v}(t) + \frac{\Delta t}{2}\,\frac{\underline{f}(t)+\underline{R}(t)}{m} \\
\underline{\underline{\mathbf{H}}}(t+\Delta t) &\leftarrow \exp\left[\eta(t+\frac{1}{2}\Delta t)\;\Delta t\right] \underline{\underline{\mathbf{H}}}(t) \\
V(t+\Delta t) &\leftarrow \exp\left[3\eta(t+\frac{1}{2}\Delta t)\;\Delta t\right] V(t) \\
\underline{r}(t+\Delta t) &\leftarrow \exp\left[\eta(t+\frac{1}{2}\Delta t)\;\Delta t\right] \underline{r}(t) + \Delta t\;\underline{v}(t+\frac{1}{2}\Delta t)
\end{aligned} \tag{3.77}$$

5. RATTLE_VV1

6. FF:

$$\begin{aligned}
\underline{f}(t+\Delta t) &\leftarrow \underline{f}(t) \\
\underline{R}(t+\Delta t) &\leftarrow \underline{R}(t) \\
R_p(t+\Delta t) &\leftarrow R_p(t)
\end{aligned} \tag{3.78}$$

7. VV2:

$$\underline{v}(t+\Delta t) \leftarrow \underline{v}(t+\frac{\Delta t}{2}) + \frac{\Delta t}{2}\,\frac{\underline{f}(t)+\underline{R}(t)}{m} \tag{3.79}$$

8. RATTLE_VV2

9. Thermostat: Note $E_{kin}(t+\Delta t)$ has changed and changes inside

$$\underline{v}(t+\Delta t) \leftarrow \exp\left(-\chi\,\frac{\Delta t}{4}\right)\;\underline{v}(t+\Delta t) \tag{3.80}$$

10. Barostat: Note $E_{kin}(t+\Delta t)$ and $\mathcal{P}(t+\Delta t)$ have changed and change inside

$$\begin{aligned}
\eta(t+\tfrac{1}{2}\Delta t) &\leftarrow \exp\left(-\chi_p\,\frac{\Delta t}{8}\right)\;\eta(t+\tfrac{1}{2}\Delta t) \\
\eta(t+\tfrac{3}{4}\Delta t) &\leftarrow \eta(t+\tfrac{1}{2}\Delta t)+\frac{\Delta t}{4}\;\Bigg[3V(t+\Delta t)\frac{\mathcal{P}(t+\Delta t)-P_{\rm ext}}{p_{mass}}+ \\
&\qquad 3\frac{2E_{kin}(t+\Delta t)}{f}\frac{1}{p_{mass}}+\frac{R_p(t)}{p_{mass}}\Bigg] \\
\eta(t+\tfrac{3}{4}\Delta t) &\leftarrow \exp\left(-\chi_p\,\frac{\Delta t}{8}\right)\;\eta(t+\tfrac{3}{4}\Delta t) \\
\underline{v}(t+\Delta t) &\leftarrow \exp\left[-\left(1+\frac{3}{f}\right)\eta(t+\tfrac{3}{4}\Delta t)\;\frac{\Delta t}{2}\right]\;\underline{v}(t+\Delta t) \\
\eta(t+\tfrac{3}{4}\Delta t) &\leftarrow \exp\left(-\chi_p\,\frac{\Delta t}{8}\right)\;\eta(t+\tfrac{3}{4}\Delta t) \\
\eta(t+\Delta t) &\leftarrow \eta(t+\tfrac{3}{4}\Delta t)+\frac{\Delta t}{4}\;\Bigg[3V(t+\Delta t)\frac{\mathcal{P}(t+\Delta t)-P_{\rm ext}}{p_{mass}}+ \\
&\qquad 3\frac{2E_{kin}(t+\Delta t)}{f}\frac{1}{p_{mass}}+\frac{R_p(t)}{p_{mass}}\Bigg] \\
\eta(t+\Delta t) &\leftarrow \exp\left(-\chi_p\,\frac{\Delta t}{8}\right)\;\eta(t+\Delta t)
\end{aligned} \tag{3.81}$$

11. Thermostat: Note $E_{kin}(t+\Delta t)$ has changed and changes inside

$$\underline{v}(t+\Delta t) \leftarrow \exp\left(-\chi\,\frac{\Delta t}{4}\right)\;\underline{v}(t+\Delta t)\quad , \tag{3.82}$$

The VV flavour of the langevin barostat (and Nosé-Hoover thermostat) is implemented in the DL_POLY_4 routine NPT_L0_VV. The routine NPT_L1_VV implements the same but also incorporate RB dynamics.

Cell size and shape variations

The isotropic algorithms may be extended to allowing the cell shape to vary by defining η as a tensor, $\underline{\underline{\eta}}$ and extending the Langevin pressure variable R_p to a stochastic (Langevin) tensor $\underline{\underline{\mathbf{R_p}}}$:

$$\left\langle R_{p,i}(t)\;R_{p,j}(t')\right\rangle = 2\;\chi_p\;p_{mass}\;k_BT\;\delta_{ij}\;\delta(t-t')\quad , \tag{3.83}$$

which is drawn from Gaussian distribution of zero mean and unit variance, Gauss$(0,1)$, scaled by $\sqrt{\frac{2\;\chi_p\;p_{mass}\;k_BT}{\Delta t}}$. k_B is the Boltzmann constant, T the target temperature and p_{mass} the barostat mass. **Note** that $\underline{\underline{\mathbf{R_p}}}$ has to be symmetric and only 6 independent components must be generated each timestep.

The equations of motion are written in the same fashion as is in the isotropic algorithm with slight modifications (as now the equations with η are extended to matrix forms)

$$
\begin{aligned}
\frac{d}{dt}\underline{r}(t) &= \underline{v}(t) + \underline{\underline{\eta(\mathbf{t})}} \cdot \underline{r}(t) \\
\frac{d}{dt}\underline{v}(t) &= \frac{\underline{f}(t) + \underline{R}(t)}{m} - \left[\chi\,\underline{\underline{\mathbf{1}}} + \underline{\underline{\eta}}(t) + \frac{\mathtt{Tr}\left[\underline{\underline{\eta}}(t)\right]}{f}\,\underline{\underline{\mathbf{1}}}\right] \cdot \underline{v}(t) \\
\frac{d}{dt}\underline{\underline{\eta}}(t) &= \frac{\underline{\underline{\sigma}}(t) - P_{\rm ext}\;V(t)\;\underline{\underline{\mathbf{1}}}}{p_{mass}} + \frac{2E_{kin}(t)}{f}\frac{\underline{\underline{\mathbf{1}}}}{p_{mass}} - \chi_p \underline{\underline{\eta}}(t) + \frac{\underline{\underline{\mathbf{R_p}}}}{p_{mass}} \\
p_{mass} &= \frac{(f+3)}{3}\;\frac{k_B\;T_{\rm ext}}{(2\pi\;\chi_P)^2} \\
\frac{d}{dt}\underline{\underline{\mathbf{H}}}(t) &= \underline{\underline{\eta}}(t) \cdot \underline{\underline{\mathbf{H}}}(t) \\
\frac{d}{dt}V(t) &= \mathtt{Tr}[\underline{\underline{\eta}}(t)]\;V(t) \quad .
\end{aligned}
\tag{3.84}
$$

where $\underline{\underline{\sigma}}$ is the stress tensor (equation (3.70)) and $\underline{\underline{\mathbf{1}}}$ is the identity matrix.

The conserved quantity these generate is:

$$
\mathcal{H}_{{\rm N}\underline{\underline{\sigma}}{\rm T}} = \mathcal{H}_{\rm NVE} + \frac{p_{mass}\;\mathtt{Tr}[\underline{\underline{\eta}} \cdot \underline{\underline{\eta}}^T]}{2} + P_{\rm ext}V(t) \quad . \tag{3.85}
$$

the VV algorithmic equations are, therefore, written in the same fashion as in the isotropic case with slight modifications. For the VV couched algorithm these are of the following sort

$$
\begin{aligned}
\underline{\underline{\eta}}(t) &\leftarrow \exp\left(-\chi_p\,\frac{\Delta t}{8}\right)\,\underline{\underline{\eta}}(t) \\
\underline{\underline{\eta}}(t+\frac{1}{4}\Delta t) &\leftarrow \underline{\underline{\eta}}(t) + \\
&\quad \frac{\Delta t}{4}\left[\frac{\underline{\underline{\sigma}}(t) - P_{\rm ext}\;V(t)\;\underline{\underline{\mathbf{1}}}}{p_{mass}} + \frac{2E_{kin}(t)}{f}\frac{\underline{\underline{\mathbf{1}}}}{p_{mass}} + \frac{\underline{\underline{\mathbf{R_p}}}(t)}{p_{mass}}\right] \\
\underline{v}(t) &\leftarrow \exp\left[-\left(\underline{\underline{\eta}}(t+\frac{1}{4}\Delta t) + \frac{1}{f}\mathtt{Tr}\left[\underline{\underline{\eta}}(t+\frac{1}{4}\Delta t)\right]\right)\,\frac{\Delta t}{2}\right] \cdot \underline{v}(t) \\
\underline{r}(t+\Delta t) &\leftarrow \exp\left[\underline{\underline{\eta}}(t+\frac{1}{2}\Delta t)\;\Delta t\right] \cdot \underline{r}(t) + \Delta t\;\underline{v}(t+\frac{1}{2}\Delta t)
\end{aligned}
\tag{3.86}
$$

This ensemble is optionally extending to constant normal pressure and constant surface area, NP$_n$AT [87], by semi-isotropic constraining of the barostat equation of motion to:

$$
\frac{d}{dt}\eta_{\alpha\beta}(t) = \begin{cases} \frac{\sigma_{zz}(t) - P_{\rm ext}\;V(t)}{p_{mass}} + \frac{2E_{kin}(t)}{f\;p_{mass}} - \chi_p\eta_{zz}(t) + \frac{R_{p,zz}(t)}{p_{mass}} & : \quad (\alpha=\beta)=z \\ 0 \quad ; \quad \eta_{\alpha\beta}(0) = 0 & : \quad (\alpha,\beta) \neq z \quad . \end{cases} \tag{3.87}
$$

Similarly, this ensemble is optionally extending to constant normal pressure and constant surface tension, NP$_n\gamma$T [87], by semi-isotropic constraining of the barostat equation of motion to:

$$
\frac{d}{dt}\eta_{\alpha\beta}(t) = \begin{cases} \frac{\sigma_{\alpha\alpha}(t) - [P_{\rm ext} - \gamma_{\rm ext}/h_z(t)]\;V(t)}{p_{mass}} + \frac{2E_{kin}(t)}{f\;p_{mass}} - \chi_p\eta_{\alpha\alpha}(t) + \frac{R_{p,\alpha\alpha}(t)}{p_{mass}} & : \quad (\alpha=\beta)=x,y \\ & : \\ \frac{\sigma_{zz}(t) - P_{\rm ext}\;V(t)}{p_{mass}} + \frac{2E_{kin}(t)}{f}\frac{1}{p_{mass}} - \chi_p\eta_{zz}(t) + \frac{R_{p,zz}(t)}{p_{mass}} & : \quad (\alpha=\beta)=z \\ 0 \quad ; \quad \eta_{\alpha\beta}(0) = 0 & : \quad (\alpha\neq\beta)=x,y,z \end{cases} , \tag{3.88}
$$

where $\gamma_{\rm ext}$ is the user defined external surface tension and $h_z(t) = V(t)/A_{xy}(t)$ is the instantaneous hight of the MD box (or MD box volume over area). The instnatneous surface tension is defined as

$$
\gamma_\alpha(t) = -h_z(t)\,[\sigma_{\alpha\alpha}(t) - P_{\rm ext}] \quad . \tag{3.89}
$$

The case $\gamma_{\text{ext}} = 0$ generates the NPT anisotropic ensemble for the orthorhombic cell (imcon=2 in CONFIG, see Appendix B). This can be considered as an "orthorhombic" constraint on the NσT ensemble. The constraint can be strengthened further, to a "semi-orthorhombic" one, by imposing that the MD cell change isotropically in the (x, y) plane which leads to the following modification in the N$P_n\gamma$T set of equatons

$$\frac{d}{dt}\eta_{\alpha\alpha}(t) = \frac{\left[\sigma_{xx}(t) + \sigma_{yy}(t)\right]/2 - \left[P_{\text{ext}} - \gamma_{\text{ext}}/h_z(t)\right]\ V(t)}{p_{mass}} + \frac{2\ E_{kin}(t)}{f\ p_{mass}} - \\ - \chi_p\eta_{\alpha\alpha}(t) + \frac{R_{p,xx}(t) + R_{p,yy}(t)}{2\ p_{mass}} \quad : \quad (\alpha = \beta) = x, y \quad . \tag{3.90}$$

The VV flavour of the non-isotropic Langevin barostat (and Nosé-Hoover thermostat) is implemented in the DL_POLY_4 routine NST_L0_VV. The routine NST_L1_VV implements the same but also incorporate RB dynamics.

3.5.3 Berendsen Barostat

With the Berendsen barostat the system is made to obey the equation of motion at the beginning of each step

$$\frac{d\mathcal{P}(t)}{dt} = \frac{P_{\text{ext}} - \mathcal{P}(t)}{\tau_P} \quad , \tag{3.91}$$

where $\mathcal{P}$ is the instantaneous pressure (equation (3.69)) and τ_P is the barostat relaxation time constant.

Cell size variations

In the isotropic implementation, at each step the MD cell volume is scaled by a factor η, and the coordinates and cell vectors by $\eta^{1/3}$,

$$\eta(t) = 1 - \frac{\beta\Delta t}{\tau_P}\left(P_{\text{ext}} - \mathcal{P}(t)\right) \tag{3.92}$$

where β is the isothermal compressibility of the system. In practice β is a specified constant which DL_POLY_4 takes to be the isothermal compressibility of liquid water. The exact value is not critical to the algorithm as it relies on the ratio τ_P/β. τ_P is a specified time constant for pressure fluctuations, supplied by the user.

It is worth noting that the barostat and the thermostat are independent and fully separable.

The VV implementation of the Berendsen algorithm only requires iterations if bond or PMF constraints are present (13 until satisfactory convergence of the constraint forces is achieved). These are with respect to the pressure (i.e. $\eta(t)$) in the first part, VV1+RATTLE_VV1. The second part is conventional, VV2+RATTLE_VV2, as at the end the velocities are scaled by a factor of χ.

1. VV1:

$$\begin{aligned} \underline{v}(t + \frac{1}{2}\Delta t) &\leftarrow \underline{v}(t) + \frac{\Delta t}{2}\frac{\underline{f}(t)}{m} \\ \underline{r}(t + \Delta t) &\leftarrow \eta(t)^{1/3}\ \underline{r}(t) + \Delta t\ \underline{v}(t + \frac{1}{2}\Delta t) \\ \underline{\underline{\mathbf{H}}}(t + \Delta t) &\leftarrow \eta(t)^{1/3}\ \underline{\underline{\mathbf{H}}}(t) \\ V(t + \Delta t) &\leftarrow \eta(t)\ V(t) \end{aligned} \tag{3.93}$$

2. RATTLE_VV1

3. Barostat:

$$\eta(t) = 1 - \frac{\beta\Delta t}{\tau_P}\left(P_{\text{ext}} - \mathcal{P}(t)\right) \tag{3.94}$$

4. FF:
$$\underline{f}(t+\Delta t) \leftarrow \underline{f}(t) \tag{3.95}$$

5. VV2:
$$\underline{v}(t+\Delta t) \leftarrow \underline{v}(t+\frac{1}{2}\Delta t) + \frac{\Delta t}{2}\,\frac{\underline{f}(t+\Delta t)}{m} \tag{3.96}$$

6. RATTLE_VV2

7. Thermostat:
$$\begin{aligned} \chi(t+\Delta t) &\leftarrow \left[1+\frac{\Delta t}{\tau_T}\left(\frac{\sigma}{E_{kin}(t+\Delta t)}-1\right)\right]^{1/2} \\ \underline{v}(t+\Delta t) &\leftarrow \underline{v}(t+\Delta t)\;\chi \quad . \end{aligned} \tag{3.97}$$

where $\underline{\underline{\mathbf{H}}}$ is the cell matrix whose columns are the three cell vectors $\underline{a}, \underline{b}, \underline{c}$.

The Berendsen algorithms conserve total momentum but not energy.

The VV flavour of the Berendsen barostat (and thermostat) is implemented in the DL_POLY_4 routine NPT_B0_VV. The routines NPT_B1_VV implements the same but also incorporate RB dynamics.

Cell size and shape variations

The extension of the isotropic algorithm to anisotropic cell variations is straightforward. A tensor $\underline{\underline{\eta}}$ is defined as
$$\underline{\underline{\eta}}(t) = \underline{\underline{\mathbf{1}}} - \frac{\beta\Delta t}{\tau_P}(P_{\rm ext}\;\underline{\underline{\mathbf{1}}} - \underline{\underline{\sigma}}(t)/V(t)) \quad , \tag{3.98}$$
where where $\underline{\underline{\sigma}}$ is the stress tensor (equation (3.70)) and $\underline{\underline{\mathbf{1}}}$ is the identity matrix. Then new cell vectors and volume are given by
$$\begin{aligned} \underline{\underline{\mathbf{H}}}(t+\Delta t) &\leftarrow \underline{\underline{\eta}}(t)\cdot\underline{\underline{\mathbf{H}}}(t) \\ V(t+\Delta t) &\leftarrow \mathtt{Tr}[\underline{\underline{\eta}}(t)]\;V(t) \quad . \end{aligned} \tag{3.99}$$
and the velocity updates as
$$\mathtt{VV1}: \quad \underline{r}(t+\Delta t) \leftarrow \underline{\underline{\eta}}(t)\cdot\underline{r}(t) + \Delta t\;\underline{v}(t+\frac{1}{2}\Delta t)$$

This ensemble is optionally extending to constant normal pressure and constant surface area, NP$_n$AT [87], by semi-isotropic constraining of the barostat equation of motion to:
$$\eta_{\alpha\delta}(t) = \begin{cases} 1-\frac{\beta\Delta t}{\tau_P}\left[P_{\rm ext}-\sigma_{zz}(t)/V(t)\right] & : \quad (\alpha=\delta)=z \\ 1 & : \quad (\alpha=\delta)=x,y \\ 0 & : \quad (\alpha\neq\delta) \quad . \end{cases} \tag{3.100}$$

Similarly, this ensemble is optionally extending to constant normal pressure and constant surface tension, NP$_n\gamma$T [87], by semi-isotropic constraining of the barostat equation of motion to:
$$\eta_{\alpha\delta}(t) = \begin{cases} 1-\frac{\beta\Delta t}{\tau_P}\left[P_{\rm ext}-\gamma_{\rm ext}\;V(t)/h_z(t)-\sigma_{\alpha\alpha}(t)/V(t)\right] & : \quad (\alpha=\delta)=x,y \\ & : \\ 1-\frac{\beta\Delta t}{\tau_P}\left[P_{\rm ext}-\sigma_{zz}(t)/V(t)\right] & : \quad (\alpha=\delta)=z \\ 0 & : \quad (\alpha\neq\delta) \quad , \end{cases} \tag{3.101}$$

where $\gamma_{\rm ext}$ is the user defined external surface tension and $h_z(t) = V(t)/A_{xy}(t)$ is the instantaneous hight of the MD box (or MD box volume over area). One defines the instantaneous surface tension as given in equation (3.89). The case $\gamma_{\rm ext} = 0$ generates the NPT anisotropic ensemble for the orthorhombic cell (imcon=2 in CONFIG, see Appendix B). This can be considered as an "orthorhombic" constraint on the NσT ensemble. The constraint can be strengthened further, to a "semi-orthorhombic" one, by imposing that the MD cell change isotropically in the (x, y) plane which leads to the following change in the equations above

$$\eta_{\alpha\alpha}(t) = 1 - \frac{\beta\Delta t}{\tau_P}\left[P_{\rm ext} - \gamma_{\rm ext}\,\frac{V(t)}{h_z(t)} - \frac{\sigma_{xx}(t)+\sigma_{yy}(t)}{2\,V(t)}\right] \quad : \quad (\alpha = \delta) = x, y \quad . \tag{3.102}$$

The VV flavour of the non-isotropic Berendsen barostat (and thermostat) is implemented in the DL_POLY_4 routine NST_B0_VV. The routine NST_B1_VV implements the same but also incorporate RB dynamics.

3.5.4 Nosé-Hoover Barostat

DL_POLY_4 uses the Melchionna modification of the Nosé-Hoover algorithm [94] in which the equations of motion involve a Nosé-Hoover thermostat and a barostat in the same spirit. Additionally, as shown in [95], a modification allowing for coupling between the thermostat and barostat is also introduced.

Cell size variation

For isotropic fluctuations the equations of motion are:

$$\begin{aligned}
\frac{d}{dt}\underline{r}(t) &= \underline{v}(t) + \eta(t)\,(\underline{r}(t) - \underline{R}_0(t)) \\
\frac{d}{dt}\underline{v}(t) &= \frac{\underline{f}(t)}{m} - [\chi(t)+\eta(t)]\,\underline{v}(t) \\
\frac{d}{dt}\chi(t) &= \frac{2E_{kin}(t) + p_{mass}\,\eta(t)^2 - 2\sigma - k_B\,T_{\rm ext}}{q_{mass}} \\
q_{mass} &= 2\,\sigma\,\tau_T^2 \\
\frac{d}{dt}\eta(t) &= 3V(t)\frac{\mathcal{P}(t) - P_{\rm ext}}{p_{mass}} - \chi(t)\eta(t) \\
p_{mass} &= (f+3)\,k_B\,T_{\rm ext}\,\tau_P^2 \\
\frac{d}{dt}\underline{\underline{\mathbf{H}}}(t) &= \eta(t)\,\underline{\underline{\mathbf{H}}}(t) \\
\frac{d}{dt}V(t) &= [3\eta(t)]\,V(t) \quad ,
\end{aligned} \tag{3.103}$$

where η is the barostat friction coefficient, $\underline{R}_0(t)$ the system centre of mass at time t, q_{mass} the thermostat mass, τ_T a specified time constant for temperature fluctuations, σ the target thermostat energy (equation (3.41)), p_{mass} the barostat mass, τ_P a specified time constant for pressure fluctuations, $\mathcal{P}$ the instantaneous pressure (equation (3.69)) and V the system volume. $\underline{\underline{\mathbf{H}}}$ is the cell matrix whose columns are the three cell vectors $\underline{a}, \underline{b}, \underline{c}$.

The conserved quantity is, to within a constant, the Gibbs free energy of the system:

$$\mathcal{H}_{\rm NPT} = \mathcal{H}_{\rm NVE} + \frac{q_{mass}\,\chi(t)^2}{2} + \frac{p_{mass}\,\eta(t)^2}{2} + P_{\rm ext}V(t) + (f+1)\,k_B\,T_{\rm ext}\int_o^t \chi(s)ds \quad , \tag{3.104}$$

where f is the system's degrees of freedom - equation (3.7).

The VV implementation of the Nosé-Hoover algorithm only requires iterations if bond or PMF constraints are present (5 until satisfactory convergence of the constraint forces is achieved). These are with respect to the pressure (i.e. $\eta(t)$) in the first part, VV1+RATTLE_VV1. The second part is conventional, VV2+RATTLE_VV2, as at the end the velocities are scaled by a factor of χ.

1. Thermostat: Note $E_{kin}(t)$ changes inside

$$\begin{aligned}
\chi(t+\frac{1}{8}\Delta t) &\leftarrow \chi(t)+\frac{\Delta t}{8}\,\frac{2E_{kin}(t)+p_{mass}\;\eta(t)^2-2\sigma-k_B\;T_{\rm ext}}{q_{mass}} \\
\underline{v}(t) &\leftarrow \exp\left(-\chi(t+\frac{1}{8}\Delta t)\,\frac{\Delta t}{4}\right)\,\underline{v}(t) \\
\chi(t+\frac{1}{4}\Delta t) &\leftarrow \chi(t+\frac{1}{8}\Delta t)+\frac{\Delta t}{8}\,\frac{2E_{kin}(t)+p_{mass}\;\eta(t)^2-2\sigma-k_B\;T_{\rm ext}}{q_{mass}}
\end{aligned} \tag{3.105}$$

2. Barostat: Note $E_{kin}(t)$ and $\mathcal{P}(t)$ have changed and change inside

$$\begin{aligned}
\eta(t) &\leftarrow \exp\left(-\chi(t+\frac{1}{4}\Delta t)\,\frac{\Delta t}{8}\right)\,\eta(t) \\
\eta(t+\frac{1}{4}\Delta t) &\leftarrow \eta(t)+\frac{\Delta t}{4}\,\frac{3\,[\mathcal{P}(t)-P_{\rm ext}]\,V(t)}{p_{mass}} \\
\eta(t+\frac{1}{4}\Delta t) &\leftarrow \exp\left(-\chi(t+\frac{1}{4}\Delta t)\,\frac{\Delta t}{8}\right)\,\eta(t+\frac{1}{4}\Delta t) \\
\underline{v}(t) &\leftarrow \exp\left[-\eta(t+\frac{1}{4}\Delta t)\,\frac{\Delta t}{2}\right]\,\underline{v}(t) \\
\eta(t+\frac{1}{4}\Delta t) &\leftarrow \exp\left(-\chi(t+\frac{1}{4}\Delta t)\,\frac{\Delta t}{8}\right)\,\eta(t+\frac{1}{4}\Delta t) \\
\eta(t+\frac{1}{2}\Delta t) &\leftarrow \eta(t+\frac{1}{4}\Delta t)+\frac{\Delta t}{4}\,\frac{3\,[\mathcal{P}(t)-P_{\rm ext}]\,V(t)}{p_{mass}} \\
\eta(t+\frac{1}{2}\Delta t) &\leftarrow \exp\left(-\chi(t+\frac{1}{4}\Delta t)\,\frac{\Delta t}{8}\right)\,\eta(t+\frac{1}{2}\Delta t)
\end{aligned} \tag{3.106}$$

3. Thermostat: Note $E_{kin}(t)$ has changed and changes inside

$$\begin{aligned}
\chi(t+\frac{3}{8}\Delta t) &\leftarrow \chi(t+\frac{1}{4}\Delta t)+\frac{\Delta t}{8}\,\frac{2E_{kin}(t)+p_{mass}\;\eta(t+\frac{1}{2}\Delta t)^2-2\sigma-k_B\;T_{\rm ext}}{q_{mass}} \\
\underline{v}(t) &\leftarrow \exp\left(-\chi(t+\frac{3}{8}\Delta t)\,\frac{\Delta t}{4}\right)\,\underline{v}(t) \\
\chi(t+\frac{1}{2}\Delta t) &\leftarrow \chi(t+\frac{3}{8}\Delta t)+\frac{\Delta t}{8}\,\frac{2E_{kin}(t)+p_{mass}\;\eta(t+\frac{1}{2}\Delta t)^2-2\sigma-k_B\;T_{\rm ext}}{q_{mass}}
\end{aligned} \tag{3.107}$$

4. VV1:

$$\begin{aligned}
\underline{v}(t+\frac{1}{2}\Delta t) &\leftarrow \underline{v}(t)+\frac{\Delta t}{2}\,\frac{\underline{f}(t)}{m} \\
\underline{\underline{\mathbf{H}}}(t+\Delta t) &\leftarrow \exp\left[\eta(t+\frac{1}{2}\Delta t)\;\Delta t\right]\,\underline{\underline{\mathbf{H}}}(t) \\
V(t+\Delta t) &\leftarrow \exp\left[3\eta(t+\frac{1}{2}\Delta t)\;\Delta t\right]\,V(t) \\
\underline{r}(t+\Delta t) &\leftarrow \exp\left[\eta(t+\frac{1}{2}\Delta t)\;\Delta t\right]\,(\underline{r}(t)-\underline{R}_0(t))+\Delta t\;\underline{v}(t+\frac{1}{2}\Delta t)+\underline{R}_0(t)
\end{aligned} \tag{3.108}$$

5. RATTLE_VV1

6. FF:

$$\underline{f}(t+\Delta t) \leftarrow \underline{f}(t) \tag{3.109}$$

7. VV2:

$$\underline{v}(t+\Delta t) \leftarrow \underline{v}(t+\frac{\Delta t}{2}) + \frac{\Delta t}{2}\,\frac{\underline{f}(t)}{m} \tag{3.110}$$

8. RATTLE_VV2

9. Thermostat: Note $E_{kin}(t+\Delta t)$ has changed and changes inside

$$\begin{aligned}
\chi(t+\frac{5}{8}\Delta t) &\leftarrow \chi(t+\frac{1}{2}\Delta t) + \frac{\Delta t}{8}\,\frac{2E_{kin}(t+\Delta t) + p_{mass}\;\eta(t+\frac{1}{2}\Delta t)^2 - 2\sigma - k_B\;T_{\text{ext}}}{q_{mass}} \\
\underline{v}(t+\Delta t) &\leftarrow \exp\left(-\chi(t+\frac{5}{8}\Delta t)\;\frac{\Delta t}{4}\right)\;\underline{v}(t+\Delta t) \\
\chi(t+\frac{3}{4}\Delta t) &\leftarrow \chi(t+\frac{5}{8}\Delta t) + \frac{\Delta t}{8}\,\frac{2E_{kin}(t+\Delta t) + p_{mass}\;\eta(t+\frac{1}{2}\Delta t)^2 - 2\sigma - k_B\;T_{\text{ext}}}{q_{mass}}
\end{aligned} \tag{3.111}$$

10. Barostat: Note $E_{kin}(t+\Delta t)$ and $\mathcal{P}(t+\Delta t)$ have changed and change inside

$$\begin{aligned}
\eta(t+\frac{1}{2}\Delta t) &\leftarrow \exp\left(-\chi(t+\frac{3}{4}\Delta t)\;\frac{\Delta t}{8}\right)\;\eta(t+\frac{1}{2}\Delta t) \\
\eta(t+\frac{3}{4}\Delta t) &\leftarrow \eta(t+\frac{1}{2}\Delta t) + \frac{\Delta t}{4}\;\frac{3\left[\mathcal{P}(t+\Delta t) - P_{\text{ext}}\right]V(t+\Delta t)}{p_{mass}} \\
\eta(t+\frac{3}{4}\Delta t) &\leftarrow \exp\left(-\chi(t+\frac{3}{4}\Delta t)\;\frac{\Delta t}{8}\right)\;\eta(t+\frac{3}{4}\Delta t) \\
\underline{v}(t+\Delta t) &\leftarrow \exp\left[-\eta(t+\frac{3}{4}\Delta t)\;\frac{\Delta t}{2}\right]\;\underline{v}(t+\Delta t) \\
\eta(t+\frac{3}{4}\Delta t) &\leftarrow \exp\left(-\chi(t+\frac{3}{4}\Delta t)\;\frac{\Delta t}{8}\right)\;\eta(t+\frac{3}{4}\Delta t) \\
\eta(t+\Delta t) &\leftarrow \eta(t+\frac{3}{4}\Delta t) + \frac{\Delta t}{4}\;\frac{3\left[\mathcal{P}(t+\Delta t) - P_{\text{ext}}\right]V(t+\Delta t)}{p_{mass}} \\
\eta(t+\Delta t) &\leftarrow \exp\left(-\chi(t+\frac{3}{4}\Delta t)\;\frac{\Delta t}{8}\right)\;\eta(t+\Delta t)
\end{aligned} \tag{3.112}$$

11. Thermostat: Note $E_{kin}(t+\Delta t)$ has changed and changes inside

$$\begin{aligned}
\chi(t+\frac{7}{8}\Delta t) &\leftarrow \chi(t+\frac{3}{4}\Delta t) + \frac{\Delta t}{8}\,\frac{2E_{kin}(t+\Delta t) + p_{mass}\;\eta(t+\Delta t)^2 - 2\sigma - k_B\;T_{\text{ext}}}{q_{mass}} \\
\underline{v}(t+\Delta t) &\leftarrow \exp\left(-\chi(t+\frac{7}{8}\Delta t)\;\frac{\Delta t}{4}\right)\;\underline{v}(t+\Delta t) \\
\chi(t+\Delta t) &\leftarrow \chi(t+\frac{7}{8}\Delta t) + \frac{\Delta t}{8}\,\frac{2E_{kin}(t+\Delta t) + p_{mass}\;\eta(t+\Delta t)^2 - 2\sigma - k_B\;T_{\text{ext}}}{q_{mass}} \\
\underline{v}(t+\Delta t) &\leftarrow \underline{v}(t+\Delta t) - \underline{V}_0(t+\Delta t)\quad ,
\end{aligned} \tag{3.113}$$

where $\underline{V}_0(t+\Delta t)$ is the c.o.m. velocity at timestep $t+\Delta t$ and $\underline{\underline{\mathbf{H}}}$ is the cell matrix whose columns are the three cell vectors $\underline{a}, \underline{b}, \underline{c}$.

The VV flavour of the Nosé-Hoover barostat (and thermostat) is implemented in the DL_POLY_4 routine NPT_H0_VV. The routine NPT_H1_VV implements the same but also incorporate RB dynamics.

Cell size and shape variation

The isotropic algorithmscmay be extended to allowing the cell shape to vary by defining η as a tensor, $\underline{\underline{\eta}}$. The equations of motion are written in the same fashion as is in the isotropic algorithm with slight modifications (as now the equations with η are extended to matrix forms)

$$\begin{aligned}
\frac{d}{dt}\underline{r}(t) &= \underline{v}(t) + \underline{\underline{\eta}}(t) \cdot (\underline{r}(t) - \underline{R}_0(t)) \\
\frac{d}{dt}\underline{v}(t) &= \frac{\underline{f}(t)}{m} - \left[\chi(t)\ \underline{\underline{\mathbf{1}}} + \underline{\underline{\eta}}(t)\right] \cdot \underline{v}(t) \\
\frac{d}{dt}\chi(t) &= \frac{2E_{kin}(t) + p_{mass}\ \mathtt{Tr}[\underline{\underline{\eta}}(t) \cdot \underline{\underline{\eta}}(t)^T] - 2\sigma - 3^2\ k_B\ T_{\rm ext}}{q_{mass}} \\
q_{mass} &= 2\ \sigma\ \tau_T^2 \\
\frac{d}{dt}\underline{\underline{\eta}}(t) &= \frac{\underline{\underline{\sigma}}(t) - P_{\rm ext}\ V(t)\ \underline{\underline{\mathbf{1}}}}{p_{mass}} - \chi(t)\underline{\underline{\eta}}(t) \\
p_{mass} &= \frac{(f+3)}{3}\ k_B\ T_{\rm ext}\ \tau_P^2 \\
\frac{d}{dt}\underline{\underline{\mathbf{H}}}(t) &= \underline{\underline{\eta}}(t) \cdot \underline{\underline{\mathbf{H}}}(t) \\
\frac{d}{dt}V(t) &= \mathtt{Tr}[\underline{\underline{\eta}}(t)]\ V(t)\quad ,
\end{aligned} \tag{3.114}$$

where $\underline{\underline{\sigma}}$ is the stress tensor (equation (3.70)) and $\underline{\underline{\mathbf{1}}}$ is the identity matrix. The VV algorithmic equations are, therefore, written in the same fashion as above with slight modifications in (i) the equations for the thermostat and barostat frictions, and (ii) the equations for the system volume and cell parameters. The modifications in (i) for the VV couched algorithm are of the following sort

$$\begin{aligned}
\chi(t + \frac{1}{8}\Delta t) &\leftarrow \chi(t) + \frac{\Delta t}{8}\ \frac{2E_{kin}(t) + p_{mass}\ \mathtt{Tr}[\underline{\underline{\eta}}(t) \cdot \underline{\underline{\eta}}(t)^T] - 2\sigma - 3^2\ k_B\ T_{\rm ext}}{q_{mass}} \\
\underline{v}(t) &\leftarrow \exp\left[-\underline{\underline{\eta}}(t + \frac{1}{4}\Delta t)\ \frac{\Delta t}{2}\right] \cdot \underline{v}(t) \\
\underline{\underline{\eta}}(t + \frac{1}{4}\Delta t) &\leftarrow \underline{\underline{\eta}}(t) + \frac{\Delta t}{4}\ \frac{\underline{\underline{\sigma}}(t) - P_{\rm ext}\ V(t)\ \underline{\underline{\mathbf{1}}}}{p_{mass}}\quad ,
\end{aligned} \tag{3.115}$$

The modifications in (ii) couched algorithms

$$\begin{aligned}
\underline{\underline{\mathbf{H}}}(t + \Delta t) &\leftarrow \exp\left(\underline{\underline{\eta}}(t + \frac{1}{2}\Delta t)\ \Delta t\right) \cdot \underline{\underline{\mathbf{H}}}(t) \\
V(t + \Delta t) &\leftarrow \exp\left(\mathtt{Tr}\left[\underline{\underline{\eta}}(t + \frac{1}{2}\Delta t)\right]\ \Delta t\right)\ V(t)\quad .
\end{aligned} \tag{3.116}$$

It is worth noting DL_POLY_4 uses Taylor expansion truncated to the quadratic term to approximate exponentials of tensorial terms.

The conserved quantity is, to within a constant, the Gibbs free energy of the system:

$$\mathcal{H}_{\rm N\underline{\underline{\sigma}}T} = \mathcal{H}_{\rm NVE} + \frac{q_{mass}\ \chi(t)^2}{2} + \frac{p_{mass}\ \mathtt{Tr}[\underline{\underline{\eta}} \cdot \underline{\underline{\eta}}^T]}{2} + P_{\rm ext}V(t) + (f + 3^2)\ k_B\ T_{\rm ext} \int_o^t \chi(s)ds\quad , \tag{3.117}$$

where f is the system's degrees of freedom - equation (3.7).

This ensemble is optionally extending to constant normal pressure and constant surface area, NP$_n$AT [87], by semi-isotropic constraining of the barostat equation of motion and slight amending the thermostat equation of motion and the conserved quantity to:

$$\begin{aligned}
\frac{d}{dt}\eta_{\alpha\beta}(t) &= \begin{cases} \frac{\sigma_{zz}(t)-P_{\rm ext}\ V(t)}{p_{mass}} - \chi(t)\eta_{zz}(t) & : \quad (\alpha=\beta)=z \\ 0 \quad ; \quad \eta_{\alpha\beta}(0)=0 & : \quad (\alpha,\beta)\neq z \end{cases} \\
\frac{d}{dt}\chi(t) &= \frac{2E_{kin}(t)+p_{mass}\ \mathtt{Tr}[\underline{\underline{\eta}}(t)\cdot\underline{\underline{\eta}}(t)^T]-2\sigma-k_B\ T_{\rm ext}}{q_{mass}} \\
\mathcal{H}_{\rm NP_nAT} &= \mathcal{H}_{\rm NVE}+\frac{q_{mass}\ \chi(t)^2}{2}+\frac{p_{mass}\ \mathtt{Tr}[\underline{\underline{\eta}}\cdot\underline{\underline{\eta}}^T]}{2}+P_{\rm ext}V(t)+(f+1)\ k_B\ T_{\rm ext}\int_o^t\chi(s)ds\quad .
\end{aligned} \tag{3.118}$$

Similarly, this ensemble is optionally extending to constant normal pressure and constant surface tension, NP$_n\gamma$T [87], by semi-isotropic constraining of the barostat equation of motion and slight amending the thermostat equation of motion and the conserved quantity to:

$$\begin{aligned}
\frac{d}{dt}\eta_{\alpha\beta}(t) &= \begin{cases} \frac{\sigma_{\alpha\alpha}(t)-[P_{\rm ext}-\gamma_{\rm ext}/h_z(t)]\ V(t)}{p_{mass}} - \chi(t)\eta_{\alpha\alpha}(t) & : \quad (\alpha=\beta)=x,y \\ & : \\ \frac{\sigma_{zz}(t)-P_{\rm ext}\ V(t)}{p_{mass}} - \chi(t)\eta_{zz}(t) & : \quad (\alpha=\beta)=z \\ 0 \quad ; \quad \eta_{\alpha\beta}(0)=0 & : \quad (\alpha\neq\beta)=x,y,z \end{cases} \\
\frac{d}{dt}\chi(t) &= \frac{2E_{kin}(t)+p_{mass}\ \mathtt{Tr}[\underline{\underline{\eta}}(t)\cdot\underline{\underline{\eta}}(t)^T]-2\sigma-3\ k_B\ T_{\rm ext}}{q_{mass}} \\
\mathcal{H}_{\rm NP_n\gamma T} &= \mathcal{H}_{\rm NVE}+\frac{q_{mass}\ \chi(t)^2}{2}+\frac{p_{mass}\ \mathtt{Tr}[\underline{\underline{\eta}}\cdot\underline{\underline{\eta}}^T]}{2}+P_{\rm ext}V(t)+(f+3)\ k_B\ T_{\rm ext}\int_o^t\chi(s)ds\quad .
\end{aligned} \tag{3.119}$$

where $\gamma_{\rm ext}$ is the user defined external surface tension and $h_z(t)=V(t)/A_{xy}(t)$ is the instantaneous hight of the MD box (or MD box volume over area). One defines the instantaneous surface tension as given in equation (3.89). The case $\gamma_{\rm ext}=0$ generates the NPT anisotropic ensemble for the orthorhombic cell (imcon=2 in CONFIG, see Appendix B). This can be considered as an "orthorhombic" constraint on the NσT ensemble. The constraint can be strengthened further, to a "semi-orthorhombic" one, by imposing that the MD cell change isotropically in the (x,y) plane which leads to the following changes in the equations above

$$\begin{aligned}
\frac{d}{dt}\eta_{\alpha\alpha}(t) &= \frac{[\sigma_{xx}(t)+\sigma_{yy}(t)]/2-[P_{\rm ext}-\gamma_{\rm ext}/h_z(t)]\ V(t)}{p_{mass}}-\chi(t)\eta_{\alpha\alpha}(t)\quad : \quad (\alpha=\beta)=x,y \\
\mathcal{H}_{\rm NP_n\gamma=0T} &= \mathcal{H}_{\rm NVE}+\frac{q_{mass}\ \chi(t)^2}{2}+\frac{p_{mass}\ \mathtt{Tr}[\underline{\underline{\eta}}\cdot\underline{\underline{\eta}}^T]}{2}+P_{\rm ext}V(t)+(f+2)\ k_B\ T_{\rm ext}\int_o^t\chi(s)ds\quad .
\end{aligned} \tag{3.120}$$

The VV flavour of the non-isotropic Nosé-Hoover barostat (and thermostat) is implemented in the DL_POLY_4 routine NST_H0_VV. The routine NST_H1_VV implements the same but also incorporate RB dynamics.

3.5.5 Martyna-Tuckerman-Klein Barostat

DL_POLY_4 includes the Martyna-Tuckerman-Klein (MTK) interpretation of the VV flavoured Nosé-Hoover algorithms [31] for isotropic and anisotropic cell fluctuations in which the equations of motion are only slightly augmented with respect to those for the coupled Nosé-Hoover thermostat and barostat. Compare the isotropic cell changes case, equations (3.103), to

$$\begin{aligned}
\frac{d}{dt}\underline{r}(t) &= \underline{v}(t) + \eta(t)\ \underline{r}(t) \\
\frac{d}{dt}\underline{v}(t) &= \frac{\underline{f}(t)}{m} - \left[\chi(t) + \left(1 + \frac{3}{f}\right)\eta(t)\right]\underline{v}(t) \\
\frac{d}{dt}\chi(t) &= \frac{2E_{kin}(t) + p_{mass}\ \eta(t)^2 - 2\sigma - k_B\ T_{\rm ext}}{q_{mass}} \\
q_{mass} &= 2\ \sigma\ \tau_T^2 \\
\frac{d}{dt}\eta(t) &= 3V(t)\frac{\mathcal{P}(t) - P_{\rm ext}}{p_{mass}} + 3\frac{2E_{kin}(t)}{f}\frac{1}{p_{mass}} - \chi(t)\eta(t) \\
p_{mass} &= (f+3)\ k_B\ T_{\rm ext}\ \tau_P^2 \\
\frac{d}{dt}\underline{\underline{\mathbf{H}}}(t) &= \eta(t)\ \underline{\underline{\mathbf{H}}}(t) \\
\frac{d}{dt}V(t) &= [3\eta(t)]\ V(t)\quad ,
\end{aligned} \tag{3.121}$$

and the anisotropic cell change case, equations (3.114), to

$$\begin{aligned}
\frac{d}{dt}\underline{r}(t) &= \underline{v}(t) + \underline{\underline{\eta}}(t)\cdot\underline{r}(t) \\
\frac{d}{dt}\underline{v}(t) &= \frac{\underline{f}(t)}{m} - \left[\chi(t)\ \underline{\underline{\mathbf{1}}} + \underline{\underline{\eta}}(t) + \frac{\mathtt{Tr}\left[\underline{\underline{\eta}}(t)\right]}{f}\ \underline{\underline{\mathbf{1}}}\right]\cdot\underline{v}(t) \\
\frac{d}{dt}\chi(t) &= \frac{2E_{kin}(t) + p_{mass}\ \mathtt{Tr}[\underline{\underline{\eta}}(t)\cdot\underline{\underline{\eta}}(t)^T] - 2\sigma - 3^2\ k_B\ T_{\rm ext}}{q_{mass}} \\
q_{mass} &= 2\ \sigma\ \tau_T^2 \\
\frac{d}{dt}\underline{\underline{\eta}}(t) &= \frac{\underline{\underline{\sigma}}(t) - P_{\rm ext}\ V(t)\ \underline{\underline{\mathbf{1}}}}{p_{mass}} + \frac{2E_{kin}(t)}{f}\frac{\underline{\underline{\mathbf{1}}}}{p_{mass}} - \chi(t)\underline{\underline{\eta}}(t) \\
p_{mass} &= \frac{(f+3)}{3}\ k_B\ T_{\rm ext}\ \tau_P^2 \\
\frac{d}{dt}\underline{\underline{\mathbf{H}}}(t) &= \underline{\underline{\eta}}(t)\cdot\underline{\underline{\mathbf{H}}}(t) \\
\frac{d}{dt}V(t) &= \mathtt{Tr}[\underline{\underline{\eta}}(t)]\ V(t)\quad .
\end{aligned} \tag{3.122}$$

The changes include one extra dependence to the velocity and barostat equations and removal of the centre of mass variable $\underline{R}_0(t)$ dependence in the position equation.

The modifications in for the VV couched algorithms are of the following sort

$$\begin{aligned}
\eta(t + \frac{1}{4}\Delta t) &\leftarrow \eta(t) + \frac{\Delta t}{4}\left[3V(t)\frac{\mathcal{P}(t) - P_{\rm ext}}{p_{mass}} + 3\frac{2E_{kin}(t)}{f}\frac{1}{p_{mass}}\right] \\
\underline{v}(t) &\leftarrow \exp\left[-\left(1 + \frac{3}{f}\right)\eta(t + \frac{1}{4}\Delta t)\ \frac{\Delta t}{2}\right]\underline{v}(t) \\
\underline{r}(t + \Delta t) &\leftarrow \exp\left[\eta(t + \frac{1}{2}\Delta t)\ \Delta t\right]\underline{r}(t) + \Delta t\ \underline{v}(t + \frac{1}{2}\Delta t)
\end{aligned} \tag{3.123}$$

for the isotropic cell fluctuations case and

$$\begin{aligned}
\underline{\underline{\eta}}(t+\frac{1}{4}\Delta t) &\leftarrow \underline{\underline{\eta}}(t)+\frac{\Delta t}{4}\left[\frac{\underline{\underline{\sigma}}(t)-P_{\rm ext}\ V(t)\ \underline{\underline{\mathbf{1}}}}{p_{mass}}+\frac{2E_{kin}(t)}{f}\frac{\underline{\underline{\mathbf{1}}}}{p_{mass}}\right] \\
\underline{v}(t) &\leftarrow \exp\left[-\left(\underline{\underline{\eta}}(t+\frac{1}{4}\Delta t)+\frac{1}{f}\mathtt{Tr}\left[\underline{\underline{\eta}}(t+\frac{1}{4}\Delta t)\right]\right)\frac{\Delta t}{2}\right]\cdot\underline{v}(t) \qquad (3.124)\\
\underline{r}(t+\Delta t) &\leftarrow \exp\left[\underline{\underline{\eta}}(t+\frac{1}{2}\Delta t)\ \Delta t\right]\cdot\underline{r}(t)+\Delta t\ \underline{v}(t+\frac{1}{2}\Delta t)
\end{aligned}$$

for the anisotropic cell fluctuations case.

This ensemble is optionally extending to constant normal pressure and constant surface area, NP$_n$AT [87], by semi-isotropic constraining of the barostat equation of motion and slight amending the thermostat equation of motion and the conserved quantity to:

$$\begin{aligned}
\frac{d}{dt}\eta_{\alpha\beta}(t) &= \begin{cases}\frac{\sigma_{zz}(t)-P_{\rm ext}\ V(t)}{p_{mass}}+\frac{2E_{kin}(t)}{f\ p_{mass}}-\chi(t)\eta_{zz}(t) & : \ (\alpha=\beta)=z \\ 0\ \ ;\ \ \eta_{\alpha\beta}(0)=0 & : \ (\alpha,\beta)\neq z\end{cases} \\
\frac{d}{dt}\chi(t) &= \frac{2E_{kin}(t)+p_{mass}\ \mathtt{Tr}[\underline{\underline{\eta}}(t)\cdot\underline{\underline{\eta}}(t)^T]-2\sigma-k_B\ T_{\rm ext}}{q_{mass}} \qquad (3.125)\\
\mathcal{H}_{\rm NP_nAT} &= \mathcal{H}_{\rm NVE}+\frac{q_{mass}\ \chi(t)^2}{2}+\frac{p_{mass}\ \mathtt{Tr}[\underline{\underline{\eta}}\cdot\underline{\underline{\eta}}^T]}{2}+P_{\rm ext}V(t)+(f+1)\ k_B\ T_{\rm ext}\int_o^t\chi(s)ds\ .
\end{aligned}$$

Similarly, this ensemble is optionally extending to constant normal pressure and constant surface tension, NP$_n\gamma$T [87], by semi-isotropic constraining of the barostat equation of motion and slight amending the thermostat equation of motion and the conserved quantity to:

$$\begin{aligned}
\frac{d}{dt}\eta_{\alpha\beta}(t) &= \begin{cases}\frac{\sigma_{\alpha\alpha}(t)-[P_{\rm ext}-\gamma_{\rm ext}/h_z(t)]\ V(t)}{p_{mass}}+\frac{2E_{kin}(t)}{f}\frac{1}{p_{mass}}-\chi(t)\eta_{\alpha\alpha}(t) & : \ (\alpha=\beta)=x,y \\ & : \\ \frac{\sigma_{zz}(t)-P_{\rm ext}\ V(t)}{p_{mass}}+\frac{2E_{kin}(t)}{f\ p_{mass}}-\chi(t)\eta_{zz}(t) & : \ (\alpha=\beta)=z \\ 0\ \ ;\ \ \eta_{\alpha\beta}(0)=0 & : \ (\alpha\neq\beta)=x,y,z\end{cases} \\
\frac{d}{dt}\chi(t) &= \frac{2E_{kin}(t)+p_{mass}\ \mathtt{Tr}[\underline{\underline{\eta}}(t)\cdot\underline{\underline{\eta}}(t)^T]-2\sigma-3\ k_B\ T_{\rm ext}}{q_{mass}} \qquad (3.126)\\
\mathcal{H}_{\rm NP_n\gamma T} &= \mathcal{H}_{\rm NVE}+\frac{q_{mass}\ \chi(t)^2}{2}+\frac{p_{mass}\ \mathtt{Tr}[\underline{\underline{\eta}}\cdot\underline{\underline{\eta}}^T]}{2}+P_{\rm ext}V(t)+(f+3)\ k_B\ T_{\rm ext}\int_o^t\chi(s)ds\ ,
\end{aligned}$$

where $\gamma_{\rm ext}$ is the user defined external surface tension and $h_z(t)=V(t)/A_{xy}(t)$ is the instantaneous hight of the MD box (or MD box volume over area). One defines the instantaneous surface tension as given in equation (3.89). The case $\gamma_{\rm ext}=0$ generates the NPT anisotropic ensemble for the orthorhombic cell (imcon=2 in CONFIG, see Appendix B). This can be considered as an "orthorhombic" constraint on the NσT ensemble. The constraint can be strengthened further, to a "semi-orthorhombic" one, by imposing that the MD cell change isotropically in the (x,y) plane which leads to the following changes in the equations above

$$\begin{aligned}
\frac{d}{dt}\eta_{\alpha\alpha}(t) &= \frac{[\sigma_{xx}(t)+\sigma_{yy}(t)]/2-[P_{\rm ext}-\gamma_{\rm ext}/h_z(t)]\ V(t)}{p_{mass}}+\frac{2\ E_{kin}(t)}{f\ p_{mass}}-\chi(t)\eta_{\alpha\alpha}(t)\ : \\
&\qquad : \ (\alpha=\beta)=x,y \qquad (3.127)\\
\mathcal{H}_{\rm NP_n\gamma=0T} &= \mathcal{H}_{\rm NVE}+\frac{q_{mass}\ \chi(t)^2}{2}+\frac{p_{mass}\ \mathtt{Tr}[\underline{\underline{\eta}}\cdot\underline{\underline{\eta}}^T]}{2}+P_{\rm ext}V(t)+(f+2)\ k_B\ T_{\rm ext}\int_o^t\chi(s)ds\ .
\end{aligned}$$

Although the Martyna-Tuckerman-Klein equations of motion have same conserved quantities as the Nosé-Hoover's ones they are proven to generate ensembles that conserve the phase space volume and thus have well defined conserved quantities even in presence of forces external to the system [95], which is not the case for Nosé-Hoover NPT and N$\underline{\underline{\sigma}}$T ensembles.

The NPT and N$\underline{\underline{\sigma}}$T versions of the MTK ensemble are implemented in the DL_POLY_4 routines NPT_M0_VV and NST_M0_VV. The corresponding routines incorporating RB dynamics are NPT_M1_VV, and NST_M1_VV.

3.6 Rigid Bodies and Rotational Integration Algorithms

3.6.1 Description of Rigid Body Units

A rigid body unit is a collection of point entities whose local geometry is time invariant. One way to enforce this in a simulation is to impose a sufficient number of bond constraints between the atoms in the unit. However, in many cases this may be either problematic or impossible. Examples in which it is impossible to specify sufficient bond constraints are

1. linear molecules with more than 2 atoms (e.g. CO_2)
2. planar molecules with more than three atoms (e.g. benzene).

Even when the structure *can* be defined by bond constraints the network of bonds produced may be problematic. Normally, they make the iterative SHAKE/RATTLE procedure slow, particularly if a ring of constraints is involved (as occurs when one defines water as a constrained triangle). It is also possible, inadvertently, to over constrain a molecule (e.g. by defining a methane tetrahedron to have 10 rather than 9 bond constraints) in which case the SHAKE/RATTLE procedure will become unstable. In addition, massless sites (e.g. charge sites) cannot be included in a simple constraint approach making modelling with potentials such as TIP4P water impossible.

All these problems may be circumvented by defining rigid body units, the dynamics of which may be described in terms of the translational motion of the centre of mass (COM) and rotation about the COM. To do this we need to define the appropriate variables describing the position, orientation and inertia of a rigid body, and the rigid body equations of motion[*].

The mass of a rigid unit M is the sum of the atomic masses in that unit:

$$M = \sum_{j=1}^{N_{sites}} m_j \quad , \tag{3.128}$$

where m_j is the mass of an atom and the sum includes all sites (N_{sites}) in the body. The position of the rigid unit is defined as the location of its centre of mass $\underline{R}$:

$$\underline{R} = \frac{1}{M} \sum_{j=1}^{N_{sites}} m_j \underline{r}_j \quad , \tag{3.129}$$

where $\underline{r}_j$ is the position vector of atom j. The rigid body translational velocity $\underline{V}$ is defined by:

$$\underline{V} = \frac{1}{M} \sum_{j=1}^{N_{sites}} m_j \underline{v}_j \tag{3.130}$$

and its angular momentum $\underline{J}$ can then be defined by the expression:

$$\underline{J} = \sum_{j=1}^{N_{sites}} m_j \left(\underline{d}_j \times \left[\underline{v}_j - \underline{V} \right] \right) \quad , \tag{3.131}$$

[*] An alternative approach is to define "basic" and "secondary" particles. The basic particles are the minimum number needed to define a local body axis system. The remaining particle positions are expressed in terms of the COM and the basic particles. Ordinary bond constraints can then be applied to the basic particles provided the forces and torques arising from the secondary particles are transferred to the basic particles in a physically meaningful way.

where $\underline{v}_j$ is the velocity of atom j and $\underline{d}_j$ is the displacement vector of the atom j from the COM, is given by:

$$\underline{d}_j = \underline{r}_j - \underline{R} \quad . \tag{3.132}$$

The net translational force $\underline{F}$ acting on the rigid body unit is the vector sum of the forces acting on the atoms of the body:

$$\underline{F} = \sum_{j=1}^{N_{sites}} \underline{f}_j \tag{3.133}$$

and the torque vector $\underline{\tau}$ acting on the body in the universal frame of reference is given by:

$$\underline{\tau} = \sum_{j=1}^{N_{sites}} \underline{d}_j \times \underline{f}_j \quad , \tag{3.134}$$

where $\underline{f}_j$ is the force on a rigid unit site.

A rigid body also has associated with it a rotational inertia matrix $\underline{\underline{\mathbf{I}}}$, whose components are given by:

$$I^{\alpha\beta} = \sum_{j=1}^{N_{sites}} m_j (d_j^2 \delta_{\alpha\beta} - d_j^\alpha r_j^\beta) \quad , \tag{3.135}$$

and COM stress and virial respectively written down as:

$$\sigma^{\alpha\beta} = \sum_{j=1}^{N_{sites}} d_j^\alpha f_j^\beta$$
$$\mathcal{W} = -\sum_{j=1}^{N_{sites}} \underline{d}_j \cdot \underline{f}_j \quad , \tag{3.136}$$

where $\underline{d}_j$ is the displacement vector of the atom j from the COM, and is given by:

$$\underline{d}_j = \underline{r}_j - \underline{R} \quad . \tag{3.137}$$

The rigid body angular velocity $\underline{\omega}$ is the right dot product of the inverse of the moment inertia, $\underline{\underline{\mathbf{I}}}$, and the angular momentum, $\underline{J}$,:

$$\underline{\omega} = \underline{\underline{\mathbf{I}}}^{-1} \cdot \underline{J} \quad . \tag{3.138}$$

It is common practice in the treatment of rigid body motion to define the position $\underline{R}$ of the body in a universal frame of reference (the so called laboratory or inertial frame), but to describe the moment of inertia tensor in a frame of reference that is localised in the rigid body and changes as the rigid body rotates. Thus the local body frame is taken to be that in which the rotational inertia tensor $\hat{\underline{\underline{\mathbf{I}}}}$ is diagonal and the components satisfy $I_{xx} \geq I_{yy} \geq I_{zz}$. In this local frame (the so called *Principal Frame*) the inertia tensor is therefore constant.

The orientation of the local body frame with respect to the space fixed frame is described via a four dimensional unit vector, the quaternion:

$$\underline{q} = [q_0, q_1, q_2, q_3]^T \quad , \tag{3.139}$$

and the rotational matrix $\underline{\underline{\mathbf{R}}}$ to transform from the local body frame to the space fixed frame is the unitary matrix:

$$\underline{\underline{\mathbf{R}}} = \begin{pmatrix} q_0^2 + q_1^2 - q_2^2 - q_3^2 & 2\ (q_1\ q_2 - q_0\ q_3) & 2\ (q_1\ q_3 + q_0\ q_2) \\ 2\ (q_1\ q_2 + q_0\ q_3) & q_0^2 - q_1^2 + q_2^2 - q_3^2 & 2\ (q_2\ q_3 - q_0\ q_1) \\ 2\ (q_1\ q_3 - q_0\ q_2) & 2\ (q_2\ q_3 + q_0\ q_1) & q_0^2 - q_1^2 - q_2^2 + q_3^2 \end{pmatrix} \tag{3.140}$$

so that if $\hat{\underline{d}}_j$ is the position of an atom in the local body frame (with respect to its COM), its position in the universal frame (w.r.t. its COM) is given by:

$$\underline{d}_j = \underline{\underline{\mathbf{R}}} \cdot \hat{\underline{d}}_j \quad . \tag{3.141}$$

With these variables defined we can now consider the equations of motion for the rigid body unit.

3.6.2 Integration of the Rigid Body Equations of Motion

The equations of translational motion of a rigid body are the same as those describing the motion of a single atom, except that the force is the total force acting on the rigid body i.e. $\underline{F}$ in equation (3.133) and the mass is the total mass of the rigid body unit i.e. M in equation (3.128). These equations can be integrated by the standard Verlet VV algorithm described in the previous sections. Thus we need only consider the rotational motion here.

The rotational equation of motion for a rigid body relates the torque to the change in angular momentum:

$$\underline{\tau} = \frac{d}{dt}\underline{J} = \frac{d}{dt}\left(\underline{\underline{\mathbf{I}}} \cdot \underline{\omega}\right) \quad . \tag{3.142}$$

In a thermostat it can be written as:

$$\underline{\dot{J}}_i = \underline{\tau}_i - \underline{\omega}_i \times \underline{J}_i + \frac{\chi}{q_{mass}}\underline{J}_i \quad , \tag{3.143}$$

where i is the index of the rigid body, χ and q_{mass} are the thermostat friction coefficient and mass*. In the local frame of the rigid body and without the thermostat term, these simplify to the Euler's equations

$$\begin{aligned} \dot{\hat{\omega}}_x &= \frac{\hat{\tau}_x}{\hat{I}_{xx}} + (\hat{I}_{yy} - \hat{I}_{zz})\;\hat{\omega}_y\;\hat{\omega}_z \\ \dot{\hat{\omega}}_y &= \frac{\hat{\tau}_y}{\hat{I}_{yy}} + (\hat{I}_{zz} - \hat{I}_{xx})\;\hat{\omega}_z\;\hat{\omega}_z \\ \dot{\hat{\omega}}_z &= \frac{\hat{\tau}_z}{\hat{I}_{zz}} + (\hat{I}_{xx} - \hat{I}_{yy})\;\hat{\omega}_x\;\hat{\omega}_y \quad . \end{aligned} \tag{3.144}$$

The vectors $\underline{\hat{\tau}}$ and $\underline{\hat{\omega}}$ are the torque and angular velocity acting on the body transformed to the local body frame. Integration of $\underline{\hat{\omega}}$ is complicated by the fact that as the rigid body rotates, so does the local reference frame. So it is necessary to integrate equations (3.144) simultaneously with an integration of the quaternions describing the orientation of the rigid body. The equation describing this is:

$$\begin{pmatrix} \dot{q}_0 \\ \dot{q}_1 \\ \dot{q}_2 \\ \dot{q}_3 \end{pmatrix} = \frac{1}{2}\begin{pmatrix} q_0 & -q_1 & -q_2 & -q_3 \\ q_1 & q_0 & -q_3 & q_2 \\ q_2 & q_3 & q_0 & -q_1 \\ q_3 & -q_2 & q_1 & q_0 \end{pmatrix}\begin{pmatrix} 0 \\ \hat{\omega}_x \\ \hat{\omega}_y \\ \hat{\omega}_z \end{pmatrix} \quad . \tag{3.145}$$

Rotational motion in DL_POLY_4 is handled by two different methods. For the LFV implementation, the Fincham Implicit Quaternion Algorithm (FIQA) is used [96]. The VV implementation uses the NOSQUISH algorithm of Miller *et al.* [24]. The implementation NOSQUSH is coded in NO_SQUISH both contained within QUATERNION_CONTAINER.

The LFV implementation begins by integrating the angular velocity equation in the local frame:

$$\underline{\hat{\omega}}(t + \frac{\Delta t}{2}) = \underline{\hat{\omega}}(t - \frac{\Delta t}{2}) + \Delta t\;\underline{\underline{\hat{\mathbf{I}}}}^{-1} \cdot \underline{\dot{\hat{\omega}}}(t) \quad . \tag{3.146}$$

The new quaternions are found using the FIQA algorithm. In this algorithm the new quaternions are found by solving the implicit equation:

$$\underline{q}(t + \Delta t) = \underline{q}(t) + \frac{\Delta t}{2}\left(\underline{\underline{\mathbf{Q}}}\;[\underline{q}(t)] \cdot \underline{\hat{w}}(t) + \underline{\underline{\mathbf{Q}}}\;[\underline{q}(t + \Delta t)] \cdot \underline{\hat{w}}(t + \Delta t)\right) \quad , \tag{3.147}$$

* It is worth noting that in DL_POLY_4 all degrees of freedom, translational (both the free particles' ones and the RBs' COMs ones) and rotational, are considered equal and thus treated in the same manner in all available thermostats!

where $\underline{\hat{w}} = [0, \underline{\hat{\omega}}]^T$ and $\underline{\underline{\mathbf{Q}}}[\underline{q}]$ is:

$$\underline{\underline{\mathbf{Q}}} = \frac{1}{2} \begin{pmatrix} q_0 & -q_1 & -q_2 & -q_3 \\ q_1 & q_0 & -q_3 & q_2 \\ q_2 & q_3 & q_0 & -q_1 \\ q_3 & -q_2 & q_1 & q_0 \end{pmatrix} \quad . \tag{3.148}$$

The above equation is solved iteratively with

$$\underline{q}(t + \Delta t) = \underline{q}(t) + \Delta t \; \underline{\underline{\mathbf{Q}}}[\underline{q}(t)] \cdot \underline{\hat{w}}(t) \tag{3.149}$$

as the first guess. Typically, no more than 3 or 4 iterations are needed for convergence. At each step the normalisation constraint:

$$\|\underline{q}(t + \Delta t)\| = 1 \tag{3.150}$$

is imposed.

While all the above is enough to build LFV implementations, the VV implementations, based on the NOSQUISH algorithm of Miller *et al.* [24], also require treatment of the quaternion momenta as defined by:

$$\begin{pmatrix} p_0 \\ p_1 \\ p_2 \\ p_3 \end{pmatrix} = 2 \begin{pmatrix} q_0 & -q_1 & -q_2 & -q_3 \\ q_1 & q_0 & -q_3 & q_2 \\ q_2 & q_3 & q_0 & -q_1 \\ q_3 & -q_2 & q_1 & q_0 \end{pmatrix} \begin{pmatrix} 0 \\ \hat{I}_{xx} \; \hat{\omega}_x \\ \hat{I}_{yy} \; \hat{\omega}_y \\ \hat{I}_{zz} \; \hat{\omega}_z \end{pmatrix} \quad , \tag{3.151}$$

and quaternion torques as defined by:

$$\begin{pmatrix} \Upsilon_0 \\ \Upsilon_1 \\ \Upsilon_2 \\ \Upsilon_3 \end{pmatrix} = 2 \begin{pmatrix} q_0 & -q_1 & -q_2 & -q_3 \\ q_1 & q_0 & -q_3 & q_2 \\ q_2 & q_3 & q_0 & -q_1 \\ q_3 & -q_2 & q_1 & q_0 \end{pmatrix} \begin{pmatrix} 0 \\ \hat{\tau}_x \\ \hat{\tau}_y \\ \hat{\tau}_z \end{pmatrix} \quad . \tag{3.152}$$

It should be noted that vectors $\underline{p}$ and $\underline{\Upsilon}$ are 4-component vectors. The quaternion momenta are first updated a half-step using the formula:

$$\underline{p}(t + \frac{\Delta t}{2}) \leftarrow \underline{p}(t) + \frac{\Delta t}{2} \underline{\Upsilon}(t) \quad . \tag{3.153}$$

Next a sequence of operations is applied to the quaternions and the quaternion momenta in the order:

$$e^{i\mathcal{L}_3(\delta t/2)} \; e^{i\mathcal{L}_2(\delta t/2)} \; e^{i\mathcal{L}_1(\delta t)} \; e^{i\mathcal{L}_2(\delta t/2)} \; e^{i\mathcal{L}_3(\delta t/2)} \quad , \tag{3.154}$$

which preserves the symplecticness of the operations (see reference [31]). Note that δt is some submultiple of Δt. (In DL_POLY_4 the default is $\Delta t = 10\delta t$.) The operators themselves are of the following kind:

$$\begin{aligned} e^{i\mathcal{L}(\delta t)} \; \underline{q} &= \cos(\zeta_k \delta t) \; \underline{q} + \sin(\zeta_k \delta t) \; P_k \; \underline{q} \\ e^{i\mathcal{L}(\delta t)} \; \underline{p} &= \cos(\zeta_k \delta t) \; \underline{p} + \sin(\zeta_k \delta t) \; P_k \; \underline{p} \quad , \end{aligned} \tag{3.155}$$

where P_k is a permutation operator with $k = 0, \ldots, 3$ with the following properties:

$$\begin{aligned} P_0 \; \underline{q} &= \{ \; q_0, \; q_1, \; q_2, \; q_3\} \\ P_1 \; \underline{q} &= \{-q_1, \; q_0, \; q_3, -q_2\} \\ P_2 \; \underline{q} &= \{-q_2, -q_3, \; q_0, \; q_1\} \\ P_3 \; \underline{q} &= \{-q_3, \; q_2, -q_1, \; q_0\} \quad , \end{aligned} \tag{3.156}$$

and the angular velocity ζ_k is defined as:

$$\zeta_k = \frac{1}{4 I_k} \underline{p}^T P_k \; \underline{q} \quad . \tag{3.157}$$

Equations (3.154-3.156) represent the heart of the NOSQUISH algorithm and are repeatedly applied (10 times in DL_POLY_4). The final result is the quaternion updated to the full timestep value i.e. $\underline{q}(t+\Delta t)$. These equations form part of the first stage of the VV algorithm (VV1).

In the second stage of the VV algorithm (VV2), new torques are used to update the quaternion momenta to a full timestep:

$$\underline{p}(t+\Delta t) \leftarrow \underline{p}(t+\frac{\Delta t}{2}) + \frac{\Delta t}{2}\underline{\Upsilon}(t+\Delta t) \quad . \tag{3.158}$$

3.6.3 Thermostats and Barostats coupling to the Rigid Body Equations of Motion

In the presence of rigid bodies in the atomic system the system's instantaneous pressure, equation (3.69):

$$\mathcal{P}(t) = \frac{[2E_{kin}(t) - \mathcal{W}_{\text{atomic}}(t) - \mathcal{W}_{\text{COM}}(t) - \mathcal{W}_{\text{constrain}}(t-\Delta t) - \mathcal{W}_{\text{PMF}}(t-\Delta t)]}{3V(t)} \tag{3.159}$$

and stress, equation (3.70):

$$\underline{\underline{\sigma}}(t) = \underline{\underline{\sigma}}_{kin}(t) + \underline{\underline{\sigma}}_{\text{atomic}}(t) + \underline{\underline{\sigma}}_{\text{COM}}(t) + \underline{\underline{\sigma}}_{\text{constrain}}(t-\Delta t) + \underline{\underline{\sigma}}_{\text{PMF}}(t-\Delta t) \tag{3.160}$$

are augmented to include the RBs' COM virial and stress contributions. **Note** that the kinetic energy and stress in the above also include the contributions of the RBs' COMs kinetic energy and stress!

In DL_POLY_4 all degrees of freedom, translational and rotational, are considered equal and thus treated in the same manner in all available thermostats. Similarly, in the same spirit of equi-partitioning, all translational degrees of freedom, the free particles' ones and the RBs' COMs ones are considered equal and thus treated in the same manner in all available barostats. Based on these considerations, it is straightforward to couple the rigid body equations of motion to a thermostat and/or barostat. The thermostat is coupled to both the translational and rotational degrees of freedom and thus the translational and rotational velocities (momenta) are thermostated in the same operational manner as the purely atomic ones. The barostat, however, is coupled only to the translational degrees of freedom and does not contribute to the rotational motion of the RBs, thus only the RBs' COMs positions and momenta are subjected to the same barostat driven algorithmic operations as those of the free particles' positions and momenta. Therefore, if we notion the change of the system's degrees of freedom as:

$$f \rightarrow F = f + f^{RB(\mathbf{tra})} + f^{RB(\mathbf{rot})} \tag{3.161}$$

then all equations of motion defining the ensembles as described in this chapter are subject to the following notional changes in order to include the RB contributions:

$$\begin{array}{rcccl} \sigma(f) & \rightarrow & \sigma(F) & = & \sigma(f + f^{RB(\mathbf{tra})} + f^{RB(\mathbf{rot})}) \\ \mathcal{H}(f) & \rightarrow & \mathcal{H}(F) & = & \mathcal{H}(f + f^{RB(\mathbf{tra})} + f^{RB(\mathbf{rot})}) \\ p_{mass}(f) & \rightarrow & p_{mass}(F - f^{RB(\mathbf{rot})}) & = & p_{mass}(f + f^{RB(\mathbf{tra})}) \\ \eta(f) & \rightarrow & \eta(F - f^{RB(\mathbf{rot})}) & = & \eta(f + f^{RB(\mathbf{tra})}) \\ \underline{\underline{\eta}}(f) & \rightarrow & \underline{\underline{\eta}}(F - f^{RB(\mathbf{rot})}) & = & \underline{\underline{\eta}}(f + f^{RB(\mathbf{tra})}) \quad , \end{array} \tag{3.162}$$

where f refers to the degrees of freedom in the system (see equation (3.7)), σ is the system target energy (see equation (3.41)), $\mathcal{H}$ is the conserved quantity of the ensemble (if there is such defined), E_{kin} (!includes RB COM kinetic energy too) and E_{rot} are respectively the kinetic and rotational energies of the system, p_{mass} is the barostat mass, and η and $\underline{\underline{\eta}}$ are the barostat friction coefficient or matrix of coefficients respectively.

There are two slight technicalities with the Evans and Andersen ensembles that are worth mentioning.

Since both the translational and rotational velocities contribute towards temperature, equation (3.20), showing the derivation of the thermostat friction in the Evans ensemble by imposing a Gaussian constraint on the system's instantaneous temperature, changes to:

$$\begin{aligned}
\frac{d}{dt}\mathcal{T} = 0 \qquad \propto \qquad & \frac{d}{dt}\left(\frac{1}{2}\sum_{i}^{FP} m_i \underline{v}_i^2 + \frac{1}{2}\sum_{j}^{RB} M_j \underline{V}_j^2 + \frac{1}{2}\sum_{j}^{RB} \underline{\hat{\omega}}_j^T \cdot \underline{\underline{\hat{\mathbf{I}}}}_j \cdot \underline{\hat{\omega}}_j\right) = 0 \\
\left\{\sum_{i}^{FP} \underline{v}_i(t) \cdot \underline{f}_i(t) + \sum_{j}^{RB} \underline{V}_j(t) \cdot \underline{F}_j(t) + \sum_{j}^{RB} \underline{\hat{\omega}}_j(t) \cdot \underline{\hat{\tau}}(t)\right\} - & \\
\chi(t)\left\{\sum_{i}^{FP} m_i \underline{v}_i^2(t) + \sum_{j}^{RB} M_j \underline{V}_j^2(t) + \sum_{j}^{RB} \underline{\hat{\omega}}_j^T(t) \cdot \underline{\underline{\hat{\mathbf{I}}}}_j \cdot \underline{\hat{\omega}}_j(t)\right\} &= 0 \\
\chi(t) = \frac{\left\{\sum_i^{FP} \underline{v}_i(t) \cdot \underline{f}_i(t) + \sum_j^{RB} \underline{V}_j(t) \cdot \underline{F}_j(t) + \sum_j^{RB} \underline{\hat{\omega}}_j(t) \cdot \underline{\hat{\tau}}(t)\right\}}{2\left[E_{kin}(t) + E_{rot}(t)\right]} & \quad ,
\end{aligned} \tag{3.163}$$

where where $\mathcal{T}$ is the instantaneous temperature defined in equation (3.6) and E_{kin} in the final expression contains both the kinetic contribution form the free particles and the RBs' COMs.

In the case of the Andersen ensemble, if a Poisson selected particle constitutes a RB then the whole RB is Poisson selected. Poisson selected RBs' translational and angular velocities together with Poisson selected FPs' velocities sample the same Gaussian distribution isokinetically (Boltzmann distribution), where the isokineticity to target temperature is dependent upon the total of the Poisson selected FPs' and RBs' degrees of freedom.

Chapter 4

Coarse Graining Functionality

Scope of Chapter

This chapter describes the coarse-graining functionality available in DL_POLY_4.

4.1 User-Defined Coarse-Grain Models with Tabulated Force-Fields

One can use DL_POLY_4 for preparing and running simulations of (numerically) coarse-grained (CG) models by using tabulated effective force-fields (FF) derived from either **(i)** potentials of mean force (PMF); or **(ii)** iteratively optimised CG models.

In outline, systematic coarse-graining (SCG) of an atomistic system implies the application of a geometrical projection, or "mapping", of the original system onto a considerably reduced set of degrees of freedom (DoF), whence referred to as a "coarse-grained" model. The procedure is to recover the average configurational (often termed "physical") and topological (often termed "chemical") force-field related properties of the original full-atom (FA) model. Integrating out degrees of freedom ultimately leads to loss of information. However, if this is done selectively and consistently, the intrinsic information pertaining to the dominating interactions within the FA system (that govern its behaviour towards phenomena of interest) and the most important thermodynamic properties are retained. Then the greatly reduced phase-space of the CG model system allows for efficient access to much longer length and time scales for investigating microscopic phenomena using of the well-established classical MD machinery. The reduced DoF mapping leads to the generation of an effective CG FF in terms of the effective interaction potentials and forces between the CG particles, which are often tabulated numerically.

The initial coarse-grain mapping of the original FA trajectory can be done with the aid of DL_CGMAP tool – http://www.ccp5.ac.uk/software/. After the CG mapped trajectory has been obtained, the relevant distribution analysis and the Boltzmann Inversion procedure (producing tabulated PMF:s) can be performed by using DL_POLY_4 in "replay history" mode, with the corresponding directives in CONTROL file. For further iterative optimisation of the CG model the user is advised to use DL_POLY_4 as simulation engine within the framework of VOTCA package – http://www.votca.org/, which provides a handful of various systematic coarse-graining methodologies. For more details the user should refer to the manuals of these tools.

This section describes how to use DL_POLY_4 for two SCG tasks:

- Post-simulation analysis of the intramolecular (bonded and angular) mean-force interactions, based on the calculation of the corresponding intramolecular distributions.
- Preparation and use of the tabulated intramolecular potentials.

Note: Although these steps are also applicable to atomistic systems, which can be useful for benchmarking purposes, below we shall assume that the CG mapping has been done and the following data files have been generated for the CG mapped model system: CONTROL, FIELD, CONFIG, and, alternatively, HISTORY.

4.2 Intramolecular Probability Distribution Function (PDF) Analysis

Albeit the distribution analysis can be performed on the fly, in the course of a simulation run, for CG purposes it has to be done on an existing CG mapped trajectory, by invoking the **replay history** and **analysis** directives, see Section 10.1.1. To trigger the PDF collection and subsequent output of PMF:s for any of the following intramolecular interactions: bonds, angles, dihedrals and/or inversions, the CONTROL file must contain any combination of the options demonstrated in the example below. **Note** that such analyses will only be carried out if the desired intramolecular types of interactions are defined within FIELD!

```
TITLE: EXAMPLE OF DL_POLY_4 PDF ANALYSIS DIRECTIVES SNIPPET

# DIRECTIVES TO INVOKE INTRAMOLECULAR PDF ANALYSIS BY TYPE
analyse  bonds      sample every 100  nbins 250  rmax 5.0
analyse  angles     sample every 100  nbins 360  # [ 0 : pi]
analyse  dihedrals  sample every 100  nbins 720  # [-pi: pi]
analyse  inversions  sample every 100  nbins 360  # [ 0 : pi]

# DIRECTIVES TO INVOKE INTRAMOLECULAR PDF ANALYSIS FOR ALL TYPES
analyse  all         sample every 100  nbins 1000  rmax 5.0

# DIRECTIVES TO INVOKE PRINTING FOR ANY DEFINED
# INTER-(RDF->VDW) & INTRA-(bonded) MOLECULAR PDF ANALYSIS
print analysis
```

The **analyse** directive acts in a similar manner to that of the **rdf** directive outlining **(i)** the frequency of sampling (in steps/frames) and **(ii)** the number of bins over the cutoff interval of the specified interaction. It is only for bonded pairs that this cutoff needs specifying, whereas for angles the possible ranges are known *a priory* by definition. If no cutoff is supplied for bonds, then it defaults to 2 Å. It is worth noting that, for the sake of accuracy, the number of bins per PDF must be larger than or equal to $N_{\rm min} = \texttt{Nint}\left(\frac{\rm PDF\ cutoff}{\Delta_{\rm max}}\right)$, where the max bin size, $\Delta_{\rm max}$, is defined internally for each type of PDF (see `setup_module.f90`). Otherwise, it will default to $\texttt{max}(N_{\rm min}, N_{\rm tab})$, where $N_{\rm tab}$ is the number of bins on the grid defined in the corresponding tabulated force-field data file (TAB*), if it is provided, otherwise $N_{\rm tab} = 0$. In particular, for bonds $\Delta_{\rm max}$ is 0.01 Å and for angles it is $0.2\pi/180$, i.e. 0.2 degree.

If **analyse all** option is used in conjunction with any specific directive, then it triggers the analysis on all PDFs while enforcing on the individually targeted PDF(s) the following parameters: **(i)** its sampling interval (frames) – only if it is smaller than, and **(ii)** its grid number – only if it is larger than those specified for the targeted PDFs. In the case of bonds it will also enforce the grid range **rmax** – only if it is larger than that specified in the **analyse bonds** directive. Hence, the **analyse all** directive allows to quickly override and/or unify the sampling frequencies and max grid numbers for all PDFs, provided its parameters improve on the accuracy of the collected data (compared to the specifications for the individually targeted PDF:s).

While the statistics are always collected and stored for the future use in the (binary) REVIVE file for all targeted PDF:s (see Section 10.2.10), using the **print analysis** directive will instruct DL_POLY_4 to additionally print the data in the OUTPUT and *DAT files, the latter containing each *type* of PDF:s separately (the asterisk stands for one of the following: BND, ANG, DIH, or INV). As a result, apart from OUTPUT, three data files will be created for each of the targeted distribution types: BNDDAT, BNDPMF & BNDTAB – for bonds, ANGDAT, ANGPMF & ANGTAB – for angles, DIHDAT, DIHPMF & DIHTAB – for dihedrals, and INVDAT, INVPMF & INVTAB – for inversions; the *PMF and *TAB files containing tabulated data for the respective potential of mean force and force/virial (*TAB files bearing the data from *PMF files but *resampled onto a finer grid*; see the last paragraph in this section).

Partial examples of the *DAT files for bonds and angles are given below.

```
[user@host]$ more BNDDAT
# TITLE: Hexane FA OPLSAA -> CG mapped with 3 beads (A-B-A)
# BONDS: Probability Density Functions (PDF) := histogram(bin)/hist_sum(bins)/dr_bin
# bins, cutoff, frames, types:        250    5.000       2285          1
#
# r(Angstroms)  PDF_norm(r)  PDF_norm(r)/dVol(r)   @   dr_bin =  0.02000
#

# type, index, instances: A        B                  1       2000
    0.01000  0.000000E+00  0.000000E+00
    0.03000  0.000000E+00  0.000000E+00
    0.05000  0.000000E+00  0.000000E+00
...
    4.95000  0.000000E+00  0.000000E+00
    4.97000  0.000000E+00  0.000000E+00
    4.99000  0.000000E+00  0.000000E+00
```

```
[user@host]$ more ANGDAT
# TITLE: Hexane FA OPLSAA -> CG mapped with 3 beads (A-B-A)
# ANGLES: Probability Density Functions (PDF) := histogram(bin)/hist_sum(bins)/dTheta_bin
# bins, cutoff, frames, types:       360       180       2285          1
#
# Theta(degrees)  PDF_norm(Theta)  PDF_norm(Theta)/sin(Theta)   @   dTheta_bin =  0.50000
#

# type, index, instances: A        B        A                 1       1000
    0.25000  0.000000E+00  0.000000E+00
    0.75000  0.000000E+00  0.000000E+00
    1.25000  0.000000E+00  0.000000E+00
...
  178.75000  1.380569E-02  6.328564E-01
  179.25000  8.368490E-03  6.393238E-01
  179.75000  2.901532E-03  6.649842E-01
```

One can see that all the header lines are commented out due to starting with the hash symbol, #. Nonetheless, the header contains some useful information. The title is, as usual, placed in the first line, which is followed by an explanatory line with the definition of a normalised PDF. The third line provides the four most important descriptors: the number of *bins* on the histogram grid, the *cutoff* interval (absolute value of the span) over which the distributions are sampled, the number of *frames* (samples) used, and the number of unique unit *types* analysed (where "unit" is one of the following: bonds, angles, dihedrals or inversions). The last explanatory line in the header, found in between two empty commented-out lines, defines the meaning of the columns and, at the end, the grid bin size.

The PDF histograms within a *DAT file are separated by uncommented empty lines, which makes it possible to directly import and plot all the data as separate lines in [Xm]Grace 2D plotter. For clarity, the data of each histogram are preceded by a commented-out line specifying the *type*, *index* and number of *instances* of the analysed interaction unit (the latter being counted over the entire system).

In all *DAT files the first column bears the *bin-centered* abscissa values (distance or angle), the second column is the distribution histogram normalized to unity, i.e. its *integral* equals 1, whereas the third column, if any, contains the PDF data corrected for the volumetric (entropic) degeneracy of the grid points. Thus, it is the data of the last column found that are used for calculating PMF $\sim -\ln(\text{PDF})$.

The OUTPUT file will contain copies of the first and second columns of all collected PDF:s, but normalised so that the figures in the second column *sum up* to 1, which can be checked by examining the third column as it bears the running sum of a PDF histogram.

In addition to the PDF:s, DL_POLY_4 also calculates the respective PMF:s and pairwise force dependencies (virial for bonds), which are stored in the *PMF and, upon resampling onto a *bin-edge* grid, *TAB files. **It is important to note** that, unlike the *PMF files containing the bare $-\ln(\text{PDF})$ data (converted to the requested energy units) *on the same grid as the PDF histograms*, the force-field data in *TAB files are *resampled* onto $\max(N_{\text{min}}, N_{\text{tab}})$ grid points located at the bin edges (as expected by DL_POLY_4 when reading the potential and force tables), where N_{tab} is the grid number for the respective intramolecular unit type read-in from its TAB* file, if provided (otherwise $N_{\text{tab}} = 0$). Thus, the *TAB files obey the DL_POLY_4 format for numerically defined intramolecular force-field tables (TAB*, see below) and, hence, can be directly used as such upon renaming: BNDTAB → TABBND, ANGTAB → TABANG, DIHTAB → TABDIH and INVTAB → TABINV. **The user is, however, strongly advised to check the quality of the obtained tabulated force-fields before using those as input for a CG simulation.** Albeit DL_POLY_4 uses a simple smoothing algorithm for noise-reduction in PMF:s and implements capping of the forces in the regions of zero-valued PDF:s, in undersampled regions the PDF and PMF data are likely to suffer from inaccuracy and increased noise which, most often, require extra attention and re-fitting manually.

The general format of the above discussed files is shown in Section 10.2.16

4.3 Setting up Tabulated Intramolecular Force-Field Files

For a user-defined, e.g. coarse-grained, model system the effective potentials must be provided in a tabulated form. For non-bonded short-range (VdW) interactions the TABLE file must be prepared as described in Section 10.1.7. However, the tabulated data format for intramolecular interactions (bonds, bending angles, dihedral and inversion angles in a polymer) differs from that of the TABLE file and assumes three columns: abscissa (distance in Å or angle in degrees), and two ordinates: potential and force data (virial for distance dependent interactions – e.g. bonds, and force for angle dependent interactions – e.g. angles). Shown below are examples of TABBND and TABANG files, corresponding to the above PDF examples. **Note** that the PMF and force data have been resampled onto a finer grid with points located *at bin edges*.

```
[user@host]$ more TABBND
# TITLE: Hexane FA OPLSAA -> CG mapped with 3 beads (A-B-A)
# 5.0 500

# A B
 1.00000e-02  -9.0906600e+02   1.3954000e+00
 2.00000e-02  -9.1046200e+02   2.7908000e+00
 3.00000e-02  -9.1185700e+02   4.1862000e+00
...

[user@host]$ more TABANG
# TITLE: Hexane FA OPLSAA -> CG mapped with 3 beads (A-B-A)
# 1000

# A B A
 1.80000e-01   8.8720627e+01   6.9119576e-01
 3.60000e-01   8.8596227e+01   6.9119227e-01
 5.40000e-01   8.8471827e+01   6.9118704e-01
...
```

The input tables for bonds, angles, dihedrals and inversions are named TABBND, TABANG, TABDIH and TABINV, correspondingly. The format of these files is fixed in terms of the line, or *record*, order. In particular, the initial two header lines must contain a title and a record with the grid specification, and each of the following blocks of tabulated data must be preceded by an empty line and a one-line descriptor record containing the (white-space delimited) names of the atoms making up the given intermolecular interaction unit (same as a *unit type* in *DAT files). These descriptor lines can be commented out or not, i.e. having # as the first symbol would not affect the reading operation, but it would ease importing of the data for plotting and manipulating in [Xm]Grace software.

In all TAB* files the number of grid points (bins) must be specified on the second line (commented-out or not). For angles (TABANG, TABDIH, TABINV) no other information needs to be provided as their ranges are pre-determined: $0 < \Theta$ in TABANG & TABINV ≤ 180 and $-180 < \Theta$ in TABDIH ≤ 180. In the TABBND file, however, the "bond cutoff", $r_{\max}$ (Å), must be precede the grid number.

Note that all potential and force data are to be provided in the same energy units as specified by the user in the FIELD file (see Section 10.1.3) with distances in Å and angles in degrees. All the data related to angles are internally transformed and handled by DL_POLY_4 with angles measured in *radians*.

Finally, in order to instruct DL_POLY_4 to use tabulated intramolecular force-fields read from the TAB* files the user has to specify in the FIELD file the keyword **tab** or **-tab** for each intramolecular interaction thereby chosen for tabulation (similarly to how it is described in Section 10.1.3). The dash symbol (-) in front of the keyword **tab** is only valid for bonds and angles, and is interpreted in the same manner as in Table 10.5 and Table 10.6.

Note that VOTCA package is also capable of collecting both intra- and inter-molecular stats and producing correct TAB* files, provided the FIELD and HISTORY files exist (albeit VOTCA saves the distributions in a format different from *DAT files).

Below we summarise the sequence of operations the user has to follow in order to perform the CG distribution analysis and prepare the TAB* files for a newly coarse-grained system.

- Perform CG mapping of the original FA system with the aid of DL_CGMAP or VOTCA (in the case of using VOTCA follow its manual; the remainder of the list describes using DL_POLY_4 only);
- Move the data files for the newly CG-mapped system (FIELD_CG, CONFIG_CG, HISTORY_CG) into a separate directory under the standard names (FIELD, CONFIG, HISTORY) and create the corresponding CONTROL file containing the **analysis** and **replay history** directives.
- As no TAB* files are yet available for the CG system at this stage, one can either **(i)** create the initial TAB* files padded with zeros, or **(ii)** use a FIELD file with fictitious records for all the interactions to be tabulated, with the interaction keywords and parameters chosen arbitrarily in accord with Table 10.5, Table 10.6, Table 10.7 and Table 10.8. **Note** that DL_CGMAP (as well as VOTCA) creates FIELD_CG files that already contain interaction descriptors with the **tab** keyword(s) in place, so if the route *(ii)* is chosen, the user needs to replace those records with the fictitious ones (it is advisory to store the initial FIELD_CG for the future use, when the actual TAB* files are ready).
- Run DL_POLY_4 with the **replay history**, **rdf** and/or **analysis** options invoked in the CONTROL file, which will result in creation of the targeted inter- and intra-molecular PDF data files (RDFDAT, BNDDAT, ANGDAT, DIHDAT, INVDAT) and the respective PMF files as described above. **Note** that only and only when the **rdf** and **analysis** options are both active then the VDWPMF and VDWTAB files (derived from RDF:s) will be produced, along with RDFDAT. They are structured in the same manner and format as their intramolecular counterparts. The user can then convert the VDWTAB file into a correctly formatted TABLE file by using the utility called `pmf2tab.f` (subject to compilation; found in DL_POLY_4 directory `utility`) as follows.

  ```
  [user@host]$ pmf2tab.exe < VDWTAB
  ```

- Check the data for accuracy and amend the tabulated force-fields. Redo the analysis on coarser/finer grid(s), if necessary.
- When satisfied with the created TAB* files, run a DL_POLY_4 simulation for the prepared CG (or simply user-defined) model system.

Chapter 5

Two-Temperature Model

Scope of Chapter

This chapter describes the two-temperature model functionality available in DL_POLY_4.

5.1 Introduction

Traditionally, the modelling of radiation damage has been confined to cases in which the radiation (i.e. a projectile) collides elastically with the nuclei of the target material. Collisions of this type are said to be dominated by nuclear stopping (energy transfer from the projectile to the nuclei). However, there are a broad range of scenarios (high energy collision cascades, swift heavy ion irradiation, and laser excitation) in which a significant portion (high energy collision cascades) or all (lasers and swift heavy ions) of the energy is transferred to the electrons in the target material. These cases, in which electronic stopping cannot be neglected, are impossible to account for using traditional molecular dynamics simulations (which assume equilibrium conditions between the target nuclei and electrons). This chapter describes the implementation of a hybrid continuum-atomistic implementation within DL_POLY_4 that incorporates these electronic excitations.

The model is based on the traditional two-temperature model (TTM) [97]. This model splits the nuclei and electrons into two separate but interacting subsystems, each evolving according to a modified version of Fourier's heat equation. The continuum implementation of the nuclei (assumed to be in a lattice) in this form of the model is unable to track individual atomistic trajectories, thus information on superheating, recrystallisation, and pressure waves is lost. These limitations were overcome by Duffy and Rutherford [85, 98], by replacing the continuum representation of the nuclei with an MD cell (the details of which will be described in this chapter). This hybrid continuum-atomistic model, a combination of TTM and MD known as the two-temperature molecular dynamics (2T-MD) model, has been successfully used to model high energy cascades in iron [99], ultrafast laser irradiation of gold nanofilms [100], and swift heavy ion irradiation of silicon [101].

5.2 Methodology

Electronic subsystem

The electronic temperature is treated as a continuum system, and evolves according to Fourier's law of heat conduction:

$$C_e(T_e)\frac{\partial T_e}{\partial t} - \nabla.[\kappa_e \nabla T_e] = -G_{ep}(T_e - T_a) + G_s T'_a + A(r,t), \tag{5.1}$$

where $C_e(T_e)$ is the electronic volumetric heat capacity (equal to the product of specific heat capacity and density), κ_e is the electronic thermal conductivity, $G_{ep}(T_e)$ the electron-phonon coupling, G_s the electronic stopping term (which is significant for atomistic/ionic temperatures T'_a greater than a velocity cutoff, as defined later), and $A(r,t)$ the temporal and spatial dependent source term. This equation describes how energy evolves in the electronic system as follows: energy is dumped into the electronic system via the source term (for swift heavy ions and laser excitation), $A(r,t)$, the electronic volume-specific heat determines the electronic temperature (T_e) rise due to this deposition, the thermal conductivity describes how energy dissipates throughout the electronic subsystem, and the electron-phonon coupling determines energy transfer from the electrons to the MD cell (and is proportional to the temperature difference between T_e and the atomistic temperature, T_a). Equation (5.1) can be equated to the more general heat diffusion equation,

$$\frac{\partial T}{\partial t} - \alpha \nabla^2 T = \frac{\dot{q}}{C}, \tag{5.2}$$

T is temperature, t is time, $\dot{q}$ is a heat source or sink, C is heat capacity, and α is thermal diffusivity. This partial differential equation is solved using Euler's method, which is a space-centred, forward-in-time integration algorithm. For a temperature T_i^n (at time step n and grid point i), the forward-in-time implementation can be Taylor-expanded and rearranged to:

$$\left(\frac{\partial T}{\partial t}\right)_i = \frac{T_i^{n+1} - T_i^n}{\Delta t} - \frac{\Delta t}{2}\left(\frac{\partial^2 T}{\partial t^2}\right)_i - \frac{\Delta t^2}{6}\left(\frac{\partial^3 T}{\partial t^3}\right)_i - \cdots \approx \frac{T_i^{n+1} - T_i^n}{\Delta t}, \tag{5.3}$$

for equally-spaced timesteps Δt, and leads to a truncation error $O(\Delta t)$. The one-dimensional space-centred integration is

$$\left(\frac{\partial T}{\partial x}\right)_n = \frac{T_{i+1}^n - T_{i-1}^n}{\Delta x} - \frac{\Delta x^2}{6}\left(\frac{\partial^3 T}{\partial x^3}\right)_i - ... \approx \frac{T_{i+1}^n - T_{i-1}^n}{\Delta x} \tag{5.4}$$

for equally spaced grid lengths Δx, and leads to a truncation error of $O(\Delta x^2)$. The second derivative can be calculated as follows:

$$\begin{aligned}\left(\frac{\partial^2 T}{\partial x^2}\right)_n &= \left[\frac{\partial}{\partial x}\frac{\partial T}{\partial x}\right]_n \\ &= \lim_{\Delta x \to 0}\left(\frac{\text{forward difference - backwards difference}}{\Delta x}\right) \\ &\approx \frac{\frac{T_{i+1}^n - T_i^n}{\Delta x} - \frac{T_i^n - T_{i-1}^n}{\Delta x}}{\Delta x} \\ &= \frac{T_{i+1}^n - 2T_i^n + T_{i-1}^n}{\Delta x^2} \end{aligned} \tag{5.5}$$

Inserting these numerical solutions into Equation (5.2), the one-dimensional heat diffusion equation can be expressed via a finite-difference scheme as

$$\frac{T_i^{n+1} - T_i^n}{\Delta t} - \alpha\left(\frac{T_{i+1}^n - 2T_i^n + T_{i-1}^n}{\Delta x^2}\right) = \frac{\dot{q}}{C}. \tag{5.6}$$

Rearranging for T_i^{n+1} gives

$$T_i^{n+1} = T_i^n + \Delta t\left[\alpha\left(\frac{T_{i+1}^n - 2T_i^n + T_{i-1}^n}{\Delta x^2}\right) + \frac{\dot{q}}{C}\right], \tag{5.7}$$

which is also known as the one-dimensional explicit finite-difference solution to Fourier's law of heat conduction. This scheme is illustrated in Figure 5.1: it is explicit as the temperature at time $n+1$ *explicitly* depends on the temperature at time n, and is evidently forward-in-time and space-centred.

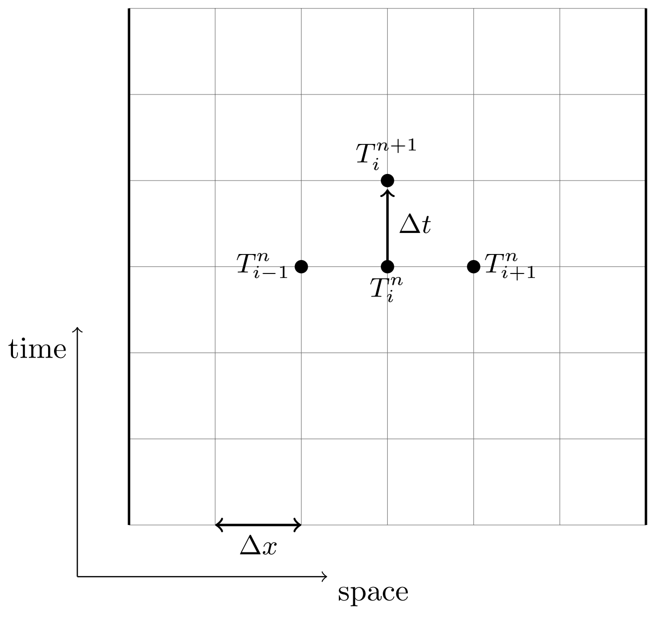


Figure 5.1: One-dimensional finite-difference schematic (boundary nodes indicated by dark vertical lines)

The timestep and lattice spacing, Δt and Δx respectively, must be chosen carefully to ensure the stability of this algorithm, and this is provided by defining the Fourier mesh number, F, as

$$F = \alpha \frac{\Delta t}{\Delta x^2} \tag{5.8}$$

This can be thought of as the ratio of timestep to the time it takes to equilibrate a region of length Δx. In this one-dimensional case, the value of F must satisfy $0 < F < \frac{1}{2}$, or else the algorithm becomes unstable and oscillates wildly.

In three-dimensions, if $\Delta x = \Delta y = \Delta z$, the finite-difference solution becomes

$$\begin{aligned} T^{n+1}_{i,j,k} &= T^n_{i,j,k} + \Delta t \left[\alpha \left(\frac{T^n_{i+1,j,k} + T^n_{i-1,j,k} + T^n_{i,j+1,k} + T^n_{i,j-1,k} + T^n_{i,j,k+1} + T^n_{i-1,j,k-1} - 6T^n_{i,j,k}}{\Delta x^2} \right) + \frac{\dot{q}}{C} \right] \\ &= T^n_{i,j,k} + F[T^n_{i+1,j,k} + T^n_{i-1,j,k} + T^n_{i,j+1,k} + T^n_{i,j-1,k} + T^n_{i,j,k+1} + T^n_{i-1,j,k-1} - 6T^n_{i,j,k}] + \Delta t \frac{\dot{q}}{C}, \end{aligned} \tag{5.9}$$

with a new stability criteria of $0 < F < \frac{1}{6}$. Thus, the size of the timestep must satisfy $\Delta t < \frac{\Delta x^2}{6\alpha}$. Equation (5.9) applies under the assumption that the thermal diffusivity α is a constant value, i.e. $\nabla.[\alpha \nabla T] = \alpha \nabla^2 T$, but the more general (and hence more complicated) case, where α can vary spatially, takes the form

$$\begin{aligned} T^{n+1}_{i,j,k} &= \frac{\Delta t}{\Delta x^2} \left(\frac{\kappa\left[\frac{1}{2}(T^n_{i+1,j,k} + T^n_{i,j,k})\right]}{C(T^n_{i,j,k})} (T^n_{i+1,j,k} - T^n_{i,j,k}) + \frac{\kappa\left[\frac{1}{2}(T^n_{i-1,j,k} + T^n_{i,j,k})\right]}{C(T^n_{i,j,k})} (T^n_{i-1,j,k} - T^n_{i,j,k}) \right) \\ &+ \frac{\Delta t}{\Delta y^2} \left(\frac{\kappa\left[\frac{1}{2}(T^n_{i,j+1,k} + T^n_{i,j,k})\right]}{C(T^n_{i,j,k})} (T^n_{i,j+1,k} - T^n_{i,j,k}) + \frac{\kappa\left[\frac{1}{2}(T^n_{i,j-1,k} + T^n_{i,j,k})\right]}{C(T^n_{i,j,k})} (T^n_{i,j-1,k} - T^n_{i,j,k}) \right) \\ &+ \frac{\Delta t}{\Delta z^2} \left(\frac{\kappa\left[\frac{1}{2}(T^n_{i,j,k+1} + T^n_{i,j,k})\right]}{C(T^n_{i,j,k})} (T^n_{i,j,k+1} - T^n_{i,j,k}) + \frac{\kappa\left[\frac{1}{2}(T^n_{i,j,k-1} + T^n_{i,j,k})\right]}{C(T^n_{i,j,k})} (T^n_{i,j,k-1} - T^n_{i,j,k}) \right) \\ &+ \Delta t \frac{\dot{q^n_{i,j,k}}}{C^n_{i,j,k}}. \end{aligned} \tag{5.10}$$

Here the electronic thermal conductivity has an explicit spatial dependence. To simplify this relationship, κ can be assumed to be constant locally, and is taken to be the average value between the current and neighbouring cells. An adaptive timestep is also utilised, so at each timestep a fraction of the 'worst case scenario' for the Fourier mesh number, F, is chosen, ensuring the stability of the electronic subsystem.

Various boundary condition choices are available for the edge cells in Figure 5.1, which surround the simulation in all three dimensions. These are:

- *Dirichlet Boundary Conditions*: Also known as infinite-flux boundary conditions, the edge cell is fixed at a finite temperature, $T = T_0$, where T_0 is the target (system) emperature. Dirichlet BCs are usually chosen for cascade simulations.
- *Neumann Boundary Conditions*: Also known as zero-flux boundary conditions, the temperature of the edge cell is taken to be the value of its corresponding neighbour, thus $\frac{dT}{dx} = 0$ in this region.
- *Robin Boundary Conditions*: Also known as partial or variable flux boundary conditions, the temperature of the edge cell is taken to be a fixed proportion of the neighbouring cell's temperature. Thus $\frac{dT}{dx} = -k(T - T_0)$, where k is the fraction of the neighbouring temperature that is 'targeted'.

The electronic energy contained in a voxel representing an electronic temperature grid point can be calculated by integrating the volumetric heat capacity between a datum temperature (e.g. system temperature) and the local electronic temperature, i.e.

$$E^j_e = \int_{T_0}^{T^j_e} C_e dT. \tag{5.11}$$

A quantity of electronic energy can be added to a voxel by setting the local electronic temperature to a new value, such that the integral of volumetric heat capacity between the original and new temperatures gives the required energy.

The atomic temperatures T_a and T'_a can be calculated from kinetic energies of atoms in each voxel, i.e. for cell j:

$$T_a^j = \frac{\sum_{p \in \vec{x}^j} m_p v_p^2}{3k_B N} \tag{5.12}$$

$$T_a'^j = \frac{\sum_{p \in \vec{x}^j, |v_p| > v_{cut}} m_p v_p^2}{3k_B N'} \tag{5.13}$$

where v_{cut} is the cut-off velocity beyond which electronic stopping is significant, N is the total number of atoms in the voxel and N' is the number of atoms in the voxel with velocities greater than v_{cut}. To account for centre-of-mass drift, the atomic velocities $\underline{v}_p$ should be corrected by the voxel's velocity, calculated by $\underline{v}_{com}^j = \frac{\sum_{p \in \vec{x}^j} m_p \underline{v}_p}{\sum_{p \in \vec{x}^j} m_p}$.

If there are too few (or no) atoms in a voxel, it is considered to be *inactive* as no definable ionic temperatures can be calculated. Equation (5.10) does not have to be applied to inactive voxels (setting electronic temperatures to the datum value T_0 and source terms to zero), while temperature gradient terms involving inactive voxels in the same equation can be omitted for their neighbours.

MD system

The principal idea is to modify the MD equations of motion according to Langevin dynamics, which describes the movement of particles in a viscous medium. This viscous medium will represent the electronic subsystem, and the modified equation of motion takes the form

$$\frac{d\underline{v}_p(t)}{dt} = \frac{\underline{f}_p(t) + \underline{R}_p(t)}{m_p} - \chi_p \, \underline{v}_p(t), \tag{5.14}$$

where m_p and $\vec{v}_p$ are the mass and velocity of atom p at time t, $\underline{f}_p$ is the deterministic force on atom p due to the interatomic potential, $\underline{R}_p(t)$ is a stochastic force with random magnitude and orientation and $m_p \chi_p \underline{v}_p(t)$ is the frictional force due to the electrons. These last two terms in Equation (5.14) are the Langevin modifications to Newton's second law, which allow energy to be lost and gained by the MD system.

The stochastic force $\underline{R}_p(t)$ returns energy from the electrons to the ions and is formulated as $\underline{R}_p(t) = \sqrt{\Gamma} \vec{A}_p(t)$, where $\vec{A}_p(t)$ is a three-dimensional vector with components randomly distributed in $[-1, 1]$, and Γ is the stochastic friction coefficient. $\underline{R}_p(t)$ must satisfy two important time-averaged conditions:

$$\langle \underline{R}_p(t) \rangle = 0, \tag{5.15}$$

$$\langle \underline{R}_p(t) \cdot \underline{R}_q(t') \rangle \propto \delta_{pq} \delta(t - t') \tag{5.16}$$

The first condition states that over a significant period of time, $\underline{R}_p(t)$ must not behave as a net source or sink. Equation (5.16) is known as the fluctuation-dissipation theorem, which describes how the drag felt by a particle as it moves through a viscous medium can give rise to Brownian motion. In the standard homogeneous Langevin thermostat, every atom in the MD simulation is thermostatted to a target temperature. The inhomogeneous case allows for each atom to be thermostatted to the electronic temperature of the corresponding continuum electronic cell. This leads to the stochastic friction term

$$\Gamma = \frac{6 m_p \chi_{ep}^j k_B T_e^j}{\Delta t}, \tag{5.17}$$

where χ_{ep}^j is the electron-phonon friction of the j^{th} electronic finite-element cell, T_e^j is the electronic temperature of the corresponding cell, k_B is the Boltzmann constant, and Δt is the timestep. The electron-phonon friction term is thus calculated at each point in the finite electronic temperature grid:

$$\chi_{ep} = \frac{G_{ep}\Delta V}{3k_B N}, \tag{5.18}$$

where ΔV is the volume of the cell ($\Delta x \Delta y \Delta z$), G_{ep} is the electron-phonon coupling constant of the material, and N is the number of atoms in the cell.

The friction term in Equation (5.14) is made up of two forms of energy loss: the previous discussed electron-phonon friction, and electronic stopping, which is inelastic electron scattering of ballistic atoms. The total electron friction coefficient χ_p for atom p is given by

$$\chi_p = \begin{cases} \chi_{ep} + \chi_{es} & : \quad |v_p| > v_{cut} \\ \chi_{ep} & : \quad |v_p| \leq v_{cut} \end{cases}, \tag{5.19}$$

where χ_{es} is the electronic stopping friction, $\underline{v}_p$ is the velocity of atom p, and v_{cut} is the cut-off velocity for which electronic stopping becomes significant. The electronic stopping friction term can be calculated in a similar fashion to the electron-phonon term:

$$\chi_{es} = \frac{G_s \Delta V}{3k_B N'} \tag{5.20}$$

where N' is the number of atoms in the cell with velocities greater than v_{cut}. Note that this term is set to zero when $N' = 0$.

From Equations (5.14), (5.18) and (5.20), the differences between the contributions from electron-phonon coupling and electronic stopping are evident. Electron-phonon coupling allows energy to flow to and from the lattice (depending on the temperature gradient between ions and electrons), whereas electronic stopping acts solely as an energy loss mechanism for the lattice. Figure 5.2 illustrates these processes for swift heavy

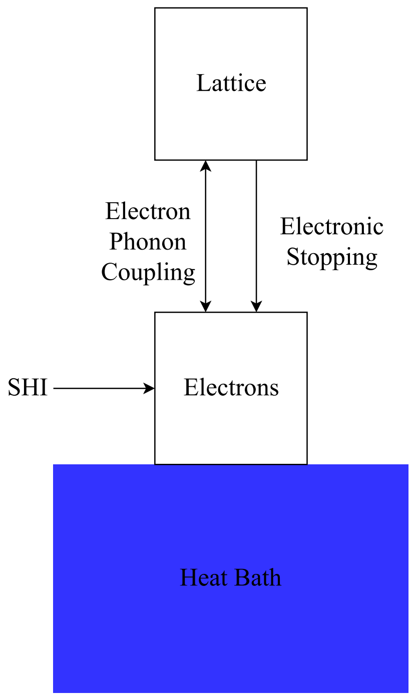


Figure 5.2: Schematic of thermodynamic coupling and processes in 2T-MD model

ion simulations, and highlights how the MD cell is now thermostatted to a heat bath. The lattice will reach local equilibrium with the electron gas, which is thermostatted to the heat bath, thus eventually driving both subsystems to the chosen ambient temperature. Energy can only be removed from the system via the electron gas; this is justified due to how slow lattice heat diffusion is in comparison to electronic heat diffusion.

It is possible to use the inhomogeneous Langevin thermostat (Equation (5.14)) on its own without coupling it to the electronic temperature grid, but still enhancing the total Langevin friction term for atoms with velocities greater than a cut-off value[102]. In this case, the stochastic friction coefficient Γ is modified to use the system temperature T_0 instead of a local electronic temperature and to take advantage of the enhanced friction coefficient when electronic stopping applies, i.e.

$$\Gamma = \frac{6 m_p \chi_p k_B T_0}{\Delta t}. \tag{5.21}$$

5.3 Simulation setup

There are three distinct types of irradiation that can be simulated using the TTM (2T-MD) implementation in DL_POLY_4: swift heavy ions, laser excitation, and high-energy cascades. These are conducted by splitting the MD cell into discrete coarse-grained lattice ionic temperature (CIT) voxels, and discretising the electronic system into coarse-grained electronic temperature (CET) voxels. Energy can thus be exchanged between the voxels and subsequently passed to or from the atoms within each respective CIT. The volume of each CIT voxel must contain a sufficient number of atoms so that thermal fluctuations of ions are negligible and an ionic temperature can be defined: a good general choice is a cube of length 10 Å in each direction. There is more flexibility in choosing the number of CET voxels, as long as an integer number of these overlap with the CIT grid: to simplify the connections between the CET and CIT grids, equal-sized voxels for both systems will be assumed from here on.

Cascades

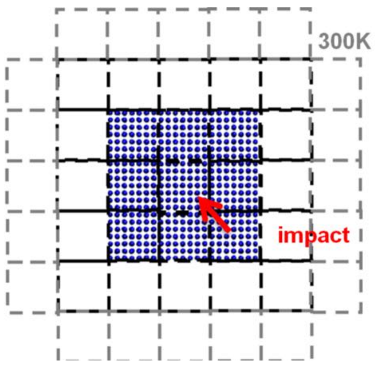


Figure 5.3: Schematic of cascade simulation setup

High-energy cascades require no initial energy deposition into the electronic system (i.e. $\frac{dE}{dx} = 0$): instead, an ion is initialised with a very high velocity. The electronic temperature (CET) voxels extend further than the ionic temperature (CIT) voxels in all directions, with open (Dirichlet) or semi-open (Robin) boundary conditions in all dimensions to represent thermal electronic conduction into the bulk, allowing the electronic temperature to converge towards the initial system temperature T_0. (Figure 5.3 gives an example schematic of this simulation setup.) Stochastic boundary conditions can be applied in the ionic system to dampen the shock wave generated by the displacement spike.

Swift heavy ions

Swift heavy ion systems can be modelled using an initial Gaussian spatial energy deposition into the electronic system (i.e. $\frac{dE}{dx} > 0$) with either Gaussian or exponentially decaying temporal distribution in electronic temperature. The size of the electronic temperature (CET) grid in the z-direction is set equal to the size of the ionic temperature (CIT) grid in the same dimension, while the CET voxels are extended over the corresponding CIT voxels in the x- and y-directions. Boundary conditions can be set with no energy flux

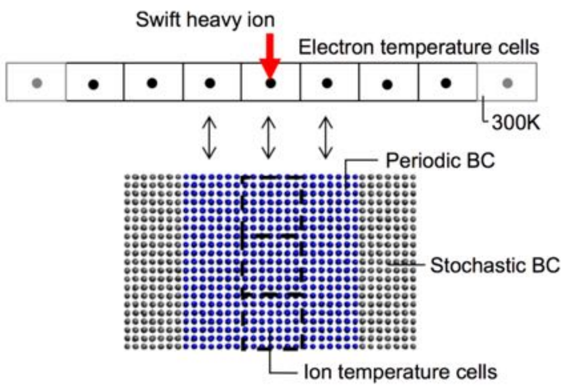


Figure 5.4: Simulation setup for swift heavy ion impact.

in the z-direction and open or semi-open boundary conditions in x- and y-directions. (Figure 5.4 gives a schematic of this simulation setup.) Stochastic boundary conditions can be applied to the lattice system in lateral directions only to represent non-negligible phononic thermal conductivity in semiconductors into the builk. Similarly, while electronic thermal conduction in the lateral directions is allowed, conduction parallel to impact is not. This reflects the fact that the simulation represents a small cross-section of the evolution of a micron-sized track.

Laser excitation

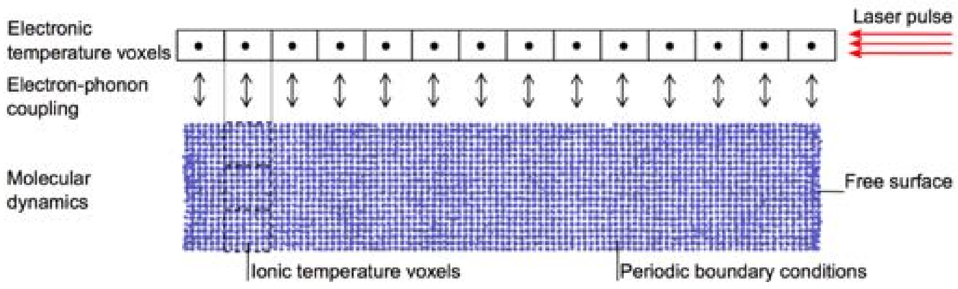


Figure 5.5: Simulation setup for laser irradiation.

Laser excitation systems can be modelled with an initial homogeneous spatial energy deposition into the electronic system (either in all three directions or in x- and y-directions with exponential decay in the z-direction) with either Gaussian or exponentially decaying temporal distribution in electronic temperature. (The energy deposition can be specified for the fully homogeneous case either by setting $\frac{dE}{dx} > 0$ or by giving values for the absorbed fluence and penetration depth from the laser.) The size of the electronic temperature

(CET) grid is set to the same size as the ionic temperature (CIT) grid, with zero-flux (Neumann) boundary conditions in all directions. This setup (shown in Figure 5.5) represents a homogeneous laser excitation with the simulated part as a small section of a larger photoexcited sample.

It is possible in such simulations for voxels to become empty due to displacement of atoms from the laser source. These voxels are omitted from electronic heat diffusion calculations, setting their electronic temperatures to the background value T_0 and their source terms to zero. Their associated spatial gradients in Equation (5.10) are also omitted for neighbouring voxels.

5.4 Implementation

TTM with MD (2T-MD) has been implemented in DL_POLY_4 to take advantage of the domain decomposition scheme used by the code, by dividing up the coarse-grained ionic (CIT) and electronic (CET) temperature voxels as evenly as possible among the processors based on location. This avoids the need for each processor to hold copies of the entire CIT and CET grids and provides good to excellent parallel scalability for larger scale problems.

Coarse-grained ionic temperature (CIT) voxels are divided among processors with overlapping voxels between two or more processors assigned to the first processor in each direction. A boundary halo of voxels is also included to allow communication of contributions to voxel momenta, kinetic energies and atom counters between processors for calculations of ionic temperatures. Since ionic temperatures are only needed for finite-difference calculations of Equation (5.1), some of these communications only need to be applied in one direction for each dimension.

The coarse-grained electronic temperature (CET) grid is considered as integer multiples of the ionic temperature grid, with equal numbers in both directions of each dimension. While this may provide more CET voxels than requested by the user, the application of boundary conditions in the correct places means that the finite-difference calculations can be carried out in superfluous voxels without affecting the result. The centre of the CET grid is located at the same place as the CIT grid, matching the two up precisely: the electron-phonon, electronic stopping and energy deposition source terms are only applied in these CET voxels. Communications of electronic temperature can be carried out both within each 'copy' of the ionic temperature grid and between them: these need to be applied for each iteration (timestep) of the finite-difference solver.

Communications to and from boundary halos for both CIT and CET grids make use of MPI derived data types, which allow for single MPI send and receive calls for grid values without needing to pack and unpack data. This is the same communication technique used in DL_MESO for its lattice Boltzmann equation code[103] and has been shown to give near-perfect parallel scaling to thousands of processors.

Functionality and directives

All directives in the CONTROL file beginning with **ttm** will switch on the two-temperature model (2T-MD) as described above. If no other information is provided, DL_POLY_4 will use default values for certain required properties, but some information *must* be provided: if this information is unavailable, DL_POLY_4 will terminate. The list of TTM directives is given as part of Section 10.1.1.2: more details about these directives are given below.

The inhomogeneous Langevin thermostat can be activated using the directive **ensemble nvt ttm** or **ensemble nvt inhomo** in the CONTROL file, specifying the electron-phonon friction term (χ_{ep}, in ps^{-1}), electronic stopping friction term (χ_{es}, in ps^{-1}) and the cutoff atomic velocity for electronic stopping (v_{cut}, in Å ps^{-1}). This thermostat is required for 2T-MD calculations but can also be used independently: this CONTROL file directive therefore does not automatically switch on the two-temperature model, but the thermostat will be selected automatically if TTM is otherwise switched on. Default values for χ_{ep}, χ_{es} and v_{cut} will be used in this case if they have not been specified, although if a standard NVT Langevin ensemble has been selected, its thermostat friction term χ will be used for χ_{ep}.

By default, the inhomogeneous Langevin thermostat will be applied only to particle thermal velocities, i.e. velocities that have been corrected to remove the centre-of-mass flow calculated from its coarse-grained ionic temperature (CIT) voxel. This can be overridden either by the directive **ttm thvelz** to only correct the z-direction velocity component and use total velocity components in the x- and y-directions, or by the directive **ttm nothvel** to omit all velocity corrections and use total particle velocities. A warning message will also be printed if the **no vom** option is not used, as removal of total centre-of-mass motion may affect the dynamics of systems with electronic stopping effects.

The number of coarse-grained electronic temperature (CET) voxels is specified in the CONTROL file with the directive **ttm ncet**: by default, a grid of 50 × 50 × 50 will be used if this information is not supplied by the user. The number of coarse-grained ionic temperature (CIT) voxels in the z-direction is specified in CONTROL using the directive **ttm ncit**: the default number is 10, and the number of voxels in x- and y-directions will be determined automatically based on system size. The number of CET voxels must be at least the same as CIT or larger: too few CET voxels will cause DL_POLY_4 to terminate.

The volumetric heat capacity C_e can be obtained for TTM calculations in four different forms with their corresponding CONTROL file directives:

- A temperature-independent constant value (**ttm ceconst**)

$$C_e = C_0\rho$$

- A linear function of temperature up to a maximum at the Fermi temperature T_f (**ttm celin**)

$$C_e(T) = C_0\rho \max\left(T/T_f, 1\right)$$

- A hyperbolic tangent function of temperature (**ttm cetanh**)

$$C_e(T) = A\rho \tanh\left(10^{-4}BT\right)$$

- A tabulated function of temperature supplied in a Ce.dat file (**ttm cetab**).

Note that the values given for the first three options are specific heat capacities (C_0 and A given in k_B per atom), which are converted to volumetric heat capacities by multiplying by the atomic density $\rho = \frac{N}{\Delta V}$. The atomic density is assumed to be constant throughout the system, and this value can be set in one of three ways: (1) a value can be calculated from the provided configuration at the start (assuming all ionic temperature cells are active), (2) the user can specifiy a value using the **ttm atomdens** directive in the CONTROL file, or (3) after energy deposition to the electronic temperature grid, the value can be calculated dynamically from active CITs when using the **ttm dyndens** directive. In the latter case, the system or user-specified atomic density is used during energy deposition. Tabulated volumetric heat capacities are given in the Ce.dat file as J m^{-3} K^{-1}, which are converted to k_B Å^{-3}. If no heat capacity information is supplied, DL_POLY_4 will assume a constant volumetric heat capacity of 1 k_B per atom by default. In all cases, the electronic energy of a given voxel can be determined from the product of cell volume and the integral of the volumetric heat capacity between the system temperature T_0 and its current electronic temperature T_e.

If the system is metallic (specified by the CONTROL directive **ttm metal**), a thermal conductivity needs to be supplied: no default value is provided by DL_POLY_4. Four options are available with their corresponding CONTROL file directives:

- An infinitely large value (**ttm keinf**)

$$K_e = \infty$$

- A temperature-independent constant value (**ttm keconst**)

$$K_e = K_0$$

- A Drude-model (linear) function of temperature (**ttm kedrude**)

$$K_e(T) = K_0 \frac{T}{T_0}$$

- A tabulated function of temperature supplied in a Ke.dat file (**ttm ketab**).

All values (constants or tabulated) are supplied in W m^{-1} K^{-1}. In the case of infinitely large conductivity, all heat diffusion in the electronic subsystem is instantaneous and all active CET voxels without source terms will be at the same mean electronic temperature. The Drude model uses the electronic temperature for a given voxel, while the tabulated function will use either the ionic temperature for overlapping cells or the system temperature for CET voxels beyond the CIT system.

If the system is non-metallic (specified by **ttm nonmetal** in CONTROL), a thermal diffusivity needs to be supplied: no default value is provided by DL_POLY_4. Three options are available with their corresponding CONTROL file directives:

- A temperature-independent constant value (**ttm deconst** or **ttm diff**)

$$D_e = D_0$$

- A reciprocal function of temperature up to the Fermi temperature T_f (**ttm derecip**)

$$D_e(T) = D_0 \frac{T_0}{\min{(T, T_f)}}$$

- A tabulated function of temperature supplied in a De.dat file (**ttm detab**).

All values (constant and tabulated) are supplied in m^2 s^{-1} and subsequently converted to Å^2 ps^{-1}.

The electron-phonon coupling friction term χ_{ep} can either be held at the constant value given in **ensemble nvt ttm**, or it can be dynamically varied according to electronic temperature. The CONTROL file directive **ttm varg** can be used to specify that the electron-phonon coupling constant G_{ep} is supplied in tabulated form from a g.dat file, with values of G_{ep} given in W m^{-3} K^{-1} and converted to χ_{ep} values (using Equation (5.18) with the mean value for the number of atoms per voxel, $N = \rho \Delta V$) in ps^{-1}. Two variants of dynamic coupling are available: homogeneous coupling uses the mean electronic temperature to calculate a system-wide value of chi_{ep}, while heterogeneous coupling uses local values of electronic temperature (based on values in CET voxels) to calculate chi_{ep} for each atom.

Boundary conditions to the electronic temperature system are applied using the **ttm bcs** directive in the CONTROL file. Along with the Dirichlet, Neumann and Robin boundary conditions described earlier in this chapter, periodic boundary conditions can also be applied. Combinations of Dirichlet or Robin boundary conditions in x- and y-directions with Neumann boundary conditions in the z-direction are also available. These boundary conditions have no direct connection to any boundary conditions used for the MD system: conditions for the latter should be chosen carefully by the user to give the desired effects.

An energy deposition ($A(r,t)$) can be applied using one of two CONTROL file directives:

- **ttm dedx**: specifies the electron stopping power of a projectile entering the electronic system ($\frac{dE}{dx}$ in eV/nm);
- **ttm laser**: specifies the absorbed fluence (F in mJ cm^{-2}) and penetration depth (l_p in nm) of a laser.

The energy deposition can be split into spatial and temporal parts, i.e. $A(r,t) = f(r,z)g(t)\Delta V$. Depositions can be applied spatially in one of three ways:

- **ttm sflat**: specifies a homogeneous distribution in x-, y- and z-directions

  $$f(r) = \frac{dE}{dV} = \frac{dE}{dx}\frac{1}{L_x L_y}$$

  where L_x and L_y are the MD system dimensions in x- and y-directions;

- **ttm sgauss**: specifies a Gaussian distribution in x- and y-directions from the centre of the system, i.e.

  $$f(r) = \frac{dE}{dx}\frac{1}{2\pi\sigma^2}e^{-\frac{r^2}{2\sigma^2}}$$

  with standard deviation σ, extending up to a cut-off distance (given in multiples of σ);

- **ttm laser** F l_p **zdep**: specifies a homogeneous distribution in x- and y-directions with exponential decay in the z-direction (from the centre of the system)

  $$f(r,z) = \frac{F}{l_p}\exp\left(-\frac{|z|}{l_p}\right)$$

  using the penetration depth l_p as the exponential scaling factor.

In the first two cases, spatial depositions are homogeneous in the z-direction. Unless the **zdep** option is invoked, laser depositions are homogeneous in all three directions (similar to the **ttm sflat** case), with the ratio of absorbed fluence to penetration depth equal to $\frac{dE}{dV}$. The deposition can be applied temporally in one of four ways:

- **ttm delta**: specifies a Dirac delta function in time

  $$g(t) = \delta(t - t_0)$$

  which is approximated as an energy injection during a single diffusion timestep;

- **ttm pulse**: specifies a square pulse function in time

  $$g(t) = \begin{cases} \frac{1}{\tau}, & t - t_0 < \tau \\ 0, & t < t_0 \text{ and } t - t_0 \geq \tau \end{cases}$$

  over a period τ;

- **ttm gauss**: specifies a Gaussian function in time

  $$g(t) = \frac{1}{\sigma\sqrt{2\pi}}e^{-\left(\frac{t-t_0}{\sigma}\right)^2}$$

  over a period of a multiple of standard deviations σ;

- **ttm nexp**: specifies a decaying exponential function in time

  $$g(t) = e^{-\frac{t-t_0}{\tau}}$$

  over a period of a multiple of τ.

The exponential function is scaled to ensure the correct total energy (within a tolerance of ±1%) is deposited to the electronic system. Energy depositions are achieved by determining the required increases in electronic temperature to add the required energy to the CET voxel.

To calculate ionic temperatures in CIT voxels, a minimum number of atoms is required: this can be specified by the CONTROL file directive **ttm amin**, although a default value of 1 is used if this is not supplied by the user. Any CIT voxel that includes this number of atoms will be considered as active and an ionic temperature can be calculated from the mean kinetic energy of atoms in the voxel (after centre-of-mass motion is removed). If the voxel does not contain enough atoms, it is considered inactive: no electron-phonon coupling is applied to the corresponding electronic temperature cell, although thermal diffusion is still applied as normal. All voxels within the CIT grid are checked at each MD timestep and can be reactivated once the minimum number of atoms can be found. If the **ttm dyndens** option is also specified, the average cell density (number of atoms per unit volume) will be recalculated after energy deposition by only considering active CIT voxels: note that the system-based or user-specified atomic density will be used before and during energy deposition.

If the **ttm redist** option is in use, any recently deactivated voxels have their electronic energy transferred to neighbouring active voxels by increasing their electronic temperatures: these are used for localised Dirichlet (constant temperature) boundary conditions during thermal diffusion in the single MD timestep. The corresponding electronic temperature cell is also deactivated: its electronic temperature is set to the system temperature and the voxel is excluded from thermal diffusion calculations by setting the electronic temperature gradient between itself and any neighbours to zero. This option requires at least one electronic temperature cell beyond the ionic system in each direction to ensure electronic energy from deactivated cells is not lost: if this condition does not hold, the energy transfer option is switched off.

To equilibrate the electronic system prior to any energy deposition, the CONTROL file directive **ttm offset** can be used to allow the electronic and ionic systems to settle: this switches off the electron-phonon terms in both electronic heat diffusion calculations and the inhomogeneous Langevin thermostat, only applying electronic stopping effects. The **ttm oneway** directive restricts electron-phonon coupling to coarse-grained ionic temperature (CIT) voxels and their atoms when the electronic temperature is higher than the ionic temperature: both the electron-phonon coupling term in the thermal diffusion equation and the inhomogeneous Langevin thermostat are switched off if this condition is not met.

The CONTROL file directives **ttm stats** and **ttm traj** switches on outputs of statistical information and limited temperature trajectory snapshots of the two-temperature model simulation. Ionic temperatures (minimum, maximum, mean and sum) are supplied at user-defined intervals in the file PEAK_I, while electronic temperatures and energies are supplied in the PEAK_E file. Ionic and electronic temperature profiles along the y-direction at the centre of the xz plane are given in the LATS_I and LATS_E respectively at user-defined intervals.

Overview of program structure

The following modules, both modified and added, incorporate the two-temperature molecular dynamics model in DL_POLY_4:

- DL_POLY.F90: modified to initialise TTM_MODULE.F90;
- READ_CONTROL.F90: modified to include the additional control directives required to activate and parameterise 2T-MD;
- TTM_MODULE.F90: deal with initial array allocation, locations of electronic temperature voxel boundaries etc.;
- TTM_TRACK_MODULE.F90: deal with initial energy deposition to the electronic temperature grid;
- TTM_UTILS.F90: contains functions to calculate properties for heat diffusion calculations, boundary conditions etc.;

- TTM_TABLE_SCAN.F90: scans for any-temperature dependent electronic parameter data files;
- TTM_TABLE_SCAN.F90: reads in tabulated parameters for the electronic system;
- TTM_SYSTEM_INIT.F90: reads in any electronic temperature files (DUMP_E) to allow simulation restart;
- W_MD_LFV.F90 and W_MD_VV.F90: modified to calculate ionic lattice and electronic cell temperatures and call the new inhomogeneous Langevin thermostat (NVT_L2_LFV.F90 and NVT_L2_VV.F90);
- TTM_ION_TEMPERATURES.F90: calculates local ion lattice temperatures from average kinetic energies of atoms in coarse-grained ionic temperature (CIT) cells and electronic stopping terms (determining energy transferred from ions to electrons);
- TTM_THERMAL_DIFFUSION.F90: calculates local electronic temperatures by solving the heat diffusion equation using a finite-difference solver, including source and sink terms representing energy exchange with the MD cell;
- W_INTEGRATE_LFV.F90 and W_INTEGRATE_VV.F90: modified to call the new inhomogeneous Langevin thermostat (NVT_L2_LFV.F90 and NVT_L2_VV.F90);
- NVT_L2_LFV.F90 and NVT_L2_VV.F90: new inhomogeneous Langevin thermostat (both leap-frog and velocity Verlet variants);
- LANGEVIN_FORCES.F90: modified to allow scaling of Langevin random forces by local electronic temperature instead of system temperature;
- TTM_SYSTEM_REVIVE.F90: writes file with electronic temperatures (DUMP_E) to allow the MD system with TTM to be restarted.

Chapter 6

Heat Flux

Scope of Chapter

This chapter describes the heat flux functionality available in DL_POLY_4.

6.1 Heat Flux

6.1.1 Introduction

It is possible to use DL_POLY_4 to calculate the heat flux of a material with two-body interactions in an MD simulation. The heat flux can subsequently be used as a means to calculate the thermal conductivity of a material via the Green-Kubo relation. Currently, the heat flux is only viable for two-body interactions, but valid for any two-body interactions.

To enable the calculation of heat flux add the **heat_flux** keyword into the CONTROL file.

6.1.2 Theory

The heat flux for two-body interactions is defined as:

$$\underline{J} = \frac{1}{V}\left[\sum_{i}^{N} e_i \underline{v}_i - \sum_{i}^{N} \underline{\underline{\mathbf{S}}}_i \underline{v}_i\right] \tag{6.1}$$

where $\underline{\mathbf{J}}$ is the heat flux, V is the volume of the cell, N is the number of particles, e is the energy, $\underline{v}$ is the velocity, $\underline{\underline{\mathbf{S}}}$ is the stress. All subscript i refer to the particle.

The thermal conductivity can then be as an auto-correlation of the heat flux over a run:

$$\kappa = \frac{V}{k_B T^2} \int_0^\infty \langle \underline{J}(0)\underline{J}(t)\rangle \,\mathrm{d}t \tag{6.2}$$

where κ is the thermal conductivity, V is the volume of the cell, k_B is the Boltzmann constant, J is the heatflux.

6.1.3 Implementation

For the purposes of calculating per-particle SPME interactions, the long-range electrostatics forces and energies are calculated differently for the heat flux case. From the SPME equations, we can calculate a per particle contribution via:

$$\Omega^{ABC} = \sum_j \omega_j^{ABC}$$

$$\omega_j^{ABC} = \frac{1}{2\pi V}\sum_{n_1 n_2 n_3=-\infty}^{\infty}\sum_{k_1}^{K_1-1}\sum_{k_2}^{K_2-1}\sum_{k_3}^{K_3-1} Q_j^{ABC}(\mathbf{k},\mathbf{n})\sum_{\mathbf{m}\neq 0}\left[\prod_\mu b_\mu(\mathbf{m})e^{2\pi i\frac{m_\mu k_\mu}{K_\mu}}\right] f(\mathbf{m})$$

$$Q_j^{ABC}(\mathbf{k},\mathbf{n}) = q_j \left[\frac{K_\alpha}{2\pi i}\right]^A \frac{\partial^A}{\partial u_{\alpha j}^A} M_n(u_{\alpha j} - k_\alpha - n_\alpha K_\alpha)\times$$

$$\left[\frac{K_\beta}{2\pi i}\right]^B \frac{\partial^B}{\partial u_{\beta j}^B} M_n(u_{\beta j} - k_\beta - n_\beta K_\beta)\times$$

$$\left[\frac{K_\gamma}{2\pi i}\right]^C \frac{\partial^C}{\partial u_{\gamma j}^C} M_n(u_{\gamma j} - k_\gamma - n_\gamma K_\gamma)$$

where ω is the per-particle contribution for particle j, q is the charge, other values are defined in the SPME section (see: 2.4.1.5)

6.1.4 File

The heat flux method creates a file called HEATFLUX which contains the relevant data structured as: STEP PRESSURE VOLUME HEAT-FLUX

Chapter 7

Exdenting DL_POLY_4 to reactive systems: the Empirical Valence Bond method

Scope of Chapter

This chapter presents the theoretical fundamentals of the Empirical Valence Bond (EVB) method and its implementation into the DL_POLY_4 code. This feature allows to perform molecular dynamics simulations of reactive processes via the use of standard (non-reactive) force-fields.

7.1 Framework and motivation

A key component of DL_POLY_4 is the Force Field (FF) to model the interactions between atoms. As already described in previous chapters, such atomic interactions are often modelled by relatively simple functional forms with parameters either fitted to experimental data or derived from quantum mechanical calculations. In most of the classical FFs available, functional forms and fitted parameters remain unchanged during the course of the molecular dynamics (MD) simulation. Indeed, this is the type of FFs that DL_POLY_4 can handle. In reactive processes, however, the nature of the interactions inevitable changes due to the formation of new chemical species. For this reason, standard FFs (thence DL_POLY_4) are not suitable to simulate chemical reactions. We shall refer to such FFs as non-reactive.
An alternative to simulate chemical reactions is offered by the so called Reactive FFs (RFFs). In contrast to standard FFs where interactions are modelled for a particular state with a given topology and chemistry, RFFs are designed to model the interatomic interactions valid for multiple states that are chemically different. The task of designing RFFs, however, is very challenging and requires a high level of expertise to tackle a multi-dimensional problem, where the modelled interactions are often expressed by complicated functional forms with many strongly coupled parameters that are optimised via the use of sophisticated tools. Even though RFFs have evolved considerably in the last years, a general parametrization is not yet available and, instead, parameters have to be tuned to specific chemical systems and environments.
Within this framework and to the purpose of extending the applicability of DL_POLY_4 to simulate reactive processes, the Empirical Valence Bond (EVB) method [104] offers an appealing alternative for computational implementation and development. In contrast to facing the challenges of building RFFs, the EVB method defines a suitable matrix using computed quantities of the participating chemical states, where each state is modelled by a non-reactive FF. Via the definition of appropriate coupling terms and matrix diagonalization at each time step, it is possible to obtain potential energy landscapes that account for the change in chemistry when sampling conformations between the participating, chemically different, states.
In contrast to RFFs, the advantage of the EVB method lies in the large availability of standard non-reactive FFs libraries. In addition, despite the initial task to calibrate the coupling terms against reference data, research has demonstrated that these couplings are invariant to the surrounding electrostatics, making it possible to simulate the same reactive unit in different environments. For further details about the applications of the EVB method, we refer the user to ref. [105].
The fundamentals of the EVB method are presented in the next section. Strategies to calibrate EVB-FFs are discussed in section 7.3. The computational implementation of the EVB method is described in section 7.4. Finally, section 7.5 provides a guideline to users on how to prepare the settings for EVB simulations with DL_POLY_4.

7.2 The EVB method

Let us assume an atomic system composed of N_p particles with positions described by the set of vectors $\{\mathbf{R}\}$. The non-reactive force field (FF) for the chemical state m is described by the configurational energy $E_c^{(m)}(\{\mathbf{R}\})$ and the set of forces $\vec{F}_J^{(m)}(\{\mathbf{R}\})$, where the index J runs over the total number of particles. The configurational energy function $E_c^{(m)}(\{\mathbf{R}\})$ has the decomposition of eq. (2.1). In the following, however, we shall omit the presence of external fields, such as electric or magnetic. In the current notation, we shall use indexes m and k for the chemical states (and FFs), I and J for atoms and Greek letters for Cartesian coordinates. Indexes in parenthesis are used to emphasize the particular chemical state.
The purpose of the EVB method is to couple N_F non-reactive force fields to obtain a reactive potential. These FFs are coupled through the Hamiltonian $\hat{H}_{\mathrm{EVB}}$ with a matrix representation $H_{\mathrm{EVB}} \in \mathcal{R}^{N_F \times N_F}$ that has the following components

$$H_{\mathrm{EVB}}^{mk}(\{\mathbf{R}\}) = \begin{cases} E_c^{(m)}(\{\mathbf{R}\}) & m = k \\ C_{mk}(\epsilon_{mk}) & m \neq k \end{cases} \tag{7.1}$$

where each diagonal element corresponds to the configurational energy $E_c^{(m)}(\{\mathbf{R}\})$ of the non-reactive FF that models the interactions as if the system was in the chemical state (m), whereas the off-diagonal terms C_{mk} are the couplings between states m and k. For convenience in the notation, we shall omit hereinafter the dependence on the set of coordinates $\{\mathbf{R}\}$ for the particles. Even though there are different possible choices for the coupling terms, in the above definition we have set C_{mk} to depend on $\epsilon_{mk} = E_c^{(m)} - E_c^{(k)} = -[E_c^{(k)} - E_c^{(m)}] = -\epsilon_{km}$, where ϵ_{mk} is commonly referred to as energy gap and defines a possible reaction coordinate for the reactive process [106]. Since the H_{EVB} matrix is Hermitian by construction and the C_{mk} terms are real, the condition of $C_{mk} = C_{km}$ must be imposed to the off-diagonal elements. Diagonalization of H_{EVB} leads to N_F possible eigenvalues $\{\lambda_1, ..., \lambda_{N_F}\}$ with

$$H_{\mathrm{EVB}}\Psi_{\lambda_m} = \lambda_m \Psi_{\lambda_m}, \qquad m = 1, ..., N_F. \tag{7.2}$$

The EVB energy, E_{EVB}, is defined as the lowest eigenvalue

$$E_{\mathrm{EVB}} = min(\lambda_1, ..., \lambda_{N_F}) \tag{7.3}$$

with the corresponding normalized EVB eigenvector

$$\Psi_{\mathrm{EVB}} = \Psi_{min(\lambda_1,...,\lambda_{N_F})}. \tag{7.4}$$

and

$$E_{\mathrm{EVB}} = \langle \Psi_{\mathrm{EVB}} | \hat{H}_{\mathrm{EVB}} | \Psi_{\mathrm{EVB}} \rangle. \tag{7.5}$$

Since the eigenvector Ψ_{EVB} is real and normalized we have

$$\sum_{k=1}^{N_F} |\Psi_{\mathrm{EVB}}^{(k)}|^2 = 1 \tag{7.6}$$

from which we can interpret $|\Psi_{\mathrm{EVB}}^{(k)}|^2$ as the fraction of the chemical state (k) being part of the EVB state. The eigenvector Ψ_{EVB} can also be represented as a column vector $\in \mathcal{R}^{N_F \times 1}$ where $\Psi_{\mathrm{EVB}}^{(k)}$ is the element of the k-row. Thus, eq. (7.5) is expressed as a matrix multiplication

$$E_{\mathrm{EVB}} = \sum_{m,k=1}^{N_F} \tilde{\Psi}_{\mathrm{EVB}}^{(m)} H_{\mathrm{EVB}}^{mk} \Psi_{\mathrm{EVB}}^{(k)} \tag{7.7}$$

where $\tilde{\Psi}_{\mathrm{EVB}}$ is the transpose of Ψ_{EVB}. The resulting EVB force over the particle J, $\vec{F}_J^{\mathrm{EVB}}$, follows from the Hellman-Feynman theorem

$$\begin{aligned}\vec{F}_J^{\mathrm{EVB}} &= -\nabla_{\vec{R}_J} E_{\mathrm{EVB}} = -\langle \Psi_{\mathrm{EVB}} | \nabla_{\vec{R}_J} \hat{H}_{\mathrm{EVB}} | \Psi_{\mathrm{EVB}} \rangle \\ &= \sum_{\alpha=x,yz} F_{J\alpha}^{\mathrm{EVB}}\ \breve{\alpha}\end{aligned} \tag{7.8}$$

where $\breve{\alpha}$ corresponds to each of the orthonormal Cartesian vectors and

$$F_{J\alpha}^{\mathrm{EVB}} = -\langle \Psi_{\mathrm{EVB}} | \frac{\partial \hat{H}_{\mathrm{EVB}}}{\partial_{R_{J\alpha}}} | \Psi_{\mathrm{EVB}} \rangle. \tag{7.9}$$

From eq. (7.1) the matrix components of the operator $\frac{\partial \hat{H}_{\mathrm{EVB}}}{\partial_{R_{J\alpha}}}$ are given as follows

$$\frac{\partial H_{\mathrm{EVB}}^{mk}}{\partial R_{J\alpha}} = \begin{cases} \frac{\partial E_c^{(m)}}{\partial R_{J\alpha}} = -F_{J\alpha}^{(m)} & m = k \\ \\ \dfrac{dC_{mk}}{\partial R_{J\alpha}} = \dfrac{dC_{mk}(\epsilon_{mk})}{d\epsilon_{mk}} \dfrac{\partial \epsilon_{mk}}{\partial R_{J\alpha}} & m \neq k \\ \qquad = \dfrac{dC_{mk}(\epsilon_{mk})}{d\epsilon_{mk}} \left[\dfrac{\partial E_c^{(m)}}{\partial J\alpha} - \dfrac{\partial E_c^{(k)}}{\partial J\alpha} \right] & \\ \qquad = C'_{mk}[F_{J\alpha}^{(k)} - F_{J\alpha}^{(m)}] & \end{cases} \tag{7.10}$$

where $C'_{mk} = \frac{dC_{mk}(\epsilon_{mk})}{d\epsilon_{mk}}$ and $F_{J\alpha}^{(k,m)}$ is the α component of the total configurational force over particle J in the chemical state (k, m). Similarly to eq. (7.7), eq. (7.9) can be expressed as a matrix multiplication

$$F_{J\alpha}^{\text{EVB}} = -\sum_{m,k=1}^{N_F} \tilde{\Psi}_{\text{EVB}}^{(m)} \left(\frac{\partial H_{\text{EVB}}^{mk}}{\partial R_{J\alpha}} \right) \Psi_{\text{EVB}}^{(k)}. \tag{7.11}$$

The above equations define the standard EVB force field (EVB-FF). Even though the EVB formalism was first developed to compute molecular systems, EVB is also applicable to extended systems, customarily modelled using the supercell approximation and periodic boundary conditions (PBCs). However, the application of the EVB method to NPT ensembles requires the computation of the EVB stress tensor, which cannot be derived using the standard formulation [105]. To circumvent this limitation, we propose to make use of the well-known relation between the configurational energy and the configurational stress tensor [71]

$$\frac{\partial E_c^{(k)}}{\partial h_{\alpha\beta}} = -V \sum_{\gamma=x,y,z} \sigma_{\alpha\gamma}^{c(k)} h_{\beta\gamma}^{-1} \tag{7.12}$$

where h is the set of lattice vectors of the supercell with volume V=det(h). Multiplying to the left by $h_{\nu\beta}$ and summing over β we obtain the inverse relation to eq. (7.12)

$$\sigma_{\alpha\beta}^{c(k)} = -\frac{1}{V} \sum_{\gamma=x,y,z} h_{\beta\gamma} \frac{\partial E_c^{(k)}}{\partial h_{\alpha\gamma}} \tag{7.13}$$

which can be used to define the EVB stress tensor

$$\sigma_{\alpha\beta}^{\text{EVB}} = -\frac{1}{V} \sum_{\gamma=x,y,z} h_{\beta\gamma} \frac{\partial E_{\text{EVB}}}{\partial h_{\alpha\gamma}}. \tag{7.14}$$

Similar to the definition of the EVB force, we evaluate $\partial E_{\text{EVB}}/\partial h_{\alpha\gamma}$ using the eq. (7.5) and the Hellman-Feynman theorem

$$\frac{\partial E_{\text{EVB}}}{\partial h_{\alpha\beta}} = \langle \Psi_{\text{EVB}} | \frac{\partial \hat{H}_{\text{EVB}}}{\partial h_{\alpha\beta}} | \Psi_{\text{EVB}} \rangle. \tag{7.15}$$

The matrix components of the operator $\frac{\partial \hat{H}_{\text{EVB}}}{\partial_{h_{\alpha\beta}}}$ follow from the definition of the EVB matrix (7.1) and the use of relation (7.12)

$$\frac{\partial H_{\text{EVB}}^{mk}}{\partial h_{\alpha\beta}} = \begin{cases} \frac{\partial E_c^{(m)}}{\partial h_{\alpha\beta}} = -V \sum_\gamma \sigma_{\alpha\gamma}^{c(m)} h_{\beta\gamma}^{-1} & m = k \\ \frac{dC_{mk}}{\partial h_{\alpha\beta}} = \frac{dC_{mk}(\epsilon_{mk})}{d\epsilon_{mk}} \frac{\partial \epsilon_{mk}}{\partial h_{\alpha\beta}} & m \neq k \\ \qquad = \frac{dC_{mk}(\epsilon_{mk})}{d\epsilon_{mk}} \left[\frac{\partial E_c^{(m)}}{\partial h_{\alpha\beta}} - \frac{\partial E_c^{(k)}}{\partial h_{\alpha\beta}} \right] & \\ \qquad = -VC'_{mk} \sum_\gamma [\sigma_{\alpha\gamma}^{c(m)} - \sigma_{\alpha\gamma}^{c(k)}] h_{\beta\gamma}^{-1}. & \end{cases}$$

Finally, the EVB stress tensor of eq. (7.14) can be expressed as a matrix multiplication

$$\sigma_{\alpha\beta}^{\text{EVB}} = -\frac{1}{V} \sum_{\gamma=x,y,z} h_{\beta\gamma} \sum_{m,k=1}^{N_F} \tilde{\Psi}_{\text{EVB}}^{(m)} \left(\frac{\partial H_{\text{EVB}}^{mk}}{\partial h_{\alpha\beta}} \right) \Psi_{\text{EVB}}^{(k)}. \tag{7.16}$$

These expressions provide an alternative to compute the stress tensor σ^{EVB} from the configurational stress tensors of each non-reactive FF, $\sigma_{\alpha\gamma}^{c(k)}$. It is important to note that the presented scheme to compute σ^{EVB}

can only be derived if one uses functional forms for C_{mk} that depend on the energy differences ϵ_{mk}, for which one can evaluate $\frac{\partial E_c^{(m)}}{\partial h_{\alpha\beta}} - \frac{\partial E_c^{(m)}}{\partial h_{\alpha\beta}}$ and use relation (7.12) with the computed configurational stress tensor for each chemical state. In contrast, if the choice was to use coupling terms that do not depend on ϵ_{mk} but other degrees of freedom such as spatial coordinates, it is not clear how to derive an expression for σ^{EVB}.
Similarly to the stress tensor, the inability to compute individual contributions of the EVB force [105] prevents the evaluation of the virial using the standard formulation, and the usual decomposition of the virial depending of the type of interaction under consideration. Within the presented formalism, we compute the virial $\mathcal{V}_{\text{EVB}}$ from $\sigma_{\alpha\beta}^{\text{EVB}}$ as follows

$$\mathcal{V}_{\text{EVB}} = -\sum_{\alpha=x,y,z} \sigma_{\alpha\alpha}^{\text{EVB}}. \tag{7.17}$$

The instantaneous total stress tensor, σ^T, is given by the following general expression

$$\sigma^T = \sigma^{\text{kin}} + \sigma^{\text{EVB}} + \sigma^{\text{RB}} + \sigma^{\text{bc}} \tag{7.18}$$

where σ^{kin}, σ^{RB} and σ^{bc} are the contributions to the stress tensor from the kinetic energy, rigid bodies (RB) and bond constraints (bc), respectively. The EVB method only accounts for the configurational interactions, as described. The kinetic stress tensor is computed as usual from the instantaneous velocities of the particles. For a particle that is part of a rigid body, the only possible interactions are intermolecular non-bonded interactions (such as coulombic and van der Waals interactions) with other neighboring particles that are not part of the same rigid body. Following the computation of the EVB forces via eq. (7.9), the contribution to the stress from the rigid bodies is analogously to eq. (3.136)

$$\sigma_{\alpha\beta}^{\text{RB}} = \sum_{\mathcal{B}=1}^{N_{\text{RB}}} \sum_{I=1}^{\eta_{\mathcal{B}}} F_{I_{\mathcal{B}},\alpha}^{\text{EVB}} d_{I_{\mathcal{B}},\beta} \tag{7.19}$$

where $\vec{F}_{I_{\mathcal{B}}}$ is the total force over particle I of rigid body $\mathcal{B}$ and $\vec{d}_{I_{\mathcal{B}}}$ the vector distance from atom $I_{\mathcal{B}}$ to the center of mass of the rigid body $\mathcal{B}$. In the above expression, index $\mathcal{B}$ runs over all the rigid bodies. Each rigid body is composed of $\eta_{\mathcal{B}}$ particles. Since, by definition, the topology of rigid bodies remain unaltered during the simulation, the use of RBs within in the present framework is meaningful only to model the environment interacting reactive EVB site. A common example is the use of rigidly constrained water molecules to model a solution.
Contributions to the stress tensor from bond constraints, $\sigma_{\alpha\beta}^{\text{bc}}$, are obtained using the SHAKE/RATTLE algorithm (sec. 3.2) during the course of the simulation. This algorithm is independent of the EVB formalism, and corrects for the dynamics of the constrained particles. Finally, frozen particles do not contributed to the stress tensor and are not considered in the formalism. It is important to note that the topology defined via the setting of RBs, frozen atoms and bond constraints must be the consistent for all the coupled FFs, as they impose well defined conditions for the dynamics. For example, if a group of atoms form a rigid body, they must remain a rigid body independently of chemical state under consideration.

7.3 Calibrating EVB force fields

The quality of EVB for the description of reactive processes depends on the choice for the coupling terms C_{mk}, particularly to reproduce accurate interactions at the intermediate region between chemical states m and k where the change of chemistry occurs. For the implementation of the EVB method in DL_POLY_4, we have used functional forms C_{mk} that depend on the energy differences $\epsilon_{mk} = E_c^{(m)} - E_c^{(k)}$ to compute the stress tensor as described in Sec. 7.2. We have implemented two functional forms for the coupling terms. One is just setting the coupling term to be a constant:

$$C_{mk}(\epsilon_{mk}) = \mathcal{A}_{1,mk} \tag{7.20}$$

and the other possibility is to use Gaussian type of function,

$$C_{mk}(\epsilon_{mk}) = \mathcal{A}_{1,mk}\, e^{-\left(\frac{\epsilon_{mk}-\mathcal{A}_{2,mk}}{\mathcal{A}_{3,mk}}\right)^2} + \mathcal{A}_{4,mk}. \tag{7.21}$$

To determine the parameters for the coupling terms, it is necessary to consider a path that connects the reference geometries for states m and k. A convenient path is the minimum energy path (MPE) at zero-temperature, ζ_{mk}, obtained either via Density Functional Theory (DFT) or quantum chemistry (QC) methods to reproduce the change of chemistry between the states. The corresponding energy profile for this trajectory, $\tilde{E}_{\zeta_{mk}}$, is used as a reference, and the aim is to fit the coupling parameters such that E_{EVB} coincides with $\tilde{E}_{\zeta_{mk}}$ along ζ_{mk}. If we consider another state l, for example, it is expected that along ζ_{mk} the values for $E_c^{(l)}$ will be exceedingly large in comparison with $E_c^{(m)}$ and $E_c^{(k)}$ ($|\epsilon_{lk}| \gg 1$ and $|\epsilon_{lm}| \gg 1$), from which $C_{ml}(\epsilon_{ml}) \approx \mathcal{A}_{4,ml}$ and $C_{kl}(\epsilon_{kl}) \approx \mathcal{A}_{4,kl}$. One can initially set $\mathcal{A}_{4,kl} = \mathcal{A}_{4,ml} = 0$ for all $l \neq m, k$ and the coupling term C_{ml} is computed as follows

$$C^2_{mk}(\epsilon_{mk}) = \left[\tilde{E}_{\zeta_{mk}} - E^{(m)}_{c,\zeta_{mk}}\right]\left[\tilde{E}_{\zeta_{mk}} - E^{(k)}_{c,\zeta_{mk}}\right] \tag{7.22}$$

where $E^{(m)}_{c,\zeta_{mk}}$ and $E^{(k)}_{c,\zeta_{mk}}$ are the conformational energies for states m and k along ζ_{mk}, while ϵ_{mk} is in turn a implicit function of ζ_{mk}

$$\epsilon_{mk}(\zeta_{mk}) = E^{(m)}_{c,\zeta_{mk}} - E^{(k)}_{c,\zeta_{mk}}. \tag{7.23}$$

To find the parameters for the coupling C_{mk}, one has to plot the values obtained from eq. (7.22) as a function of ϵ_{mk} and fit the parameters using the functional form of eqs. (7.21) or (7.20), depending on the user's choice. This procedure is enough when coupling two FFs. For more than two fields, however, we have assumed $\mathcal{A}_{4,ln} = 0$ for $l \neq n \neq m, k$. Thus, in order to fit the parameters for the rest of the coupling terms of the EVB matrix, one should consider all the possible remaining MEPs between the states. For the pair l, p, for example, one can proceed in a similar way by setting all $\mathcal{A}_4$ elements to zero, but this time C_{mk} will not be necessarily zero. Depending on the number of coupled FFs, different but more complicated expressions like eq. (7.22) can be derived. Details are beyond the scope of this chapter.
The procedure to fit the coupling terms necessarily requires the use of force-fields i) consistent with the level of theory that is used to compute the explicit electronic problem for the reaction and ii) accurate enough far from the reference geometry for which they were fitted. Ultimately, meeting these requirements is a non-trivial challenge, particularly for large systems. We refer the user to Ref. [105] (and references therein) for a more detail discussion of the available strategies to calibrate EVB potentials.
Finally, depending on the non-reactive FF and the result from a DFT/QC simulation, one may want to shift the configuration energies $E_c^{(m)}$ by $\Delta E^{(m)}_{shift}$. This is particularly convenient to correct the relative energy between the involved chemical states. We have implemented this feature as input parameters in the SETEVB file (see Sec. 7.5).

7.4 Computational implementation

In the standard format, DL_POLY_4 reads the initial coordinates, velocities and forces from the CONFIG file. Each particle is labelled according to its specification in the FIELD file, which contains the information of the FF type and parameters for the interactions between the particles. Settings for the MD simulation are specified in the CONTROL file. Initially, the code was modified to allow i) reading multiple (N_F) CONFIG and FIELD files, ii) allocating arrays of dimension N_F for the relevant quantities, iii) checking consistency of specification between all force fields and initial coordinates (including any possible constraint such as rigid bodies), iv) reading EVB settings such as coupling terms and v) preventing the execution if there are MD or FF options that are not consistent with a EVB simulation. With regards to this last point, not all type of interactions in the energy decomposition of eq. (2.1) are suitable to describe reactive interactions. For example, three-body, four-body, Tersoff and metallic interactions are, by construction, not designed to account for different chemical states. Thus, such interactions should only be used to model the surrounding atomic environment interacting with the EVB site.
Regarding the EVB method in itself, modifications to the code required to allow for the computations of energies, forces, stress tensor and virials for each of the N_F force-fields separately. From the computed configurational energy of each FF and the choice of the functional forms for the coupling terms, the EVB

matrix (7.1) is built and diagonalized, and the lowest eigenvalue and the corresponding vector are assigned to E_{EVB} and Ψ_{EVB}, respectively. Matrix (7.10) is computed for each particle's Cartesian components and the resulting EVB force is obtained via the matrix multiplication of eq. (7.11). From the stress tensors computed for each FF, matrix (7.16) is built for all the $\alpha\beta$ terms and the $\alpha\beta$ component of the EVB stress tensor obtained via eq. (7.16), and the total virial from eq. (7.17). Such EVB calculations are conducted for each time step taking advantage of the domain decomposition as implemented in DL_POLY_4.
All the N_F force fields are computed in a loop architecture, i.e. one after the other, before being coupled via the EVB method. This means that all the available processors are used to compute each force-field, in contrast to the alternative strategy of dividing processors for each force field. For extended systems, this choice is convenient given the relative high computational cost of the long range Coulombic part in comparison with all the other contributions to the configurational energy. This loop structure increases the computational time by a multiplicative factor of approximately N_F with respect to the required time to compute only a single force field.

7.5 Setting EVB calculations

Setting input files and parameters for the EVB simulation of N_F coupled FFs in DL_POLY_4 requires of:
• the CONTROL file with the directive *evb* N_F
• N_F CONFIG files with the same ionic coordinates. The labelling of each atom in the CONFIG file must be consistent its FIELD file.
• N_F FIELD files with the interaction parameters to describe each of the coupled chemical states. Very important: the FFs descriptors for the reactive part of the potential must be specified before the descriptors for the non-reactive part.
• the SETEVB file.

To avoid problems, users are advised to check consistency between CONFIG and FIELD files for each of the chemical states separately, as for any standard simulation with DL_POLY_4. It is important to remark that the numbering and coordinates for all the atoms should be same of all CONFIG files, and all CONFIG files must have the same number of atoms. For example, atom 1 with tag A in CONFIG (labelling consistent with FIELD file) should be also atom 1 in CONFIG2, even though it might have a different tag B (labelling assigned in FIELD2).
The file SETEVB is compulsory for EVB simulations. For a EVB site described by N_F fields, the SETEVB file must contain all the settings specified via the structure details in Table 7.1.
The definition of the N_F values of *evbtypemols* in the SETEVB file requires of particular care. As described in table 7.1, these values indicate how many of the first defined type-of-molecules for each FIELD files are used to describe the EVB reactive site. To further clarify on this statement, let us consider a single EVB reactive unit interacting with non-reactive water molecules. Such a reactive unit is described by two-coupled FFs. In the chemical state 1, the reactive site is a single fragment described by the first type-of-molecule in the FIELD file, while the second type-of-molecule describes each of the surrounding water molecules. In the chemical state 2, the reactive site is composed of two molecular fragments, described by the first two type-of-molecules in the FIELD2 file, while now the third type-of-molecule describes the surrounding water. Consequently, *Molecular types* is set to 2 and 3 for files FIELD and FIELD2, respectively, and the specification in the SETEVB must be: *evbtypemols* 1 2.
The definition of constraints is only valid for atoms that are not part of the reactive EVB site. In addition, constraints must be kept consistent between FIELD files. For example, if a bond-constraint is set in the FIELD file for atoms X and Y, this bond-constraint should also be defined for the other FIELD files. Similarly with ridig-bodies, tethers, core-shells and frozen atoms. This requirement is crucial to ensure correctness in the dynamics of the system as forces over constrained atoms must be corrected to comply with the constraint. In case there is an inconsistency found, the code will abort the execution.
For the non-reactive part of the system (non-EVB atoms), it is also important to make sure that the specification for labels, mass and charges for all non-EVB atoms is the same for all FIELD files. Likewise, all

intermolecular (Tersoff, metallic, three-body, four-body), intramolecular (bond, angle, dihedral and inversion) and vdW interactions between these non-EVB atoms must be the same for all FIELD files. If any of these requirements is not fulfilled, DL_POLY_4 aborts the execution and print an error message that (hopefully) will guide the user to identify and fix the inconsistency.
Finally, the EVB implementation offers the possibility to restart the simulation, as N_F REVCON files are written. Analogous to the standard restart calculation, the user must copy the REVIVE file to REVOLD, while each REVCON (REVCON2, ..., REVCONN_F) file must be copied to the corresponding CONFIG (CONFIG2, ..., CONFIGN_F) file. To restart, the user must add the word *restart* in CONTROL file.
Additional points for further consideration:

- all FIELD files must have the same units
- Replay calculations are not allowed for EVB
- Simulations with four-body interactions are prevented
- external electric and magnetic fields are not possible within the EVB formalism (see ref. [105])

Table 7.1: Description for the settings of the SETEVB file

Setting	Description
evbtypemols	(Compulsory) Indicates how many of the first type-of-molecules specified in each of the N_F FIELD files are used to describe the EVB reactive site. See 7.5
evbcoupl	(Compulsory) Specifies the information for coupling parameters. Since $C_{mk} = C_{km}$ only $N_F(N_F-1)/2$ of these lines are needed. If the specification for any pair is repeated the simulation is stopped. The syntax for the specification is as follows, depending if one sets a coupling term to be a constant (*const*) or use the Gaussian functional form (*gauss*): FF-Pair / Type / Parameters (in units of the FIELD file) *evbcoul* m k *const* $\mathcal{A}_{1,mk}$ *evbcoul* m k *gauss* $\mathcal{A}_{1,mk}$ $\mathcal{A}_{2,mk}$ $\mathcal{A}_{3,mk}$ $\mathcal{A}_{4,mk}$ The order for m and k is irrelevant. Execution will stop if: • evbcoul is misspelled. • $m=k$, $m,k<1$ or $m,k>N_F$. • Input type different from *const* or *gauss*, misspelling including. • missing $\mathcal{A}_{mk}$ parameters. • the specification of any pair is repeated.
evbshift	(Compulsory) specifies the energy shift for a given FF. The syntax for this is specification as follows: FF / $\Delta E^{(m)}_{shift}$ (in units of the FIELD file) *evbshift* m value (either positive or negative) Execution will stop if: • evbshift is misspelled. • $m<1$ or $m>N_F$. • missing $\Delta E^{(m)}_{shift}$ parameters. • the specification for a given FF is repeated.
evbpop	(Optional) If present, the N_F computed values of $\|\Psi^{(k)}_{\mathrm{EVB}}\|^2$ are printed (index (k) for all FFs) at each time step in file POPEVB (only after equilibration). POPEVB is not overwritten upon *restart*.

As an illustrative example of the SETEVB file, we consider the case of a single reactive malonaldehyde molecule in non-reactive water.

```
#Coupling terms for EVB calculations
evbtypemols   1   1
evbcoupl      1   2     const 49.0 # In units of kcal/mol
evbshift      1   0.0   # In units of kcal/mol
evbshift      2   0.0   # In units of kcal/mol
```

In this case, we have two possible conformations for the molecule, each conformation described by a different FF (see section 6 of ref. [105]). In contrast, the FF for the water molecules is the same independently of the malonaldehyde conformation. As per guidance above, the FIELD files must first specify the FF descriptors for the malonaldehyde molecule, followed by the FF descriptor for the surrounding water. For both FIELDS we have 2 types of molecules, the malonaldehyde molecule for the first type and N water molecules for the second type. Therefore, directive *evbcoupl* must be set to 1 1. For the EVB coupling, we must specify the *evbcoul* with the involved fields (1 and 2 in this case), the type of coupling (*const*) and the parameter (49.0) in units of the FIELD files. Finally, in the present case, both conformations for malonaldehyde are energetically equivalent and the energy shift *evb* must be the same for both directives $evbshift$. Depending on the used potentials and the system to be computed, one may want to introduce an asymmetry in the FFs.

Chapter 8

Construction and Execution

Scope of Chapter

This chapter describes how to compile a working version of DL_POLY_4 and run it.

8.1 Constructing DL_POLY_4: an Overview

8.1.1 Constructing the Standard Versions

DL_POLY_4 was designed as a package of useful subroutines rather than a single program, which means that users are to be able to construct a working simulation program of their own design from the subroutines available, which is capable of performing a specific simulation. However, we recognise that many, perhaps most, users will be content with creating a standard version that covers all of the possible applications with DL_POLY_4 native functionalities and may be a few external ones for this reason we have only provided the necessary tools to assemble such a version. The methods of creating the standard versions is described in detail in this chapter, however a brief step-by-step description follows.

1. DL_POLY_4 is supplied as a ZIP compressed file. This must uncompressed to create the DL_POLY_4 directory (Section 1.4).
2. The next step is to compile DL_POLY_4 either by using the traditional Makefiles or by creating OS customised ones using **cmake**. In either case **make** will be used to produce a binary executable (see Section 8.2), which as a default will be named DLPOLY.Z and located in the *execute* subdirectory if compiling in traditional mode or in the place of using **cmake**.
3. DL_POLY_4 also has a Java GUI. The files for this are stored in the subdirectory *java*. Compilation of this is simple and requires running the javac compiler and the jar utility. Details for these procedures are provided in the GUI manual [21].
4. To run the executable for the first time you require the files CONTROL, FIELD and CONFIG; and possibly a few tabulated files (TABLE, TABEAM, etc.). These must be present in the directory from which the program is executed. (See Section 10.1 for the description of the input files.)
5. Executing the program will produce the files OUTPUT, STATIS, REVCON and REVIVE; and optionally a constellation of others in the executing directory depending on user options in CONTROL (HISTORY, RDFDAT, ZDNDAT, MSDTMP, REFERENCE, DEFECTS, etc.). (See Section 10.2 for the description of the output files.)

This simple procedure is enough to create a standard version to run most simulations. There may however be some difficulty with array sizes. DL_POLY_4 contains features which allocate arrays after scanning the input files for a simulation. Sometimes these initial estimates are insufficient for a long simulation when, for example, the system volume changes markedly during the simulation or when a system is artificially constructed to have a non-uniform density. Usually, simply restarting the program will cure the problem, but sometimes, especially when the local atom density is somewhat higher than the global one or the system undergoes some form of clustering and the distribution of bonded-like interactions is far from uniform, it may be necessary to amend the array sizes in accordance with the error message obtained. To trigger lengthening of the density dependent global arrays the user may use the **densvar** option in the CONTROL (Section 10.1.1) file. However, lengthening these arrays will require a larger amount of memory from the execution machine for the simulation, which it may not be able to provide. See Section 11.2.2 for more insight on the DL_POLY_4 source code structure.

8.1.2 Constructing Non-standard Versions

In constructing a non-standard DL_POLY_4 simulation program, the first requirement is for the user to write a program to function as the root segment. The root segment /VV/DL_POLY is placed in the *source* directory and contains the set-up and close-down calls for a molecular dynamics simulation. It is the routine that first opens the OUTPUT file (Section 10.2), which provides the summary of the job. The root program calls the "molecular dynamics cycle" routines implementing the VV. These routines contain major routines required to perform the simulation, control the normal "molecular dynamics cycle" and monitor the *cpu* and *memory* usage. They also bring about a controlled termination of the program if the *cpu* usage approaches the allotted job time within a pre-set closure time and/or if the *memory* usage approaches the allocated limit for density dependent arrays. Users are recommended to study the aforementioned root directories as a model for other implementations of the package they may wish to construct. Some advise on hierarchies of all the DL_POLY_4 subroutines can be found in Section 11.2.2.

Should additional functionality be added to DL_POLY_4 by the user, the SET_BOUNDS routine (and its support subroutines) may need modifying to allow specification of the dimensions of any new arrays.

Any molecular dynamics simulation performs five different kinds of operation: initialisation; forces calculation; integration of the equations of motion; calculation of system properties; and job termination. It is worth considering these operations in turn and to indicate which DL_POLY_4 routines are available to perform them. We do not give a detailed description, but provide only a guide. Readers are recommended to examine the different routines described in the DL_POLY_4 User Manual for further details (particularly regarding further dependencies i.e. additional routines that may be called).

The following outline assumes a system containing flexible molecules held together by rigid bonds.

Initialisation requires firstly that the program determine what platform resources are made available to the specific simulation job. This is done by the DL_POLY_4 routine MAP_DOMAINS in DOMAINS_MODULE that attempts to allocate and map the resources (nodes in parallel) in compliance with the DD strategy. MAP_DOMAINS is called within the routine SET_BOUNDS, which also sets the necessary limits for various simulation array sizes and all global variables as declared in SETUP_MODULE to convenient values based on a rough scan through the CONFIG, CONTROL, FIELD and optionally TABLE and TABEAM (Section 10.1) files. The routine also calls the READ_CONFIG routine to obtain atomic positions and optionally velocities and forces from the CONFIG file. After allocation of all necessary simulation arrays and variables (with compulsory initialisation to "zero" value), the job control information is required; this is obtained by the routine READ_CONTROL, which reads the CONTROL file. The description of the system to be simulated – the types of atoms and molecules present and the intermolecular forces – are obtained by the READ_FIELD routine, which reads the FIELD file. The SYSTEM_INIT routine is called next to initialise various simulation arrays and variables with the data available so far and detects if the job is a restart of previous simulation run. If so it reads the REVOLD (Section 10.1.6) to supply some arrays and variables with the necessary values as saved from the previous job. The domain halo is constructed immediately afterwards by the routine SET_HALO_PARTICLES. After gathering all these data, bookkeeping and exclusion arrays are created for the intramolecular and site related interactions (core-shell, constraint and tether units) by the BUILD_BOOK_INTRA and BUILD_EXCL_INTRA routines. Lastly, the thermodynamic properties of the system are checked and set by the SET_TEMPERATURE routine (which also generates the initial velocities if required to do so).

The calculation of the pair-like forces is carried out in the TWO_BODY_FORCES routine and represents the main part of any simulation. For calculation of the two-body contributions to the atomic forces, the Verlet neighbour list is constructed by the LINK_CELL_PAIRS routine using link-cell lists. Special measures are taken so that the list excludes: (i) pairs of atoms that are both in a *frozen state* as well as (ii) pairs in which one of the atoms has the other in its *exclusion list*. The last is built by BUILD_EXCL_INTRA where the specifications of bond-like interactions in the FIELD file are processed. Various other subroutines are then called to calculate specific contributions by different interactions. For example; VDW_FORCES for the short-range (van der Waals) forces (Section 2.3.1), METAL_LRC, METAL_LD_COMPUTE and METAL_FORCES for the metal interactions (Section 2.3.2), and EWALD_SPME_FORCES, EWALD_REAL_FORCES, EWALD_FRZN_FORCES and EWALD_EXCL_FORCES for the Coulombic forces (Section 2.4).

Higher order intermolecular, site-related and intramolecular forces require the routines
TERSOFF_FORCES, THREE_BODY_FORCES, FOUR_BODY_FORCES,
CORE_SHELL_FORCES or CORE_SHELL_RELAX, TETHERS_FORCES,
BONDS_FORCES, ANGLES_FORCES, DIHEDRALS_FORCES and INVERSIONS_FORCES.
The routines

EXTERNAL_FIELD_APPLY and EXTERNAL_FIELD_CORRECT are required if the simulated system has an external force field (e.g. electrostatic field) operating.

To help with equilibration simulations, routines such as CAP_FORCES, ZERO_K_OPTIMISE and MINIMISE_RELAX are sometimes required to reduce the magnitude of badly equilibrated forces and to steer the MD system towards an equilibrium state.

Integration of the equations of motion is handled by one of the routines listed and described in Chapter 3.

As mentioned elsewhere, DL_POLY_4 does not contain many routines for computing system properties during a simulation. Radial distributions may be calculated however, by using the routines RDF_COLLECT, RDF_EXCL_COLLECT, RDF_FRZN_COLLECT a and RDF_COMPUTE. Similarly, Z-density distributions may be calculated by using the routines Z_DENSITY_COLLECT and Z_DENSITY_COMPUTE, while velocity autocorrelation functions may be calculated using the routines VAF_COLLECT and VAF_COMPUTE. Ordinary thermodynamic quantities are calculated by the routine STATISTICS_COLLECT, which also writes the STATIS file (Section 10.2.17). Routine TRAJECTORY_WRITE writes the HISTORY (Section 10.2.1) file for later (postmortem) analysis. Routine DEFECTS_WRITE writes the DEFECTS (Section 10.2.3) file for later (postmortem) analysis. Routine MSD_WRITE writes the MSDTMP (Section 10.2.2) file for later (postmortem) analysis. Routine RSD_WRITE writes the RSDDAT (Section 10.2.4) file for later (postmortem) analysis.

Job termination is handled by the routine STATISTICS_RESULT which writes the final summaries in the OUTPUT file and dumps the restart files REVIVE and REVCON (Sections 10.2.10 and 10.2.9 respectively).

8.2 Compiling and Running DL_POLY_4

When you have obtained DL_POLY_4 from Daresbury Laboratory and unpacked it, your next task will be to compile it.

CMake - http://www.cmake.org/ - is an open-source, cross-platform family of tools designed to build, test and package software. CMake is used to control the software compilation process using simple platform and compiler independent configuration files, and generate native makefiles and workspaces that can be used in a compiler environment of your choice. The suite of CMake tools was created by Kitware in response to the need for a powerful, cross-platform build environment for open-source projects such as ITK, VTK, KDE, etc...

In order to build a DL_POLY_4 executable with cmake there are two stages - **Stage 1**: generating the build files (e.g. makefiles); and **Stage 2**: building the code (e.g. make).

For the examples within this section we assume that DL_POLY_4 has been downloaded, the archived contents extracted and we are stepped in the the main folder (root). All commands and discussions that follow are relative to this root folder of DL_POLY_4.

Full instructions can be also found online at `https://ccp5.gitlab.io/dlpoly-setup/`

- **Stage 1**: Configure and generate the build files.

  One can pass different options to the build system to generate the build files. It is important to determine these before one choses the ones they consider relevant for their purposes. Finding out all available options for DL_POLY_4 can be done in the following ways:

  a) by using *cmake*:

```
[10:38:02 alin@abaddon:~/playground/dl-poly]: mkdir myBuild
[10:38:11 alin@abaddon:~/playground/dl-poly]: cd myBuild/
[10:38:13 alin@abaddon:~/playground/dl-poly/myBuild]: cmake .. -L
....
-- Cache values
```

```
BUILDER:STRING=
BUILD_SHARED_LIBS:BOOL=OFF
BUILD_TESTING:BOOL=OFF
CMAKE_BUILD_TYPE:STRING=
CMAKE_INSTALL_PREFIX:PATH=/usr/local
DOCS:BOOL=OFF
HOST:STRING=abaddon.dl.ac.uk
MPI_NPROCS:STRING=8
WITH_COVERAGE:BOOL=OFF
WITH_EXTRATIME:BOOL=OFF
WITH_FORCHECK:BOOL=OFF
WITH_KIM:BOOL=OFF
WITH_MPI:BOOL=ON
WITH_NETCDF:BOOL=OFF
WITH_OPENMP:BOOL=OFF
WITH_PHI:BOOL=OFF
WITH_PLUMED:BOOL=OFF
```

Do notice that some of the options are already filled in... *cmake* already tried its best to find some of the available information about your system and picked the some defaults, a.g. **WITH_MPI:BOOL=ON**.

b) by reading the DL_POLY_4 option file:
All available options and their description are stored in CMAKE/DLPOLYBUILDOPTIONS.CMAKE.

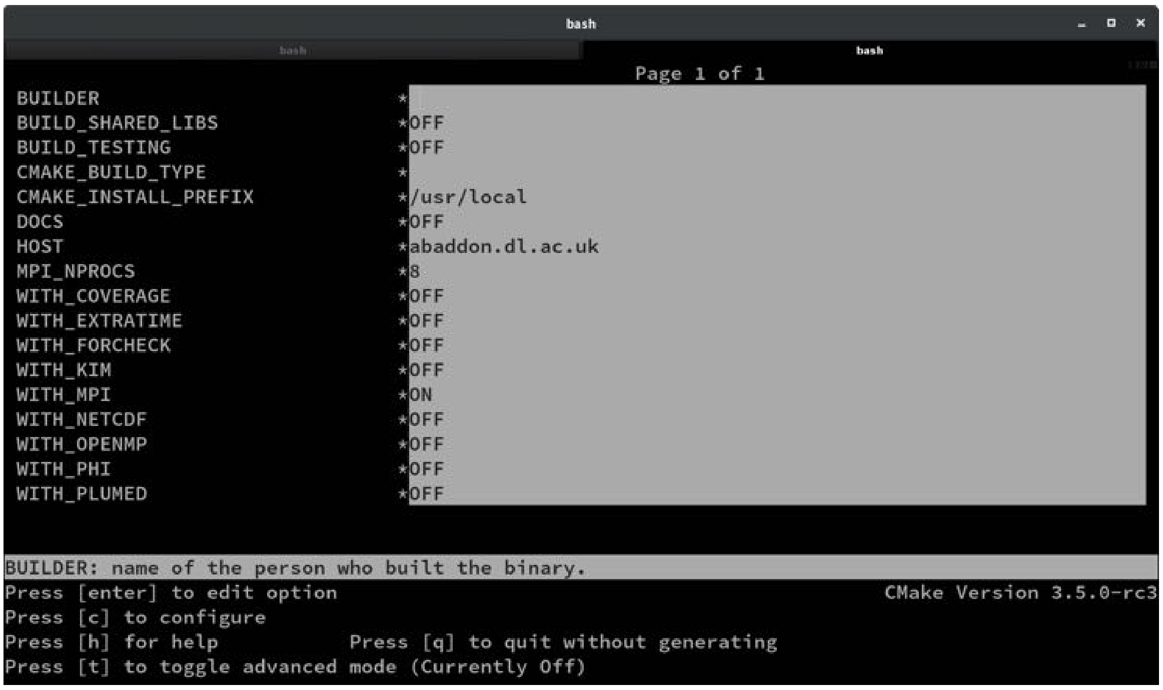
```
[10:46:01 alin@abaddon:~/playground/dl-poly]: cat cmake/DLPOLYBuildOptions.cmake
option(WITH_MPI "build a MPI version" ON)
option(WITH_OPENMP "build an OpenMP version" OFF)
option(WITH_PHI "build an executable for Xeon Phi version" OFF)
option(WITH_NETCDF "build using netcdf support version" OFF)
option(WITH_EXTRATIME "activate extra timing information" OFF)
option(WITH_KIM "Build with KIM support" OFF)
option(WITH_PLUMED "Build with PLUMED support" OFF)
option(BUILD_TESTING "Build with Testing support" OFF)
option(WITH_COVERAGE "Build with instrumentation for code coverage" OFF)
option(WITH_FORCHECK "Build with forcheck for code" OFF)
option(DOCS "Doxygen Documentation" OFF)
option(BUILD_SHARED_LIBS "Build with shared libraries" OFF)

set(MPI_NPROCS 8 CACHE STRING "number of MPI processes to be used for code
   coverage and tests")
cmake_host_system_information(RESULT AH QUERY FQDN)
set(HOST "${AH}" CACHE STRING "name of the hostname we build on.")
set(BUILDER "" CACHE STRING "name of the person who built the binary.")
```

c) by using *ccmake*:
Press <c> to configure and <e> if errors appears. A typical output is shown in Figure 8.1.

```
[10:38:02 alin@abaddon:~/playground/dl-poly]: mkdir myBuild
[10:38:11 alin@abaddon:~/playground/dl-poly]: cd myBuild/
[10:38:13 alin@abaddon:~/playground/dl-poly/myBuild]: ccmake ..
```

d) by using *cmake-gui*:
A typical output is shown in Figure 8.2.

```
[10:38:02 alin@abaddon:~/playground/dl-poly]: mkdir myBuild
```



```
                                        bash
              bash                                      bash
                                           Page 1 of 1
 BUILDER                          *
 BUILD_SHARED_LIBS                *OFF
 BUILD_TESTING                    *OFF
 CMAKE_BUILD_TYPE                 *
 CMAKE_INSTALL_PREFIX             */usr/local
 DOCS                             *OFF
 HOST                             *abaddon.dl.ac.uk
 MPI_NPROCS                       *8
 WITH_COVERAGE                    *OFF
 WITH_EXTRATIME                   *OFF
 WITH_FORCHECK                    *OFF
 WITH_KIM                         *OFF
 WITH_MPI                         *ON
 WITH_NETCDF                      *OFF
 WITH_OPENMP                      *OFF
 WITH_PHI                         *OFF
 WITH_PLUMED                      *OFF


BUILDER: name of the person who built the binary.
Press [enter] to edit option                                 CMake Version 3.5.0-rc3
Press [c] to configure
Press [h] for help           Press [q] to quit without generating
Press [t] to toggle advanced mode (Currently Off)
```

Figure 8.1: Typical ccmake output for DL_POLY_4

```
[10:38:11 alin@abaddon:~/playground/dl-poly]: cd myBuild/
[10:38:13 alin@abaddon:~/playground/dl-poly/myBuild]: cmake-gui ..
```

One may also choose to pass the command line options via **-DOPTION=value**. Explicit compiler specification can be achieved by setting the environment variable **FC** (e.g. using Intel ifort **FC=ifort**). Compiler flags can be altered via **FFLAGS**, (e.g. **FFLAGS="-O3 -xHost"**). Once one is happy with the choices made then they are ready to move to Stage 2.

- **Stage 2**: Build the executable.

  Building is as simple as typing *make -jX*, where X is the number of desired compilation threads to work in parallel. Once the build process is successful one can find the DL_POLY_4 executable in the folder *bin* (freshly generated if it did not exists before). One can then copy or link the executable to any accessible to them place on the system they wish.

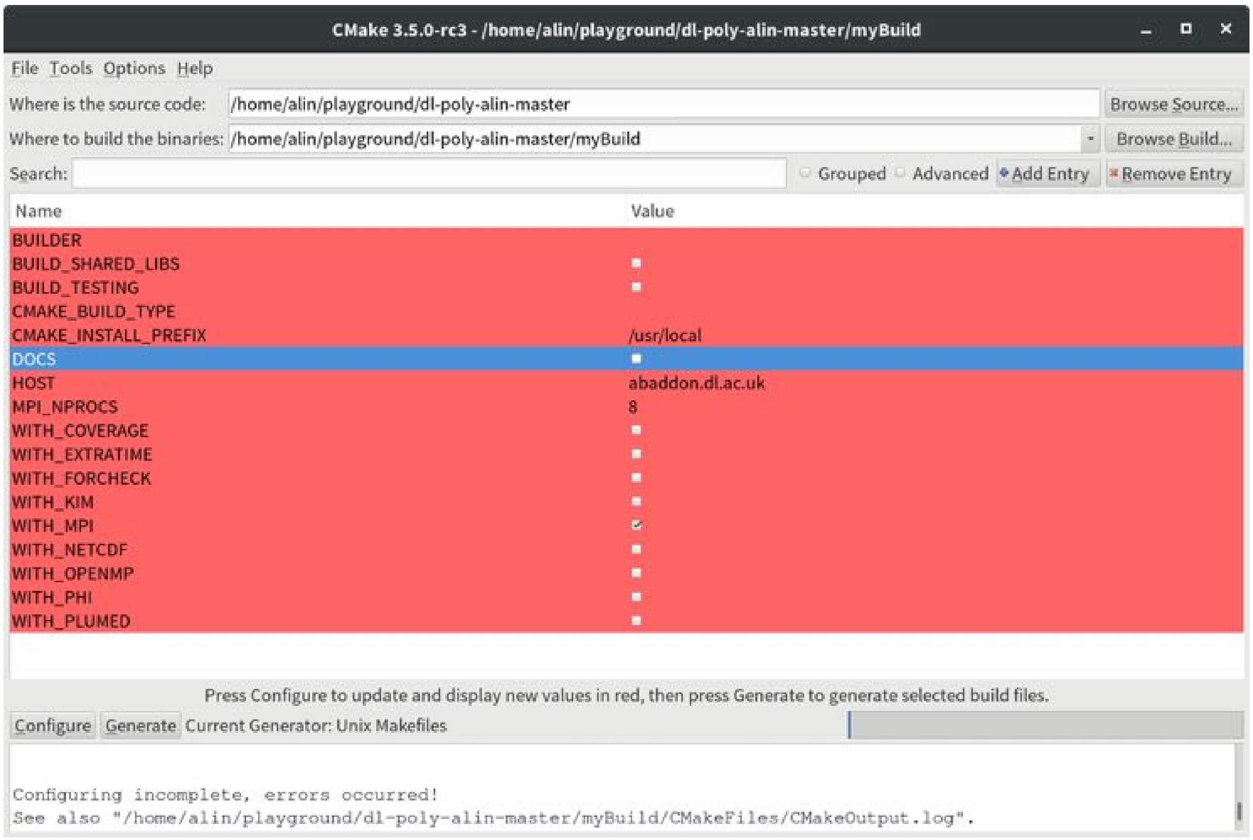
Examples of different builds. Assuming that the OS environment is set up properly so that access paths to the MPI library (and any other needed libraries) and gfortran is the default FORTRAN90 compiler.

- **Pure MPI**

```
[alin@abaddon: ...dl-poly]: mkdir myBuild
[alin@abaddon: ...dl-poly]: cd myBuild/
[alin@abaddon: ...dl-poly/myBuild]: FFLAGS="-O3 -mtune=native" cmake ..
[alin@abaddon: ...dl-poly/myBuild]: make -j10
```

- **Serial**

```
[alin@abaddon: ...dl-poly]: mkdir myBuild
[alin@abaddon: ...dl-poly]: cd myBuild/
```



Where is the source code: /home/alin/playground/dl-poly-alin-master

Where to build the binaries: /home/alin/playground/dl-poly-alin-master/myBuild

Search:

Name	Value
BUILDER	
BUILD_SHARED_LIBS	☐
BUILD_TESTING	☐
CMAKE_BUILD_TYPE	
CMAKE_INSTALL_PREFIX	/usr/local
DOCS	☐
HOST	abaddon.dl.ac.uk
MPI_NPROCS	8
WITH_COVERAGE	☐
WITH_EXTRATIME	☐
WITH_FORCHECK	☐
WITH_KIM	☐
WITH_MPI	✓
WITH_NETCDF	☐
WITH_OPENMP	☐
WITH_PHI	☐
WITH_PLUMED	☐

Press Configure to update and display new values in red, then press Generate to generate selected build files.

Current Generator: Unix Makefiles

```
Configuring incomplete, errors occurred!
See also "/home/alin/playground/dl-poly-alin-master/myBuild/CMakeFiles/CMakeOutput.log".
```

Figure 8.2: Typical cmake-gui output for DL_POLY_4

```
[alin@abaddon: ...dl-poly/myBuild]: FFLAGS="-O3 -mtune=native" cmake .. -DWITH_MPI=
    OFF
[alin@abaddon: ...dl-poly/myBuild]: make -j10
```

- **Pure MPI+NETCDF**

```
[alin@abaddon: ...dl-poly]: mkdir myBuild
[alin@abaddon: ...dl-poly]: cd myBuild/
[alin@abaddon: ...dl-poly/myBuild]: FFLAGS="-O3 -mtune=native" cmake .. -
    DWITH_NETCDF=ON
[alin@abaddon: ...dl-poly/myBuild]: make -j10
```

- **Pure MPI+KIM**

```
[alin@abaddon: ...dl-poly]: mkdir myBuild
[alin@abaddon: ...dl-poly]: cd myBuild/
[alin@abaddon: ...dl-poly/myBuild]: FFLAGS="-O3 -mtune=native" cmake .. -DWITH_KIM=
    ON
[alin@abaddon: ...dl-poly/myBuild]: make -j10
```

- **Pure MPI+PLUMED**

```
[alin@abaddon: ...dl-poly]: mkdir myBuild
[alin@abaddon: ...dl-poly]: cd myBuild/
[alin@abaddon: ...dl-poly/myBuild]: FFLAGS="-O3 -mtune=native" cmake .. -
    DWITH_PLUMED=ON
[alin@abaddon: ...dl-poly/myBuild]: make -j10
```

- **Pure MPI+Testing**

  ```
  [alin@abaddon: ...dl-poly]: mkdir myBuild
  [alin@abaddon: ...dl-poly]: cd myBuild/
  [alin@abaddon: ...dl-poly/myBuild]: FFLAGS="-O3 -mtune=native" cmake .. -
     DBUILD_TESTING=ON
  [alin@abaddon: ...dl-poly/myBuild]: make -j10
  ```

  To run the tests one can run *make test* or *ctest*. For a complete list of tests available, run *ctest -N*. If specific test is desired to run then use *ctest -R TESTNAME*.

For a list of complete options to build use *make help*. The list will be different depending on the options used to configure.

8.2.1 Note on the Interpolation Scheme

In DL_POLY_4 two-body-like contributions (van der Waals, metal and real space Ewald summation) to energy and force are evaluated by interpolation of tables constructed at the beginning of execution. The DL_POLY_4 interpolation scheme is based on a 3-point linear interpolation in r. **Note** that a 5-point linear interpolation in r is ised in DL_POLY_4 for interpolation of the EAM (metal) forces from EAM table data (TABEAM).

The number of grid points (`mxgrvdw`) required for interpolation in r to give good energy conservation in a simulation is:

$$\texttt{mxgrid} = \texttt{Max}(\texttt{mxgrid}, 1004, \texttt{Nint}(r_{\texttt{cut}}/\delta r_{\texttt{max}}) + 4) \quad ,$$

where $r_{\texttt{cut}}$ is the main cutoff beyond which the contributions from the short-range-like interactions are negligible, and $\delta r_{\texttt{max}} = 0.01$ Å. is the default grid bin for real space grids.

8.2.2 Running

To run the DL_POLY_4 executable (DLPOLY.Z) you will initially require at least three input data files, which you must provide in the *execute* sub-directory, (or whichever sub-directory you will execute the run). The first of these is the CONTROL file (Section 10.1.1), which indicates to DL_POLY_4 what kind of simulation conditions you want to run, how much data you want to gather and for how long you want the job to run. The second file you need is the CONFIG file (Section 10.1.2). This contains the atom positions and, depending on how the file was created (e.g. whether this is a configuration created from 'scratch' or the end point of a previous run), the velocities and forces also. The third file required is the FIELD file (Section 10.1.3), which specifies the atomic properties (such as charge and mass), molecular stoichiometry and intermolecular topology and interactions, and finally intermolecular interactions and external fields. Sometimes one or a few more extra files may also be required: MPOLES (Section 10.1.4) - which contains the specification of higher order charge distributions (multipolar momenta); TABLE (Section 10.1.7), TABEAM (Section 10.1.8), TABBND, TABANG, TABDIH and TABINV Files (Section 10.1.9); which contain potential and force arrays for particular type of interaction that is not supplied with an explicit analytical for in DL_POLY_4 (usually because they are too complex, e.g. spline potentials, , non-analytic functionals as in TEABEAM, etc.). Other optional files may also be required such as REFERENCE (Section 10.1.5) - similar to the CONFIG file it contains the "perfect" crystalline structure of the system used as a reference to detect instantaneous interstitial and vacancy defects during radiation damage events, HISTORY (Section 10.2.1) - used for replaying a previously generated trajectory so that various observables could be recreated (e.g. RDF, Z-density profiles, etc.).

Examples of input files are found in the *data* sub-directory, which can be copied into the *execute* subdirectory using the *select* macro found in the *execute* sub-directory.

A successful run of DL_POLY_4 will generate several data files, which appear in the *execute* sub-directory. The most obvious one is the file OUTPUT (Section 10.2.6), which provides an effective summary of the job run: the input information; starting configuration; instantaneous and rolling-averaged thermodynamic data; minimisation information, final configurations; radial distribution functions (RDFs); Z-density profiles and job timing data. The OUTPUT file is human readable. Also present will be the restart files REVIVE (Section 10.2.10) and REVCON (Section 10.2.9). REVIVE contains the accumulated data for a number of thermodynamic quantities and statistical accumulators (RDFs, fluctuations, etc.), and is intended to be used as the input file for a following run. It is *not* human readable. The REVCON file contains the *restart configuration*, i.e. the final positions, velocities and forces of the atoms when the run ended and is human readable. The STATIS file (Section 10.2.17) contains a catalogue of instantaneous values of thermodynamic and other variables, in a form suitable for temporal or statistical analysis.

There are quite a few other optional files, for which more detailed description and formating can be found in the relevant Section 10.2. It is, however, worth mentioning that these are generated upon specific user instructions in CONTROL, so their specific functional description and activation information detail can be found in Section 10.1.1.

8.2.3 Parallel I/O

Many users that have suffered loss of data in the OUTPUT, especially running in parallel and when an error occurs on parallel architectures. In such circumstances the OUTPUT may be empty or incomplete, despite being clear that the actual simulation has progressed well beyond what has been printed in OUTPUT. Ultimately, this is due to OS's I/O buffers not being flushed as a default by the particular OS when certain kind of errors occurs, especially MPI related. The safest way to avoid loss of information in such circumstances is to write the OUTPUT data to the default output channel (”the screen”). There is an easy way to do this in DL_POLY_4, which is to use the **l_scr** keyword in the CONTROL file. The batch daemon will then place the output in the standard output file, which can then be of use to the user, or alternatively on many batch systems the output can be redirected into another file, allowing an easier following of the job progress over time. This latter technique is also useful on interactive systems where simply printing to the screen could lead to large amounts of output. However, such situations could be easily avoided by redirecting the output using the ”>” symbol, for instance: ”mpirun -n 4 DLPOLY.Z > OUTPUT”.

It is also worth noting that the use of large batch and buffer numbers can speed up enormously the performance of the parallel I/O, for example putting in CONTROL (see Section 10.1.1):

```
io read mpiio         128 10000000 1000000
io write mpiio        512 10000000 1000000
```

at large processor count jobs (over 1000). However, this help comes at a price as larger batches and buffers also requires more memory. So at smaller processor counts the job will abort at the point of trying to use some of the allocated arrays responsible for these.

More information about DL_POLY_4 parallel I/O can be found in the following references [107, 108, 109].

8.2.4 Restarting

The best approach to running DL_POLY_4 is to define from the outset precisely the simulation you wish to perform and create the input files specific to this requirement. The program will then perform the requested simulation, but may terminate prematurely through error, inadequate time allocation or computer failure. Errors in input data are your responsibility, but DL_POLY_4 will usually give diagnostic messages to help you sort out the trouble. Running out of job time is common and provided you have correctly specified the job time variables (using the **close time** and **job time** directives - see Section 10.1.1) in the CONTROL file, DL_POLY_4 will stop in a controlled manner, allowing you to restart the job as if it had not been interrupted.

To restart a simulation after normal termination you will again require the original CONTROL file (*augment it to include the* **restart** *directive and/or extend the length and duration of the new targeted MD run*), the FIELD (and TABLE and/or TABEAM) file, and a CONFIG file, which is the exact copy of the REVCON file created by the previous job. You will also require a new file: REVOLD (Section 10.1.6), which is an exact copy of the previous REVIVE file. If you attempt to restart DL_POLY_4 without this additional file available, the job will most probably fail. **Note** that DL_POLY_4 will append new data to the existing STATIS and HISTORY files if the run is restarted, other output files will be **overwritten**.

In the event of machine failure, you should be able to restart the job in the same way from the surviving REVCON and REVIVE files, which are dumped at regular intervals to meet just such an emergency. In this case check carefully that the input files are intact and use any extra files; such as STATIS, HISTORY, etc.; with caution - there may be duplicated, mangled or missing records. The reprieve processing capabilities of DL_POLY_4 are not foolproof - the job may crash while these files are being written from memory to disk on a parallel architecture for example, but they can help a great deal. You are advised to keep backup copies of these files, noting the times they were written, to help you avoid going right back to the start of a simulation.

You can also extend a simulation beyond its initial allocation of timesteps, provided you still have the REVCON and REVIVE files. These should be copied to the CONFIG and REVOLD files respectively and the directive **timesteps** adjusted in the CONTROL file to the new total number of steps required for the simulation. For example if you wish to extend a 10000 step simulation by a further 5000 steps use the directive **timesteps 15000** in the CONTROL file and include the **restart** directive.

Further to the full restart option, there is an alternative **restart scale** directive that will reset the temperature at start or **restart noscale** that will keep the current kinetics intact. /bf Note that these two options are not correct *restart*s but rather modified *start*s as they make no use of REVOLD file and will reset internal accumulators to zero at start.

Note that all these options are mutually exclusive!

If none of the restart options is specified velocities are generated anew with Gaussian distribution of the target kinetic energy based on the provided temperature in the CONTROL file.

8.2.5 Optimising the Starting Structure

The preparation of the initial structure of a system for a molecular dynamics simulation can be difficult. It is quite likely that the structure created does not correspond to one typical of the equilibrium state for the required state point, for the given force field employed. This can make the simulation unstable in the initial stages and can even prevent it from proceeding.

For this reason DL_POLY_4 has available a selection of structure relaxation methods. Broadly speaking, these are energy minimisation algorithms, but their role in DL_POLY_4 is not to provide users with true structural optimisation procedures capable of finding the ground state structure. They are simply intended to help users improve the quality of the starting structure prior to a statistical dynamical simulation, which implies usage during the equilibration period only!

The available algorithms are:

1. ‘Zero’ temperature molecular dynamics. This is equivalent to a dynamical simulation at low temperature. At each time step the molecules move in the direction of the computed forces (and torques), but are not allowed to acquire a velocity larger than that corresponding to a temperature of 10 Kelvin. The subroutine that performs this procedure is ZERO_K_OPTIMISE.

2. Conjugate Gradients Method (CGM) minimisation. This is nominally a simple minimisation of the system configuration energy using the conjugate gradients method [80]. The algorithm coded into DL_POLY_4 is an adaptation that allows for rotation and translation of rigid bodies. Rigid (constraint) bonds however are treated as stiff harmonic springs - a strategy which we find does allow the bonds

to converge within the accuracy required by SHAKE. The subroutine that performs this procedure is MINIMISE_RELAX which makes use of, MINIMISE_MODULE.

3. 'Programmed' energy minimisation, involving both MD and CGM. This method combines the two as minimisation is invoked by user-defined intervals of (usually low temperature) dynamics, in a cycle of minimisation - dynamics - minimisation etc., which is intended to help the structure relax from overstrained conditions (see Section 10.1.1). When using the programmed minimisation DL_POLY_4 writes (and rewrites) the file CFGMIN 10.2.5, which represents the lowest energy structure found during the programmed minimisation. CFGMIN is written in CONFIG file format (see section 10.1.2) and can be used in place of the original CONFIG file.

It should be noted that none of these algorithms permit the simulation cell to change shape. It is only the atomic structure that is relaxed. After which it is assumed that normal molecular dynamics will commence from the final structure.

Also worth noting is that some dynamics related options can be used in assistance with the optimisation algorithms to reach a better state (close to equlibrium) faster. Such would be the force capping (the **cap** option applied at each step) and velocity rescaling (the **scale** option) at regular intervals of steps that will only be applied during equlibration. Last but not least using any combination of these will be otimal and safe only with *safe* integrators. For *safe* equlibration work we strongly recommend only NVE and Berendsen couched NV/P/σT as appropriate! In the case of liquid and soft matter systems NσT Berendsen integrator must be either avoided or used with **orth** or **orth semi** constraints!

Notes on the Minimisation Procedures

1. The zero temperature dynamics is really dynamics conducted at 10 Kelvin. However, the dynamics has been modified so that the velocities of the atoms are always directed along the force vectors. Thus the dynamics follows the steepest descent to the (local) minimum. From any given configuration, it will always descend to the same minimum.

2. The conjugate gradient procedure has been adapted to take account of the possibilities of constraint bonds and rigid bodies being present in the system. If neither of these is present, the conventional unadapted procedure is followed.

   (a) In the case of rigid bodies, atomic forces are resolved into molecular forces and torques. The torques are subsequently transformed into an equivalent set of atomic forces which are perpendicular both to the instantaneous axis of rotation (defined by the torque vector) and to the cylindrical radial displacement vector of the atom from the axis. These modified forces are then used in place of the original atomic forces in the conjugate gradient scheme. The atomic displacement induced in the conjugate gradient algorithm is corrected to maintain the magnitude of the radial position vector, as required for circular motion.

   (b) With regard to constraint bonds, these are replaced by stiff harmonic bonds to permit minimisation. This is not normally recommended as a means to incorporate constraints in minimisation procedures as it leads to ill conditioning. However, *if the constraints in the original structure are satisfied*, we find that provided only small atomic displacements are allowed during relaxation it is possible to converge to a minimum energy structure. Furthermore, provided the harmonic springs are stiff enough, it is possible afterwards to satisfy the constraints exactly by further optimising the structure using the stiff springs alone, without having a significant affect on the overall system energy.

   (c) Systems with independent constraint bonds and rigid bodies may also be minimised by these methods.

3. Of the three minimisation strategies available in DL_POLY_4, only the programmed minimiser is capable of finding more than one minimum without the user intervening.

4. Finally, we emphasise once again that the purpose of the minimisers in DL_POLY_4 is to help improve the quality of the starting structure and we believe they are adequate for that purpose. We do not recommend them as general molecular structure optimisers. They may however prove useful for relaxing crystal structures to 0 Kelvin for the purpose of identifying a true crystal structure.

Do examine the CONTROL file Section 10.1.1) for more information.

8.2.6 Simulation Efficiency and Performance

Although the DL_POLY_4 underlining parallelisation strategy (DD and link-cells, see Section 11.1.1) is extremely efficient, it cannot always provide linear parallelisation speed gain with increasing processor count for a fixed size system. Nevertheless, it will always provide speedup of the simulation (i.e. there still is a sufficient speed gain in simulations when the number of nodes used in parallel is increased). The simplest explanation why this is is that increasing the processor count for a fixed size system decreases not only the work- and memory-load per processor but also the ratio size of domain to size of halo (both in counts of link cells). When this ratio falls down to values close to one and below, the time DL_POLY_4 spends on inevitable communication (MPI messages across neighbouring domains to refresh the halo data) increases with respect to and eventually becomes prevalent to the time DL_POLY_4 spends on numeric calculations (integration and forces). In such regimes, the **overall** DL_POLY_4 efficiency falls down since processors spend more time on staying idle while communicating than on computing.

It is important that the user recognises when DL_POLY_4 becomes vulnerable to decreased efficiency and what possible measures could be taken to avoid this. DL_POLY_4 calculates and reports the major and secondary link-cell algorithms ($M_x \cdot M_y \cdot M_z$) employed in the simulations immediately after execution. M_x (analogously for M_y and M_z) is the integer number of the ratio of the width of the system domains in x-direction (i.e. perpendicular to the (y,z) plane) to the major and secondary (coming from three- and/or four-body and/or Tersoff interactions) short-range cutoffs specified for the system:

$$\begin{aligned} M_x &= \texttt{Nint}\left[\frac{W_x/P_x}{\texttt{cutoff}}\right] \\ W_x &= \texttt{MD box width} \perp \texttt{plane}(y,z) \\ P_x &= \#(\texttt{nodes})_{x-\texttt{direction}} \quad , \end{aligned} \tag{8.1}$$

where x, y and z represent the directions along the MD cell lattice vectors. Every domain (node) of the MD cell is loaded with $(M_x+2)\cdot(M_y+2)\cdot(M_z+2)$ link-cells of which $M_x \cdot M_y \cdot M_z$ belong to that domain and the rest are a halo image of link-cells forming the surface of the immediate neighbouring domains. In this respect, if we define performance efficiency as minimising communications with respect to maximising computation (minimising the halo volume with respect to the node volume), best performance efficiency will require $M_x \approx M_y \approx M_z \approx M$ and $M \gg 1$. The former expression is a necessary condition and only guarantees good communication distribution balancing. Whereas the latter, is a sufficient condition and guarantees prevalence of computation over communications.

DL_POLY_4 issues a built-in warning when a link-cell algorithms has a dimension less than three (i.e. less than three link-cells per domain in given direction). A useful rule of thumb is that parallelisation speed-up inefficiency is expected when the ratio

$$R = \frac{M_x \cdot M_y \cdot M_z}{(M_x+2)\cdot(M_y+2)\cdot(M_z+2) - M_x \cdot M_y \cdot M_z} \tag{8.2}$$

is close to or drops below one. In such cases there are three strategies for improving the situation that can be used singly or in combination. As obvious from equation (8.1) these are: **(i)** decrease the number of nodes used in parallel, **(ii)** decrease the cutoff and **(iii)** increase system size. It is crucial to note that increased parallelisation efficiency remains even when the link-cell algorithm is used inefficiently. However, DL_POLY_4 will issue an error message and cease execution if it detects it cannot fit a link-cell per domain as this is the minimum the DL_POLY_4 link-cell algorithm can work with - $(1 \cdot 1 \cdot 1)$ corresponding to ratio $R = 1/26$.

It is worth outlining in terms of the $\mathcal{O}(\texttt{computation} \; ; \; \texttt{communication})$ function what the rough scaling performance is like of the most computation and communication intensive parts of DL_POLY_4 in an MD timestep.

(a) Domain hallo re-construction in SET_HALO_PARTICLES, METAL_LD_SET_HALO and DEFECTS_REFERENCE_SET_HALO - $\mathcal{O}\,(\mathcal{N}/P \; ; \; \mathcal{N}/R)$

(b) Verlet neighbourlist construction by link-cells in LINK_CELL_PAIRS - $\mathcal{O}\,(\mathcal{N}/P \; ; \; 0)$, may take up to 40% of the time per timestep

(c) Calculation of k-space contributions to energy and forces from SMPE by EWALD_SPME_FORCES (depends on PARALLEL_FFT which depends on GPFA_MODULE) - $\mathcal{O}\,(\mathcal{N} \; log \; \mathcal{N} \; ; \; (\mathcal{N} \; log \; P)/P)$, may take up to 40% of the time per timestep

(d) Particle exchange between domains, involving construction and connection of new out of domain topology when bonded-like interactions exist, by RELOCATE_PARTICLES - $\mathcal{O}\left(\mathcal{N} \; ; \; (P/\mathcal{N})^{1/3}\right)$

(e) Iterative bond and PMF constraint solvers:
CONSTRAINTS_SHAKE_VV, CONSTRAINTS_RATTLE_VV, CONSTRAINTS_SHAKE_LFV and PMF_SHAKE_VV, PMF_RATTLE_VV, PMF_SHAKE_LFV - $\mathcal{O}\left(\mathcal{N} \; ; \; (P/\mathcal{N})^{1/3}\right)$

where $\mathcal{N}$ is the number of particles, $P = P_x \cdot P_y \cdot P_z$ the total number of domains in the MD cell and the rest of the quantities are as defined in equations (8.1-8.2).

Performance may also affected by the fluctuations in the inter-node communication, due to unavoidable communication traffic when a simulation job does not have exclusive use of all machine resources. Such effects may worsen the performance much, especially when the average calculation time is of the same magnitude as or less than the average communication time (i.e. nodes spend more time communicating rather than computing).

8.3 A Guide to Preparing Input Files

The CONFIG file and the FIELD file can be quite large and unwieldy particularly if a polymer or biological molecule is involved in the simulation. This section outlines the paths to follow when trying to construct files for such systems. The description of the DL_POLY_4 force field in Chapter 2 is essential reading. The various utility routines mentioned in this section are described in greater detail in the DL_POLY_Classic User Manual. Many of these have been incorporated into the DL_POLY GUI [21] and may be conveniently used from there.

8.3.1 Inorganic Materials

The utility GENLAT can be used to construct the CONFIG file for relatively simple lattice structures. Input is interactive. The FIELD file for such systems are normally small and can be constructed by hand. Otherwise, the input of force field data for crystalline systems is particularly simple, if no angular forces are required (notable exceptions to this are zeolites and silicate glasses - see below). Such systems require only the specification of the atomic types and the necessary pair forces. The reader is referred to the description of the DL_POLY_4 FIELD file for further details (Section 10.1.3).

DL_POLY_4 can simulate zeolites and silicate (or other) glasses. Both these materials require the use of angular forces to describe the local structure correctly. In both cases the angular terms are included as *three-body terms*, the forms of which are described in Chapter 2. These terms are entered into the FIELD file with the pair potentials.

An alternative way of handling zeolites is to treat the zeolite framework as a kind of macromolecule (see below). Specifying all this is tedious and is best done computationally: what is required is to determine the nearest image neighbours of all atoms and assign appropriate bond and valence angle potentials. What must be avoided at all costs is specifying the angle potentials *without* specifying bond potentials. In this case DL_POLY_4 will automatically cancel the non-bonded forces between atoms linked via valence angles and the system will collapse. The advantage of this method is that the calculation is likely to be faster than using three-body forces. This method is not recommended for amorphous systems.

8.3.2 Macromolecules

To set up force fields for macromolecules, or indeed any covalent molecules, it is best to use DL_FIELD - http://www.ccp5.ac.uk/DL_FIELD/ . It is a program application tool developed to facilitate the construction of force field models for biological molecules and other molecules with complex geometries. For instance proteins, carbohydrates, polymers and networked molecules such as graphenes and organic cages. **Although created to assist DL_POLY_4, DL_FIELD is a separate program suite that requires separate registration!**

The primary functions of DL_FIELD are as follows:

1. **Force field model converter:** DL_FIELD converts the users atom models, supplied in PDB file format, into input files that are recognisable and ready to run with DL_POLY_Classic and DL_POLY_4 programs with minimum users intervention. This basically involves the conversion of the users atomic configuration in simple xyz coordinates into identifiable atom types base on a particular user-selectable potential schemes and then automatically generate the DL_POLY configuration file (CONFIG), the force field file (FIELD) and a generic control file (CONTROL).

2. **Force field editor:** DL_FIELD allows the user to edit or modify parameters of a particular force field scheme in order to produce a customised scheme that is specific to a particular simulation model. In addition, the standard force field model framework can also be easily modified. For instance, introduction of pseudo points and rigid body implementation to an otherwise standard potential scheme such as CHARMM or AMBER, etc.

3. **Force field library repertoire:** DL_FIELD contains a range of popular potential schemes (see below), all described in a single DL_FIELD format that are also easily recognisable by the user for maintenance purposes. Users can easily expand the existing library to include other new molecules.

Force Field Schemes

The available force field schemes are as follows:

CHARMM - proteins, ethers, some lipids and carbohydrates.

AMBER - proteins and Glycam for carbohydrates.

OPLSAA - proteins

DREIDING - General force field for covalent molecules.

PCFF - Polyorganics and other covalent molecules.

Model Construction

DL_FIELD does not have feature to construct molecular models. This can be achieved by either using DL_POLY GUI [21] or any other standard molecular building packages. The output files must be converted into the PDB format. In the case of proteins, these structures are usually obtained from data banks such as PDB. These *raw* PBD files must first be preprocessed by the user before they are in a *readable* format for DL_FIELD. To ensure this, it is advisable that users take into consideration the following steps:

1. Decide on inclusion/exclusion of and if necessary manually delete molecular residues that involve multiple occupancies in crystalline structures.

2. Usually, hydrogen atoms are not assigned in the raw PDB file. The molecules must therefore be pre-filled with hydrogen atoms (protonated) by using any standard packages available. The user must ensure that proper care is taken of terminal residues which must also be appropriately terminated.

3. Decide on the various charge states of some amino acids, such as histidine (HIS), lysine (LYS), glutamic acid (GLU), etc., by adding or deleting the appropriate hydrogen atoms. Force field schemes such as CHARMM will have different three-letter notations for amino acids of different charge states, DL_FIELD will automatically identify these differences and assign the appropriate potential parameters accordingly.

4. For cysteine (CYS) molecules with disulphide bonds, thiolate hydrogen atoms must be removed. DL_FIELD will automatically define disulphide bonds between the molecules, provided the S-S distance is within a sensible value.

5. DL_FIELD does not solvate the molecules and it is the user's responsibility to add water by using any standard package available (for example the DL_POLY GUI [21]).

Fore more details or further information, please consult the DL_FIELD manual and website
http://www.ccp5.ac.uk/DL_FIELD/ .

8.3.3 Adding Solvent to a Structure

The utility WATERADD adds water from an equilibrated configuration of 256 SPC water molecules at 300 K to fill out the MD cell. The utility SOLVADD fills out the MD box with single-site solvent molecules from a fcc lattice. The FIELD files will then need to be edited to account for the solvent molecules added to the file.

Hint: to save yourself some work in entering the non-bonded interactions variables involving solvent sites to the FIELD file put two bogus atoms of each solvent type at the end of the CONNECT_DAT file (for AMBER force-fields) the utility AMBFORCE will then evaluate all the non-bonded variables required by DL_POLY_4. Remember to delete the bogus entries from the CONFIG file before running DL_POLY_4.

8.3.4 Analysing Results

DL_POLY_4 is not designed to calculate every conceivable property you might wish from a simulation. Apart from some obvious thermodynamic quantities and radial distribution functions, it does not calculate anything beyond the atomic trajectories. You must therefore be prepared to post-process the HISTORY file if you want other information. There are some utilities in the DL_POLY_4 package to help with this, but the list is far from exhaustive. In time, we hope to have many more. Our users are invited to submit code to the DL_POLY_4*public* library to help with this.

The utilities available are described in the DL_POLY_Classic User Manual. Users should also be aware that many of these utilities are incorporated into the DL_POLY GUI [21].

8.3.5 Choosing Ewald Sum Variables

8.3.5.1 Ewald sum and SPME

This section outlines how to optimise the accuracy of the Smoothed Particle Mesh Ewald sum parameters for a given simulation..

As a guide to beginners DL_POLY_4 will calculate reasonable parameters if the **ewald precision** directive is used in the CONTROL file (see Section 10.1.1). A relative error (see below) of 10^{-6} is normally sufficient so the directive

ewald precision 1d-6

will make DL_POLY_4 evaluate its best guess at the Ewald parameters α, `kmaxa`, `kmaxb` and `kmaxc`, or their doubles if **ewald** rather than **spme** is specified. (The user should note that this represents an *estimate*, and there are sometimes circumstances where the estimate can be improved upon. This is especially the case when the system contains a strong directional anisotropy, such as a surface.) These four parameters may also be set explicitly by the **ewald sum** directive in the CONTROL file. For example the directive

ewald sum 0.35 6 6 8

which is equvalent to

spme sum 0.35 12 12 16

would set $\alpha = 0.35$ Å^{-1}, `kmaxa` $= 12$, `kmaxb` $= 12$ and `kmaxc` $= 16$[*]. The quickest check on the accuracy of the Ewald sum is to compare the coulombic energy (U) and virial ($\mathcal{W}$) in a short simulation. Adherence to the relationship $U = -\mathcal{W}$, shows the extent to which the Ewald sum is correctly converged. These variables can be found under the columns headed `eng_cou` and `vir_cou` in the OUTPUT file (see Section 10.2.6).

The remainder of this section explains the meanings of these parameters and how they can be chosen. The Ewald sum can only be used in a three dimensional periodic system. There are five variables that control the accuracy: α, the Ewald convergence parameter; $r_{\rm cut}$ the real space force cutoff; and the `kmaxa`, `kmaxb` and `kmaxc` integers that specify the dimensions of the SPME charge array (as well as FFT arrays). The three integers effectively define the range of the reciprocal space sum (one integer for each of the three axis directions). These variables are not independent, and it is usual to regard one of them as pre-determined and adjust the others accordingly. In this treatment we assume that $r_{\rm cut}$ (defined by the **cutoff** directive in the CONTROL file) is fixed for the given system.

The Ewald sum splits the (electrostatic) sum for the infinite, periodic, system into a damped real space sum and a reciprocal space sum. The rate of convergence of both sums is governed by α. Evaluation of the real space sum is truncated at $r = r_{\rm cut}$ so it is important that α be chosen so that contributions to the real space sum are negligible for terms with $r > r_{\rm cut}$. The relative error (ϵ) in the real space sum truncated at $r_{\rm cut}$ is given approximately by

$$\epsilon \approx {\rm erfc}(\alpha\ r_{\rm cut})/r_{\rm cut} \approx \exp[-(\alpha\ r_{\rm cut})^2]/r_{\rm cut} \quad , \tag{8.3}$$

which reciprocally gives an estimate for α for a given ϵ:

$$\alpha \approx \frac{\sqrt{|\ln(\epsilon\ r_{\rm cut})|}}{r_{\rm cut}} \quad . \tag{8.4}$$

The recommended value for α is $3.2/r_{\rm cut}$ or greater (too large a value will make the reciprocal space sum very slowly convergent). This gives a relative error in the energy of no greater than $\epsilon = 4 \times 10^{-5}$ in the real space sum. When using the directive **ewald precision** DL_POLY_4 makes use of a more sophisticated approximation:

$$ {\rm erfc}(x) \approx 0.56\ \exp(-x^2)/x \tag{8.5}$$

[*] **Important note**: As the SPME method substitues the standard Ewald the values of `kmaxa`, `kmaxb` and `kmaxc` are the double of those in the prescription of the standard Ewald since they specify the sides of a cube, not a radius of convergence.

to solve recursively for α, using equation 8.3 to give the first guess.

The relative error in the reciprocal space term is approximately

$$\epsilon \approx \exp(-k_{max}^2/4\alpha^2)/k_{max}^2 \tag{8.6}$$

where

$$k_{max} = \frac{2\pi}{L}\frac{\texttt{kmax}}{2} \tag{8.7}$$

is largest k-vector considered in reciprocal space, L is the width of the cell in the specified direction and `kmax` is an integer.

For a relative error of 4×10^{-5} this means using $k_{max} \approx 6.2\ \alpha$. `kmax` is then

$$\texttt{kmax} > 6.4\ L/r_{\rm cut}. \tag{8.8}$$

In a cubic system, $r_{\rm cut} = L/2$ implies `kmax` $= 14$. In practice the above equation slightly over estimates the value of `kmax` required, so optimal values need to be found experimentally. In the above example `kmax` $= 10$ or 12 would be adequate.

If you wish to set the Ewald parameters manually (via the **ewald sum** or **spme sum** directives) the recommended approach is as follows. Preselect the value of $r_{\rm cut}$, choose a working a value of α of about $3.2/r_{\rm cut}$ and a large value for the `kmax` (say 20 20 20 or more). Then do a series of ten or so *single* step simulations with your initial configuration and with α ranging over the value you have chosen plus and minus 20%. Plot the Coulombic energy (-$\mathcal{W}$) versus α. If the Ewald sum is correctly converged you will see a plateau in the plot. Divergence from the plateau at small α is due to non-convergence in the real space sum. Divergence from the plateau at large α is due to non-convergence of the reciprocal space sum. Redo the series of calculations using smaller `kmax` values. The optimum values for `kmax` are the smallest values that reproduce the correct Coulombic energy (the plateau value) and virial at the value of α to be used in the simulation. Note that one needs to specify the three integers (`kmaxa`, `kmaxb`, `kmaxc`) referring to the three spatial directions, to ensure the reciprocal space sum is equally accurate in all directions. The values of `kmaxa`, `kmaxb` and `kmaxc` must be commensurate with the cell geometry to ensure the same minimum wavelength is used in all directions. For a cubic cell set `kmaxa` = `kmaxb` = `kmaxc`. However, for example, in a cell with dimensions 2A = 2B = C, (ie. a tetragonal cell, longer in the c direction than the a and b directions) use 2`kmaxa` = 2`kmaxb` = `kmaxc`.

If the values for the kmax used are too small, the Ewald sum will produce spurious results. If values that are too large are used, the results will be correct but the calculation will consume unnecessary amounts of cpu time. The amount of cpu time increases proportionally to `kmaxa` × `kmaxb` × `kmaxc`.

It is worth noting that the working values of the k-vectors may be larger than their original values depending on the actual processor decomposition. This is to satisfy the requirement that the k-vector/FFT transform down each direction per domain is a multiple of 2, 3 and 5 only, which is due to the GPFA code (single 1D FFT) which the DaFT implementation relies on. This allowes for greater flexiblity than the power of 2 multiple restriction in DL_POLY_4 predicessor, DL_POLY_3. As a consequence, however, execution on different processor decompositions may lead to different working lengths of the k-vectors/FFT transforms and therefore slightly different SPME forces/energies whithin the same level of SPME/Ewald precision/accuracy specified. **Note** that although the number of processors along a dimension of the DD grid may be any number, numbers that have a large prime as a factor will lead to inefficient performance!

8.4 Warning and Error Processing

8.4.1 The DL_POLY_4 Internal Warning Facility

DL_POLY_4 contains a number of various in-built checks scattered throughout the package which detect a range of possible inconsistencies or errors. In all cases, such a check fails the subroutine WARNING is called, resulting in an appropriate message that identifies the inconsistency. In some cases an inconsistency is resolved by DL_POLY_4 supplying a default value or DL_POLY_4 assuming a priority of one directive over the another (in clash of mutually exclusive directives). However, in other cases this cannot be done and controlled termination of the program execution is called by the subroutine ERROR. In any case appropriate diagnostic message is displayed notifying the user of the nature of the problem.

8.4.2 The DL_POLY_4 Internal Error Facility

DL_POLY_4 contains a number of in-built error checks scattered throughout the package which detect a wide range of possible errors. In all cases, when an error is detected the subroutine ERROR is called, resulting in an appropriate message and termination of the program execution (either immediately, or after some additional processing). In some case, if the cause for error is considered to be mendable it is corrected and the subroutine WARNING results in an appropriate message.

Users intending to insert new error checks should ensure that all error checks are performed *concurrently* on *all* nodes, and that in circumstances where a different result may obtain on different nodes, a call to the global status routine GCHECK is made to set the appropriate global error flag on all nodes. Only after this is done, a call to subroutine ERROR may be made. An example of such a procedure might be:

```
Logical :: safe
safe = (test_condition)
Call gcheck(safe)
If (.not.safe) Call error(message_number)
```

In this example it is assumed that the logical operation *test_condition* will result in the answer *.true.* if it is safe for the program to proceed, and *.false.* otherwise. The call to ERROR requires the user to state the `message_number` is an integer which used to identify the appropriate message to be printed.

A full list of the DL_POLY_4 error messages and the appropriate user action can be found in Appendix D of this document.

Chapter 9

New Control Format

Scope of Chapter

This chapter describes the changes from old-style to new-style control with information for users and developers.

9.1 Introduction

As of DLPOLY 4.11, there is a new refactored form of control (henceforth new-style). The primary motivation behind this change is an overall improvement in consistency of keywords for the purpose of allowing easier automation of DLPOLY jobs. The revisions confer several additional benefits, however, for both users and developers. These include, but are not limited to:

- More easily extensible hash-table based control
- New control parameter allows definition of defaults, internal units and a description
- Searchable keyword help for keyword description
- Consistent "keyword value unit" scheme for all keywords
- Automated and generalised unit parsing and conversion scheme
- More standardised naming scheme
- Only reads control file once in one location
- Decomposed reading routines for easier handling and addition of new parameters
- Writing routines for parameters independent of reading
- Warnings during reading are directed to the top of output file
- Restructured indentation-based parameter output for easier parsing

The standard form of the new control is that of:

```
keyword value [unit]
```

All new-style control parameters are of this form.

Values are **required** if a keyword is present. Units are **required** for non-dimensionless data.

9.1.1 Keywords

Keywords in new-style control only have one value and attempt to only affect one thing, this means that what, in old-style, might be a single keyword will be subdivided into multiple parameters in new-style. An example of this is the ensemble parameter, which previously might be rendered as:

```
ensemble nvt hoover 1.0
```

will, in new-style, be rendered:

```
ensemble nvt
ensemble_method hoover
ensemble_thermostat_coupling 1.0 ps
```

9.1.2 Value types

New-style control divides control parameters into distinct classes of parameters depending on how they should be handled by the parser. These are int, float, bool, string, option and vector (3,6), however, these are easily extensible and in future more may be added by developers.

9.1.2.1 Int

These values, identified by the DATA_INT enumeration, are simple integer values. There is also a special case for unit-converted integer values for steps values (see: §9.1.3).

```
vaf_binsize 21
```

9.1.2.2 Floats

These values, identified by the DATA_FLOAT enumeration, are generally dimensioned real data and will be converted between input and internal units when read.

```
analyse_max_dist 2.0 ang
```

9.1.2.3 Vector

These values, identified by the DATA_VECTOR3 or DATA_VECTOR6 enumerations, are connected sets of data which may be either floats or ints

```
pressure_tensor [ 1.0 2.0 3.0 4.0 5.0 6.0 ] GPa
ewald_kvec [ 32 64 32 ]
```

9.1.2.4 Bool

These values, identified by the DATA_BOOL enumeration, are binary options which are set by "On" or "Off"

```
vdw_force_shift ON
```

9.1.2.5 String

These values, identified by the DATA_STRING enumeration, are arbitrary strings of characters usually used for setting filepaths, however, they may have special options such as SCREEN or NONE to specify override their function.

```
io_file_config CONTROL.new
io_file_output SCREEN
```

9.1.2.6 Option

These values, identified by the DATA_OPTION enumeration, would otherwise be indistinguishable from strings, however, they are differentiated by the fact that there are number of expected values to switch between.

```
coul_method dddp
```

9.1.3 Units

The automatic units conversion allows the user to enter any dimensionally correct physical unit as input to allow ease and complete flexibility. Units can be entered in a natural manner with decimal prefixes.

Units are **case insensitive**, however decimal prefixes are **case sensitive**.

Units can be combined using a full stop (period) [.] for product or slash [/] for quotients or raised to an exponent with a caret [^]

```
2.0 e.V
1.0 m/s
3.0 ang^3
```

for 2 electron-volts, 1 metre per second & 3 cubic Ångströms respectively.

Decimal prefixes are applied directly to the unit they affect.

```
2.0 GPa
3.0 ang/ps
```

for 2 Gigapascals & 3 Ångströms per picosecond respectively.

The special unit "steps" is derived from the timestep parameter and will be automatically converted to/from to allow consistent run-lengths.

```
timestep 2.0 fs
time_run 30 steps
time_run 60.0 fs
```

will mean the calculation will perform 30 steps of 2 fs (60fs) and alternatively 60fs regardless of the timestep.

9.2 Adding new keywords

New keywords should be added to the parameters hash in `initialise_control` in the style:

```
call table%set("<keyword>", control_parameter( &
     key = "<keyword>", &
     name = "<human-readable-full-name>", &
     val = "<default-value>", &
     units = "<units-of-default>", &
     internal_units = "<units-to-use-internally>", &
     description = "<description-for-help>", &
     data_type = <data-type>))
```

where values in `<>` are to be filled in, and `data-type` is one of `DATA_INT`, `DATA_FLOAT`, `DATA_STRING`, `DATA_BOOL`, `DATA_OPTION`, `DATA_VECTOR3`, `DATA_VECTOR6` and other relevant data is filled in.

If your data is unitless, you can remove the `units` and `internal_units` entries and they will default to unitless.

Keywords to be parsed in `initialise_control` are grouped into named blocks for ease of maintaining these, ensure your keyword is appropriately grouped either into one of these or its own relevant block.

Once the data exists in the parameters table (through `initialise_control`) it is ready to be read in and searched for through the help functions.

The next step is to retrieve the parsed keyword, there are various functions to subdivide reading to increase maintainability and reduce argument lists to workable levels. Within an appropriate read function, call the following function:

```
call params%retrieve("<keyword>", <storage>)
```

where `<storage>` is the variable (of an appropriate type) to store the data. Any necessary unit or data conversion will be performed by the retrieval automatically. If the keyword is not present in control, it will default to "¡default-value¿ ¡default-units¿" as specified in the table entry.

Note: Only floats, vectors and ints in units of steps will act upon units.

Following this, the information should be added to the write function corresponding to the read function for ease of maintainability. It should be noted that written data should be appropriately indented.

Note: Should you be writing a lot of information, it may be best to hide the information printing behind the print level via:

```
Call info(message, .true., level=N)
```

where higher N requires the `print_level` variable to be a higher value (default=2).

9.3 Going from old to new

For most cases to go from old to new, it should be a simple case of using the dlpoly-py tool (Available from: https://gitlab.com/drFaustroll/dlpoly-py/) and using the `old2new` tool in the tools directory through:

```
<path-to-old2new>/old2new.py CONTROL
```

which will create/overwrite `CONTROL.new`.

Old Keyword	New Keyword(s)
first line (taken to be title)	title
l_scr	io_file_output SCREEN
l_tor	io_file_revcon NONE io_file_revive NONE
l_eng	output_energy ON
l_rout	io_write_ascii_revive ON
l_rin	io_read_ascii_revive ON
l_print	print_level
l_dis	initial_minimum_separation
l_fast	unsafe_comms ON
adf *i j*	adf_calculate ON adf_frequency *i* steps adf_precision *j*
analyse all (sampling) (every) *f* **nbins** *n* **rmax** *r*	analyse_all ON

	analyse_frequency *f* steps analyse_max_dist *r* ang analayse_num_bins *n*
ana (**bon** | **ang** | **dih** | **inv**) (sampling) (every) *f* **nbins** *n*	analyse_(bonds|angles|dihedrals|inversion) ON analyse_frequency_(bonds|angles|dihedrals|inversion) *f* steps analyse_num_bins_(bonds|angles|dihedrals|inversion) *n*
binsize *f*	rdf_binsize *f* ang zden_binsize *f* ang
cap (forces) *f*	equilibration_force_cap *f* k_B.temp/ang
close time *f*	time_close *f* s
job time *f*	time_job *f* s **Note: Defaults to 1e304**
coord *i j f*	coord_calculate ON coord_ops (icoord|ccoord|full) coord_start *j* steps coord_interval *f* steps
collect	record_equilibration
coul|**distan**|**reaction**|**shift** **shift damp** α **reaction damp** α **shift precision** *f* **reaction precision** *f*	coul_method (dddp|pairwise|reaction_field|force_shifted) coul_damping α 1/ang coul_precision *f*
cutoff *f* ($\equiv$ **rcut** *f*)	cutoff *f* ang
defects *i j f*	defects_calculate ON defects_start *j* steps defects_interval *f* steps defects_distance *f* ang
delr *f* ($\equiv$ **rpad** $4f$)	removed (see: padding)
densvar *f*	density_variance *f* %
displacements *i j f*	displacements_calculate ON displacements_start *j* steps displacements_interval *f* steps displacements_distance *f* ang
dump *n*	data_dump_frequency *n* steps
ensemble (**nve**|**nvt**|**npt**|**nst**) **evans**	ensemble (nve|nvt|npt|nst) ensemble_method evans

langevin f	ensemble_method langevin ensemble_thermostat_friction f 1/ps
andersen f_1 f_2	ensemble_method andersen ensemble_thermostat_coupling f_1 ps ensemble_thermostat_softness f_2
berendsen f	ensemble_method berendsen ensemble_thermostat_coupling f ps
hoover f	ensemble_method (hoover|nose|nose-hoover) ensemble_thermostat_coupling f ps
gst f_1 f_2	ensemble_method (gentle|gst) ensemble_thermostat_coupling f_1 ps ensemble_thermostat_friction f_2 1/ps
ttm|**inhomo** f_1 f_2 f_3	ensemble_method ttm ttm_e-phonon_friction f_3 1/ps ttm_e-stopping_friction f_2 1/ps ttm_e-stopping_velocity f_3 ang/ps
dpd s1 *gamma*	ensemble_method dpd ensemble_dpd_order (1|first) ensemble_dpd_drag *gamma* Da/ps
dpd s2 *gamma*	ensemble_method dpd ensemble_dpd_order (2|second) ensemble_dpd_drag *gamma* Da/ps
epsilon f	coul_dielectric_constant f
equilibration (steps) f	time_equilibration f steps
ewald precision f	coul_method ewald ewald_precision f
ewald (sum) α k_1 k_2 k_3	coul_method ewald ewald_alpha α ewald_kvec [k_1 k_2 k_3]
exclude	coul_extended_exclusion ON
finish	removed
heat_flux	heat_flux ON
impact i j E x y z	impact_part_index i impact_time j steps impact_energy E ke.V impact_direction [x y z]
nfold i j k	nfold [i j k]
no elec	coul_method off
no index	ignore_config_indices ON
no strict	strict_checks OFF

no topology	print_topology_info OFF
no vafaveraging	vaf_averaging OFF
no vdw	vdw_method OFF
no vom	fixed_com ON
metal direct	metal_direct ON
metal sqrtrho	metal_sqrtrho ON
minimise *string* n f s	minimisation_criterion (off|force|energy|distance)
optimise *string* f s	minimisation_frequency n steps
	minimisation_tolerance f (internal_f|internal_e|ang)
	minimisation_step_length s ang
msdtmp i j	msd_calculate ON
	msd_start i steps
	msd_interval i steps
padding f ($\equiv$ **rpad** f)	padding f ang
plumed *string* (on|off)	plumed (ON|OFF)
plumed input $<$*filename*$>$	plumed_input $<$*filename*$>$
plumed log $<$*filename*$>$	plumed_log $<$*filename*$>$
plumed precision *int-val*	plumed_precision *int-val*
plumed restart *string* (yes|no)	plumed_restart (ON|OFF)
polarisation *scheme/type* **thole** f	polarisation_model (charmm|default)
	polarisation_thole f
pressure f	pressure_hydrostatic f katm
pressure **tensor** xx yy zz xy xz yz	pressure_tensor [xx yy zz xy xz yz] katm
	pressure_perpendicular xx yy zz katm
pp_dump	write_per_particle ON
print (every) n	print_frequency n steps
print analysis	removed
print rdf	rdf_print ON
print vaf	vaf_print ON
print zden	zden_print ON
pseudo *string* f_1 f_2	pseudo_thermostat_method (off|langevin-direct|langevin|gaussian|direct)
	pseudo_thermostat_width f_1 ang
	pseudo_thermostat_temperature f_2 K
quaternion (tolerance) f	removed

rdf (sampling) (every) *f*	rdf_frequency *f* steps
regauss (every) *n*	regauss_frequency *n* steps
replay (history)	Now command line option
restart (|noscale|scale)	restart (clean|continue|rescale|noscale)
rlxtol *f* *s*	rlx_tol *f* internal_f rlx_cgm_step *s* ang
rvdw (cutoff) *f*	vdw_cutoff *f* ang
scale (temperature) (every) *f*	rescale_frequency *f* steps
seed n_1 n_2 n_3	random_seed [n_1 n_2 n_3]
slab	removed
stack (size) *n*	stack_size *n* steps
stats (every) *n*	stats_frequency *n* steps
steps *n*	time_run *n* steps
subcelling (threshold) (density) *f*	subcell_threshold *f*
temperature *f*	temperature *f* K
trajectory *i* *j* *k*	traj_calculate ON traj_start *i* steps traj_interval *j* steps traj_key (pos|pos-vel|pos-vel-force|compressed)
ttm amin *n*	ttm_min_atoms *n*
ttm bcs *Q*	ttm_boundary_condition (periodic|dirichlet|neumann|robin)
	ttm_boundary_xy (ON|OFF)
	ttm_boundary_heat_flux *f* %
ttm ceconst *f*	ttm_heat_cap_model (constant|tanh|linear|tabulated)
ttm cetab	ttm_heat_cap *f*|f_1 internal_e/amu/K
ttm celin f_1 f_2	ttm_fermi_temp f_2 K
ttm cetanh f_1 f_2	ttm_temp_term f_2 K^-1
ttm deconst|diff *f*	ttm_diff_model (constant|reciprocal|tabulated)
ttm derecip f_1 f_2	ttm_diff *f*|f_1 m^2/s
ttm detab	ttm_fermi_temp f_2 K
ttm dedx *f*	ttm_stopping_power *f* e.V/nm
ttm dyndens	ttm_dens_model (constant|dynamic)
ttm atomdens *f*	ttm_dens *f* ang^-3
ttm keconst *f*	ttm_elec_cond_model (infinite|constant|drude|tabulated)
ttm kedrude *f*	ttm_elec_cond *f* W/m/K
ttm keinf	

ttm ketab	
ttm delta	ttm_temporal_dist delta
ttm pulse f	ttm_temporal_duration $f \vert f_1$ ps
ttm gauss f_1 f_2	ttm_temporal_cutoff f_2 ps
ttm nexp f_1 f_2	
ttm sflat	ttm_spatial_dist flat
ttm sgauss f_1 f_2	ttm_spatial_dist gaussian
ttm sigma f_1 f_2	ttm_spatial_sigma f_1 nm
	ttm_spatial_cutoff f_2 nm
ttm laser f_1 f_2	ttm_spatial_dist laser
ttm laser f_1 f_2 **zdep**	ttm_laser_type (flat|exponential)
	ttm_fluence f_1 mJ/cm^2
	ttm_penetration_depth f_2 nm
ttm metal	ttm_metal ON
ttm nonmetal	ttm_metal OFF
ttm ncit n	ttm_num_ion_cells n
ttm ncet n_1 n_2 n_3	ttm_num_elec_cell [n_1 n_2 n_3]
ttm offset f	ttm_time_offset f ps
ttm oneway	ttm_oneway ON
ttm redistribute	ttm_redistribute ON
ttm thvelz	ttm_com_correction (full|zdir|off)
ttm nothvel	
ttm stats n	ttm_stats_frequency n steps
ttm traj n	ttm_traj_frequency n steps
ttm varg homogeneous	ttm_variable_ep homo
ttm varg heterogeneous	ttm_variable_ep hetero
vaf (sampling) (every) i (bin) (size) n	
	vaf_calculate ON
	vaf_frequency i steps
	vaf_binsize n
timestep f	timestep f ps
variable timestep f	timestep f ps
	timestep_variable ON
maxdis f	timestep_variable_max_dist f ang
mindis f	timestep_variable_min_dist f ang
mxstep f	timestep_variable_max_delta f ps
vdw direct	vdw_method (tabulated|direct|ewald|off)
vdw mixing *rule*	vdw_mix_method (Lorentz-Berthelot|Fender-Hasley|Hogervorst|Waldman-Hagler|Tang-Toennies|Functional)
vdw shift	vdw_force_shift ON
zden (sampling) (every) f	zden_calculate ON
	zden_frequency f steps
zero (fire) (every n)	reset_temperature_interval n steps

Chapter 10

Data Files

Scope of Chapter

This chapter describes all the input and output files for DL_POLY_4, examples of which are to be found in the *data* sub-directory.

10.1 The INPUT Files

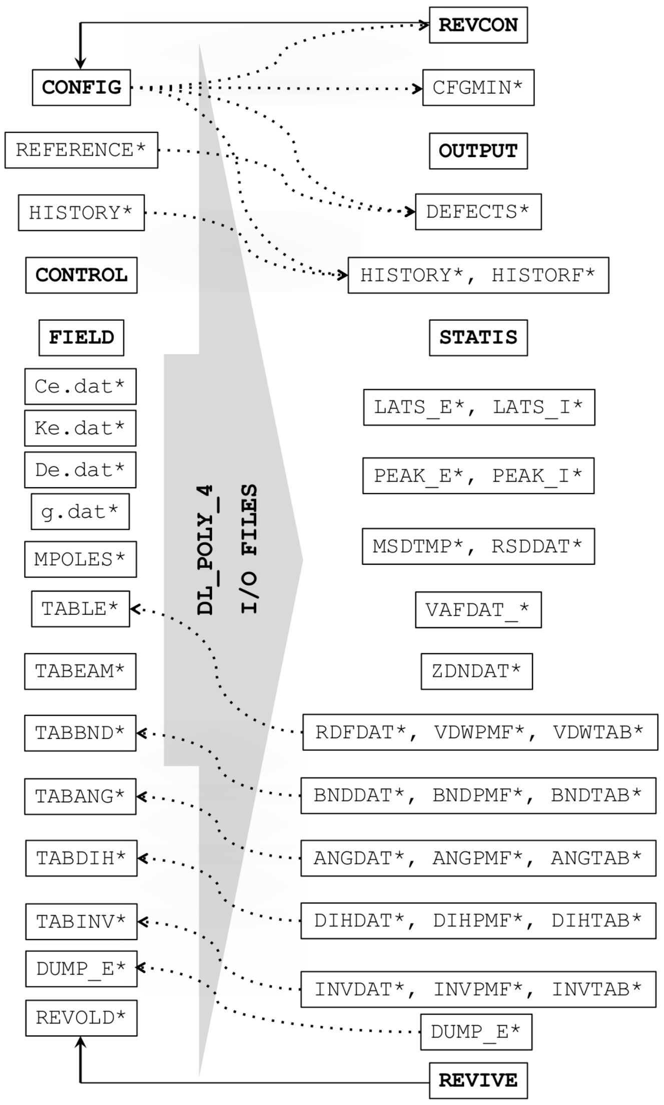


Figure 10.1: DL_POLY_4 input (left) and output (right) files. **Note**: files marked with an asterisk are non-mandatory.

DL_POLY_4 may require many input files. However, only CONTROL, CONFIG and FIELD are mandatory. The MPOLES and TAB* are complimentary to FIELD and are only required when opted within. HISTORY is required when an old trajectory is opted for re-play in CONTROL. REFERENCE is optionally required when defect detection is switched on in CONTROL. REVOLD is required only when the job represents a continuation of a previous job. In the following sections we describe the form and content of these files.

It is worth noting that historically DL_POLY used hard-coded names for different I/O files as shown in Figure 10.1. **This is no longer the case!** Upon instructions in the CONTROL many I/O file name can be overridden with specific, user-defined filenames (see I/O Control Options). Even the CONTROL file can be named differently but in this case the alternative name **must be passed as a command line argument** to the DL_POLY_4 executable (usually named **DLPOLY.Z**). Thus the DL_POLY_4 engine can be efficiently embedded and utilised within external frameworks for user-definable work-flows.

10.1.1 The CONTROL File

The CONTROL file is read by the subroutine READ_NEW_CONTROL and defines the control variables for running a DL_POLY_4 job. Keywords are character strings that appear as the first entry on a data record (or line) and which invoke a particular operation or provide numerical parameters. Keywords can appear in any order in the CONTROL file and some of the directives are mandatory (for example the **timestep** directive that defines the timestep), others are optional.

This way of constructing the file is very convenient, but it has inherent dangers. It is, for example, quite easy to specify contradictory directives, or invoke algorithms that do not work together. By large DL_POLY_4 tries to sort out these difficulties and print helpful error messages, but it does not claim to be fully foolproof. It is important to think carefully about a simulation beforehand and ensure that DL_POLY_4 is being asked to do something that is physically reasonable. It should also be remembered that the present capabilities the package may not allow the simulation required and it may be necessary for you yourself to add new features.

An example CONTROL file appears below. The directives and keywords appearing are described in the following section.

```
title DL_POLY_4 CONTROL DIRECTIVES

# I/O REDIRECT
io_file_output my-dl_poly-run
io_file_field FIELD-dl_field.out
io_file_config CONFIG-CDS-generated
io_file_history new-trajectory
io_file_revive REVIVE-2ps-run
io_file_revcon REVCON-100k-output
io_file_revold REVIVE-1ps-run

# SYSTEM REPLICATION & IMPACT OPTION
nfold [ 10 10 10 ]
impact_part_index 1
impact_time  2000.0 steps
impact_energy  7.5 ke.V
impact_direction [  1.0  2.0  3.0 ]

# DENSITY VARIATION ARRAY BOOST
density_variance  10.0 %

# INDEX AND VERIFICATION BYPASS AND NO TOPOLOGY REPORTING
ignore_config_indices ON
strict_checks OFF
print_topology_info OFF

# INTERACTIONS BYPASS
vdw_method OFF
coul_method OFF

# APPLY MIXING TO ALLOWED & AVAILABLE VDW CROSS INTERACTIONS
vdw_mix_method Lorentz-Berthelot

# DIRECT CALCULATION OF VDW/METAL INTERACTIONS INSTEAD OF
# EVALUATION BY SPLINING OVER TABULATED VALUES IN MEMORY
```

```
vdw_method direct
metal_direct ON

# FORCE-SHIFT VDW INTERACTIONS SO THAT ENERGY AND FORCE
# CONTRIBUTIONS FALL SMOOTHLY TO ZERO WHEN APPROACHING R_CUT
vdw_force_shift ON

# RANDOM NUMBER GENERATOR SEEDING
random_seed [  100  200  300 ]

# RESTART OPTIONS
restart noscale
data_dump_frequency  1000 steps

# SYSTEM TARGET TEMPERATURE AND PRESSURE
pressure_hydrostatic  0.001 katm
temperature  300.0 K

# SYSTEM CUTOFFS AND ELECTROSTATICS
vdw_cutoff  8.0 ang
padding  0.35 ang
cutoff  10.0 ang
subcelling_threshold_density   50.0 %
coul_extended_exclusion ON
coul_dielectric_constant 1.0
coul_method ewald
ewald_precision 1e-05

# RELAXED SHELL MODEL TOLERANCE
rlx_tol  1.0

# CONSTRANTS ITERATION LENGTH and TOLERANCE
shake_max_iter 250
shake_tolerance  1e-05 ang

# INTEGRATION FLAVOUR, ENSEMBLE AND PSEUDO THERMOSTAT
ensemble nst
ensemble_method berendsen
ensemble_thermostat_coupling  0.5 ps
ensemble_barostat_coupling  1.5 ps
pseudo_thermostat_method langevin
pseudo_thermostat_width  2.0 ang
pseudo_thermostat_temperature  150.0 K

# INTEGRATION TIMESTEP
timestep  0.001 ps
timestep_variable ON
timestep_variable_min_dist  0.03 ang
timestep_variable_max_dist  0.1 ang
timestep_variable_max_delta  0.005 ps

# SIMULATION & EQUILIBRATION LENGTH
```

```
time_run  10000.0 steps
time_equilibration  1000.0 steps

# EQUILIBRATION DIRECTIVES
reset_temperature_interval  1.0 steps
equilibration_force_cap  500.0 k_B.temp/ang
rescale_frequency  5.0 steps
regauss_frequency 3.0 steps
minimisation_criterion energy
minimisation_tolerance  0.001 internal_e
minimisation_frequency  20.0 steps
minimisation_step_length  1e-05 ang

# STATISTICS
record_equilibration ON
stack_size  50.0 steps
stats_frequency  10.0 steps

# OUTPUT
print_frequency  20.0 steps

# HISTORY
traj_calculate ON
traj_start  20.0 steps
traj_interval  30.0 steps
traj_key pos

# DEFECTS TRAJECTORY - DEFECTS
defects_calculate ON
defects_start  40.0 steps
defects_interval  15.0 steps
defects_distance  0.75 ang

# DISPLACEMENTS TRAJECTORY - RSDDAT
displacements_calculate ON
displacements_start  70.0 steps
displacements_interval  10.0 steps
displacements_distance  0.25 ang

# MSDTMP
msd_calculate ON
msd_start  1000.0 steps
msd_frequency  100.0 steps

# INTRAMOLECULAR PDF ANALYSIS BY TYPE IF PRESENT
analyse_bonds ON
analyse_frequency_bonds 100 steps
analyse_num_bins_bonds 250

analyse_angles ON
analyse_frequency_angles 100 steps
analyse_num_bins_angles 360
```

```
analyse_dihedrals ON
analyse_frequency_dihedrals 100 steps
analyse_num_bins_dihedrals 720

analyse_inversions ON
analyse_frequency_inversions 100 steps
analyse_num_bins_inversions 360

analyse_all ON
analyse_frequency 100 steps
analyse_num_bins 1000
analyse_max_dist 5.0 ang


# RDF & Z-DENSITY
rdf_print ON
rdf_calculate ON
rdf_frequency  7.0 steps
rdf_binsize 0.05
zden_print ON
zden_calculate ON
zden_frequency  7.0 steps
zden_binsize 0.05

# EMPIRICAL VALENCE BOND (EVB) DIRECTIVE
evb 2

# EXECUTION TIME
time_job  1000.0 s
time_close  10.0 s
```

10.1.1.1 The CONTROL File Format

The file is free-formatted and not case-sensitive. Every line is treated as a command sentence (record). Commented records (beginning with a # or !) and blank lines are not processed and may be added to aid legibility (see example above). Records must be limited in length to 200 characters. Records are read in as up to three words "keyword value unit". A word must not exceed 256 characters in length.

Additional annotation must be rendered as a comment.

The first keyword (i.e. not comment) in the CONTROL file is must be title and a suiltable title.

10.1.1.2 The CONTROL File Directives

Note that in some cases additional keywords, shown in brackets "(...)", may also be supplied in the directives, or directives may be used in a long form. However, it is strongly recommended that the user uses only the **bold** part of these directives.

The **MAIN LIST** of directives available is as follows:

title	STRING	Run title
simulation_method	OPTION	Set simulation method, options: MD, EVB, FFS (default = md)

random_seed	VECTOR3	Set random seed (default = 1 2 3)
density_variance	FLOAT	Set expected density variance for determining maximum array sizes (default = 0.0 `%`)
data_dump_frequency	FLOAT	Set data dumping frequency for restarts (default = 1000 `steps`)
subcell_threshold	FLOAT	Set subcelling threshold density for setting minimum particles per link-cell (default = 50.0)
time_run	FLOAT	Set duration of simulation (inc. equilibration) (default = 0 `steps`)
time_equilibration	FLOAT	Set equilibration duration (default = 0 `steps`)
time_job	FLOAT	Set total job time before attempted safe closure (default = -1.0 `hr`)
time_close	FLOAT	Estimated closure time for finite-time jobs (default = -1.0 `min`)
stats_frequency	FLOAT	Set frequency of stats sampling to statis file (default = 0 `steps`)
stack_size	FLOAT	Set rolling average stack to n timesteps (default = 0 `steps`)
record_equilibration	BOOL	Include equilibration in output (default = off)
print_per_particle_contrib	BOOL	Calculate and print per-particle contributions to energy, force and stress to file every stats step (default = off)
print_probability_distribution	BOOL	Calculate and print probability distribution (enforces RDF print) (default = off)
analyse_all	BOOL	Enable analysis for all bonds, angles, dihedrals and inversions (default = off)
analyse_angles	BOOL	Enable analysis for all angles (default = off)
analyse_bonds	BOOL	Enable analysis for all bonds (default = off)
analyse_dihedrals	BOOL	Enable analysis for all dihedrals (default = off)
analyse_inversions	BOOL	Enable analysis for all inversions (default = off)
analyse_frequency	FLOAT	Set global frequency of data analysis (default = 1 `steps`)
analyse_frequency_bonds	FLOAT	Set frequency of bonds data analysis (default = 1 `steps`)
analyse_frequency_angles	FLOAT	Set frequency of angles data analysis (default = 1 `steps`)
analyse_frequency_dihedrals	FLOAT	Set frequency of dihedrals data analysis (default = 1 `steps`)
analyse_frequency_inversions	FLOAT	Set frequency of inversions data analysis (default = 1 `steps`)
analyse_max_dist	FLOAT	Set cutoff for bonds analysis (default = 2.0 `ang`)
analyse_num_bins	INT	Set global number of bins to be used in bonding analysis (default = -1)
analyse_num_bins_bonds	INT	Set number of bins to be used in bond analysis (default = 0)
analyse_num_bins_angles	INT	Set number of bins to be used in angle analysis (default = 0)
analyse_num_bins_dihedrals	INT	Set number of bins to be used in dihedral analysis (default = 0)
analyse_num_bins_inversions	INT	Set number of bins to be used in inversion analysis (default = 0)
msd_calculate	BOOL	Enable calculation of MSD (default = off)
msd_print	BOOL	Enable printing of MSD (default = off)
msd_start	FLOAT	Start timestep for dumping MSD configurations (default = 0 `steps`)
msd_frequency	FLOAT	Interval between dumping MSD configurations (default = 1 `steps`)
traj_calculate	BOOL	Enable calculation of trajectory (default = off)

traj_key	OPTION	Set trajectory output, options: pos, pos-vel, pos-vel-force, compressed (default = pos)
traj_start	FLOAT	Start timestep for dumping trajectory configurations (default = 0 `steps`)
traj_interval	FLOAT	Interval between dumping trajectory configurations (default = 1 `steps`)
defects_calculate	BOOL	Enable calculation of defects (default = off)
defects_start	FLOAT	Start timestep for dumping defects configurations (default = 0 `steps`)
defects_interval	FLOAT	Interval between dumping defects configurations (default = 1 `steps`)
defects_distance	FLOAT	Set cutoff for deviation to be considered by defects as interstitial (default = 0.75 `ang`)
defects_backup	BOOL	Enable defects backup (default = off)
displacements_calculate	BOOL	Enable calculation of displacements (default = off)
displacements_start	FLOAT	Start timestep for dumping displacements configurations (default = 0 `steps`)
displacements_interval	FLOAT	Interval between dumping displacements configurations (default = 1 `steps`)
displacements_distance	FLOAT	Set cutoff for qualifying as displacement (default = 0.75 `ang`)
coord_calculate	BOOL	Enable calculation of coordination numbers (default = off)
coord_ops	OPTION	Set Coordops, options: icoord: only dumps the coordination of each atom at i; CCOORD: only dumps coordination of each atom at i; FULL: dumps the coordination of each atom every j steps (default = icoord)
coord_start	FLOAT	Start timestep for dumping coordination configurations (default = 0 `steps`)
coord_interval	FLOAT	Interval between dumping coordination configurations (default = 100 `steps`)
adf_calculate	BOOL	Enable calculation of ADF (default = off)
adf_frequency	FLOAT	Set frequency of ADF sampling (default = 100 `steps`)
adf_precision	FLOAT	Set precision of angular distribution bins in ADF analysis (default = 0.0)
rdf_calculate	BOOL	Enable calculation of RDF (default = off)
rdf_print	BOOL	Enable printing of RDF (default = on)
rdf_frequency	FLOAT	Set frequency of RDF sampling (default = 1 `steps`)
rdf_binsize	FLOAT	Set number of bins to be used in RDF analysis (default = 0.05 `ang`)
rdf_error_analysis	OPTION	Enable RDF error analysis, options: Off, Jackknife, Block (default = off)
rdf_error_analysis_blocks	INT	Set number of RDF error analysis blocks (default = 1)
zden_calculate	BOOL	Enable calculation of ZDen (default = off)
zden_print	BOOL	Enable printing of ZDen (default = on)
zden_frequency	FLOAT	Set frequency of ZDen sampling (default = 1 `steps`)
zden_binsize	FLOAT	Set number of bins to be used in ZDen analysis (default = 0.05 `ang`)
vaf_calculate	BOOL	Enable calculation of VAF (default = off)
vaf_print	BOOL	Enable printing of VAF (default = on)
vaf_frequency	FLOAT	Set frequency of VAF sampling (default = 1 `steps`)
vaf_binsize	INT	Set number of bins to be used in VAF analysis (default = 0)
vaf_averaging	BOOL	Ignore time-averaging of VAF, report all calculated VAF to VAFDAT files and final profile to OUTPUT (default = on)

currents_calculate	BOOL	Enable calculation of currents (default = off)
heat_flux	BOOL	Enable calculation of heat flux (default = off)
write_per_particle	BOOL	Enable dumping of per-particle information (default = off)
print_frequency	FLOAT	Set frequency of printing results to output (default = 0 `steps`)
io_units_scheme	OPTION	Set I/O units scheme, options: internal, si, atomic (*unused*) (default = internal)
io_units_length	OPTION	Set I/O units for length (*unused*) (default = internal_l)
io_units_time	OPTION	Set I/O units for time (*unused*) (default = internal_t)
io_units_mass	OPTION	Set I/O units for mass (*unused*) (default = internal_m)
io_units_charge	OPTION	Set I/O units for charge (*unused*) (default = internal_q)
io_units_energy	OPTION	Set I/O units for energy (*unused*) (default = internal_e)
io_units_pressure	OPTION	Set I/O units for pressure (*unused*) (default = internal_p)
io_units_force	OPTION	Set I/O units for force (*unused*) (default = internal_f)
io_units_velocity	OPTION	Set I/O units for velocity (*unused*) (default = internal_v)
io_units_power	OPTION	Set I/O units for power (*unused*) (default = internal_e/internal_t)
io_units_surface_tension	OPTION	Set I/O units for surface tension (*unused*) (default = internal_f/internal_l)
io_units_emf	OPTION	Set I/O units for electromotive force (*unused*) (default = internal_e/internal_q)
io_read_method	OPTION	Set I/O read method, possible read methods: mpiio, direct, netcdf, master (default = mpiio)
io_read_readers	INT	Set number of parallel I/O readers (default = 0 `(Automatic)`)
io_read_batch_size	INT	Set I/O reader batch size (default = 0 `(Automatic)`)
io_read_buffer_size	INT	Set I/O reader buffer size (default = 0 `(Automatic)`)
io_read_error_check	BOOL	Enable extended error checking on read (default = off)
io_read_ascii_revold	BOOL	Read human-readable (ASCII) REVOLD file (default = off)
io_write_method	OPTION	Set I/O write method, possible write methods: mpiio, direct, netcdf, master (default = mpiio)
io_write_writers	INT	Set number of parallel I/O writers (default = 0 `(Automatic)`)
io_write_batch_size	INT	Set I/O writer batch size (default = 0 `(Automatic)`)
io_write_buffer_size	INT	Set I/O writer buffer size (default = 0 `(Automatic)`)
io_write_sorted	BOOL	Enable sorted output for atomic data (default = on)
io_write_error_check	BOOL	Enable extended error checking on write (default = off)
io_write_netcdf_format	OPTION	Set netcdf write format, options: amber, 32bit, 32-bit, 64-bit, 64bit (default = 64bit)
io_write_ascii_revive	BOOL	Write REVIVE as a human-readable (ASCII) file (default = off)
io_file_output	STRING	Set output filepath, special options: SCREEN, NONE (default = OUTPUT)
io_file_config	STRING	Set input configuration filepath (default = CONFIG)
io_file_field	STRING	Set input field filepath (default = FIELD)
io_file_statis	STRING	Set output statistics filepath, special options: NONE (default = STATIS)
io_file_history	STRING	Set output history filepath, special options: NONE (default = HISTORY)
io_file_historf	STRING	Set output historf filepath, special options: NONE (default = HISTORF)

io_file_revive	STRING	Set output revive filepath, special options: NONE (default = REVIVE)
io_file_revold	STRING	Set output revold filepath, special options: NONE (default = REVOLD)
io_file_revcon	STRING	Set output revcon filepath, special options: NONE (default = REVCON)
io_file_rdf	STRING	Set output RDF filepath, special options: NONE (default = RDFDAT)
io_file_msd	STRING	Set output MSD filepath, special options: NONE (default = MSDTMP)
io_file_tabbnd	STRING	Set input TABBND filepath (default = TABBND)
io_file_tabang	STRING	Set input TABANG filepath (default = TABANG)
io_file_tabdih	STRING	Set input TABDIH filepath (default = TABDIH)
io_file_tabinv	STRING	Set input TABINV filepath (default = TABINV)
io_file_tabvdw	STRING	Set input TABVDW filepath (default = TABVDW)
io_file_tabeam	STRING	Set input TABEAM filepath (default = TABEAM)
output_energy	BOOL	Output final energy e_tot in output file (default = off)
ignore_config_indices	BOOL	Ignore indices as defined in CONFIG and use read order instead (default = off)
print_topology_info	BOOL	Print topology information in output file (default = off)
print_level	INT	Disable unnecessary printing, levels: 0 - silent, 1 - quiet, 2 - standard, 3 - full (default = 1)
timer_depth	INT	Do not display timers beyond this many levels in final timer output (default = 4)
timer_per_mpi	BOOL	Write timings for each MPI process individually (default = off)
timestep	FLOAT	Set calculation timestep or initial timestep for variable timestep calculations (default = 0.0 `internal_t`)
timestep_variable	BOOL	Enable variable timestep (default = off)
timestep_variable_min_dist	FLOAT	Set minimum permissible distance for variable timestep (default = 0.03 `ang`)
timestep_variable_max_dist	FLOAT	Set maximum permissible distance for variable timestep (default = 0.1 `ang`)
timestep_variable_max_delta	FLOAT	Set maximum timestep delta for variable timestep (default = 0.0 `internal_t`)
ensemble	OPTION	Set ensemble constraints, options: NVE, PMF, NVT, NPT, NST (default = NVE)
ensemble_method	OPTION	Set ensemble method, options NVT: Evans, Langevin, Andersen, Berendsen, Hoover, gentle, ttm, dpds1, dpds2. NP—ST: Langevin, Berendsen, Hoover, MTK.
ensemble_thermostat_coupling	FLOAT	Set thermostat relaxation/decorrelation times (use ensemble_thermostat_friction for langevin) (default = 0.0 `ps`)
ensemble_dpd_order	OPTION	Set dpd method, options: off, first, second (default = off)
ensemble_dpd_drag	FLOAT	Set DPD drag coefficient (default = 0.0 `Da/ps`)
ensemble_thermostat_friction	FLOAT	Set thermostat friction for langevin and gentle stochastic thermostats (default = 0.0 `ps^-1`)
ensemble_thermostat_softness	FLOAT	Set thermostat softness for Andersen thermostat (default = 0.0)
ensemble_barostat_coupling	FLOAT	Set barostat relaxation/decorrelation times (use ensemble_barostat_friction for langevin) (default = 0.0 `ps`)
ensemble_barostat_friction	FLOAT	Set barostat friction (default = 0.0 `ps^-1`)

ensemble_semi_isotropic	OPTION	Enable semi-isotropic barostat constraints, options: area, tension, orthorhombic (default = off)
ensemble_semi_orthorhombic	BOOL	Enable semi-orthorhombic barostat constraints (default = off)
ensemble_tension	FLOAT	Set tension in NPngT calctulation (default = 0.0 `N/m`)
pressure_tensor	VECTOR6	Set the target pressure tensor for NsT calculations (default = 0.0 0.0 0.0 0.0 0.0 0.0 `katm`)
pressure_hydrostatic	FLOAT	Set the target hydrostatic pressure (1/3Tr[P]) for NPT calculations (default = 0.0 `katm`)
pressure_perpendicular	VECTOR3	Set the target pressure as x, y, z perpendicular to cell faces for NPT calculations (default = 0.0 0.0 0.0 `katm`)
temperature	FLOAT	Set the initial temperature or target temperature (for thermostats) (default = 0.0 `K`)
pseudo_thermostat_method	OPTION	Set pseudo thermostat method, possible options: Off, Langevin-Direct, Langevin, Gauss, Direct (default = off)
pseudo_thermostat_width	FLOAT	Set the width of thermostatted boundaries for pseudo thermostats (default = 2.0 `ang`)
pseudo_thermostat_temperature	FLOAT	Set the temperature of the pseudo thermostat (default = 0.0 `K`)
impact_part_index	INT	Set particle index for impact simulations (default = 0)
impact_time	FLOAT	Set time for impact in impact simulations (default = 0.0 `internal_t`)
impact_energy	FLOAT	Set impact energy for impact simulations (default = 0.0 `ke.V`)
impact_direction	VECTOR3	Direction vector for impact simulations (default = 1.0 1.0 1.0)
ttm_calculate	BOOL	Enable calculation of two-temperature model (default = off)
ttm_num_ion_cells	INT	Set number of coarse-grained ion temperature cells (CIT) (default = 10)
ttm_num_elec_cells	VECTOR3	Set number of coarse-grained electronic temperature cells (CET) (default = 50 50 50)
ttm_metal	BOOL	Specifies parameters for metallic system are required for two-temperature model, i.e. thermal conductivity (default = off)
ttm_heat_cap_model	OPTION	Sets model for specific heat capacity in TTM, options: const, linear, tabulated, tanh
ttm_heat_cap	FLOAT	Sets constant, scale or maximum heat capcity in TTM (default = 0.0 `internal_e/internal_m/K`)
ttm_temp_term	FLOAT	Set Fermi temperature in TTM, for tanh (default = 0.0 `K^-1`)
ttm_fermi_temp	FLOAT	Set Fermi temperature in TTM, for linear (default = 0.0 `K`)
ttm_elec_cond_model	OPTION	Set electronic conductivity model in TTM, options: Infinite, constant, drude, tabulated
ttm_elec_cond	FLOAT	Set electronic conductivity in TTM (default = 0.0 `W/m/K`)
ttm_diff_model	OPTION	Set diffusion model in TTM, options: constant, recip, tabulated
ttm_diff	FLOAT	Set TTM thermal diffusivity (default = 0.0 `m^2/s`)
ttm_dens_model	OPTION	Set density model in TTM, options are: constant, dynamic
ttm_dens	FLOAT	Set constant density in TTM (default = 0.0 `ang^-3`)
ttm_min_atoms	INT	Minimum number of atoms needed per ionic temperature cell (default = 0)
ttm_stopping_power	FLOAT	Electronic stopping power of projectile entering electronic system (default = 0.0 `e.V/nm`)

ttm_spatial_dist	OPTION	Set the spatial distribution of TTM, options: flat, gaussian, flat-laser, exp-laser
ttm_spatial_sigma	FLOAT	Set the sigma for spatial distributions of TTM (default = 1.0 `nm`)
ttm_spatial_cutoff	FLOAT	Set the cutoff for spatial distributions of TTM (default = 5.0 `nm`)
ttm_fluence	FLOAT	Initial energy deposition into electronic system by laser for TTM (default = 0.0 `mJ/cm^2`)
ttm_penetration_depth	FLOAT	Set laser penetration depth for TTM (default = 0.0 `nm`)
ttm_laser_type	OPTION	Set laser deposition type. options: flat, exponential (default = flat)
ttm_temporal_dist	OPTION	Set temporal distribution for TTM, options: gaussian, exponential, delta, square
ttm_temporal_duration	FLOAT	Set duration of energy deposition for TTM (gaussian, exponential, square) (default = 0.001 `ps`)
ttm_temporal_cutoff	FLOAT	Set temporal cutoff for TTM (gaussian, exponential) (default = 5.0 `ps`)
ttm_variable_ep	OPTION	Set electron-phonon coupling for TTM, options: homo, hetero
ttm_boundary_condition	OPTION	Set boundary conditions for TTM, options: periodic, dirichlet, neumann, robin
ttm_boundary_xy	BOOL	Fix Neumann (zero-flux) boundary in Z (default = off)
ttm_boundary_heat_flux	BOOL	Set boundary heat flux in Robin boundaries for TTM (default = 96 `%`)
ttm_time_offset	FLOAT	Set electron-ion coupling offset for TTM (default = 0.0 `ps`)
ttm_oneway	BOOL	Enable one-way electron-phonon coupling when electronic temperature is greater than ionic temperature (default = off)
ttm_stats_frequency	FLOAT	Frequency of write to TTM PEAK_E and PEAK_I (default = 0 `steps`)
ttm_traj_frequency	FLOAT	Frequency of write to TTM LATS_E and LATS_I (default = 0 `steps`)
ttm_com_correction	OPTION	Apply inhomogeneous Langevin thermostat to selected directions in TTM, options: full, zdir, off (default = full)
ttm_redistribute	BOOL	Redistribute electronic energy in newly-deactivated temperature cells to nearest active neighbours
ttm_e-phonon_friction	FLOAT	Set TTM electron-phonon friction (default = 0.0 `ps^-1`)
ttm_e-stopping_friction	FLOAT	Set TTM electron-stopping friction (default = 0.0 `ps^-1`)
ttm_e-stopping_velocity	FLOAT	Set TTM electron-stopping velocity (default = 0.0 `ang/ps`)
rlx_cgm_step	FLOAT	Set CGM stepping for relaxed shell model (default = -1.0 `ang`)
rlx_tol	FLOAT	Set force tolerance for relaxed shell model (default = 1.0 `internal_f`)
shake_max_iter	INT	Set maximum number of SHAKE/RATTLE iterations (default = 250)
shake_tolerance	FLOAT	Set accepted SHAKE/RATTLE tolerance (default = 1e-6 `ang`)
dftb	BOOL	Enable DFTB (default = off)
fixed_com	BOOL	Remove net centre of mass momentum (default = on)
reset_temperature_interval	FLOAT	Interval between temperature zeroing during equilibration for minimisation (default = -1 `steps`)

regauss_frequency	FLOAT	Set the frequency of temperature regaussing (default = -1 `steps`)
rescale_frequency	FLOAT	Set the frequency of temperature rescaling (default = -1 `steps`)
equilibration_force_cap	FLOAT	Set force cap clamping maximum force during equilibration (default = 1000.0 `k_b.temp/ang`)
minimisation_criterion	OPTION	Set minimisation criterion, options: off, force, energy, distance (default = off)
minimisation_tolerance	FLOAT	Set minimisation tolerance, units: determined by criterion (default = 0.0)
minimisation_step_length	FLOAT	Set minimisation tolerance (default = -1.0 `ang`)
minimisation_frequency	FLOAT	Set minimisation frequency (default = 0 `steps`)
initial_minimum_separation	FLOAT	Turn on the check on minimum separation distance between VNL pairs at re/start (default = -1.0 `internal_l`)
restart	OPTION	Set restart settings, possible options: Clean, Continue, Rescale, Noscale (default = clean)
nfold	VECTOR3	Expand cell before running (default = 1 1 1)
cutoff	FLOAT	Set the global cutoff for real-speace potentials (default = 1.0 `internal_l`)
padding	FLOAT	Set padding for sizing of Verlet neighbour lists (default = 0.0 `internal_l`)
coul_damping	FLOAT	Calculate electrostatics using Fennell damping (Ewald-like) with given alpha (default = 0.0 `1/Ang`)
coul_dielectric_constant	FLOAT	Set dielectric constant relative to vacuum (default = 1.0)
coul_extended_exclusion	BOOL	Enable extended coulombic exclusion affecting intra-molecular interactions (default = off)
coul_method	OPTION	Set method for electrostatics method, options: off, spme, dddp, pairwise, reaction_field, force_shifted (default = off)
coul_precision	FLOAT	Calculate electrostatics using Fennell damping (Ewald-like) with given precision (default = 0.0)
ewald_precision	FLOAT	Set Ewald parameters to calculate within given precision for Ewald calculations (default = 1.0e-6)
ewald_alpha	FLOAT	Set real-recip changeover location for Ewald calculations (default = 0.0 `ang^-1`)
ewald_kvec	VECTOR3	Set number of k-space samples for Ewald calculations (default = 0 0 0)
ewald_kvec_spacing	FLOAT	Calculate k-vector samples for an even sampling of given spacing in Ewald calculations (default = 0.0 `ang^-1`)
ewald_nsplines	INT	Set number of B-Splines for Ewald SPME calculations, min=3 (default = 8)
polarisation_model	OPTION	Enable polarisation, options: default, CHARMM (default = default)
polarisation_thole	FLOAT	Set global atomic damping factor (default = 1.3)
metal_direct	BOOL	Enable direct (non-tabulated) calculation of metallic forces (default = off)
metal_sqrtrho	BOOL	Enable metal sqrtrho interpolation option for EAM embeding function in TABEAM (default = off)
vdw_method	OPTION	Set method for Van der Waal's calculations, options: off, direct, tabulated, ewald (default = tabulated)
vdw_cutoff	FLOAT	Set cut-off for Van der Waal's potentials (default = 0.0 `internal_l`)

vdw_mix_method	OPTION	Enable VdW mixing, possible mixing schemes: Off, Lorentz-Berthelot, Fender-Hasley, Hogervorst, Waldman-Hagler, Tang-Toennies, Functional (default = off)
vdw_force_shift	BOOL	Enable force shift corrections to Van der Waals' forces (default = off)
plumed	BOOL	Enabled plumed dynamics (default = off)
plumed_input	STRING	Set plumed input file
plumed_log	STRING	Set plumed log file
plumed_precision	INT	Set plumed numerical precision (4=single, 8=double) (default = 8)
plumed_restart	BOOL	Restart plumed dynamics (default = on)
strict_checks	BOOL	Enforce strict checks such as: good system cutoff, particle index contiguity, disable non-error warnings, minimisation information (default = on)
unsafe_comms	BOOL	Do not ensure checks of logicals are enforced in parallel (default = off)
dftb_test	BOOL	Do not perform a DLPOLY run, instead run dftb tests (default = off)

10.1.1.3 Further Comments on the CONTROL File

1. All keywords in new-style control files must be unique, double specification will result in an error. This is to protect users from unexpected behaviour.
2. The following parameters are **mandatory**:
   (a) ensemble
   (b) ensemble_method
   (c) cutoff
   (d) timestep
3. Some directives are optional. If not specified DL_POLY_4 may give default values if necessary. (defaults are specified above in the list of directives) However fail-safe DL_POLY_4 is, ensure parameters are appropriate for the system of interest and defaults should only be used after it is known that they are valid.
4. The **time_run** and **time_equilibration** directives have a default of zero. If not used or used with their default values a "dry run" is performed. This includes force generation and system dump (REVCON and REVIVE) and, depending on the rest of the options, may include; velocity generation, force capping, application of the CGM minimiser, application of the pseudo thermostat, and dumps of HISTORY, DEFECTS, RDFDAT, ZDNDAT and MSDTMP. **Note** that, since no actual dynamics is to be performed, the **temperature** and **pressure** directives do not play any role and are therefore not necessary.
5. If the CGM minimiser, **minimise_frequency**, is specified with zero frequency, it is only applied at timestep zero if **time_equilibration** $\geq$ **time_run** (i.e. optimise structure at start only!). In this way it can be used as a configuration optimiser at the beginning of the equilibration period or when a "dry run" (**time_run** = 0) is performed (i.e. equilibrate without any actual dynamics!). **Note** that the default CGM search algorithm stepping uses a step that is proportional to the square of the instantaneous value of the **timestep** and thus its usage in may lead to algorithm's instability and failure in the cases when optimisation is applied before any dynamics occurs in a model system in an unphysical state (i.e. much away from equilibrium) and/or with an ill/badly defined forcefield for the state. Hence, special care (specifying the optional CGM stepping for example) should be taken

when the option is used, especially in a "dry run" mode with a timestep value too large for the model system state.

6. The **timestep_variable** option requires the user to specify an initial guess for a reasonable timestep for the system. The simulation is unlikely to retain this as the operational timestep however, as the latter may change in response to the dynamics of the system. The option is used in conjunction with the default values of **timestep_variable_max_dist** (0.10 Å) and **timestep_variable_min_dist** (0.03 Å), which can also be optionally altered if used as directives (note the rule that **timestep_variable_max_dist** $>$ 2.5 **timestep_variable_min_dist** applies). Also, an additional **timestep_variable_max_delta** (in ps) control can be applied. These serve as control values in the variable timestep algorithm, which calculates the greatest distance a particle has travelled in any timestep during the simulation. If the maximum distance is exceeded, the timestep variable is halved and the step repeated. If the greatest move is less than the minimum allowed, the timestep variable is doubled and the step repeated provided it does not exceed the user specified **timestep_variable_max_delta**. If it does then it scales to **timestep_variable_max_delta** and the step is repeated. In this way the integration timestep self-adjusts in response to the dynamics of the system. **Note** that this option is abandoned when used in conjunction with DPD thermostats (**ensemble_method dpd**), since it cannot be applied efficiently and furthermore it is not well defined in a DPD sense either.

7. The starting options for a simulation are governed by the keyword **restart**. If this is **not** specified in the control file, the simulation will start as new. When specified, it will continue a previous simulation (**restart**) provided all needed restart files are in place and not corrupted. If they are not in place or are found corrupted, it will start a new simulation without initial temperature scaling of the previous configuration (**restart noscale**).

8. The **ensemble nst** keyword is also used in the NσT ensembles extension to NP$_n$AT and NP$_n\gamma$T ones. **Note** that these semi-isotropic ensembles are only correct for infinite interfaces placed perpendicularly to the z axis! This means that the interface is homogeneous (unbroken) and continuous in the (x,y) plane of the MD cell, which assumes that that two of the cell vectors have a cross product only in the z direction. (For example, if the MD box is defined by its lattice vectors ($\underline{a}, \underline{b}, \underline{c}$) then $\underline{a} \times \underline{b} = \pm(0, 0, 1)$.) It is the users' responsibility to ensure this holds for their model system.

9. The **reset_temperature_interval** directive, enables a "zero temperature" optimisation. In this case a crude energy minimiser is used to help the system relax before each integration of the equations of motion. The function of the minimiser can be summarised as

$$\underline{v}_i \leftarrow \begin{cases} 0 & : \; \underline{v}_i \cdot \underline{f}_i < 0 \\ \underline{f}_i \, \dfrac{\underline{v}_i \cdot \underline{f}_i}{\underline{f}_i \cdot \underline{f}_i} & : \; \underline{v}_i \cdot \underline{f}_i \geq 0 \end{cases} \tag{10.1}$$

for systems with free particles only, where $\underline{v}_i$ and $\underline{f}_i$ are the force and velocity of particle i. The algorithm is extended in the case of RBs by including

$$\begin{aligned} \underline{V}_j &\leftarrow \begin{cases} 0 & : \; \underline{V}_j \cdot \underline{F}_j < 0 \\ \underline{F}_j \, \dfrac{\underline{V}_j \cdot \underline{F}_j}{\underline{F}_j \cdot \underline{F}_j} & : \; \underline{V}_j \cdot \underline{F}_j \geq 0 \end{cases} \\ \underline{\omega}_j &\leftarrow \begin{cases} 0 & : \; \underline{\omega}_j \cdot \underline{\tau}_j < 0 \\ \underline{\tau}_j \, \dfrac{\underline{\omega}_j \cdot \underline{\tau}_j}{\underline{\tau}_j \cdot \underline{\tau}_j} & : \; \underline{\omega}_j \cdot \underline{\tau}_j \geq 0 \end{cases} , \end{aligned} \tag{10.2}$$

where $\underline{V}_j$ and $\underline{F}_j$ are the velocity and force of the RB's centre of mass, and $\underline{\omega}_j$ and $\underline{\tau}_j$ are the angular velocity and torque of RB j. Measures are taken to conserve the MD cell momentum and the thermostat's instantaneous kinetic energy.

This must not be thought of as a true energy minimization method. **Note** that this optimisation is only applied when the simulation runs in equilibration mode.

The algorithm is developed in the DL_POLY_4 routine ZERO_K_OPTIMISE.

10. The **impact** directives will not be activated if the particle index is beyond the one of the last particle. The option will fail in a controlled manner at application time if the particle is found to be in a frozen state or the shell of an ion or part of a rigid body. During application the center of mass momentum is re-zeroed to prevent any drifts. The user must take care to have the impactinitiated after any possible equilibration. Otherwise, the system will be thermostatted and the impact energy dissipated during the equilibration.

11. The **pseudo_thermostat** directives are intended to be used in highly non-equilibrium simulations when users are primarily interested in the structural changes in the core of the simulated system as the the MD cell boundaries of the system are coupled to a thermal bath.

    The thermal bath can be used with several types of temperature scaling algorithms

    (a) Langevin (stochastic thermostat),
    (b) Gauss
    (c) Direct (direct thermostat).

    The user is also required to specify the width of the pseudo thermostat, **pseudo_thermostat_width**, which must be larger than 2 Å and less than or equal to a quarter of minimum width of the MD cell. The thermostat is an pseudo_thermostat_width thick buffer layer attached on the inside at the MD cell boundaries.

    The temperature of the bath is specified by the user, **pseudo_thermostat_temperature**, which must be larger than 1 Kelvin.

    - **langevin**
      The stochasticity of the Langevin thermostat emulates an infinite environment around the MD cell, providing a means for "natural" heat exchange between the MD system and the heath bath thus aiding possible heat build up in the system. In this way the instantaneous temperature of the system is driven naturally towards the bath temperature. Every particle within the thermostat buffer layer is coupled to a viscous background and a stochastic heat bath, such that

$$\begin{aligned} \frac{d\underline{r}_i(t)}{dt} &= \underline{v}_i(t) \\ \frac{d\underline{v}_i(t)}{dt} &= \frac{\underline{f}_i(t) + \underline{R}_i(t)}{m_i} - \chi(t)\,\underline{v}_i(t) \quad , \end{aligned} \tag{10.3}$$

      where $\chi(t)$ is the friction parameter from the dynamics in the the MD cell and $R(t)$ is stochastic force with zero mean that satisfies the fluctuation-dissipation theorem:

$$\left\langle R_i^\alpha(t)\ R_j^\beta(t')\right\rangle = 2\ \chi(t)\ m_i\ k_B T\ \delta_{ij}\ \delta_{\alpha\beta}\ \delta(t-t') \quad , \tag{10.4}$$

      where superscripts denote Cartesian indices, subscripts particle indices, k_B is the Boltzmann constant, T the bath temperature and m_i the particle's mass. The algorithm is implemented in routine PSEUDO and has two stages:

      – Generate random forces on all particles within the thermostat. Here, care must be exercised to prevent introduction of non-zero net force when the random forces are added to the system force field.

  - Rescale the kinetic energy of the thermostat bath so that particles within have Gaussian distributed kinetic energy with respect to the target temperature and determine the (Gaussian constraint) friction within the thermostat:

$$\chi(t) = Max\left(0, \frac{\sum_i [\vec{f}_i(t) + \vec{R}_i(t)] \cdot \vec{v}_i(t)}{\sum_i m_i \ \vec{v}_i^2(t)}\right) \quad . \tag{10.5}$$

    Care must be exercised to prevent introduction of non-zero net momentum. (Users are reminded to use for target temperature the temperature at which the original system was equilibrated in order to avoid simulation instabilities.)

  The effect of this algorithm is to relax the buffer region of the system on a local scale and to effectively dissipate the incomingexcess kinetic energy from the rest of the system, thus emulating an infinite-like environment surrounding the MD cell. The thermostat width matters as the more violent the events on the inside of the MD cell, the bigger width may be needed in order to ensure safe dissipation of the excess kinetic energy.

- **Gaussian**
  - Rescale the kinetic energy of the thermostat bath so that particles within have Gaussian distributed kinetic energy with respect to the target temperature.
- **direct**

  The Direct thermostat is the simplest possible model allowing for heat exchange between the MD system and the heath bath. All (mass, non-frozen) particles within the bath have their kinetic energy scaled to 1.5 $k_B T$ at the end of each time step during the simulation. Care is exercised to prevent introduction of non-zero net momentum when scaling velocities. (Users are reminded to use for target temperature the temperature at which the original system was equilibrated in order to avoid simulation instabilities.) Due to the "unphysical" nature of this temperature control the thermostat width does not matter to the same extent as in the case of the Langevin thermostat.

Note that embedding a thermostat in the MD cell walls is bound to produce **wrong ensemble averages**, and instantaneous pressure and stress build-ups at the thermostat boundary. Therefore, ensembles lose their meaning as such and so does the conserved quantity for true ensembles.

The algorithms are developed in the DL_POLY_4 routine PSEUDO_VV.

12. The **defects_calculate** option will trigger reading of REFERENCE (see Section 10.1.5), which defines a reference MD cell with particles' positions defining the crystalline lattice sites. If REFERENCE is not found the simulation will either:

- halt if the simulation has been restarted, i.e. is a continuation of an old one - the **restart** option is used in CONTROL and the REVOLD (see Section 10.1.6) file has been provided.
- recover using CONFIG (see Section 10.1.2) if it is a new simulation run, i.e **restart** option is not used in CONTROL or REVOLD has not been provided.

The actual defect detection is based on comparison of the simulated MD cell to the reference MD cell based on a user defined site-interstitial cutoff, R_{def},

$$\text{Min}\,[0.3, r_{\text{cut}}/3] \;\; \mathring{A} \;\leq\; R_{def} \;\leq\; \text{Min}\,[1.2, r_{\text{cut}}/2] \;\; \mathring{A}$$

with a default value of Min $[0.75, r_{\text{cut}}/3]$ Å. (If the supplied value exceeds the limits the simulation execution will halt). If a particle, p, is located in the vicinity of a site, s, defined by a sphere with its centre at this site and a radius, R_{def}, then the particle is a *first hand* claimee of s, and the site is not vacant. Otherwise, the site is presumed vacant and the particle is presumed a general interstitial. If a site, s, is claimed and another particle, $p\prime$, is located within the sphere around it, then $p\prime$ becomes an interstitial associated with s. After all particles and all sites are considered, it is clear which sites are vacancies. Finally, for every claimed site, distances between the site and its *first hand* claimee and interstitials are compared and the particle with the shortest one becomes the *real* claimee. If a *first hand* claimee of s is not the *real* claimee it becomes an interstitial associated with s. At this stage it is clear which particles are interstitials. The sum of interstitials and vacancies gives the total number of defects in the simulated MD cell.

Frozen particles and particles detected to be shells of polarisable ions are not considered in the defect detection.

Note that the algorithm cannot be applied safely if R_{def} is larger than half the shortest interatomic distance within the reference MD cell since a particle may; (i) claim more than one site, (ii) be an interstitial associated with more than one site, or both (i) and (ii). On the other hand, low values of R_{def} are likely to lead to slight overestimation of defects.

If the simulation and reference MD cell have the same number of atoms then the total number of interstitials is always equal to the total number of defects.

13. The **displacements_calculate** option will trigger dump of atom displacements based on a qualifying cutoff in a trajectory like manner. Displacements of atoms from their original position at the end of equilibration (the start of statistics), $t = 0$, is carried out at each timestep.

14. The tolerance and stepping for relaxed shell model **rlx_tol**, is a last resort option to aid shell relaxation of systems with very energetic and/or rough potential surface. Users are advised to use it with caution, should there really need be, as the use of high values for the tolerance (default of 1) may result in physically incorrect dynamics and small stepping (default $Max(k_{core-shell})/2$) to expensive force evaluations.

15. The choice of reaction field electrostatics (directive **coul_method reaction_field**) relies on the specification of the relative dielectric constant external to the cavity. This is specified by the **coul_dielectric_consta** directive.

16. DL_POLY_4 uses two different potential cutoffs. These are as follows:

    (a) r_{cut} - the universal cutoff set by **cutoff**. It applies to the real space part of the electrostatics calculations and to the van der Waals potentials if no other cutoff is applied.

    (b) r_{vdw} - the user-specified cutoff for the van der Waals potentials set by **vdw_cutoff**. If not specified its value defaults to r_{cut}.

17. Constraint algorithms in DL_POLY_4, SHAKE/RATTLE (see Section 3.2), use default iteration precision of 10^{-6} and limit of iteration cycles of 250. Users may experience that during optimisation of a new built system containing constraints simulation may fail prematurely since a constraint algorithm failed to converge. In such cases directives **shake_max_iter** (to increase) and **shake_tolerance** (to decrease) may be used to decrease the strain in the system and stablise the simulation numerics until equilibration is achieved.

18. DL_POLY_4's DD strategy assumes that the local (per domain/node or link cell) density of various system entities (i.e. atoms, bonds, angles, etc.) does not vary much during a simulation and some limits for these are assumed empirically. This may not the case in extremely non-equilibrium simulations, where the assumed limits are prone to be exceeded or in some specific systems where these do not hold from the start. A way to tackle such circumstances and avoid simulations crash (by controlled termination) is to use the **density_variance** f option. In the SET_BOUNDS subroutine DL_POLY_4 makes assumptions at the beginning of the simulation and corrects the lengths of bonded-like interaction lists arrays (`mxshl, mxcons, mxrgd, mxteth, mxbond, mxangl, mxdihd, mxinv`) as well as the lengths of link-cell (`mxlist`) and domain (`mxatms, mxatdm`) lists arrays when the option is activated with $f > 0$. Greater values of f will correspond to allocation bigger global arrays and larger memory consumption by DL_POLY_4 during the simulation. **Note** that this option may demand more

memory than available on the computer architecture. In such cases DL_POLY_4 will terminate with an array allocation failure message.

19. As a default, DL_POLY_4 does not store statistical data during the equilibration period. If the directive **record_equilibration** is used, equilibration data will be incorporated into the overall statistics.

20. The **vaf_calculate** directive switches on velocity autocorrelation function (VAF) calculations for individual atomic species in DL_POLY_4 after equilibration or immediately at start if the directive **record_equilibration** is used. It controls how often VAF profiles are started what the size of each profile (in timesteps). Overlapping profiles are possible and require more memory to store them (and initial velocities) while they are being calculated. By default DL_POLY_4 will report time-averaged VAF profiles. This can be overridden using the **vaf_averaging** directive, which will instead report individual 'instantaneous' VAF profiles.

21. The **no vom** option will trigger a default bypass of the GETVOM routine which will return zero and thus no COM removal will happen. **Note that this will lead to COM momentum accumulation for many though not all ensembles!**. Such accumulation will propagate to the generation of flow in the MD cell and ultimately suppress the thermal motion of the particles in the system, leading to the so called "frozen ice cube effect"! It is worth nothing that this option must be turned on for the correct application of stochastic dynamics via the langevin temperature control (NVT Langevin)! If the option is not applied then the dynamics will lead to peculiar thermalisation of different atomic species to mass- and system size-dependent temperatures.

22. The **padding** option will add extra distance, $r_{\rm pad}$, if larger than $f \geq {\rm Min}[0.05, 0.5\%.r_{\rm cut}]$ Å, to the major cutoff, $r_{\rm cut}$, to construct a larger link-cell width, $r_{\rm lnk} = r_{\rm cut} + r_{\rm pad}$, which will trigger a construction of a larger Verlet neighbour list (VNL) while at the same time facilitate its conditional update, rather at every timestpe. The VNL conditaional update is check at the end of each tiemstep and triggered only when the most travelled particle has moved a distance larger than $r_{\rm pad}/2$. It is worth noting that padding is at expense of extra memory but if used wisely it could improve time to solution from 10% to 100% depending on force-field complexity. If it is too large or too small (that is why the $f \geq {\rm Min}[0.05, 0.5\%.r_{\rm cut}]$ Å limit) it will lead to performace degradation. It is recomended that $r_{\rm pad}$ is set up at a value of $\approx 1 \div 5\%$ of the cutoff, $r_{\rm cut}$, as long as the major link-cell algorithm uses a link-cell decomposition that not worse than $4 \otimes 4 \otimes 4$ per domain. For such setups, in practice, one may expect average compute speedups [*] of the order of $10 \div 30\%$ for force-fields involving the Ewald summation methodology and $60 \div 100\%$ for force-fields without electrostatics evaluations involving the Ewald summation methodology.

23. The **subcelling_threshold_density** option will set the threshold density of particles per link cell below which subcelling (decreasing link-cell size) stops. The default is 50 although experimenting with values between 10 and 100 may provide sufficient information about a performance sweetspot. The comparison condition is carried out on the average density per sub-link cell of each domain. Thus subcelling takes into account only the per domain density variation. Obviously, the subcelling threshhold is limited to 1 particle per cell for the sake of safety and sanity reasons.

24. The **coul_extended_exclusion** option will make sure that for all *conventional* (no distance restraints) intra-molecular interactions (bonds, angles, digedrals, inversions) as well as for CB and RB units any intra-core-shell interactions fall within the list of excluded interactions. This is not a default behaviour. The option is also triggered by the **polarisation** directvies.

25. The various **ttm** options all have the effect of switching on the two-temperature model (TTM), which assumes the use of the inhomogeneous Langevin NVT ensemble. It is not possible to specify any other ensemble for TTM-based systems, so the inhomogeneous Langevin NVT ensemble will always be used

[*] I.e. I/O effects are excluded from comparison with a default simulation and comparisons are carried over a few hundreds of timesteps. This is usually accounting for over 90% of the time to solution.

in this case, with default values for the friction terms and cut-off velocity if these are not specified. No implementation of the inhomogeneous Langevin ensemble currently exists for rigid bodies. To model metallic systems, the thermal conductivity must be specified either as infinitely large, a constant value, a linear function of electronic temperature (based on the Drude model) or as a tabulated function of temperature, while non-metallic systems require a constant thermal diffusivity. Any energy deposition is applied after equilibration. An additional restart file, DUMP_E, consisting of electronic temperatures for each electronic temperature cell (CET), is produced along with REVCON and REVIVE when TTM is switched on.

Users are advised to study the example CONTROL files appearing in the *data* sub-directory to see how different files are constructed.

10.1.2 The CONFIG File

The CONFIG file contains the dimensions of the unit cell, the key for periodic boundary conditions and the atomic labels, coordinates, velocities and forces. This file is read by the subroutine READ_CONFIG (optionally by SCAN_CONFIG) in the SET_BOUNDS routine. The first few records of a typical CONFIG file are shown below:

```
IceI structure 6x6x6 unit cells with proton disorder
         2         3              276
  26.988000000000000   0.000000000000000   0.000000000000000
 -13.494000000000000  23.372293600000000   0.000000000000000
   0.000000000000000   0.000000000000000  44.028000000000000
       OW        1
    -2.505228382        -1.484234330        -7.274585343
    0.5446573999        -1.872177437       -0.7702718106
     3515.939287         13070.74357         4432.030587
       HW        2
    -1.622622646        -1.972916834        -7.340573742
     1.507099154        -1.577400769         4.328786484
     7455.527553        -4806.880540        -1255.814536
       HW        3
    -3.258494716        -2.125627191        -7.491549620
     2.413871957        -4.336956694         2.951142896
    -7896.278327        -8318.045939        -2379.766752
       OW        4
    0.9720599243E-01    -2.503798635        -3.732081894
     1.787340483        -1.021777575        0.5473436377
     9226.455153         9445.662860         5365.202509

etc.
```

10.1.2.1 The CONFIG File Format

The file is free-formatted and case sensitive for the atom species names. Every line is treated as a command sentence (record). However, line records are limited to 72 characters in length. Records are read in words, as a word must not exceed 40 characters in length. Words are recognised as such by separation by one or more space characters. The first record in the CONFIG file is a header (up to 72 characters long) to aid identification of the file. Blank and commented lines are not allowed.

10.1.2.2 Definitions of Variables in the CONFIG File

record 1

`header`	a72	title line

record 2

`levcfg`	integer	CONFIG file key. See Table 10.2 for permitted values
`imcon`	integer	Periodic boundary key. See Table 10.3 for permitted values
`megatm`	integer	Optinal, total number of particles (crystalographic entities)

record 3 omitted if `imcon` = 0

`cell(1)`	real	x component of the *a* cell vector in Å
`cell(2)`	real	y component of the *a* cell vector in Å
`cell(3)`	real	z component of the *a* cell vector in Å

record 4 omitted if `imcon` = 0

`cell(4)`	real	x component of the *b* cell vector in Å
`cell(5)`	real	y component of the *b* cell vector in Å
`cell(6)`	real	z component of the *b* cell vector in Å

record 5 omitted if `imcon` = 0

`cell(7)`	real	x component of the *c* cell vector in Å
`cell(8)`	real	y component of the *c* cell vector in Å
`cell(9)`	real	z component of the *c* cell vector in Å

Note that **record 2** may contain more information apart from the mandatory as listed above. If the file has been produced by DL_POLY_4 then it also contains other items intended to help possible parallel I/O reading. Also, it is worth mentioning that the periodic boundary conditions (PBC), as specified in Table 10.3 and described in detail in Appendix B, refer generally to a **triclinic type of super-cell**, for which there are **no symmetry assumptions! Records 3, 4 and 5** contain the Cartesian components of the super-cell's lattice vectors in Å. DL_POLY_4 can only tract triclinic type of super-cells as the only types of super-cell shapes that are commensurate with the domain decomposition (DD) parallelisation strategy of it. However, this is not a restriction for the replicated data (RD) parallelisation that DL_POLY_Classic adopts and thus it can also accept truncated octahedral and rhombic dodecahedral periodic boundaries.

Subsequent records consists of blocks of between 2 and 4 records depending on the value of the `levcfg` variable. Each block refers to one atom. The atoms **do not** need to be listed sequentially in order of increasing index. Within each block the data are as follows:

record i

`atmnam`	a8	atom name
`index`	integer	atom index

record ii

`xxx`	real	x coordinate in Å
`yyy`	real	y coordinate in Å
`zzz`	real	z coordinate in Å

record iii included only if `levcfg` > 0

`vxx`	real	x component of velocity in Å/picosecond
`vyy`	real	y component of velocity in Å/picosecond
`vzz`	real	x component of velocity in Å/picosecond

record iv included only if `levcfg` > 1

`fxx`	real	x component of force in Å·Dalton/picosecond^2
`fyy`	real	y component of force in Å·Dalton/picosecond^2
`fzz`	real	z component of force in Å·Dalton/picosecond^2

Note that on **record i** only the atom name is strictly mandatory, any other items are not read by DL_POLY_Classic but may be added to aid alternative uses of the file, for example alike DL_POLY GUI

[21], DL_POLY_Classic assume that the atoms' indices are in a sequentially ascending order starting form 1. However, DL_POLY_4 needs the index or the **no ind**ex option needs to be specified in the CONTROL file! It is worth mentioning that DL_POLY_4 (as well as DL_POLY_Classic) assumes that the origin of Cartesian system with respect to which the particle positions are specified is the middle of MD cell. Also, as both the cell vectors and the particles' positions are specified in Å, there is a **fine connection** between them! This would not be the case if the particles' positions were kept in reduced space with fractional coordinates. Last but not least, it is worth pointing out that composite entities, such as velocities and forces, have their units expressed as composites of the default DL_POLY units as shown in Section 1.3.7.

Table 10.2: CONFIG File Key (record 2)

`levcfg`	meaning
0	coordinates included in file
1	coordinates and velocities included in file
2	coordinates, velocities and forces included in file

Table 10.3: Periodic Boundary Key (record 2)

`imcon`	meaning
0	no periodic boundaries
1	cubic boundary conditions
2	orthorhombic boundary conditions
3	parallelepiped boundary conditions
6	x-y parallelogram boundary conditions with no periodicity in the z direction

10.1.2.3 Further Comments on the CONFIG File

The CONFIG file has the same format as the output file REVCON (Section 10.2.9). When restarting from a previous run of DL_POLY_4 (i.e. using the **restart**, **restart noscale** or **restart scale** directives in the CONTROL file - above), the CONFIG file must be replaced by the REVCON file, which is renamed as the CONFIG file. The *copy* macro in the *execute* sub-directory of DL_POLY_4 does this for you.

The CONFIG file has the same format as the optional output file CFGMIN, which is only produced when the **minimise** (**optimise**) option has been used during an equilibration simulation or a "dry run".

10.1.3 The FIELD File

The FIELD file contains the force field information defining the nature of the molecular forces. This information explicitly includes the (site) topology of the system which sequence **must** be matched (implicitly) in the crystallographic description of the system in the CONFIG file. The FIELD file is read by the subroutine READ_FIELD. (It is also read by the subroutine SCAN_FIELD in the SET_BOUNDS routine.) Excerpts from a force field file are shown below. The example is the antibiotic Valinomycin in a cluster of 146 water molecules:

```
Valinomycin Molecule with 146 SPC Waters
UNITS kcal

MOLECULES    2
Valinomycin
NUMMOLS 1
```

```
ATOMS 168
    O        16.0000     -0.4160         1
    OS       16.0000     -0.4550         1
    "          "            "         "
    "          "            "         "
    HC        1.0080      0.0580         1
    C        12.0100      0.4770         1
BONDS 78
harm   31   19 674.000     1.44900
harm   33   31 620.000     1.52600
    "    "    "    "         "
    "    "    "    "         "
harm  168   19 980.000     1.33500
harm  168  162 634.000     1.52200
CONSTRAINTS 90
   20   19    1.000017
   22   21    1.000032
    "    "      "
    "    "      "
  166  164    1.000087
  167  164    0.999968
ANGLES 312
harm   43    2   44  200.00      116.40
harm   69    5   70  200.00      116.40
  "    "    "    "     "           "
  "    "    "    "     "           "
harm   18  168  162  160.00      120.40
harm   19  168  162  140.00      116.60
DIHEDRALS 371
harm    1   43    2   44  2.3000      180.00
harm   31   43    2   44  2.3000      180.00
  "    "    "    "    "     "           "
  "    "    "    "    "     "           "
cos   149   17  161   16  10.500      180.00
cos   162   19  168   18  10.500      180.00
FINISH
SPC Water
NUMMOLS 146
ATOMS  3
    OW       16.0000     -0.8200
    HW        1.0080      0.4100
    HW        1.0080      0.4100
CONSTRAINTS  3
    1    2    1.0000
    1    3    1.0000
    2    3    1.63299
FINISH
VDW   45
C       C         lj   0.12000      3.2963
C       CT        lj   0.08485      3.2518
"       "         "     "             "
"       "         "     "             "
"         "          "      "            "
OW        OS         lj   0.15100      3.0451
OS        OS         lj   0.15000      2.9400
CLOSE
```

10.1.3.1 The FIELD File Format

The file is free-formatted and not case-sensitive (except for the site names). Every line is treated as a command sentence (record). Commented records (beginning with a #) and blank lines are not processed and may be added to aid legibility (see example above). Records must be limited in length to 200 characters. Records are read in words, as a word must not exceed 40 characters in length. Words are recognised as such by separation by one or more space characters. The contents of the file are variable and are defined by the use of **directives**. Additional information is associated with the directives.

10.1.3.2 Definitions of Variables in the FIELD File

The file divides into three sections: general information, molecular descriptions, and non-bonded interaction descriptions, appearing in that order in the file.

General information

The first viable record in the FIELD file is the title. The second is the **units** directive. Both of these are mandatory.

record 1

header	a200	field file header

record 2

units	a40	Unit of energy used for input and output

The energy units on the **units** directive are described by additional keywords:

a. **eV**, for electron-Volts

b. **kcal**/mol, for k-calories per mol

c. **kJ**/mol, for k-Joules per mol

d. **K**elvin/Boltzmann, for Kelvin per Boltzmann

e. **internal**, for DL_POLY internal units (10 Joules per mol).

If no units keyword is entered, DL_POLY internal units are assumed for both input and output. The **units** directive only affects the input and output interfaces, all internal calculations are handled using DL_POLY units. System input and output energies are read in **units per MD cell**.

Note that all energy bearing potential parameters are read in terms of the specified energy units. If such a parameter depends on an angle then the dependence is read in terms of radians although the following angle in the parameter sequence is read in terms of degrees. As to any rule, there is an exception - the electrostatic bond potential has no energy bearing parameter!

A third optional record can then be supplied before any molecular description:

record 3

multipolar order n	a50, integer	Electrostatics evaluation to order n poles

This will later trigger the parsing of the MPOLES file (see Section 10.1.4) which supplies the multipolar momenta values of the molecular sites specified in FIELD. Sites with specified pole orders, m, smaller than the one required, n, will have their $m+1$ to n order poles' momenta zeroed. Similarly, if sites have momenta of poles of higher order than the one required, n, these will not be processed by DL_POLY_4.

Note, although algorithms in DL_POLY_4 could in principle handle any high pole order summation, in practice, however, DL_POLY_4 will abort if the order is higher than hexadecapole (order 4)! For more information on this functionality refer to Section 2.4.2.

Molecular details

It is important for the user to understand that there is an organisational correspondence between the FIELD file and the CONFIG file described above. It is required that the order of specification of molecular types and their atomic constituents in the FIELD file follows the order of indices in which they appear in the CONFIG file. Failure to adhere to this common sequence will be detected by DL_POLY_4 and result in premature termination of the job. It is therefore essential to work from the CONFIG file when constructing the FIELD file. It is not as difficult as it sounds!

The entry of the molecular details begins with the mandatory directive:

molecules *n*

where n is an integer specifying the number of different *types* of molecule appearing in the FIELD file. Once this directive has been encountered, DL_POLY_4 enters the *molecular description* environment in which only molecular description keywords and data are valid.

Immediately following the **molecules** directive, are the records defining individual molecules:

1. *name-of-molecule*
   which can be any character string up to 200 characters in length. (**Note**: this is not a directive, just a simple character string.)

2. **nummols *n***
   where n is the number of times a molecule of this type appears in the simulated system. The molecular data then follow in subsequent records:

3. **atoms *n***
   where n indicates the number of atoms in this type of molecule. A number of records follow, each giving details of the atoms in the molecule i.e. site names, masses and charges. Each record carries the entries:

`sitnam`	a8	atomic site name
`weight`	real	atomic site mass (in Daltons)
`chge`	real	atomic site charge (in protons)
`nrept`	integer	repeat counter
`ifrz`	integer	'frozen' atom (if `ifrz` $>$ 0)

   The integer `nrept` need not be specified if the atom/site is not frozen (in which case a value of 1 is assumed.) A number greater than 1 specified here indicates that the next (`nrept`-1) entries in the CONFIG file are ascribed the atomic characteristics given in the current record. The sum of the repeat numbers for all atoms in a molecule should equal the number specified by the **atoms** directive.

4. **shell *n***
   where n is the number of core-shell units. Each of the subsequent n records contains:

index 1 (i)	integer	site index of core

index 2 (j)	integer	site index of shell
k_2	real	force constant of core-shell spring
k_4	real	quartic (anharmonic) force constant of spring

The spring potential is

$$U(r) = \frac{1}{2}k_2 r_{ij}^2 + \frac{1}{4!}k_4 r_{ij}^4 \quad , \tag{10.6}$$

with the force constant k_2 entered in units of `engunit`×Å^{-2} and k_4 in `engunit` Å^{-4}, where usually $k_2 >> k_4$. The `engunit` is the energy unit specified in the **units** directive.

Note that the atomic site indices referred to above are indices arising from numbering each atom in the molecule from 1 to the number specified in the **atoms** directive for this molecule. This same numbering scheme should be used for all descriptions of this molecule, including the **constraints**, **pmf**, **rigid**, **teth**, **bonds**, **angles**, **dihedrals** and **inversions** entries described below. DL_POLY_4 will itself construct the global indices for all atoms in the systems.

Note that DL_POLY_4 determines which shell model to use by scanning shells' weights provided the FIELD file (see Section 2.5). If all shells have zero weight the DL_POLY_4 will choose the relaxed shell model. If no shell has zero weight then DL_POLY_4 will choose the dynamical one. In case when some shells are massless and some are not DL_POLY_4 will terminate execution controllably and provide information about the error and possible possible choices of action in the OUTPUT file (see Section 10.2.6). Shell models' extensions are dealt according to the user specifications in the FIELD files.

This directive (and associated data records) need not be specified if the molecule contains no core-shell units.

5. **constraints *n***
where n is the number of constraint bonds in the molecule. Each of the following n records contains:

index 1	integer	first atomic site index
index 2	integer	second atomic site index
bondlength	real	constraint bond length

This directive (and associated data records) need not be specified if the molecule contains no constraint bonds. See the note on the atomic indices appearing under the **shell** directive above.

6. **pmf *b***
where b is the potential of mean force bondlength (Å). There follows the definitions of two PMF units:

(a) **pmf unit *n1***
where *n1* is the number of sites in the first unit. The subsequent *n1* records provide the site indices and weighting. Each record contains:

index	integer	atomic site index
weight	real	site weighting

(b) **pmf unit *n2***
where *n2* is the number of sites in the second unit. The subsequent *n2* records provide the site indices and weighting. Each record contains:

index	integer	atomic site index
weight	real	site weighting

This directive (and associated data records) need not be specified if no PMF constraints are present. See the note on the atomic indices appearing under the **shell** directive.

Note that if a site weighting is not supplied DL_POLY_4 will assume it is zero. However, DL_POLY_4 detects that all sites in a PMF unit have zero weighting then the PMF unit sites will be assigned the masses of the original atomic sites.

The PMF bondlength applies to the distance between the centres of the two PMF units. The centre, $\vec{R}_i$, of each unit is given by

$$\underline{R}_i = \frac{\sum_{j=1}^{n_i} w_j \; \vec{r}_j}{\sum_{j=1}^{n_j} w_j} \quad , \tag{10.7}$$

where r_j is a site position and w_j the site weighting.

Note that the PMF constraint is intramolecular. To define a constraint between two molecules, the molecules must be described as part of the same DL_POLY_4 "molecule". DL_POLY_4 allows only one type of PMF constraint per system. The value of nummols for this molecule determines the number of PMF constraint in the system.

Note that in DL_POLY_4 PMF constraints are handeled in every available ensemble.

7. **rigid *n***
where n is the number of basic rigid units in the molecule. It is followed by at least n records, each specifying the sites in a rigid unit:

m	integer	number of sites in rigid unit
site 1	integer	first site atomic index
site 2	integer	second site atomic index
site 3	integer	third site atomic index
..	..	*etc.*
site m	integer	m'th site atomic index

Up to 15 sites can be specified on the first record. Additional records can be used if necessary. Up to 16 sites are specified per record thereafter.

This directive (and associated data records) need not be specified if the molecule contains no rigid units. See the note on the atomic indices appearing under the **shell** directive above.

8. **teth *n***
where n is the number of tethered atoms in the molecule. It is followed n records specifying the tehered sites in the molecule:

tether key	a4	potential key, see Table 10.4
index 1 (i)	integer	atomic site index
variable 1	real	potential parameter, see Table 10.4
variable 2	real	potential parameter, see Table 10.4

The meaning of these variables is given in Table 10.4.

This directive (and associated data records) need not be specified if the molecule contains no flexible chemical bonds. See the note on the atomic indices appearing under the **shell** directive above.

9. **bonds *n***
where n is the number of flexible chemical bonds in the molecule. Each of the subsequent n records contains:

bond key	a4	potential key, see Table 10.5
index 1 (i)	integer	first atomic site index in bond
index 2 (j)	integer	second atomic site index in bond
variable 1	real	potential parameter, see Table 10.5

Table 10.4: Tethering Potentials

key	potential type	Variables (1-3)			functional form
harm	Harmonic	k			$U(r) = \frac{1}{2}\ k\ (r_i - r_i^{t=0})^2$
rhrm	Restraint	k	r_c		$U(r) = \frac{1}{2}\ k\ (r_i - r_i^{t=0})^2 \quad : \quad \lvert r_i - r_i^{t=0}\rvert \leq r_c$ $U(r) = \frac{1}{2}\ k\ r_c^2 + k\ r_c(\lvert r_i - r_i^{t=0}\rvert - r_c) \quad : \quad \lvert r_i - r_i^{t=0}\rvert > r_c$
quar	Quartic	k	k'	k''	$U(r) = \frac{k}{2}\ (r_i - r_i^{t=0})^2 + \frac{k'}{3}\ (r_i - r_i^{t=0})^3 + \frac{k''}{4}\ (r_i - r_i^{t=0})^4$

variable 2	real	potential parameter, see Table 10.5
variable 3	real	potential parameter, see Table 10.5
variable 4	real	potential parameter, see Table 10.5

The meaning of these variables is given in Table 10.5.

This directive (and associated data records) need not be specified if the molecule contains no flexible chemical bonds. See the note on the atomic indices appearing under the **shell** directive above.

10. **angles *n***
where n is the number of valence angle bonds in the molecule. Each of the n records following contains:

angle key	a4	potential key, see Table 10.6
index 1 (i)	integer	first atomic site index
index 2 (j)	integer	second atomic site index (central site)
index 3 (k)	integer	third atomic site index
variable 1	real	potential parameter, see Table 10.6
variable 2	real	potential parameter, see Table 10.6
variable 3	real	potential parameter, see Table 10.6
variable 4	real	potential parameter, see Table 10.6

The meaning of these variables is given in Table 10.6.

This directive (and associated data records) need not be specified if the molecule contains no angular terms. See the note on the atomic indices appearing under the **shell** directive above.

11. **dihedrals *n***
where n is the number of dihedral interactions present in the molecule. Each of the following n records contains:

dihedral key	a4	potential key, see Table 10.7
index 1 (i)	integer	first atomic site index
index 2 (j)	integer	second atomic site index (central site)
index 3 (k)	integer	third atomic site index
index 4 (l)	integer	fourth atomic site index
variable 1	real	first potential parameter, see Table 10.7
variable 2	real	second potential parameter, see Table 10.7
variable 3	real	third potential parameter, see Table 10.7
variable 4	real	1-4 electrostatic interaction scale factor
variable 5	real	1-4 van der Waals interaction scale factor
variable 6	real	fourth potential parameter, see Table 10.7
variable 7	real	fifth potential parameter, see Table 10.7

Table 10.5: Chemical Bond Potentials

key	potential type	Variables (1-4)				functional form
tab **-tab**	Tabulation, see Sections 2.2.1 10.1.9					tabulated potential in TABBND file
harm **-hrm**	Harmonic	k	r_0			$U(r) = \frac{1}{2}\ k\ (r_{ij} - r_0)^2$
mors **-mrs**	Morse	E_0	r_0	k		$U(r) = E_0\ [\{1 - \exp(-k\ (r_{ij} - r_0))\}^2 - 1]$
12-6 **-126**	12-6	A	B			$U(r) = \left(\frac{A}{r_{ij}^{12}}\right) - \left(\frac{B}{r_{ij}^{6}}\right)$
lj **-lj**	Lennard-Jones	ϵ	σ			$U(r) = 4\epsilon\left[\left(\frac{\sigma}{r_{ij}}\right)^{12} - \left(\frac{\sigma}{r_{ij}}\right)^{6}\right]$
rhrm **-rhm**	Restraint	k	r_0	r_c		$U(r) = \frac{1}{2}\ k\ (r_{ij} - r_0)^2 \ :\ \lvert r_{ij} - r_0\rvert \le r_c$ $U(r) = \frac{1}{2}\ k\ r_c^2 + k\ r_c(\lvert r_{ij} - r_0\rvert - r_c)\ :\ \lvert r_{ij} - r_0\rvert > r_c$
quar **-qur**	Quartic	k	r_0	k'	k''	$U(r) = \frac{k}{2}\ (r_{ij} - r_0)^2 + \frac{k'}{3}\ (r_{ij} - r_0)^3 + \frac{k''}{4}\ (r_{ij} - r_0)^4$
buck **-bck**	Buckingham	A	ρ	C		$U(r) = A\ \exp\left(-\frac{r_{ij}}{\rho}\right) - \frac{C}{r_{ij}^6}$
coul **-cul**	Coulomb	k				$U(r) = k \cdot U^{Electrostatics}(r_{ij})\ \left(= \frac{k}{4\pi\epsilon_0\epsilon}\frac{q_i q_j}{r_{ij}}\right)$
fene **-fne**	Shifted* FENE [32, 33, 34]	k	R_o	Δ		$U(r) = -0.5\ k\ R_o\ ln\left[1 - \left(\frac{r_{ij}-\Delta}{R_o^2}\right)^2\right]\ :\ r_{ij} < R_o + \Delta$ $U(r) = \infty\ :\ r_{ij} \ge R_o + \Delta$
mmst **-mst**	MM3 bond stretch [35]	k	r_o			$U(r) = k\ \delta^2\ \left[1 - 2.55\ \delta + (7/12)\ 2.55^2\ \delta^2\right]\ ;\ \delta\ = r - r_o$

* Note: Δ defaults to zero if $|\Delta| > 0.5\ R_o$ or if it is not specified in the FIELD file.

Note: Bond potentials with a dash (-) as the first character of the keyword, do not contribute to the *excluded atoms list* (see Section 2). In this case DL_POLY_4 will also calculate the non-bonded pair potentials between the described atoms, unless these are deactivated by another potential specification.

Table 10.6: Valence Angle Potentials

key	potential type	Variables (1-4,5-6)				functional form†
tab **-tab**	Tabulation, see Sections 2.2.3 10.1.9					tabulated potential in TABANG file
harm **-hrm**	Harmonic	k	θ_0			$U(\theta) = \frac{k}{2}\ (\theta - \theta_0)^2$
quar **-qur**	Quartic	k	θ_0	k'	k''	$U(\theta) = \frac{k}{2}\ (\theta - \theta_0)^2 + \frac{k'}{3}(\theta - \theta_0)^3 + \frac{k''}{4}(\theta - \theta_0)^4$
thrm **-thm**	Truncated harmonic	k	θ_0	ρ		$U(\theta) = \frac{k}{2}\ (\theta - \theta_0)^2 \exp[-(r_{ij}^8 + r_{ik}^8)/\rho^8]$
shrm **-shm**	Screened harmonic	k	θ_0	ρ_1	ρ_2	$U(\theta) = \frac{k}{2}\ (\theta - \theta_0)^2 \exp[-(r_{ij}/\rho_1 + r_{ik}/\rho_2)]$
bvs1 **-bv1**	Screened Vessal [36]	k	θ_0	ρ_1	ρ_2	$U(\theta) = \frac{k}{8(\theta_0-\pi)^2}\left\{\left[(\theta_0 - \pi)^2 - (\theta - \pi)^2\right]^2\right\} \times$ $\exp[-(r_{ij}/\rho_1 + r_{ik}/\rho_2)]$
bvs2 **-bv2**	Truncated Vessal [37]	k	θ_0	a	ρ	$U(\theta) = k\ (\theta - \theta_0)^2\ \left[\theta^a(\theta + \theta_0 - 2\pi)^2\right.$ $\left.+\frac{a}{2}\pi^{a-1}(\theta_0 - \pi)^3\right] \exp[-(r_{ij}^8 + r_{ik}^8)/\rho^8]$
hcos **-hcs**	Harmonic Cosine	k	θ_0			$U(\theta) = \frac{k}{2}\ (\cos(\theta) - \cos(\theta_0))^2$
cos **-cos**	Cosine	A	δ	m		$U(\theta) = A\ [1 + \cos(m\ \theta - \delta)]$
mmsb **-msb**	MM3 stretch-bend [35]	A	θ_0	r_{ij}^o	r_{jk}^o	$U(\theta) = A\ (\theta - \theta_0)\ (r_{ij} - r_{ij}^o)\ (r_{ik} - r_{ik}^o)$
stst **-sts**	Compass [38] stretch-stretch	A	r_{ij}^o	r_{jk}^o		$U(\theta) = A\ (r_{ij} - r_{ij}^o)\ (r_{ik} - r_{ik}^o)$
stbe **-stb**	Compass [38] stretch-bend	A	θ_0	r_{ij}^o		$U(\theta) = A\ (\theta - \theta_0)\ (r_{ij} - r_{ij}^o)$
cmps **-cmp**	Compass [38] all terms	A r_{ij}^o	B r_{jk}^o	C	θ_0	$U(\theta) = A\ (r_{ij} - r_{ij}^o)\ (r_{ik} - r_{ik}^o) + (\theta - \theta_0)\times$ $[B\ (r_{ij} - r_{ij}^o) + C\ (r_{ik} - r_{ik}^o)]$
mmbd **-mbd**	MM3 angle bend [35]	k	θ_0			$U(\theta) = k\ \Delta^2[1 - 1.4\cdot10^{-2}\Delta + 5.6\cdot10^{-5}\Delta^2$ $-7.0\cdot10^{-7}\Delta^3 + 2.2\cdot10^{-8}\Delta^4)]\ ;\ \Delta = \theta - \theta_0$
kky **-kky**	KKY [51]	f_k	θ_0	g_r	r_o	$U(\theta) = 2\ f_k\ \sqrt{K_{ij}\cdot K_{ik}}\ \sin^2[(\theta - \theta_0)]\,;$ $K_{ij} = 1/\left[\exp\left[g_r(r_{ij} - r_o)\right] + 1\right]$

†θ is the *i*-*j*-*k* angle.

Note: valence angle potentials with a dash (-) as the first character of the keyword, do not contribute to the *excluded atoms list* (see Section 2). In this case DL_POLY_4 will calculate the non-bonded pair potentials between the described atoms.

The meaning of the variables 1-3,6-7 is given in Table 10.7. The variables 4 and 5 specify the scaling factor for the 1-4 electrostatic and van der Waals non-bonded interactions respectively.

This directive (and associated data records) need not be specified if the molecule contains no dihedral angle terms. See the note on the atomic indices appearing under the **shell** directive above.

Table 10.7: Dihedral Angle Potentials

key	potential type	Variables (1-3,6-7)			functional form‡
tab	Tabulation, see Sections 2.2.5 2.2.6 10.1.9				tabulated potential in TABDIH file
cos	Cosine	A	δ	m	$U(\phi) = A\ [1 + \cos(m\phi - \delta)]$
harm	Harmonic	k	ϕ_0		$U(\phi) = \frac{k}{2}\ (\phi - \phi_0)^2$
hcos	Harmonic cosine	k	ϕ_0		$U(\phi) = \frac{k}{2}\ (\cos(\phi) - \cos(\phi_0))^2$
cos3	Triple cosine	A_1	A_2	A_3	$U(\phi) = \frac{1}{2}\ \{A_1\ (1 + \cos(\phi)) + A_2\ (1 - \cos(2\phi)) + A_3\ (1 + \cos(3\phi))\}$
ryck	Ryckaert-Bellemans [41]	A			$U(\phi) = A\ \{a + b\ \cos(\phi) + c\ \cos^2(\phi) + d\ \cos^3(\phi) + e\ \cos^4(\phi) + f\ \cos^5(\phi)\}$
rbf	Fluorinated Ryckaert-Bellemans [42]	A			$U(\phi) = A\ \{a + b\ \cos(\phi) + c\ \cos^2(\phi) + d\ \cos^3(\phi) + e\ \cos^4(\phi) + f\ \cos^5(\phi)) + g\ \exp(-h(\phi - \pi)^2))\}$
opls	OPLS torsion	A_0 A_3	A_1 ϕ_0	A_2	$U(\phi) = A_0 + \frac{1}{2}\ \{A_1\ (1 + \cos(\phi - \phi_0)) + A_2\ (1 - \cos(2(\phi - \phi_0))) + A_3\ (1 + \cos(3(\phi - \phi_0)))\}$

‡ϕ is the i-j-k-l dihedral angle.

12. **inversions *n***

where n is the number of inversion interactions present in the molecule. Each of the following n records contains:

inversion key	a4	potential key, see Table 10.8
index 1 (i)	integer	first atomic site index (central site)
index 2 (j)	integer	second atomic site index
index 3 (k)	integer	third atomic site index
index 4 (l)	integer	fourth atomic site index
variable 1	real	potential parameter, see Table 10.8
variable 2	real	potential parameter, see Table 10.8
variable 3	real	potential parameter, see Table 10.8

The meaning of the variables 1-2 is given in Table 10.8.

This directive (and associated data records) need not be specified if the molecule contains no inversion angle terms. See the note on the atomic indices appearing under the **shell** directive above.

Table 10.8: Inversion Angle Potentials

key	potential type	Variables (1-3)			functional form‡
tab	Tabulation, see Sections 2.2.8 10.1.9				tabulated potential in TABINV file
harm	Harmonic	k	ϕ_0		$U(\phi) = \frac{k}{2}\ (\phi - \phi_0)^2$
hcos	Harmonic cosine	k	ϕ_0		$U(\phi) = \frac{k}{2}\ (\cos(\phi) - \cos(\phi_0))^2$
plan	Planar	A			$U(\phi) = A\ [1 - \cos(\phi)]$
xpln	Extended planar	k	m	ϕ_0	$U(\phi) = \frac{k}{2}\ [1 - \cos(m\ \phi - \phi_0)]$
calc	Calcite 2.2.9	A	B		$U(u) = Au^2 + Bu^4$

‡ϕ is the i-j-k-l inversion angle.

Note that the calcite potential is not dependent on an angle ϕ, but on a *displacement* u. See Section 2.2.9 for details.

13. **finish**
This directive is entered to signal to DL_POLY_4 that the entry of the details of a molecule has been completed.

The entries for a second molecule may now be entered, beginning with the *name-of-molecule* record and ending with the **finish** directive.

The cycle is repeated until all the types of molecules indicated by the **molecules** directive have been entered.

The user is recommended to look at the example FIELD files in the *data* directory to see how typical FIELD files are constructed.

Non-bonded Interactions

Non-bonded interactions are identified by atom types as opposed to specific atomic indices. The following different types of non-bonded potentials are available in DL_POLY_4; **vdw** - van der Waals pair, **metal** - metal, **tersoff** - Tersoff, **tbp** - three-body and **fbp** - four-body. Each of these types is specified by a specific **keyword** as described bellow.

When DL_POLY_4 is cross-compiled with an OpenKIM functionality (see Section 2.10), it is possible to specify a complete model of inter-molecular interactions, by calling the OpenKIM model name provided it available in your local KIM library. Two **keyword**s are employed when using OpenKIM IMs, one to select the IM and perform necessary initialisation (**kim_init**), and the other (**kim_interactions**) to define a mapping between atom types in DL_POLY_4 to the available species in the OpenKIM IM.

1. **kim_init *model_name***
where the *model_name* is the OpenKIM model identifier, it uses the information retrieved from the OpenKIM repository to initialise and activate OpenKIM IMs for use in the DL_POLY_4.

2. **kim_interactions *site_names***
where *site_names* defines a list of unique atomic names. It defines a mapping between atom types in

DL_POLY_4 to the available species in the OpenKIM IM. For example, consider an OpenKIM IM that supports Si and C species. If the DL_POLY_4 simulation has four atoms, where the first three are Si, and the fourth is C, the *kim_interactions* would be used as:

```
kim_interactions    Si   C
```

The Si and C arguments map the DL_POLY_4 atom types to the Si and C species as defined within KIM PM.

In addition to the usual DL_POLY_4 error messages, the KIM library itself may generate errors, which should be printed to the screen. In this case, it is also useful to check the *kim.log* file for additional error information. The file *kim.log* should be generated in the same directory where DL_POLY_4 is running.

Note that although a KIM model fully describes a model system, it is still possible to specify further, complementing intra- and inter-molecular interactions in the FIELD! This is a viable option only when the model system is extended beyond what the specific KIM model is intended to describe.

By default, all the species in the DL_POLY_4 system should match to the available species in the OpenKIM IM.

Note that there is an **experimental feature**, implemented in the DL_POLY_4, where one can use a KIM IM in a hybrid style. In this case, one can create a system where part of species are interacting using a KIM IM (e.g., a machine learning model in KIM), and the rest of the species are interacting with the internal DL_POLY_4 intra- and inter-molecular interactions provided in the FIELD. While this new feature provides great flexibility, it is not fully compliant with the KIM API interface standard. Thus using this feature some of the KIM IMs might fail with the error message *unexpected species code detected* or *unknown species detected*.

3. **vdw *n***
where n is the number of pair potentials to be entered. It is followed by n records, each specifying a particular pair potential in the following manner:

atmnam 1	a8	first atom type
atmnam 2	a8	second atom type
key	a4	potential key, see Table 10.9
variable 1	real	potential parameter, see Table 10.9
variable 2	real	potential parameter, see Table 10.9
variable 3	real	potential parameter, see Table 10.9
variable 4	real	potential parameter, see Table 10.9
variable 5	real	potential parameter, see Table 10.9
variable 6	real	potential parameter, see Table 10.9
variable 7	real	potential parameter, see Table 10.9

The variables pertaining to each potential are described in Table 10.9.

Note that any pair potential not specified in the FIELD file, will be assumed to be zero.

4. **metal *n***
where n is the number of metal potentials to be entered. It is followed by n records, each specifying a particular metal potential in the following manner:

atmnam 1	a8	first atom type
atmnam 2	a8	second atom type
key	a4	potential key, see Table 10.10
variable 1	real	potential parameter, see Table 10.10
variable 2	real	potential parameter, see Table 10.10

Table 10.9: Pair Potentials

key	potential type	Variables (1-4,5-7)				functional form
tab	Tabulation, see Sections 2.3.1 10.1.7					tabulated potential in TABLE file
12-6	12-6	A	B			$U(r) = \left(\frac{A}{r^{12}}\right) - \left(\frac{B}{r^{6}}\right)$
lj	Lennard-Jones	ϵ	σ			$U(r) = 4\epsilon\left[\left(\frac{\sigma}{r}\right)^{12} - \left(\frac{\sigma}{r}\right)^{6}\right]$
ljc	LJ cohesive [45]	ϵ	σ	c		$U(r) = 4\epsilon\left[\left(\frac{\sigma}{r}\right)^{12} - c\ \left(\frac{\sigma}{r}\right)^{6}\right]$
ljf	LJ Frenkel [46]	ϵ	σ	r_c		see Eq. 2.88
nm	n-m [47, 48]	E_o	n	m	r_0	$U(r) = \frac{E_o}{(n-m)}\left[m\left(\frac{r_o}{r}\right)^{n} - n\left(\frac{r_o}{r}\right)^{m}\right]$
buck	Buckingham	A	ρ	C		$U(r) = A\ \exp\left(-\frac{r}{\rho}\right) - \frac{C}{r^{6}}$
bhm	Born-Huggins -Meyer	A D	B	σ	C	$U(r) = A\ \exp[B(\sigma - r)] - \frac{C}{r^{6}} - \frac{D}{r^{8}}$
hbnd	12-10 H-bond	A	B			$U(r) = \left(\frac{A}{r^{12}}\right) - \left(\frac{B}{r^{10}}\right)$
snm	Shifted force† n-m [47, 48]	E_o r_c‡	n	m	r_0	$U(r) = \frac{\alpha E_o}{(n-m)} \times$ $\left[m\beta^{n}\left\{\left(\frac{r_o}{r}\right)^{n} - \left(\frac{1}{\gamma}\right)^{n}\right\} - n\beta^{m}\left\{\left(\frac{r_o}{r}\right)^{m} - \left(\frac{1}{\gamma}\right)^{m}\right\}\right]$ $+\frac{nm\alpha E_o}{(n-m)}\left(\frac{r-\gamma r_o}{\gamma r_o}\right)\left\{\left(\frac{\beta}{\gamma}\right)^{n} - \left(\frac{\beta}{\gamma}\right)^{m}\right\}$
mors	Morse	E_0	r_0	k		$U(r) = E_0[\{1 - \exp(-k(r - r_0))\}^2 - 1]$
wca	Shifted* [49] Weeks-Chandler-Andersen	ϵ	σ‡	Δ		$U(r) = 4\epsilon\left[\left(\frac{\sigma}{r-\Delta}\right)^{12} - \left(\frac{\sigma}{r-\Delta}\right)^{6}\right] + \epsilon : r_{ij} < 2^{\frac{1}{6}}\ \sigma + \Delta$ $U(r) = 0 \quad : r_{ij} \geq 2^{\frac{1}{6}}\ \sigma + \Delta$
dpd	Standard DPD [50] (Groot-Warren)	A	r_c‡			$U(r) = \frac{A}{2}\ r_c\ \left(1 - \frac{r}{r_c}\right)^{2}\ :\ r < r_c$ $U(r) = 0 \quad :\ r \geq r_c$
14-7	14-7 buffered AMOEBA FF [51]	ϵ	r_o			$U(r) = \epsilon\left(\frac{1.07}{(r_{ij}/r_o)+0.07}\right)^{7}\left(\frac{1.12}{(r_{ij}/r_o)^{7}+0.12} - 2\right)$
mstw	Morse modified	E_0	r_0	k	c	$U(r) = E_0\{[1 - \exp\left(-k\left(r - r_0\right)\right)]^2 - 1\} + \frac{c}{r^{12}}$
ryd	Rydberg	a	b	ρ		$U(r) = (a + br)\exp\left(-r/\rho\right)$
zbl	ZBL	Z_1	Z_2			$U(r) = \frac{Z_1\ Z_2\ e^2}{4\pi\varepsilon_0\varepsilon_r}\sum_{i=1}^{4} b_i \exp\left(-c_i r/a\right)$

continues on next page

Pair Potentials - *continued from previous page*

key	potential type	Variables (1-4,5-7)				functional form
zbls	ZBL mixed with Morse	Z_1 E_0	Z_2 r_0	r_m k	ξ	$U(r) = f(r)\,U_{ZBL}(r) \;+$ $+\;(1 - f(r))\,U_{morse}(r)$
zblb	ZBL mixed with Buckingham	Z_1 A	Z_2 ρ	r_m C	ξ	$U(r) = f(r)\,U_{ZBL}(r) \;+$ $+\;(1 - f(r))\,U_{buckingham}(r)$
mlj	Lennard-Jones tapered with MDF	ϵ	σ	r_i		$U(r) = 4\epsilon\left[\left(\frac{\sigma}{r}\right)^{12} - \left(\frac{\sigma}{r}\right)^{6}\right] f(r)$
mbuc	Buckingham tapered with MDF	A	ρ	C	r_i	$U(r) = \left[A\ \exp\left(-\frac{r}{\rho}\right) - \frac{C}{r^6}\right] f(r)$
m126	12-6 Lennard-Jones tapered with MDF	A	B	r_i		$U(r) = \left(\frac{A}{r^{12}} - \frac{B}{r^6}\right) f(r)$

[†] Note: in this formula the terms α, β and γ are compound expressions involving the variables E_o, n, m, r_0 and r_c. See Section 2.3.1 for further details.
[‡] Note: All local potential cutoffs, r_c, default to the general van der Waals cutoff, `rvdw`, or the general domain decomposition cutoff, `rcut`, if unspecified or set to zero in the FIELD file! Similarly, if the specified value of `rvdw` (and/or `rcut`) in CONTROL is found shorter than any of r_c (including the WCA equivalent $2^{\frac{1}{6}}\ \sigma + \Delta$) values specified in FIELD then `rvdw` (and/or `rcut`) will be reset by DL_POLY_4 to the largest of all values!
[*] Note: Δ defaults to zero if $|\Delta| > 0.5\ \sigma$ or it is not specified in the FIELD file.

variable 3	real	potential parameter, see Table 10.10
variable 4	real	potential parameter, see Table 10.10
variable 5	real	potential parameter, see Table 10.10
variable 6	real	potential parameter, see Table 10.10
variable 7	real	potential parameter, see Table 10.10
variable 8	real	potential parameter, see Table 10.10
variable 9	real	potential parameter, see Table 10.10

The variables pertaining to each potential are described in Table 10.10.

5. **rdf *n***
where n is the number of RDF pairs to be entered. It is followed by n records, each specifying a particular RDF pair in the following manner:

atmnam 1	a8	first atom type
atmnam 2	a8	second atom type

By default in DL_POLY_Classic and DL_POLY_4 every **vdw** and **met** potential specifies an RDF pair. If the control option **rdf *f*** is specified in the CONTROL file then all pairs defined in **vdw** and/or **met** potentials sections will also have their RDF calculated. The user has two choices to enable the calculation of RDFs in systems with force fields that do not have **vdw** and/or **met** potentials: (i) to define fictitious potentials with zero contributions or (ii) to use **rdf *n*** option - which not only provides a neater way for specification of RDF pairs but also better memory efficiency since DL_POLY_4 will not allocate (additional) potential arrays for fictitious interactions that will not be used. (This option is not available in DL_POLY_Classic.)

Note that **rdf** and **vdw/met** are not complementary - i.e. if the former is used in FIELD none of the pairs defined by the latter will be considered for RDF calculations.

Table 10.10: Metal Potential

key	potential type	Variables (1-5,6-9)					functional form
eam	EAM						tabulated potential in TABEAM
eeam	EEAM						tabulated potential in TABEAM
2bea	2BEAM						tabulated potential in TABEAM
2bee	2BEEAM						tabulated potential in TABEAM
fnsc	Finnis-Sinclair	c_0	c_1	c_2	c	A	$U_i(r)=\frac{1}{2}\sum_{j\neq i}(r_{ij}-c)^2(c_0+c_1r_{ij}+c_2r_{ij}^2)-A\sqrt{\rho_i}$;
		d	β				$\rho_i=\sum_{j\neq i}\left[(r_{ij}-d)^2+\beta\frac{(r_{ij}-d)^3}{d}\right]$
exfs	Extended	c_0	c_1	c_2	c_3	c_4	$U_i(r)=\frac{1}{2}\sum_{j\neq i}(r_{ij}-c)^2(c_0+c_1r_{ij}+c_2r_{ij}^2+c_3r_{ij}^3+c_4r_{ij}^4)$
	Finnis-Sinclair	c	A	d	B		$-A\sqrt{\rho_i}$; $\rho_i=\sum_{j\neq i}\left[(r_{ij}-d)^2+B^2(r_{ij}-d)^4\right]$
stch	Sutton-Chen	ϵ	a	n	m	c	$U_i(r)=\epsilon\left[\frac{1}{2}\sum_{j\neq i}\left(\frac{a}{r_{ij}}\right)^n-c\sqrt{\rho_i}\right]$; $\rho_i=\sum_{j\neq i}\left(\frac{a}{r_{ij}}\right)^m$
gupt	Gupta	A	r_0	p	B	q_{ij}	$U_i(r)=\sum_{j\neq i}A\exp\left(-p\frac{r_{ij}-r_0}{r_0}\right)-B\sqrt{\rho_i}$;
							$\rho_i=\sum_{j\neq i}\exp\left(-2q_{ij}\frac{r_{ij}-r_0}{r_0}\right)$
mbpc	MBPC†	ϵ	a	m	α	$r_{\rm o}$	$U_i(r)=-\epsilon\sqrt{\rho_i}$; $\rho_i=\sum_{j\neq i}\left(\frac{a}{r_{ij}^m}\right)\frac{1}{2}\left[1+\mathrm{erf}\left(\alpha(r_{ij}-r_{\rm o})\right)\right]$

†**Note** that the parameters α and $r_{\rm o}$ must be the same for all defined potentials of this type. DL_POLY_4 will set $\alpha = {\rm Max}(0,\alpha_{pq})$ and $r_{\rm o} = {\rm Max}(0, r_{{\rm o}\_pq})$ for all defined interactions of this type between species p and q. If after this any is left undefined, i.e. zero, the undefined entities will be set to their defaults: $\alpha = 20$ and $r_{\rm o} = {\rm Min}(1.5, 0.2\ r_{\rm cut})$.

The selected RDFs are calculated in the RDF_COLLECT, RDF_EXCL_COLLECT, RDF_FRZN_COLLECT and RDF_COMPUTE by collecting distance information from all two-body pairs as encountered in the Verlet neighbour list created in the LINK_CELL_PAIRS routine within the TWO_BODY_FORCES routine. In the construction of the Verlet neighbour list, pairs of particles (part of the exclusion list) are excluded. The exclusion list contains particles that are part of:

- core-shell units
- bond constraints
- chemical bonds, that are NOT distance restraints
- valence angles, that are NOT distance restraints
- dihedrals
- inversions
- frozen particles

RDF pairs containing type(s) of particles that fall in this list will be polluted. However, there are many ways to overcome such effects.

6. **tersoff** $\boldsymbol{n}$
   where n is the number of specified Tersoff potentials. There are two types of Tersoff potential forms that cannot be mixed (used simultaneously). They are shorthanded as **ters** and **kihs** atomkeys.

   - **ters** atomkey expects $2n$ records specifying n particular Tersoff single atom type parameter sets and $n(n+1)/2$ records specifying cross atom type parameter sets in the following manner:

potential 1 : record 1		
atmnam	a8	atom type
key	a4	potential key, see Table 10.11
variable 1	real	potential parameter, see Table 10.11
variable 2	real	potential parameter, see Table 10.11
variable 3	real	potential parameter, see Table 10.11
variable 4	real	potential parameter, see Table 10.11
variable 5	real	cutoff range for this potential (Å)
potential 1 : record 2		
variable 6	real	potential parameter, see Table 10.11
variable 7	real	potential parameter, see Table 10.11
variable 8	real	potential parameter, see Table 10.11
variable 9	real	potential parameter, see Table 10.11
variable 10	real	potential parameter, see Table 10.11
variable 11	real	potential parameter, see Table 10.11
...	...	...
...	...	...
potential n **: record** $2n-1$		
...	...	...
potential n **: record** $2n$		
...	...	...
cross term 1 **: record** $2n+1$		
atmnam 1	a8	first atom type
atmnam 2	a8	second atom type
variable a	real	potential parameter, see Table 10.11
variable b	real	potential parameter, see Table 10.11
variable c	real	potential parameter, see Table 10.11
...	...	...
...	...	...

cross term $n(n+1)/2$ **: record** $2n + n(n+1)/2$		
...	...	...

- **kihs** atomkey expects $3n$ records specifying n particular Tersoff single atom type parameter sets in the following manner:

potential 1 : record 1		
atmnam	a8	atom type
key	a4	potential key, see Table 10.11
variable 1	real	potential parameter, see Table 10.11
variable 2	real	potential parameter, see Table 10.11
variable 3	real	potential parameter, see Table 10.11
variable 4	real	potential parameter, see Table 10.11
variable 5	real	cutoff range for this potential (Å)
potential 1 : record 2		
variable 6	real	potential parameter, see Table 10.11
variable 7	real	potential parameter, see Table 10.11
variable 8	real	potential parameter, see Table 10.11
variable 9	real	potential parameter, see Table 10.11
variable 10	real	potential parameter, see Table 10.11
variable 11	real	potential parameter, see Table 10.11
potential 1 : record 3		
variable 12	real	potential parameter, see Table 10.11
variable 13	real	potential parameter, see Table 10.11
variable 14	real	potential parameter, see Table 10.11
variable 15	real	potential parameter, see Table 10.11
variable 16	real	potential parameter, see Table 10.11
...	...	...
...	...	...
potential n **: record** $2n-1$		
...	...	...
potential n **: record** $2n$		
...	...	...

The variables pertaining to each potential are described in Table 10.11.

Note that the fifth variable is the range at which the particular tersoff potential is truncated. The distance is in Å.

Table 10.11: Tersoff Potential

key	potential type	Variables (1-5,6-11,a-c/12-16)						functional form
ters	Tersoff	A	a	B	b	R		Potential forms
	(single)	S	β	η	c	d	h	
	(cross)	χ	ω	δ				as shown in
kihs	KIHS	A	a	B	b	R		
		S	η	δ	c_1	c_2	c_3	Section 2.3.3
		c_4	c_5	h	α	β		

7. **tbp *n***
   where n is the number of three-body potentials to be entered. It is followed by n records, each specifying a particular three-body potential in the following manner:

atmnam 1 (i)	a8	first atom type
atmnam 2 (j)	a8	second (central) atom type
atmnam 3 (k)	a8	third atom type
key	a4	potential key, see Table 10.12
variable 1	real	potential parameter, see Table 10.12
variable 2	real	potential parameter, see Table 10.12
variable 3	real	potential parameter, see Table 10.12
variable 4	real	potential parameter, see Table 10.12
variable 5	real	cutoff range for this potential (Å)

   The variables pertaining to each potential are described in Table 10.12.

   Note that the fifth variable is the range at which the three body potential is truncated. The distance is in Å, measured from the central atom. Note that interactions defined with less than four potential parameters must provide zeros for the ones up to the cutoff one.

Table 10.12: Three-body Potentials

key	potential type	Variables (1-4)				functional form†
harm	Harmonic	k	θ_0	0	0	$U(\theta) = \frac{k}{2}\ (\theta - \theta_0)^2$
thrm	Truncated harmonic	k	θ_0	ρ	0	$U(\theta) = \frac{k}{2}\ (\theta - \theta_0)^2\ \exp[-(r_{ij}^8 + r_{ik}^8)/\rho^8]$
shrm	Screened harmonic	k	θ_0	ρ_1	ρ_2	$U(\theta) = \frac{k}{2}\ (\theta - \theta_0)^2 \exp[-(r_{ij}/\rho_1 + r_{ik}/\rho_2)]$
bvs1	Screened Vessal [36]	k	θ_0	ρ_1	ρ_2	$U(\theta) = \frac{k}{8(\theta_0-\pi)^2} \left\{ \left[(\theta_0 - \pi)^2 - (\theta - \pi)^2 \right]^2 \right\} \times \exp[-(r_{ij}/\rho_1 + r_{ik}/\rho_2)]$
bvs2	Truncated Vessal [37]	k	θ_0	a	ρ	$U(\theta) = k\ (\theta - \theta_0)^2\ \left[\theta^a (\theta - \theta_0)^2 (\theta + \theta_0 - 2\pi)^2 + \frac{a}{2}\pi^{a-1}(\theta_0 - \pi)^3 \right]\ \exp[-(r_{ij}^8 + r_{ik}^8)/\rho^8]$
hbnd	H-bond [19]	D_{hb}	R_{hb}	0	0	$U(\theta) = D_{hb}\ \cos^4(\theta) \times [5(R_{hb}/r_{jk})^{12} - 6(R_{hb}/r_{jk})^{10}]$

†θ is the i-j-k angle.

8. **fbp *n***
   where n is the number of four-body potentials to be entered. It is followed by n records, each specifying a particular four-body potential in the following manner:

atmnam 1 (i)	a8	first (central) atom type
atmnam 2 (j)	a8	second atom type
atmnam 3 (k)	a8	third atom type
atmnam 4 (l)	a8	fourth atom type
key	a4	potential key, see Table 10.13
variable 1	real	potential parameter, see Table 10.13
variable 2	real	potential parameter, see Table 10.13
variable 3	real	cutoff range for this potential (Å)

The variables pertaining to each potential are described in Table 10.13.

Note that the third variable is the range at which the four-body potential is truncated. The distance is in Å, measured from the central atom. Note that interactions idefined with less than two potential parameters must provide zeros for the ones up to the cutoff one.

Table 10.13: Four-body Potentials

key	potential type	Variables (1-2)		functional form‡
harm	Harmonic	k	ϕ_0	$U(\phi) = \frac{k}{2}\ (\phi - \phi_0)^2$
hcos	Harmonic cosine	k	ϕ_0	$U(\phi) = \frac{k}{2}\ (\cos(\phi) - \cos(\phi_0))^2$
plan	Planar	A	0	$U(\phi) = A\ [1 - \cos(\phi)]$

‡ϕ is the i-j-k-l four-body angle.

10.1.3.3 External Field

The presence of an external field is flagged by the directive:

extern

The following line in the FIELD file must contain another directive indicating what type of field is to be applied, followed by the field parameters in the following manner:

field key	a4	external field key, see Table 10.14
variable 1	real	potential parameter, see Table 10.14
variable 2	real	potential parameter, see Table 10.14
variable 3	real	potential parameter, see Table 10.14
variable 4	real	potential parameter, see Table 10.14
variable 5	real	potential parameter, see Table 10.14
variable 6	real	potential parameter, see Table 10.14

The variables pertaining to each field potential are described in Table 10.14.

Note: only one type of field can be applied at a time.

Note that external force parameters are read in terms of the specified energy units and the general DL_POLY units so that the two sides of the equation defining the field are balanced. For example, the magnetic field units, $\underline{H} = (H_1, H_2, H_3)$, in the DL_POLY FIELD scope will follow from the interaction definition as seen in Table 10.14:

$$\begin{aligned}
\underline{F} &= q\ (\underline{v} \times \underline{H}) \qquad \texttt{therefore,} \\
[H] &= \frac{[F]}{[q]\ [v]} = \frac{[m]\ [a]}{[q]\ [v]} \\
[H] &= \frac{\texttt{Dalton Å/ps}^2}{\texttt{proton Å/ps}} = \frac{\texttt{Dalton}}{\texttt{proton ps}} \qquad (10.8) \\
[H] &= 1.037837512 \times 10^4\ \texttt{Tesla} \qquad (10.9) \\
H(DL\_POLY) &= H(MKS)\ 1.037837512 \times 10^4 \quad .
\end{aligned}$$

Thus to apply a magnetic field of 1 Tesla along the y axis, one could specify in FIELD the following:

Table 10.14: External Fields

key	potential type	Variables (1-6)						functional form
elec	Electric Field	E_x	E_y	E_z				$\underline{F} = q\ \underline{E}$
oshr	Oscillating Shear	A	n					$\underline{F}_x = A\ \cos(2n\pi \cdot z/L_z)$
shrx	Continuous Shear	A	z_0					$\underline{v}_x = \frac{A}{2}\frac{\|z\|}{z}\ :\ \|z\|\ >\ z_0$
grav	Gravitational Field	G_x	G_y	G_z				$\underline{F} = m\ \underline{G}$
magn	Magnetic Field	H_x	H_y	H_z				$\underline{F} = q\ (\underline{v} \times \underline{H})$
sphr	Containing Sphere	A	R_0	n	$R_{\rm cut}$			$\underline{F} = A\ (R_0 - r)^{-n}\ :\ r > R_{\rm cut}$
zbnd	Repulsive Wall	A	z_0	$f = \pm 1$				$\underline{F} = A\ (z_0 - z)\ :\ f \cdot z > f \cdot z_0$
xpis	X-Piston	i_{gid}	j_{gid}	$P_{\rm katm}$				$\underline{F}_x = \frac{P\cdot \text{Area}(\perp \text{X-}dir)}{[\sum_{k=i}^{j} m_k]/m_k} : \forall\ k = i,..,j$
zres	Molecule in HR Zone	i_{gid}	j_{gid}	A	z_{mn}	z_{mx}		$\underline{F}_z = \begin{cases} A(z_{cm} - z_{mx}) : z_{cm} > z_{mx} \\ A(z_{mn} - z_{cm}) : z_{cm} < z_{mn} \end{cases}$
zrs−	HR Zone (push out)	i_{gid}	j_{gid}	A	z_{mn}	z_{mx}		$\underline{F}_z = \begin{cases} A(z - z_{mx}) : z \geq \frac{z_{mx}+z_{mn}}{2} \\ A(z_{mn} - z) : z < \frac{z_{mx}+z_{mn}}{2} \end{cases}$
zrs+	HR Zone (pull in)	i_{gid}	j_{gid}	A	z_{mn}	z_{mx}		$\underline{F}_z = \begin{cases} A(z - z_{mx}) : z > z_{mx} \\ A(z_{mn} - z) : z < z_{mn} \end{cases}$
osel	Osc. Electric Field	E_x	E_y	E_z	$\omega_{\rm ps^{-1}}$			$\underline{F} = q\ \underline{E}\ \sin(2\pi\omega t)$
ushr	umbrella sampling [82] harm. constraint [83]	i^A_{gid}	j^A_{gid}	i^B_{gid}	j^B_{gid}	k	R_0	$U_{AB} = \frac{k}{2}(R_{AB} - R_0)^2$

```
UNITS internal
...
external
magnetic 0  1/1.037837512e04   0
...
```

when working in DL_POLY internal units. If we worked in *unit* units then

$$\begin{aligned} H &\propto Energy \\ Energy(DL\_POLY) &= Energy(unit)\ k_{unit\to DL\_POLY} \\ H(DL\_POLY) &= H(MKS)\ 1.037837512 \times 10^4 \\ H(unit) &= H(MKS)\ \frac{1.037837512 \times 10^4}{k_{unit\to DL\_POLY}}\quad , \end{aligned} \tag{10.10}$$

with the following conversion factors values:

$$\begin{aligned} k_{eV\to DL\_POLY} &= 9648.530821 \\ k_{kcal/mol\to DL\_POLY} &= 418.4 \\ k_{kJ/mol\to DL\_POLY} &= 100.0 \\ k_{K/Boltz\to DL\_POLY} &= 0.831451115 \\ k_{DL\_POLY\to DL\_POLY} &= 1.0 \quad . \end{aligned} \tag{10.11}$$

Obviously, for eV units

$$\begin{aligned} H(unit) &= H(MKS)\,\frac{1.037837512\times 10^4}{k_{unit\to DL\_POLY}} \\ k_{eV\to DL\_POLY} &= 9648.530821 \\ H(eV) &= H(MKS)\;1.07564305 \quad , \end{aligned} \tag{10.12}$$

the FIELD file should be amended to read:

```
UNITS eV
...
external
magnetic 0  1/1.07564305   0
...
```

10.1.3.4 crd

crd i j k

This section defines the atomic pairs to use for both coordination and angular distribution calculations. The start of this section is given by the keyword **crd** followed by the i j k records:

i = The number of unique atomic pairs

i number of the following pattern: atom1 atom2 bond_length

j = The number of unique atoms you are building the pairs from

j number of the following pattern: atom minimun_displacement

k = The number of lists to build coordination and angular distribution between

k number of the following pattern: atom atom ... - atom atom ...

Example in the FIELD File

```
crd 2 3 1
B O 1.9
Si O 2.0
B 0.9
Si 1.0
O 1.0
Si B - O
```

10.1.3.5 Closing the FIELD File

The FIELD file must be closed with the directive:

close
which signals the end of the force field data. Without this directive DL_POLY_4 will abort.

10.1.4 The MPOLES File

The MPOLES file serves to define the multipolar momenta of the sites defined in FIELD (see Section 10.1.3). It is only read by DL_POLY_4 when the **mult**ipolar order n directive is specified at the top of FIELD. The file is read by the subroutine READ_MPOLES. The MPOLES file and has the same molecular topology syntax and rules as FIELD except that
(i) it only understands the stoichiometry information; and
(ii) will abort should more FIELD like detail is supplied.
Thus the molecular topology/stoichiometry information must explicitly match that in FIELD.

Example from a water system with multipolar momenta beyond the single point charges is shown below:

```
Puddle Test Case (32000 AMOEBA Waters)

MOLECULES 1

SPC WATER
NUMMOLS   32000

ATOMS     3

O  2  1  0.837  0.39
-0.51966
 0.00000   0.00000   0.14279
 0.37928   0.00000   0.00000  -0.41809   0.00000   0.03881

H  2  2  0.496  0.39
0.25983
-0.03859   0.00000  -0.05818
-0.03673   0.00000  -0.00203  -0.10739   0.00000   0.14412

FINISH

CLOSE
```

10.1.4.1 The MPOLES File Format

The file is free-formatted and not case-sensitive (except for the site names). Every line is treated as a command sentence (record). Commented records (beginning with a #) and blank lines are not processed and may be added to aid legibility (see example above). Records must be limited in length to 200 characters. Records are read in words, as a word must not exceed 40 characters in length. Words are recognised as such by separation by one or more space characters. The contents of the file are variable and are defined by the use of **directives**. Additional information is associated with the directives.

10.1.4.2 Definitions of Variables in the MPOLES File

The file divides into two sections: general information, molecular descriptions.

General information

The first viable record in the MPOLES file is the title, which is mandatory.

record 1

header	a200	field file header

Molecular details

It is important for the user to understand that there is an organisational correspondence between the FIELD file and the CONFIG file described above. It is required that the order of specification of molecular types and their atomic constituents in the FIELD file follows the order of indices in which they appear in the CONFIG file. Failure to adhere to this common sequence will be detected by DL_POLY_4 and result in premature termination of the job. It is therefore essential to work from the CONFIG file when constructing the FIELD file. It is not as difficult as it sounds!

The entry of the molecular details begins with the mandatory directive:

molecules *n*

where n is an integer specifying the number of different *types* of molecule appearing in the FIELD file. Once this directive has been encountered, DL_POLY_4 enters the *molecular description* environment in which only molecular description keywords and data are valid.

Immediately following the **molecules** directive, are the records defining individual molecules:

1. *name-of-molecule*
   which can be any character string up to 200 characters in length. (**Note**: this is not a directive, just a simple character string.)
2. **nummols *n***
   where n is the number of times a molecule of this type appears in the simulated system. The molecular data then follow in subsequent records:
3. **atoms *n***
   where n indicates the number of atoms in this type of molecule. A number of records follow, each giving details of the atoms in the molecule i.e. site names, masses and charges. Each record carries the entries:

`sitnam`	a8	atomic site name
`order`	integer	multipolar order supplied
`nrept`	integer	repeat counter
α	real(1)	*optional* atomic polarisability (in Å^3)
a	real(1)	*optional* Thole dumping factor

   The integer `nrept` may be omitted (in which case a value or 1 is assumed) **only if** no further optional directives are provided! A number greater than 1 specified here indicates that the next (`nrept`-1) entries in the CONFIG file are ascribed the atomic characteristics given in the current record. The sum of the repeat numbers for all atoms in a molecule should equal the number specified by the **atoms** directive.

   The atomic polarisability, α, and the Thole [77] dumping factor, a, are *optional* and are only parsed for the core (non-Druder) particles of core-shell units. If all polarisabilities as well as core and shell charges and associated force constants are well defined then a may default for (Druder nuclei) core particles if no Thole dumping is specified in both MPOLES and CONTROL.

   For each pole `order` specified DL_POLY_4 will read a new line that will specify the pole order momenta. If a site has a specified pole order, m, smaller than the one specified in FIELD, n, then the $m+1$ to n order poles' momenta will be zero. Similarly, if a site has momenta of poles of higher order than the one specified in FIELD, n, these will not be processed by DL_POLY_4.

   DL_POLY_4 will read same order momenta in their natural vector format:

`charge`	real(1)	scalar q (in protons)
`dipole`	real(3)	vector (x, y, z) (in protons/Å)
`quadrupole`	real(6)	vector (xx, xy, xz, yy, yz, zz) (in protons/Å^2)
`octupole`	real(10)	vector $(xxx, xxy, xxz, xyy, xyz, xzz, yyy, yyz, yzz, zzz)$ (in protons/Å^3)
`hexadecapole`	real(15)	vector $(xxxx, xxxy, xxxz, ..., yyyz, yyzz, yzzz, zzzz)$ (in protons/Å^4)

Note that the charge values supplied in FIELD will be overwritten with those supplied here!

Note, although algorithms in DL_POLY_4 could in principle handle any high pole order summation, in practice, however, DL_POLY_4 will abort if the order is higher than hexadecapole (order 4)!

4. **finish**
   This directive is entered to signal DL_POLY_4 that the entry of the details of a molecule has been completed.

   The entries for a second molecule may now be entered, beginning with the *name-of-molecule* record and ending with the **finish** directive.

   The cycle is repeated until all the types of molecules indicated by the **molecules** directive have been entered.

10.1.4.3 Closing the MPOLES File

The MPOLES file must be closed with the directive:

close
after which DL_POLY_4 will resume will processing the intermolecular part of FIELD.

10.1.5 The REFERENCE File

The REFERENCE has the same format and structure as CONFIG (see Section 10.1.2) file with the exception that `imcon` **MUST BE** $\neq 0$. REFERENCE may contain more or less particles than CONFIG does and may have particles with identities that are not defined in FIELD (see Section 10.1.3). The positions of these particles are used to define the crystalline lattice sites to which the particles in CONFIG compare during simulation when the defect detection option, **defe**cts, is used. REFERENCE is read by the subroutine DEFECTS_REFERENCE_READ.

10.1.6 The REVOLD File

This file contains statistics arrays from a previous job. It is not required if the current job is not a continuation of a previous run (i.e. if the **restart** directive is not present in the CONTROL file - see above). The file is unformatted and therefore not human readable. DL_POLY_4 normally produces the file REVIVE (see Section 10.2.10) at the end of a job which contains the statistics data. REVIVE should be copied to REVOLD before a continuation run commences. This may be done by the *copy* macro supplied in the *execute* sub-directory of DL_POLY_4.

10.1.6.1 Format

The REVOLD file is unformatted. All variables appearing are written in native working precision (see Section 8.3.5) real representation. Nominally, integer quantities (e.g. the timestep number `nstep`) are represented by the nearest real number. The contents are as follows (the dimensions of array variables are given in brackets, in terms of parameters from the SETUP_MODULE file - see Section 11.2.7).

record 1:

`nstep`	timestep of final configuration
`numacc`	number of configurations used in averages
`numrdf`	number of configurations used in RDF averages
`numzdn`	number of configurations used in Z-density averages
`time`	elapsed simulation time
`tmst`	elapsed simulation before averages were switched on
`chit`	thermostat related quantity (first)
`chip`	barostat related quantity
`cint`	thermostat related quantity (second)

record 2:

`eta`	scaling factors for simulation cell matrix elements (9)

record 3:

`stpval`	instantaneous values of thermodynamic variables (`mxnstk`)

record 4:

`sumval`	average values of thermodynamic variables (`mxnstk`)

record 5:

`ssqval`	fluctuation (squared) of thermodynamic variables (`mxnstk`)

record 6:

`zumval`	running totals of thermodynamic variables (`mxnstk`)

record 7:

`ravval`	rolling averages of thermodynamic variables (`mxnstk`)

record 8:

`stkval`	stacked values of thermodynamic variables (`mxstak`×`mxnstk`)

record 9:

`strcon`	constraint bond stress (9)

record 10:

`strpmf`	PMF constraint stress (9)

record 11:

`stress`	atomic stress (9)

record 12: (Optional)

`rdf`	RDF arrays (`mxgrdf`×`mxrdf`)

record 13: (Optional)

`usr`	umbrella sampling RDF array (`mxgusr`)

record 14: (Optional)

`zdens`	Z-density array (`mxgrdf`×`mxatyp`)

record 15: (Optional)

`vaf`	VAF arrays (sizes dependent on sampling frequency and VAF bin size)

10.1.6.2 Further Comments

Note that different versions of DL_POLY_4 may have a different order of the above parameters or include more or less such. Therefore different versions of DL_POLY_4 may render any existing REVOLD file unreadable by the code.

10.1.7 The TABLE File

The TABLE file provides an alternative way of reading in the short range potentials - in tabular form. This is particularly useful if an analytical form of the potential does not exist or is too complicated to specify in the VDW_GENERATE subroutine. The table file is read by the subroutine VDW_TABLE_READ (see Chapter 11).

The option of using tabulated potentials is specified in the FIELD file (see above). The specific potentials that are to be tabulated are indicated by the use of the **tab** keyword on the record defining the short range potential (see Table 10.9).

10.1.7.1 The TABLE File Format

The file is free-formatted but blank and commented lines are not allowed.

10.1.7.2 Definitions of Variables

record 1		
`header`	a200	file header
record 2		
`delpot`	real	mesh resolution in Å ($\texttt{delpot} = \frac{\texttt{cutpot}}{\texttt{ngrid}-4}$)
`cutpot`	real	cutoff used to define tables in Å
`ngrid`	integer	number of grid points in tables

The subsequent records define each tabulated potential in turn, in the order indicated by the specification in the FIELD file. Each potential is defined by a header record and a set of data records with the potential and force tables.

header record:		
`atom 1`	a8	first atom type
`atom 2`	a8	second atom type
potential data records: (*number of data records* = `Int`((`ngrid`+3)/4))		
`data 1`	real	data item 1
`data 2`	real	data item 2
`data 3`	real	data item 3
`data 4`	real	data item 4
force data records: (*number of data records* = `Int`((`ngrid`+3)/4))		
`data 1`	real	data item 1
`data 2`	real	data item 2
`data 3`	real	data item 3
`data 4`	real	data item 4

10.1.7.3 Further Comments

It should be noted that the number of grid points in the TABLE file should not be less than the number of grid points DL_POLY_4 is expecting. (This number is given by the parameter `mxgvdw` calculated in the SETUP_MODULE file - see Section 8.2.1 and 11.2.7.) DL_POLY_4 will re-interpolate the tables if `delpot` = $\frac{\texttt{cutpot}}{\texttt{ngrid}-4} < \texttt{dlrvdw} = \frac{\texttt{rvdw}}{\texttt{mxgvdw}-4}$ (usually when `ngrid` > `mxgvdw`), but will abort if `delpot` > `dlrvdw`.
The potential and force tables are used to fill the internal arrays `vvdw` and `gvdw` respectively (see Section 2.3.1). The contents of force arrays are derived from the potential via the formula:

$$G(r) = -r\frac{\partial}{\partial r}U(r) \quad . \tag{10.13}$$

Note, this is *not* the same as the true force.

During simulation, interactions beyond distance `cutpot` are discarded.

10.1.8 The TABEAM File

The TABEAM file contains the tabulated potential functions (no explicit analytic form) describing the EAM or EEAM metal interactions in the MD system. This file is read by the subroutine METAL_TABLE_READ (see Chapter 11).

10.1.8.1 The TABEAM File Format

The file is free-formatted but blank and commented lines are not allowed.

10.1.8.2 Definitions of Variables

record 1

`header`	a200	file header

record 2

`numpot`	integer	number of potential functions in file

For an n component alloy, `numpot` is

- $n(n+5)/2$ for the EAM potential *or*
- $3n(n+1)/2$ for the EEAM potential *or*
- $n(n+4)$ for the 2BEAM potential *or*
- $5n(n+1)/2$ for the 2BEEAM potential.

The subsequent records for an n component alloy define $n(n+1)/2$ *cross* pair potential functions - **pair**s keyword and

- EAM: n embedding functions (one for each atom type) - **embe**dding keyword, n electron density functions (one for each atom type) - **dens**ity keyword;
- EEAM: n embedding functions (one for each atom type) and n^2 for the EEAM potential (one for each non-commuting pair of atoms types);
- 2BEAM: n s-band embedding functions (one for each atom type) - **semb**edding keyword, $n(n+1)/2$ s-band density functions (one for each *cross*-pair of atoms types) - **sden**sity keyword, n d-band embedding functions (one for each atom type) - **demb**edding or **embe**edding keyword, and n d-band density functions (one for each atom types) - **dden**sity or **dens**ity keyword;
- 2BEEAM: n s-band embedding functions (one for each atom type) - **semb**edding keyword, n^2 s-band density functions (one for each non-commuting pair of atoms types) - **sden**sity keyword, n d-band embedding functions (one for each atom type) - **demb**edding or **embe**edding keyword, and n^2 d-band density functions (one for each non-commuting pair of atoms types) - **dden**sity or **dens**ity keyword.

The functions may appear in any random order in TABEAM as their identification is based on their unique keyword, defined first in the function's header record. The header record is followed by predefined number of data records **as a maximum of four data per record are read in** - allowing for incompletion of the very last record.

header record:

keyword	a4	type of EAM function: **pair**, **embe**d or **dens**ity, with 2B extension alternatives for the s-band - [**semb**ed and **sden**sity]

		and *d*-band - **demb**ed = **embe**d and **dden**sity = **dens**ity
`atom 1`	a8	first atom type
`atom 2`	a8	second atom type - only specified for **pair** potential functions and for the (*i*) **dens**ity functions in the EEAM potential case *or* (*ii*) **sden**sity functions in the 2BEAM potential case *or* (*iii*) **sden** and **dden** functions in the 2BEEAM potential case
`ngrid`	integer	number of function data points to read in
`limit 1`	real	lower interpolation limit in Å for **dens/sden/dden** and **pair** or in density units for **embe/semb/demb**
`limit 2`	real	upper interpolation limit in Å for **dens/sden/dden** and **pair** or in density units for **embe/semb/demb**

function data records: (*number of data records* = `Int`((`ngrid`+3)/4))

`data 1`	real	data item 1
`data 2`	real	data item 2
`data 3`	real	data item 3
`data 4`	real	data item 4

10.1.8.3 Further Comments

The tabled data are used to fill the internal arrays `vmet`, `dmet` and `fmet`, and optionally `dmes` and `fmes` for the 2B extensions of EAM and EEAM (see Section 2.3.2). The force arrays are generated from these (by the METAL_TABLE_DERIVATIVES routine) using a five point interpolation procedure. During simulation, interactions beyond distance $Min(r_{\rm cut}, \texttt{limit 2})$ are discarded, whereas interactions at distances shorter than `limit 1` will cause the simulation to abort. For the purpose of extrapolating the embedding functions $F(\rho)$ beyond its `limit 2` specified in the tabulated array, it is assumed that

$$F(\rho > \texttt{limit 2}) = F(\rho = \texttt{limit 2}) \quad . \tag{10.14}$$

The simulation will however abort if any local density is less than the `limit 1` for its corresponding embedding function.

It is worth noting that in the 2BEAM and 2BEEAM the *s*-band contribution is usually only for the alloy component, so that local concentrations of a single element revert to the standard EAM or EEAM! In such case, the densities functions must be zeroed in the DL_POLY_4 TABEAM file. A convenient way to do this, for example, will be data record of the type:

```
SDEN Atom1 Atom1 1 0 1
0
```

10.1.9 The TABBND, TABANG, TABDIH & TABINV Files

DL_POLY_4 allows the specification of tabulated data for intramolecular interactions:

- TABBND - for chemical bonds potentials - distance dependent
- TABANG - for bond angles potentials - angle dependent
- TABDIH - for dihedrals (torsional) potentials - angle dependent
- TABINV - for inversions potentials - angle dependent.

The files have the same formatting rules with examples shown in Section 4.3. Refer to Section 4.1 for their derivation and usage in coarse grained model systems.

10.1.9.1 Definitions of Variables

record 1		
`header`	a200	file header
record 2		
`#`	a1	a hash (#) symbol
`cutpot`	real	cutoff in Å - **only expected in TABBND** as the cutoff ranges are known for TABANG, TABDIH & TABINV
`ngrid`	integer	number of grid points in table for all potentials
record 3		
`#`	a1	a hash (#) symbol

The subsequent records define each tabulated potential in turn, in the order indicated by the specification in the FIELD file. Each potential is defined by a header record and a set of data records with the potential and force tables.

empty record:		
id record:		
`#`	a1	a hash (#) symbol
`atom 1`	a8	first atom type
`atom 2`	a8	second atom type
`atom 3`	a8	third atom type - only required for TABANG
`atom 4`	a8	forth atom type - only required for TABDIH & TABINV
interaction data records 0/1–ngrid:		
`abscissa`	real	consecutive value over the full cutoff/range in Å for TABBND and degrees for TABANG, TABDIH & TABINV
`potential`	real	potential at the abscissa grid point in **units** as specified in FIELD
`force`	real	complementary force (virial for TABBND) value

10.1.9.2 Further Comments

It should be noted that the number of grid points in the table files should not be less than the number of grid points DL_POLY_4 is expecting. For more information the reader is advised to examine SETUP_MODULE and inspect the `mxg`***int*** variables, where ***int*** refers to `bnd` for bonds, `ang` for angles, `dih` for dihedrals and `inv` for inversions.

The potential and force tables are used to fill the internal arrays `v`***int*** and `g`***int*** for the respective ***int***rmolecular potential (see Chapter 2).

The contents of force arrays for TABBND are derived from the potential via the formula:

$$G(r) = -r\frac{\partial}{\partial r}U(r) \quad . \tag{10.15}$$

Note, this is *not* the same as the true force. During simulation, interactions beyond distance `cutpot` will bring the run to a controlled termination.

10.1.10 The DUMP_E File

The DUMP_E file contains the values of coarse-grained electronic temperature (CET) cells from a previous job with the two-temperature model (TTM). It is not required if the current job is not a continuation of a previous run (i.e. if the **restart** directive is not present in the CONTROL file - see above). The two-line header consists of the following records:

record 1:

`eltsys(1)`	integer	number of CET cells in x-direction
`eltsys(2)`	integer	number of CET cells in y-direction
`eltsys(3)`	integer	number of CET cells in z-direction

record 2:

`nstep`	integer	timestep of current electronic temperature profile
`time`	real	elapsed simulation time
`depostart`	real	time when energy deposition started
`depoend`	real	time when energy deposition ended or is due to end

and each subsequent record is formatted as follows:

`x`	integer	x-coordinate of CET cell
`y`	integer	y-coordinate of CET cell
`z`	integer	z-coordinate of CET cell
`eltemp`	real	electronic temperature at current CET cell (K)

with the origin of CET cell coordinates at the centre of the grid.

The file is read by the subroutine TTM_SYSTEM_INIT. It will not be accepted by DL_POLY_4 if the number of CET cells in each direction does not match the values given in the CONTROL file or insufficient electronic temperatures are supplied. No matching up of restart timestep or elapsed simulation time between REVOLD and DUMP_E is absolutely necessary, but the user will be warned if they are not the same.

10.1.11 The Ce.dat, Ke.dat, De.dat and g.dat Files

The two-temperature model (TTM) implementation in DL_POLY_4 allows specification of the following tabulated data:

- Ce.dat - for electronic volumetric heat capacity - temperature dependent
- Ke.dat - for metallic thermal conductivity - temperature dependent
- De.dat - for non-metallic thermal diffusivity - temperature dependent
- g.dat - for electron-phonon coupling constants - temperature dependent

Each file is free-formatted and only consist of two columns: the first column gives temperature in K, while the second gives electronic volumetric heat capacity in J m^{-3} K^{-1} (Ce.dat), thermal conductivity in W m^{-1} K^{-1} (Ke.dat), thermal diffusivity in m^{2} s^{-1} (De.dat) or the electron-phonon coupling constant in W m^{-3} K^{-1} (g.dat). These files are read by the subroutine TTM_TABLE_READ if the **ttm cetab**, **ttm ketab**, **ttm detab** and/or **ttm gvar** directives are included in the CONTROL file.

10.1.12 The HISTORY/HISTROF File

The HISTORY file is usually an output file (see Section 10.2.1). However, upon specifying the **replay** option a HISTORY trajectory can be replayed and various observables recreated should there is enough information within to recover the specific observable (e.g. velocities supplied within HISTORY for regeneration of velocity autocorrelation data). Thus it can be used as input as well.

If the **replay force** option is used in CONTROL it enforce a new trajectory generation, i.e. generation of HISTORY, which necessitates the trajectory reading routine to default to reading a differently named, HISTORF (copy of HISTORY), basic trajectory file instead. However, using this option expects a full force evaluation driven by expectedly a different FIELD field file from the one used for the HISTORF generation. For more information do refer to CONTROL file directives and options (see Section 10.1.1).

10.1.13 The SETEVB File

The file SETEVB is needed for EVB simulations. If this file is not found, the execution of DL_POLY_4 is aborted. See section **??** for a detailed explanation of the input parameters for EVB calculations.

10.2 The OUTPUT Files

DL_POLY_4 may produce many output files. However only OUTPUT (an incremental summary file of the simulation), STATIS (a statistical history file), REVCON (a restart configuration file - final) and REVIVE (a restart statistics accumulators file - final) are mandatory. DUMP_E (a restart electronic temperature grid file - final) is also produced if the two-temperature model (TTM) is in use. The existence of the remaining files is optional upon user specifications in CONTROL. Some of these optional files are HISTORY, DEFECTS, MSDTMP, CFGMIN, RDFDAT, USRDAT, ZDNDAT, VDFDAT_*, LATS_E, LATS_I, PEAK_E, PEAK_I. These respectively contain: an incremental dump file of all atomic coordinates, velocities and forces; an incremental dump file of atomic coordinates of defected particles (interstitials) and sites (vacancies); an incremental dump file of of individual atomic mean square displacement and temperature; a dump file of all atomic coordinates of a minimised structure; a radial distribution function (RDF) data file; the RDF data file for the umbrella sampling (harmonic restraint); Z-density distribution data file; velocity autocorrelation function (VAF) data files (one file for each species); electronic temperature profile data file; ionic temperature profile data file; electronic temperature statistical data file; ionic temperature statistical data file.

10.2.1 The HISTORY File

The HISTORY file is the dump file of atomic coordinates, velocities and forces. Its principal use is for off-line analysis. The file is written by the subroutine TRAJECTORY_WRITE. The control variables for this file are `ltraj`, `nstraj`, `istraj` and `keytrj` which are created internally, based on information read from the **traj** directive in the CONTROL file (see Section 10.1.1). The HISTORY file will be created only if the directive **traj** appears in the CONTROL file.

The HISTORY file can become *very* large, especially if it is formatted. For serious simulation work it is recommended that the file be written to a scratch disk capable of accommodating a large data file. Alternatively, the file may be written in netCDF format instead of in ASCII (users must change ensure this functionality is available), which has the additional advantage of speed.

The HISTORY has the following structure:

record 1		
`header`	a72	file header
record 2		
`keytrj`	integer	trajectory key (see Table **??**) in last frame
`imcon`	integer	periodic boundary key (see Table 10.3) in last frame
`megatm`	integer	number of atoms in simulation cell in last frame
`frame`	integer	number configuration frames in file
`records`	integer	number of records in file

For timesteps greater than `nstraj` the HISTORY file is appended at intervals specified by the **traj** directive in the CONTROL file, with the following information for each configuration:

record i		
`timestep`	a8	the character string "timestep"
`nstep`	integer	the current time-step
`megatm`	integer	number of atoms in simulation cell (again)
`keytrj`	integer	trajectory key (again)
`imcon`	integer	periodic boundary key (again)
`tstep`	real	integration timestep (ps)
`time`	real	elapsed simulation time (ps)
record ii		
`cell(1)`	real	x component of *a* cell vector in Å
`cell(2)`	real	y component of *a* cell vector in Å
`cell(3)`	real	z component of *a* cell vector in Å
record iii		
`cell(4)`	real	x component of *b* cell vector in Å
`cell(5)`	real	y component of *b* cell vector in Å
`cell(6)`	real	z component of *b* cell vector in Å
record iv		
`cell(7)`	real	x component of *c* cell vector in Å
`cell(8)`	real	y component of *c* cell vector in Å
`cell(9)`	real	z component of *c* cell vector in Å

This is followed by the configuration for the current timestep. i.e. for each atom in the system the following data are included:

record a		
`atmnam`	a8	atomic label
`iatm`	integer	atom index
`weight`	real	atomic mass (a.m.u.)
`charge`	real	atomic charge (e)
`rsd`	real	displacement from position at $t = 0$ in Å
record b		
`xxx`	real	x coordinate
`yyy`	real	y coordinate
`zzz`	real	z coordinate
record c only for `keytrj` > 0		
`vxx`	real	x component of velocity in Å/picosecond
`vyy`	real	y component of velocity in Å/picosecond
`vzz`	real	z component of velocity in Å/picosecond
record d only for `keytrj` > 1		
`fxx`	real	x component of force in Å·Dalton/picosecond^2
`fyy`	real	y component of force in Å·Dalton/picosecond^2
`fzz`	real	z component of force in Å·Dalton/picosecond^2

Thus the data for each atom is a minimum of two records and a maximum of 4.

10.2.2 The MSDTMP File

The MSDTMP file is the dump file of individual atomic mean square displacements (square roots in Å) and mean square temperature (square roots in Kelvin). Its principal use is for off-line analysis. The file is written by the subroutine MSD_WRITE. The control variables for this file are `l_msd, nstmsd, istmsd` which are created internally, based on information read from the **msdtmp** directive in the CONTROL file (see Section 10.1.1). The MSDTMP file will be created only if the directive **msdtmp** appears in the CONTROL file.

The MSDTMP file can become *very* large, especially if it is formatted. For serious simulation work it is recommended that the file be written to a scratch disk capable of accommodating a large data file.

The MSDTMP has the following structure:

record 1		
`header`	a52	file header
record 2		
`megatm`	integer	number of atoms in simulation cell in last frame
`frame`	integer	number configuration frames in file
`records`	integer	number of records in file

For timesteps greater than `nstmsd` the MSDTMP file is appended at intervals specified by the **msdtmp** directive in the CONTROL file, with the following information for each configuration:

record i		
`timestep`	a8	the character string "timestep"
`nstep`	integer	the current time-step
`megatm`	integer	number of atoms in simulation cell (again)
`tstep`	real	integration timestep (ps)
`time`	real	elapsed simulation time (ps)

This is followed by the configuration for the current timestep. i.e. for each atom in the system the following data are included:

record a		
`atmnam`	a8	atomic label
`iatm`	integer	atom index
$\sqrt{\texttt{MSD}(t)}$	real	square root of the atomic mean square displacements (in Å)
T_{mean}	real	atomic mean temperature (in Kelvin)

10.2.3 The DEFECTS File

The DEFECTS file is the dump file of atomic coordinates of defects (see Section 10.1.5). Its principal use is for off-line analysis. The file is written by the subroutine DEFECTS_WRITE. The control variables for this file are `ldef`, `nsdef`, `isdef` and `rdef` which are created internally, based on information read from the **defects** directive in the CONTROL file (see Section 10.1.1). The DEFECTS file will be created only if the directive **defects** appears in the CONTROL file.

The DEFECTS file may become *very* large, especially if it is formatted. For serious simulation work it is recommended that the file be written to a scratch disk capable of accommodating a large data file.

The DEFECTS has the following structure:

record 1		
`header`	a72	file header
record 2		
`rdef`	real	site-interstitial cutoff (Å) in last frame
`frame`	integer	number configuration frames in file
`records`	integer	number of records in file

For timesteps greater than `nsdef` the DEFECTS file is appended at intervals specified by the **defects** directive in the CONTROL file, with the following information for each configuration:

record i

`timestep`	a8	the character string "timestep"
`nstep`	integer	the current time-step
`tstep`	real	integration timestep (ps)
`time`	real	elapsed simulation time (ps)
`imcon`	integer	periodic boundary key (see Table 10.3)
`rdef`	real	site-interstitial cutoff (Å)

record ii

`defects`	a7	the character string "defects"
`ndefs`	integer	the total number of defects
`interstitials`	a13	the character string "interstitials"
`ni`	integer	the total number of interstitials
`vacancies`	a9	the character string "vacancies"
`nv`	integer	the total number of vacancies

record iii

`cell(1)`	real	x component of *a* cell vector
`cell(2)`	real	y component of *a* cell vector
`cell(3)`	real	z component of *a* cell vector

record iv

`cell(4)`	real	x component of *b* cell vector
`cell(5)`	real	y component of *b* cell vector
`cell(6)`	real	z component of *b* cell vector

record v

`cell(7)`	real	x component of *c* cell vector
`cell(8)`	real	y component of *c* cell vector
`cell(9)`	real	z component of *c* cell vector

This is followed by the `ni` interstitials for the current timestep, as each interstitial has the following data lines:

record a

`atmnam`	a10	i_atomic label from CONFIG
`iatm`	integer	atom index from CONFIG

record b

`xxx`	real	x coordinate
`yyy`	real	y coordinate
`zzz`	real	z coordinate

This is followed by the `nv` vacancies for the current timestep, as each vacancy has the following data lines:

record a

`atmnam`	a10	v_atomic label from REFERENCE
`iatm`	integer	atom index from REFERENCE

record b

`xxx`	real	x coordinate from REFERENCE
`yyy`	real	y coordinate from REFERENCE
`zzz`	real	z coordinate from REFERENCE

10.2.4 The RSDDAT File

The RSDDAT file is the dump file of atomic coordinates of atoms that are displaced from their original position at $t = 0$ farther than a preset cutoff. Its principal use is for off-line analysis. The file is written by the subroutine RSD_WRITE. The control variables for this file are `lrsd, nsrsd, isrsd` and `rrsd` which are created internally, based on information read from the **displacements** directive in the CONTROL file (see Section 10.1.1). The RSDDAT file will be created only if the directive **defects** appears in the CONTROL file.

The RSDDAT file may become *very* large, especially if it is formatted. For serious simulation work it is recommended that the file be written to a scratch disk capable of accommodating a large data file.

The RSDDAT has the following structure:

record 1		
`header`	a72	file header
record 2		
`rdef`	real	displacement qualifying cutoff (Å) in last frame
`frame`	integer	number configuration frames in file
`records`	integer	number of records in file

For timesteps greater than `nsrsd` the RSDDAT file is appended at intervals specified by the **displacements** directive in the CONTROL file, with the following information for each configuration:

record i		
`timestep`	a8	the character string “timestep”
`nstep`	integer	the current time-step
`tstep`	real	integration timestep (ps)
`time`	real	elapsed simulation time (ps)
`imcon`	integer	periodic boundary key (see Table 10.3)
`rrsd`	real	displacement qualifying cutoff (Å)
record ii		
`displacements`	a13	the character string “displacements”
`nrsd`	integer	the total number of displacements
record iii		
`cell(1)`	real	x component of a cell vector
`cell(2)`	real	y component of a cell vector
`cell(3)`	real	z component of a cell vector
record iv		
`cell(4)`	real	x component of b cell vector
`cell(5)`	real	y component of b cell vector
`cell(6)`	real	z component of b cell vector
record v		
`cell(7)`	real	x component of c cell vector
`cell(8)`	real	y component of c cell vector
`cell(9)`	real	z component of c cell vector

This is followed by the `nrsd` displacements for the current timestep, as each atom has the following data lines:

record a		
`atmnam`	a10	atomic label from CONFIG
`iatm`	integer	atom index from CONFIG
`ratm`	real	atom displacement from its position at $t = 0$
record b		
`xxx`	real	x coordinate
`yyy`	real	y coordinate
`zzz`	real	z coordinate

10.2.5 The CFGMIN File

The CFGMIN file only appears if the user has selected the programmed minimisation option (directive **minim**ise (or **optim**ise) in the CONTROL file). Its contents have the same format as the CONFIG file (see Section 10.1.2), but contains only atomic position data and will never contain either velocity or force data (i.e. parameter `levcfg` is always zero). In addition, three extra numbers appear on the end of the second line of the file:

1. an integer indicating the number of minimisation cycles required to obtain the structure,
2. the configuration energy of the minimised configuration expressed in DL_POLY_4 units (Section 1.3.7), and
3. the configuration energy of the initial structure expressed in DL_POLY_4 units (Section 1.3.7).

10.2.6 The OUTPUT File

The job output consists of 7 sections: Header; Simulation control specifications; Force field specification; System specification; Summary of the initial configuration; Simulation progress; Sample of the final configuration; Summary of statistical data; and Radial distribution functions and Z-density profile. These sections are written by different subroutines at various stages of a job. Creation of the OUTPUT file *always* results from running DL_POLY_4. It is meant to be a human readable file, destined for hardcopy output.

10.2.6.1 Header

Gives the DL_POLY_4 version number, the number of processors in use, the link-cell algorithm in use and a title for the job as given in the header line of the input file CONTROL. This part of the file is written from the subroutines DL_POLY, SET_BOUNDS and READ_CONTROL.

10.2.6.2 Simulation Control Specifications

Echoes the input from the CONTROL file. Some variables may be reset if illegal values were specified in the CONTROL file. This part of the file is written from the subroutine READ_CONTROL.

10.2.6.3 Force Field Specification

Echoes the FIELD file. A warning line will be printed if the system is not electrically neutral. This warning will appear immediately before the non-bonded short-range potential specifications. This part of the file is written from the subroutine READ_FIELD.

10.2.6.4 System Specification

Echoes system name, periodic boundary specification, the cell vectors and volume, some initial estimates of long-ranged corrections the energy and pressure (if appropriate), some concise information on topology and degrees of freedom break-down list. This part of the file is written from the subroutines SCAN_CONFIG, CHECK_CONFIG, SYSTEM_INIT, REPORT_TOPOLOGY and SET_TEMPERATURE.

10.2.6.5 Summary of the Initial Configuration

This part of the file is written from the main subroutine DL_POLY_. It states the initial configuration of (a maximum of) 20 atoms in the system. The configuration information given is based on the value of `levcfg` in the CONFIG file. If `levcfg` is 0 (or 1) positions (and velocities) of the 20 atoms are listed. If `levcfg` is 2 forces are also written out.

10.2.6.6 Simulation Progress

This part of the file is written by the DL_POLY_4 root segment DL_POLY. The header line is printed at the top of each page as:

```
    step   eng_tot  temp_tot   eng_cfg   eng_src   eng_cou   eng_bnd   eng_ang   eng_dih   eng_tet
time(ps)    eng_pv  temp_rot   vir_cfg   vir_src   vir_cou   vir_bnd   vir_ang   vir_con   vir_tet
cpu  (s)    volume  temp_shl   eng_shl   vir_shl     alpha      beta     gamma   vir_pmf     press
```

The labels refer to :

line 1

`step`	MD step number
`eng_tot`	total internal energy of the system
`temp_tot`	system temperature (in Kelvin)
`eng_cfg`	configurational energy of the system
`eng_src`	configurational energy due to short-range potential contributions
`eng_cou`	configurational energy due to electrostatic potential
`eng_bnd`	configurational energy due to chemical bond potentials
`eng_ang`	configurational energy due to valence angle and three-body potentials
`eng_dih`	configurational energy due to dihedral inversion and four-body potentials
`eng_tet`	configurational energy due to tethering potentials

line 2

`time(ps)`	elapsed simulation time (in pico-seconds) since the beginning of the job
`eng_pv`	enthalpy of system
`temp_rot`	rotational temperature (in Kelvin)
`vir_cfg`	total configurational contribution to the virial
`vir_src`	short range potential contribution to the virial
`vir_cou`	electrostatic potential contribution to the virial
`vir_bnd`	chemical bond contribution to the virial
`vir_ang`	angular and three-body potentials contribution to the virial
`vir_con`	constraint bond contribution to the virial
`vir_tet`	tethering potential contribution to the virial

line 3

`cpu (s)`	elapsed cpu time (in seconds) since the beginning of the job
`volume`	system volume (in Å^3)
`temp_shl`	core-shell temperature (in Kelvin)
`eng_shl`	configurational energy due to core-shell potentials
`vir_shl`	core-shell potential contribution to the virial
`alpha`	angle between b and c cell vectors (in degrees)
`beta`	angle between c and a cell vectors (in degrees)
`gamma`	angle between a and b cell vectors (in degrees)
`vir_pmf`	PMF constraint contribution to the virial
`press`	pressure (in kilo-atmospheres)

Note: The total internal energy of the system (variable `tot_energy`) includes all contributions to the energy (including system extensions due to thermostats etc.). It is nominally the *conserved variable* of the system, and is not to be confused with conventional system energy, which is a sum of the kinetic and configuration energies.

The interval for printing out these data is determined by the directive **print** in the CONTROL file. At each time-step that printout is requested the instantaneous values of the above statistical variables are given in the appropriate columns. Immediately below these three lines of output the rolling averages of the same variables are also given. The maximum number of time-steps used to calculate the rolling averages is controlled by the directive **stack** in file CONTROL (see above) and listed as parameter `mxstak` in the SETUP_MODULE file (see Section 11.2.2). The default value is `mxstak` = 100.

Energy Units

The energy unit for the energy and virial data appearing in the OUTPUT is defined by the **units** directive appearing in the FIELD file. System energies are therefore read in **units** per MD cell.

Pressure Units

The unit of pressure is katms, irrespective of what energy unit is chosen.

Two-Temperature Model

If the two-temperature model is in use, information about the timestep sizes used for electronic thermal diffusivity is written immediately prior to each report of statistical variables at each molecular dynamics timestep for which printout is requested. The optimum diffusive timestep size is given in pico-seconds, along with the chosen value and the corresponding number of divisions of the MD timestep. If dynamic calculation of the average atomic density in active cells is requested, this value is included along with the number of active ionic temperature cells. Reports are also given when energy deposition starts and finishes.

10.2.6.7 Sample of Final Configuration

The positions, velocities and forces of the 20 atoms used for the sample of the initial configuration (see above) are given. This is written by the main subroutine DL_POLY.

10.2.6.8 Summary of Statistical Data

This portion of the OUTPUT file is written from the subroutine STATISTICS_RESULT. The number of time-steps used in the collection of statistics is given. Then the averages over the production portion of the run are given for the variables described in the previous section. The root mean square variation in these variables follow on the next two lines. The energy and pressure units are as for the preceding section.

Also provided in this section are estimates of the diffusion coefficient and the mean square displacement for the different atomic species in the simulation. These are determined from a *single time origin* and are therefore approximate. Accurate determinations of the diffusion coefficients can be obtained using the MSD utility program, which processes the HISTORY file (see DL_POLY_Classic User Manual).

If an NPT (N$\underline{\underline{\sigma}}$T) simulation is performed the OUTPUT file also provides the mean pressure (and stress tensor in pressure units as density) and mean simulation cell vectors. In case when extended N$\underline{\underline{\sigma}}$T ensembles are used then further mean (x, y) plain area and mean surface tension are also displayed in the OUTPUT file.

10.2.6.9 Radial Distribution Functions

If both calculation and printing of radial distribution functions have been requested (by selecting directives **rdf** and **print rdf** in the CONTROL file) radial distribution functions are printed out. This is written from the subroutine RDF_COMPUTE. First the number of time-steps used for the collection of the histograms is stated.

For each function a header line states the atom types ('a' and 'b') represented by the function. Then $r, g(r)$ and $n(r)$ are given in tabular form. $n(r)$ is the average number of atoms of type 'b' within a sphere of radius r around an atom of type 'a'. Note that a readable version of these data is provided by the RDFDAT file (below).

10.2.6.10 Umbrella Sampling Restraint RDF

If an umbrella sampling harmonic restraint is defined in the FIELD file (by selecting the **ushr** external field sectione) the RDF of the two restraint objects/fragments is printed out. This is written from the subroutine USR_COMPUTE in RDF_COMPUTE. Note that a readable version of these data is provided by the USRDAT file (below).

10.2.6.11 Z-density Profile

If both calculation and printing of Z-density profiles have been requested (by selecting directives **zden** and **print zden** in the CONTROL file Z-density profiles are printed out as the last part of the OUTPUT file. This is written by the subroutine Z_DENSITY_COMPUTE. First the number of time-steps used for the collection of the histograms is stated. Then each function is given in turn. For each function a header line states the atom type represented by the function. Then z, $\rho(z)$ and $n(z)$ are given in tabular form. Output is given from $Z = [-L/2, L/2]$ where L is the length of the MD cell in the Z direction and $\rho(z)$ is the mean number density. $n(z)$ is the running integral from $-L/2$ to z of (xy cell area) $\times\, \rho(s)\ ds$. Note that a readable version of these data is provided by the ZDNDAT file (below).

10.2.6.12 Velocity Autocorrelation Functions

If both calculation and printing of velocity autocorrelation functions have been requested (by selecting directives **vaf** and **print vaf** in the CONTROL file the velocity autocorrelation function for the system (either time-averaged or the last complete sample) is printed out as the last part of the OUTPUT file. This is written by the subroutine VAF_COMPUTE. First the details of the calculations are stated: either the number of samples used to give a time-averaged profile or the number of the last completed sample with its starting time. The absolute value of the velocity autocorrelation function for the system at $t = 0$, $C(0)$, is then stated. Then t and $Z(t)$ are given in tabular form. $Z(t) = C(t)/C(0)$ is the value of the velocity autocorrelation function, $C(t) = \langle \underline{v}_i(0) \cdot \underline{v}_i(t) \rangle$, scaled by $C(0) \equiv 3k_BT/m$. Note that a readable version of these data for individual species is provided by the VAFDAT files (below).

10.2.7 The HEATFLUX File

The HEATFLUX file contains data relevant to the calculation of heat-flux via a Green-Kubo mothod via an external convolution, the information is written as:

```
STEP STPTMP VOLUME HEAT_FLUX
```

10.2.8 The PP_CONT File

This file contains the contributions of each particle to energies, forces and stresses in a format similar to to the CONFIG file, but with ID replaced with energy, and velocities/forces with the stress 6-vector.

```
TAG ATMNAM KIN_E MASS ENERGY
STR_XX STR_YX STR_ZX
STR_XY STR_YY STR_ZY
STR_XZ STR_YZ STR_ZZ
```

10.2.9 The REVCON File

This file is formatted and written by the subroutine REVIVE. REVCON is the restart configuration file. The file is written every `ndump` time steps in case of a system crash during execution and at the termination of the job. A successful run of DL_POLY_4 will always produce a REVCON file, but a failed job may not produce the file if an insufficient number of timesteps have elapsed. `ndump` is controlled by the directive **dump** in file CONTROL (see above) and listed as parameter `ndump` in the SETUP_MODULE file (see Section 11.2.2). The default value is `ndump` = 1000. REVCON is identical in format to the CONFIG input file (see Section 10.1.2). REVCON should be renamed CONFIG to continue a simulation from one job to the next. This is done for you by the *copy* macro supplied in the *execute* directory of DL_POLY_4.

10.2.10 The REVIVE File

This file is unformatted and written by the subroutine SYSTEM_REVIVE. It contains the accumulated statistical data. It is updated whenever the file REVCON is updated (see previous section). REVIVE should be renamed REVOLD to continue a simulation from one job to the next. This is done by the *copy* macro supplied in the *execute* directory of DL_POLY_4. In addition, to continue a simulation from a previous job the **restart** keyword must be included in the CONTROL file.

The format of the REVIVE file is identical to the REVOLD file described in Section 10.1.6.

10.2.11 The DUMP_E File

This file is formatted and written by the subroutine TTM_SYSTEM_REVIVE every `ndump` time steps. It contains the electronic temperatures of all coarse-grained electronic temperature (CET) cells and can be used to restart a simulation using the two-temperature model without renaming the file.

The format of the DUMP_E is described in Section 10.1.10.

10.2.12 The RDFDAT File

This is a formatted file containing *Radial Distribution Function* (RDF) data. Its contents are as follows:

record 1		
`cfgname`	a72	configuration name
record 2		
`ntprdf`	integer	number of different RDF pairs tabulated in file
`mxgrdf`	integer	number of grid points for each RDF pair

There follow the data for each individual RDF, i.e. `ntprdf` times. The data supplied are as follows:

first record

`atname 1`	a8	first atom name
`atname 2`	a8	second atom name
following records (*mxgrdf* records)		
`radius`	real	interatomic distance (Å)
`g(r)`	real	RDF at given radius
`n(r)`	real	RDF at given radius

Note 1. The RDFDAT file is optional and appears when the **print rdf** option is specified in the CONTROL file.

Note 2. Along with the RDFDAT file, two other files will be created whenever the **print ana**lysis directive is invoked: VDWPMF & VDWTAB, both containing the data for potentials of mean force and the corresponding virials calculated based on the obtained RDF:s, i.e. PMF $\sim -\ln(\text{RDF})$ (in the energy units specified in the FIELD file). These files have a simple three column format, the same as that used for *PMF files in the case of bonded units, see Section 4.2. The purpose of these files is to provide the user with means of setting up a PMF-based force-field, for example in the case of initial coarse-graining of an atomistic system. In particular, one can convert the VDWTAB file into a correctly formatted TABLE file (Section 10.1.7) by using the utility called `pmf2tab.f` (subject to compilation; found in DL_POLY_4 directory `utility`) as follows,

```
[user@host]$ pmf2tab.exe < VDWTAB
```

see Section 4.1 for completeness.

10.2.13 The USRDAT File

record 1		
`# title`	a100	file header title
record 2		
`# header`	a100	file information header
record 3		
`# info`	a30	information to follow string
record 3		
`bins`	integer	number of bins
`cutoff`	real	cutoff in Å
`frames`	integer	number of sampled configurations
`volume`	real	average cell volue Å^3
record 4		
`#`	a1	a hash (#) symbol
following records (*mxgusr* records)		
`radius`	real	interatomic distance (Å)
`g(r)`	real	RDF at given radius

10.2.14 The ZDNDAT File

This is a formatted file containing the Z-density data. Its contents are as follows:

record 1		
`cfgname`	a72	configuration name
record 2		
`ntpatm`	integer	number of unique atom types profiled in file
`mxgrdf`	integer	number of grid points in the Z-density function

There follow the data for each individual Z-density function, i.e. `ntpatm` times. The data supplied are as follows:

first record		
`atname`	a8	unique atom name
following records (*mxgrdf* records)		
`z`	real	distance in z direction (Å)
$\rho(z)$	real	Z-density at given height `z`

Note the ZDNDAT file is optional and appears when the **print rdf** option is specified in the CONTROL file.

10.2.15 The VAFDAT Files

These are formatted files containing *Velocity Autocorrelation Function* (VAF) data. An individual file is created for each atomic species, i.e. `VAFDAT_`*atname*. Their contents are as follows:

record		
`cfgname`	a72	configuration name

There follow the data for the VAF, either a single time-averaged profile or successive profiles separated by two blank lines. The data supplied are as follows:

first record		
`atname`	a8	atom name
`binvaf`	integer	number of data points in VAF profile, *excluding* $t = 0$
`vaforigin`	real	absolute value of VAF at $t = 0$ ($C(0) \equiv 3k_BT/m$)
`vaftime0`	real	simulation time (ps) at beginning of (last) VAF profile ($t = 0$)
following records (*binvaf*+1 records)		
`t`	real	time (ps)
`Z(t)`	real	scaled velocity autocorrelation function ($C(t)/C(0)$) at given time t

Note the VAFDAT files are optional and appear when the **print vaf** option is specified in the CONTROL file.

10.2.16 The *INT*DAT, *INT*PMF & *INT*TAB Files

These files, where ***INT*** is referring to ***INT***ra-molecular interactions and ***VDW***(RDF derived inter-molecular), have very similar formatting rules with some examples shown in Section 4.2. Refer to Section 4.2 for their meaning and usage in coarse grained model systems.

record 1		
`# title`	a100	file header title
record 2		
`# header`	a100	file information header
record 3		
`# info`	a30	information to follow string
`bins`	integer	number of bins for all PDFs
`cutoff`	real	cutoff in Å for bonds and RDFs or degrees for angular intramolecular interactions
`frames`	integer	number of sampled configurations

`types`	integer	number of unique types of these interactions
record 4		
`#`	a1	a hash (#) symbol
record 5		
`# info 1`	a100	information to follow string
record 6		
`#`	a1	a hash (#) symbol

The subsequent records define each PDF potential in turn, in the order indicated by the specification in the FIELD file. Each potential is defined by a header record and a set of data records with the potential-like and force-like tables.

empty record:		
id record:		
`# info`	a25	information to follow string
`atom 1`	a8	first atom type
`atom 2`	a8	second atom type
`atom 3`	a8	third atom type - only available in ANG* files
`atom 4`	a8	forth atom type - only available in DIH* & INV* files
`index`	integer	unique index of PDF in file
`instances`	integer	instances of this unique type of PDF
interaction data records 1–bins:		
`abscissa`	real	consecutive value over the full cutoff/range in Å for BNDTAB & VDWTAB and degrees for ANGTAB, DIHTAB & INVTAB
`potential`	real	potential at the abscissa grid point in **units** as specified in FIELD
`force`	real	complementary force (virial for BNDTAB & VDWTAB) value

10.2.17 The STATIS File

The file is formatted, with integers as "i10" and reals as "e14.6". It is written by the subroutine STATISTICS_COLLECT. It consists of two header records followed by many data records of statistical data.

record 1		
`cfgname`	a72	configuration name
record 2		
`string`	a8	energy units

Data records
Subsequent lines contain the instantaneous values of statistical variables dumped from the array `stpval`. A specified number of entries of `stpval` are written in the format "(1p,5e14.6)". The number of array elements required (determined by the parameter `mxnstk` in the SETUP_MODULE file) is

$$\begin{aligned}\texttt{mxnstk} \geq\ & 28 + 9 \text{ (stress tensor elements) } + \\ & \texttt{ntpatm} \text{ (number of unique atomic sites) } + \\ & 10 \text{ (if constant pressure simulation requested) } + \\ & 2 \text{ (if iso } > \text{ 0 requested)} + 2 \text{ (if iso } > \text{ 1 requested) } + \\ & 2 * mxatdm \text{ (if msdtmp option is used)}\end{aligned}$$

The STATIS file is appended at intervals determined by the **stats** directive in the CONTROL file. The energy unit is as specified in the FIELD file with the **units** directive, and are compatible with the data appearing in the OUTPUT file. The contents of the appended information of calculated *instantaneous* observables is:

record i		
`nstep`	integer	current MD time-step
`time`	real	elapsed simulation time
`nument`	integer	number of array elements to follow
record ii `stpval`(1) – `stpval`(5)		
`engcns`	real	total extended system energy, $E_{tot}^{x} = (E_{kin} + E_{rot}) + E_{conf} + E_{consv}$ (i.e. including the conserved quantity, E_{consv})
`temp`	real	system temperature, $2\frac{E_{kin}+E_{rot}}{fk_B}$
`engcfg`	real	configurational energy, E_{conf}
`engsrc`	real	short range potential energy
`engcpe`	real	electrostatic energy
record iii `stpval`(6) – `stpval`(10)		
`engbnd`	real	chemical bond energy
`engang`	real	valence angle and 3-body potential energy
`engdih`	real	dihedral, inversion, and 4-body potential energy
`engtet`	real	tethering energy
`enthal`	real	enthalpy ($E_{tot}^{x} + \mathcal{P} \cdot V$) for NVE/T/E$_{kin}$ ensembles enthalpy ($E_{tot}^{x} + P \cdot \mathcal{V}$) for NP/$\sigma$T or NP$_n$A/$\gamma$ ensembles
record iv `stpval`(11) – `stpval`(15)		
`tmprot`	real	rotational temperature, E_{rot}
`vir`	real	total virial
`virsrc`	real	short-range virial
`vircpe`	real	electrostatic virial
`virbnd`	real	bond virial
record v `stpval`(16) – `stpval`(20)		
`virang`	real	valence angle and 3-body virial
`vircon`	real	constraint bond virial
`virtet`	real	tethering virial
`volume`	real	volume, $\mathcal{V}$
`tmpshl`	real	core-shell temperature
record vi `stpval`(21) – `stpval`(25)		
`engshl`	real	core-shell potential energy
`virshl`	real	core-shell virial
`alpha`	real	MD cell angle α
`beta`	real	MD cell angle β
`gamma`	real	MD cell angle γ
record vii `stpval`(26), `stpval`(27), `stpval`(0)		
`virpmf`	real	PMF constraint virial
`press`	real	pressure, $\mathcal{P}$
`consv`	real	extended DoF energy, E_{consv}
the next 9 entries for the stress tensor in pressure units		
`stress(1)`	real	xx component of stress tensor
`stress(2)`	real	xy component of stress tensor
`stress(3)`	real	xz component of stress tensor
`stress(4)`	real	yx component of stress tensor
`...`	real	...
`stress(9)`	real	zz component of stress tensor
the next `ntpatm` **entries**		
`amsd(1)`	real	mean squared displacement of first atom types
`amsd(2)`	real	mean squared displacement of second atom types
`...`	...	...
`amsd(ntpatm)`	real	mean squared displacement of last atom types

the next 10 entries - *if* a NPT or N$\underline{\underline{\sigma}}$T simulation is undertaken

`cell(1)`	real	x component of a cell vector
`cell(2)`	real	y component of a cell vector
`cell(3)`	real	z component of a cell vector
`cell(4)`	real	x component of b cell vector
`...`	real	...
`cell(9)`	real	z component of c cell vector
`stpipv`	real	pressure, $\mathcal{P} \cdot \mathcal{V}$

the next 2 entries - *if* NP$_n$AT simulation is undertaken with iso ¿ 0

`h_z`	real	MD cell height h_z to normal surface $\mathcal{A} \perp z$
`A⊥z`	real	MD cell normal surface $\mathcal{A} \perp z = \mathcal{V}/h_z$

the next 2 entries - *if* a Nγ_nAT simulation is undertaken with iso ¿ 1

`gamma_x`	real	surface tension γ_{n_x} on normal surface $\mathcal{A} \perp z$
`gamma_y`	real	surface tension γ_{n_y} on normal surface $\mathcal{A} \perp z$

10.2.18 The LATS_E and LATS_I Files

These are formatted files containing electronic (LATS_E) and ionic (LATS_I)temperatures at user-requested intervals along the y-direction in the centre of the system's xz-plane from two-temperature model calculations.

Each line in these files consists of a series of electronic or ionic temperatures along the y-direction – `-eltsys(2)/2` $\leq y \leq$ `+eltsys(2)/2` and `-ntsys(2)/2` $\leq y \leq$ `+ntsys(2)/2` at $x = z = 0$ – corresponding to a requested timestep. The number of values in each line will depend on the number of electronic or ionic temperature cells requested by the user.

10.2.19 The PEAK_E and PEAK_I Files

These are formatted files containing statistics from two-temperature model calculations at user-requested intervals. Each line in these files corresponds to a requested time step and the data is based upon active coarse-grained electronic (CET) and ionic (CIT) temperature grid cells.

In the PEAK_E file, the data are formatted as follows:

record i

`nstep`	integer	current MD time-step
`time`	real	elapsed simulation time
`eltemp_min`	real	minimum value of electronic temperature in system (K)
`eltemp_max`	real	maximum value of electronic temperature in system (K)
`eltemp_mean`	real	mean value of electronic temperature in system (K)
`eltemp_sum`	real	sum of electronic temperatures in system (K)
`Ue`	real	total electronic energy in system (eV)

The PEAK_I file is formatted in a similar fashion, as follows:

record i

`nstep`	integer	current MD time-step
`time`	real	elapsed simulation time
`tempion_min`	real	minimum value of ionic temperature in system (K)
`tempion_max`	real	maximum value of ionic temperature in system (K)
`tempion_mean`	real	mean value of ionic temperature in system (K)
`tempion_sum`	real	sum of ionic temperatures in system (K)

10.2.20 The POPEVB Files

This is an unformatted file to print the weight of each chemical state $|\Psi^{(k)}_{\text{EVB}}|^2$ in the total EVB state, as described in section 7.2. Values are printed at each time step only after equilibration.
The structure of the printed data is as follows:

Time (ps) $|\Psi^{(1)}_{\text{EVB}}|^2$ $|\Psi^{(2)}_{\text{EVB}}|^2$ $|\Psi^{(3)}_{\text{EVB}}|^2 \cdots\cdots |\Psi^{(N_F)}_{\text{EVB}}|^2$

where N_F is the number of force-fields coupled via the EVB simulation.

10.2.21 The ICOORD, CCOORD and ADFDAT files

ICOORD and CCOORD are output files that log coordination number data for pairs of atomic species specified by the user. To perform this analysis and output these files the user must enter the keyword **coord_calculate** (see section 10.1.1.2) into the CONTROL file and **crd** (see section 10.1.3.4) into the FIELD file.

ICOORD is a dump file that can contain 2 types of data. The top of the file contains the initial coordination of each atom and the exact atoms it is coordinated to. There is an option to write this data at set intervals (the writing step interval) or just at the initial step. The bottom of the file provides the coordination distribution statistics for each atom after each writing step interval. The coordination distribution for the [atom list] - [atom list] pairs will also be displayed here.

CCOORD is a coordination displacement file that dumps the positions of all atoms that are considered to both change their initial local atomic coordination, and move more than a set distance from their initial position, at set intervals. This procedure is described in reference [110].

ADFDAT is statistics file containing the angular distributions for the atom pairs specified.

Chapter 11

The DL_POLY_4 Parallelisation and Source Code

Scope of Chapter

This chapter we discuss the DL_POLY_4 parallelisation strategy, describe the principles used in the DL_POLY_4 modularisation of the source code and list the file structure found in the *source* subdirectory.

11.1 Parallelisation

DL_POLY_4 is a distributed parallel molecular dynamics package based on the Domain Decomposition parallelisation strategy [2, 3, 9, 10, 5, 6]. In this section we briefly outline the basic methodology. Users wishing to add new features DL_POLY_4 will need to be familiar with the underlying techniques as they are described in the above references.

11.1.1 The Domain Decomposition Strategy

The Domain Decomposition (DD) strategy [2, 3, 5] is one of several ways to achieve parallelisation in MD. Its name derives from the division of the simulated system into equi-geometrical spatial blocks or domains, each of which is allocated to a specific processor of a parallel computer. I.e. the arrays defining the atomic coordinates $\underline{r}_i$, velocities $\underline{v}_i$ and forces $\underline{f}_i$, for all N atoms in the simulated system, are divided in to sub-arrays of approximate size N/P, where P is the number of processors, and allocated to specific processors. In DL_POLY_4 the domain allocation is handled by the routine DOMAINS_MODULE and the decision of approximate sizes of various bookkeeping arrays in SET_BOUNDS. The division of the configuration data in this way is based on the location of the atoms in the simulation cell, such a *geometric* allocation of system data is the hallmark of DD algorithms. Note that in order for this strategy to work efficiently, the simulated system must possess a reasonably uniform density, so that each processor is allocated almost an equal portion of atom data (as much as possible). Through this approach the forces computation and integration of the equations of motion are shared (reasonably) equally between processors and to a large extent can be computed independently on each processor. The method is conceptually simple though tricky to program and is particularly suited to large scale simulations, where efficiency is highest.

The DD strategy underpinning DL_POLY_4 is based on the link cell algorithm of Hockney and Eastwood [111] as implemented by various authors (e.g. Pinches *et al.* [9] and Rapaport [10]). This requires that the cutoff applied to the interatomic potentials is relatively short ranged. In DL_POLY_4 the link-cell list is build by the routine LINK_CELL_PAIRS. As with all DD algorithms, there is a need for the processors to exchange 'halo data', which in the context of link-cells means sending the contents of the link cells at the boundaries of each domain, to the neighbouring processors, so that each may have all necessary information to compute the pair forces acting on the atoms belonging to its allotted domain. This in DL_POLY_4 is handled by the SET_HALO_PARTICLES routine.

Systems containing complex molecules present several difficulties. They often contain ionic species, which usually require Ewald summation methods [22, 112], and *intra*-molecular interactions in addition to *inter*-molecular forces. Intramolecular interactions are handled in the same way as in DL_POLY_Classic, where each processor is allocated a subset of intramolecular bonds to deal with. The allocation in this case is based on the atoms present in the processor's domain. The SHAKE and RATTLE algorithms [88, 23] require significant modification. Each processor must deal with the constraint bonds present in its own domain, but it must also deal with bonds it effectively shares with its neighbouring processors. This requires each processor to inform its neighbours whenever it updates the position of a shared atom during every SHAKE (RATTLE_VV1) cycle (RATTLE_VV2 updates the velocities), so that all relevant processors may incorporate this update into its own iterations. In the case of the DD strategy the SHAKE (RATTLE) algorithm is simpler than for the Replicated Data method of DL_POLY_Classic, where global updates of the atom positions (merging and splicing) are required [113]. The absence of the merge requirement means that the DD tailored SHAKE and RATTLE are less communications dependent and thus more efficient, particularly with large processor counts.

The DD strategy is applied to complex molecular systems as follows:

1. Using the atomic coordinates $\underline{r}_i$, each processor calculates the forces acting between the atoms in its domain - this requires additional information in the form of the halo data, which must be passed from the neighbouring processors beforehand. The forces are usually comprised of:

(a) All common forms of non-bonded atom-atom (van der Waals) forces
(b) Atom-atom (and site-site) coulombic forces
(c) Metal-metal (local density dependent) forces
(d) Tersoff (local density dependent) forces (for hydro-carbons) [17]
(e) Three-body valence angle and hydrogen bond forces
(f) Four-body inversion forces
(g) Ion core-shell polarasation
(h) Tether forces
(i) Chemical bond forces
(j) Valence angle forces
(k) Dihedral angle (and improper dihedral angle) forces
(l) Inversion angle forces
(m) External field forces.

2. The computed forces are accumulated in atomic force arrays $\underline{f}_i$ independently on each processor
3. The force arrays are used to update the atomic velocities and positions of all the atoms in the domain
4. Any atom which effectively moves from one domain to another, is relocated to the neighbouring processor responsible for that domain.

It is important to note that load balancing (i.e. equal and concurrent use of all processors) is an essential requirement of the overall algorithm. In DL_POLY_4 this is accomplished quite naturally through the DD partitioning of the simulated system. Note that this will only work efficiently if the density of the system is reasonably uniform. THERE ARE NO LOAD BALANCING ALGORITHMS IN DL_POLY_4 TO COMPENSATE FOR A BAD DENSITY DISTRIBUTION.

11.1.2 Distributing the Intramolecular Bonded Terms

The intramolecular terms in DL_POLY_4 are managed through bookkeeping arrays which list all atoms (sites) involved in a particular interaction and point to the appropriate arrays of parameters that define the potential. Distribution of the forces calculations is accomplished by the following scheme:

1. Every atom (site) in the simulated system is assigned a unique 'global' index number from 1 to N.
2. Every processor maintains a list of the local indices of the atoms in its domain. (This is the local atom list.)
3. Every processor also maintains a sorted (in ascending order) local list of global atom indices of the atoms in its domain. (This is the local sorted atom list.)
4. Every intramolecular bonded term U_{type} in the system has a unique index number i_{type}: from 1 to N_{type} where *type* represents a bond, angle, dihedral, or inversion. Also attached there with unique index numbers are core-shell units, bond constraint units, PMF constraint units, rigid body units and tethered atoms, their definition by site rather than by chemical type.
5. On each processor a pointer array $key_{type}(n_{type}, i_{type})$ carries the indices of the specific atoms involved in the potential term labelled i_{type} . The dimension n_{type} will be 1 if the term represents a tether, 1, 2 for a core-shell unit or a bond constraint unit or a bond, 1, 2, 3 for a valence angle and 1, 2, 3, 4 for a dihedral or an inversion, $1, .., n_{\texttt{PMF unit}_{1 \texttt{ or } 2}} + 1$ for a PMF constraint unit, or $-1,\ 0,\ 1, .., n_{\texttt{RB unit}}$ for a rigid body unit.

6. Using the *key* array, each processor can identify the global indices of the atoms in the bond term and can use this in conjunction with the local sorted atoms list *and a binary search algorithm* to find the atoms in local atom list.

7. Using the local atom identity, the potential energy and force can be calculated.

It is worth mentioning that although rigid body units are not bearing any potential parameters, their definition requires that their topology is distributed in the same manner as the rest of the intra-molecular like interactions.

Note that, at the start of a simulation DL_POLY_4 allocates individual bonded interactions to specific processors, based on the domains of the relevant atoms (DL_POLY_4 routine BUILD_BOOK_INTRA). This means that each processor does not have to handle every possible bond term to find those relevant to its domain. Also this allocation is updated as atoms move from domain to domain i.e. during the *relocation* process that follows the integration of the equations of motion (DL_POLY_4 routine RELOCATE_PARTICLES). Thus the allocation of bonded terms is effectively dynamic, changing in response to local changes.

11.1.3 Distributing the Non-bonded Terms

DL_POLY_4 calculates the non-bonded pair interactions using the link cell algorithm due to Hockney and Eastwood [111]. In this algorithm a relatively short ranged potential cutoff ($r_{\rm cut}$) is assumed. The simulation cell is logically divided into so-called link cells, which have a width not less than (or equal to) the cutoff distance. It is easy to determine the identities of the atoms in each link cell. When the pair interactions are calculated it is already known that atom pairs can only interact if they are in the same link cell, or are in link cells that share a common face. Thus using the link cell 'address' of each atom, interacting pairs are located easily and efficiently via the 'link list' that identifies the atoms in each link cell. So efficient is this process that the link list can be recreated every time step at negligible cost.

For reasons, partly historical, the link list is used to construct a Verlet neighbour list [22]. The Verlet list records the indices of all atoms within the cutoff radius ($r_{\rm cut}$) of a given atom. The use of a neighbour list is not strictly necessary in the context of link-cells, but it has the advantage here of allowing a neat solution to the problem of 'excluded' pair interactions arising from the intramolecular terms and frozen atoms (see below).

In DL_POLY_4, the neighbour list is constructed *simultaneously* on each node, using the DD adaptation of the link cell algorithm to share the total burden of the work reasonably equally between nodes. Each node is thus responsible for a unique set of non-bonded interactions and the neighbour list is therefore different on each node.

A feature in the construction of the Verlet neighbour list for macromolecules is the concept of *excluded atoms*, which arises from the need to exclude certain atom pairs from the overall list. Which atom pairs need to be excluded is dependent on the precise nature of the force field model, but as a minimum atom pairs linked via extensible bonds or constraints and atoms (grouped in pairs) linked via valence angles are probable candidates. The assumption behind this requirement is that atoms that are formally bonded in a chemical sense, should not participate in non-bonded interactions. (However, this is not a universal requirement of all force fields.) The same considerations are needed in dealing with charged excluded atoms.

The modifications necessary to handle the excluded and frozen atoms are as follows. A distributed *excluded atoms list* is constructed by the DL_POLY_4 routine BUILD_EXCL_INTRA at the start of the simulation and is then used in conjunction with the Verlet neighbour list builder LINK_CELL_PAIRS to ensure that excluded interactions are left out of the pair force calculations. Note that, completely frozen pairs of atoms are excluded in the same manner. The excluded atoms list is updated during the atom relocation process described above (DL_POLY_4 routine EXCHANGE_PARTICLES).

Once the neighbour list has been constructed, each node of the parallel computer may proceed independently to calculate the pair force contributions to the atomic forces (see routine TWO_BODY_FORCES).

The potential energy and forces arising from the non-bonded interactions, as well as metal and Tersoff interactions are calculated using interpolation tables. These are generated in the following routines: VDW_GENERATE, METAL_GENERATE, METAL_TABLE_DERIVATIVES and TERSOFF_GENERATE.

11.1.4 Modifications for the Ewald Sum

For systems with periodic boundary conditions DL_POLY_4 employs the Ewald Sum to calculate the coulombic interactions (see Section 2.4.1.5). It should be noted that DL_POLY_4 uses only the Smoothed Particle Mesh (SPME) form of the Ewald sum.

Calculation of the real space component in DL_POLY_4 employs the algorithm for the calculation of the non-bonded interactions outlined above, since the real space interactions are now short ranged (implemented in EWALD_REAL_FORCES routine).

The reciprocal space component is calculated using Fast Fourier Transform (FFT) scheme of the SMPE method [71, 114] as discussed in Section 2.4.1.5. The parallelisation of this scheme is entirely handled within the DL_POLY_4 by the 3D FFT routine PARALLEL_FFT, (using GPFA_MODULE) which is known as the Daresbury advanced Fourier Transform, due to I.J. Bush [115]. This routine distributes the SPME charge array over the processors in a manner that is completely commensurate with the distribution of the configuration data under the DD strategy. As a consequence the FFT handles all the necessary communication implicit in a distributed SPME application. The DL_POLY_4 subroutine EWALD_SPME_FORCES perfoms the bulk of the FFT operations and charge array construction, while SPME_FORCES calculates the forces.

Other routines required to calculate the Ewald sum include EWALD_MODULE,
EWALD_EXCL_FORCES,
EWALD_FRZN_FORCES and SPME_CONTAINER.

11.1.5 Metal Potentials

The simulation of metals (Section 2.3.2) by DL_POLY_4 makes use of density dependent potentials. The dependence on the atomic density presents no difficulty however, as this class of potentials can be resolved into pair contributions. This permits the use of the distributed Verlet neighbour list as outlined above. DL_POLY_4 implements these potentials in various subroutines with names beginning with METAL_.

11.1.6 Tersoff, Three-Body and Four-Body Potentials

DL_POLY_4 can calculate Tersoff, three-body and four-body interactions. Although some of these interactions have similar terms to some intramolecular ones (three-body to the bond angle and four-body to inversion angle), these are not dealt with in the same way as the normal bonded interactions. They are generally very short ranged and are most effectively calculated using a link-cell scheme [111]. No reference is made to the Verlet neighbour list nor the excluded atoms list. It follows that atoms involved these interactions can interact via non-bonded (pair) forces and ionic forces also. Note that contributions from frozen pairs of atoms to these potentials are excluded. The calculation of the Tersoff three-body and four-body terms is distributed over processors on the basis of the domain of the central atom in them. DL_POLY_4 implements these potentials in the following routines TERSOFF_FORCES, TERSOFF_GENERATE, THREE_BODY_FORCES and FOUR_BODY_FORCES.

11.1.7 Globally Summed Properties

The final stage in the DD strategy, is the global summation of different (by terms of potentials) contributions to energy, virial and stress, which must be obtained as a global sum of the contributing terms calculated on all nodes.

The DD strategy does not require a global summation of the forces, unlike the Replicated Data method used in DL_POLY_Classic, which limits communication overheads and provides smooth parallelisation to large processor counts.

11.1.8 The Parallel (DD tailored) SHAKE and RATTLE Algorithms

The essentials of the DD tailored SHAKE and RATTLE algorithms (see Section 3.2) are as follows:

1. The bond constraints acting in the simulated system are allocated between the processors, based on the location (i.e. domain) of the atoms involved.
2. Each processor makes a list of the atoms bonded by constraints it must process. Entries are zero if the atom is not bonded.
3. Each processor passes a copy of the array to the neighbouring processors which manage the domains in contact with its own. The receiving processor compares the incoming list with its own and keeps a record of the shared atoms and the processors which share them.
4. In the first stage of the algorithms, the atoms are updated through the usual Verlet algorithm, without regard to the bond constraints.
5. In the second (iterative) stage of the algorithms, each processor calculates the incremental correction vectors for the bonded atoms in its own list of bond constraints. It then sends specific correction vectors to all neighbours that share the same atoms, using the information compiled in step 3.
6. When all necessary correction vectors have been received and added the positions of the constrained atoms are corrected.
7. Steps 5 and 6 are repeated until the bond constraints are converged.
8. Finally, the change in the atom positions from the previous time step is used to calculate the atomic velocities.

The compilation of the list of constrained atoms on each processor, and the circulation of the list (items 1 - 3 above) is done at the start of the simulation, but thereafter it needs only to be done every time a constraint bond atom is relocated from one processor to another. In this respect DD-SHAKE and DD-RATTLE resemble every other intramolecular term.

Since the allocation of constraints is based purely on geometric considerations, it is not practical to arrange for a strict load balancing of the DD-SHAKE and DD-RATTLE algorithms. For many systems, however, this deficiency has little practical impact on performance.

11.1.9 The Parallel Rigid Body Implementation

The essentials of the DD tailored RB algorithms (see Section 3.6) are as follows:

1. Every processor works out a list of all local and halo atoms that are qualified as free (zero entry) or as members of a RB (unit entry.
2. The rigid body units in the simulated system are allocated between the processors, based on the location (i.e. domain) of the atoms involved.
3. Each processor makes a list of the RB and their constituting atoms that are fully or partially owned by the processors domain.

4. Each processor passes a copy of the array to the neighbouring processors which manage the domains in contact with its own. The receiving processor compares the incoming list with its own and keeps a record of the shared RBs and RBs' constituent atoms, and the processors which share them. *Note that a RB can be shared between up to* **eight** *domains!*

5. The dynamics of each RB is calculated in full on each domain but domains only update $\{\underline{r}, \underline{v}, \underline{f}\}$ of RB atoms which they own. *Note that a site/atom belongs to* **one and only one** *domain at a time (no sharing) !*

6. Strict bookkeeping is necessary to avoid multiple counting of kinetic properties. $\{\underline{r}, \underline{v}, \underline{v}\}$ updates are necessary for halo parts (particles) of partially shared RBs. For all domains the kinetic contributions from each fully or partially present RB are evaluated in full and then waited with the ratio - number of RB's sites local to the domain to total RB's sites, and then globally summed.

The compilation of the lists in items 1 - 3 above and their circulation of the list is done at the start of the simulation, but thereafter these need updating on a local level every time a RB site/atom is relocated from one processor to another. In this respect RBs topology transfer resembles every other intramolecular term.

Since the allocation of RBs is based purely on geometric considerations, it is not practical to arrange for a strict load balancing. For many systems, however, this deficiency has little practical impact on performance.

11.2 Source Code

11.2.1 Modularisation Principles

Modules in DL_POLY_4 are constructed to define parameters and variables (scalars and arrays) and/or develop methods that share much in common. The division is far from arbitrary and module interdependence is reduced to minimum. However, some dependencies exist which leads to the following division by groups in hierarchical order:

- **precision module**:: KINDS_F90

  The precision module defines the working precision `wp` of all real variables and parameters in DL_POLY_4. By default it is set to 64-bit (double) precision. If the precision is changed, the user must check whether the specific platform supports it and make sure it is allowed for in the MPI implementation. If all is OK then the code must be recompiled.

- **MPI module**:: MPI_MODULE

  The MPI module implements all MPI functional calls used in DL_POLY_4. It is only used when DL_POLY_4 is to be compiled in serial mode.

- **communication module**:: COMMS_MODULE (MPI_MODULE)

  The communication module defines MPI related parameters and develops MPI related functions and subroutines such as: initialisation and exit; global synchronisation, sum, maximum and minimum; node ID and number of nodes; simulation time. It is dependent on KINDS_F90 and on MPI_MODULE if MPI is emulated for DL_POLY_4 compilation in serial mode. The MPI_MODULE implements all MPI functional calls used in DL_POLY_4.

- **global parameters module**:: SETUP_MODULE

  The global parameters module holds all important global variables and parameters (see above). It is dependent on KINDS_F90.

- **parse module**:: PARSE_MODULE

The parse module develops several methods used to deal with textual input: `get_line strip_blanks lower_case get_word word_2_real`. Depending on the method dependencies on KINDS_F90 COMMS_MODULE SETUP_MODULE DOMAINS_MODULE are found.

- **development module**:: DEVELOPMENT_MODULE

  The development module contains several methods used to help with testing and debugging DL_POLY_4. Depending on the method dependencies on KINDS_F90 COMMS_MODULE SETUP_MODULE DOMAINS_MODULE are found.

- **netCDF module**:: NETCDF_MODULE

  The netCDF module contains all important netCDF functions and global variables in DL_POLY_4 context. It is dependent on KINDS_F90.

- **I/O module**:: IO_MODULE

  The I/O module contains all important global variables that define the I/O methods and types used in the package and contains basic routines essential for the I/O in DL_POLY_4. It is dependent on KINDS_F90.

- **domains module**:: DOMAINS_MODULE

  The domains module defines DD parameters and maps the available computer resources on a DD grid. The module does not depend on previous modules but its mapping subroutine is dependent on KINDS_F90 and COMMS_MODULE.

- **site module**:: SITE_MODULE

  The site module defines all site related arrays (FIELD) and is dependent on KINDS_F90 only. However, it also develops an allocation method that is dependent on SETUP_MODULE.

- **configuration module**:: CONFIG_MODULE

  The configuration module defines all configuration related arrays (CONFIG) and is dependent on KINDS_F90 only. However, it also develops an allocation method that is dependent on SETUP_MODULE.

- **vnl module**:: VNL_MODULE

  The Verlet neighbour list (VNL) module defines all VNL related control variables and arrays needed for the VNL conditional update functionality, and is dependent on KINDS_F90 only. However, it is assisted by a VNL_CHECK routine that is dependent on more modules.

- **defects modules**:: DEFECTS_MODULE DEFECTS1_MODULE

  The defects modules define all defects and configuration related arrays (REFERENCE) and are dependent on KINDS_F90 only. However, they also develop allocation methods that are dependent on SETUP_MODULE.

- **dpd module**:: DPD_MODULE

  The Dissipative Particle Dynamics (DPD) module defines all DPD related control variables and arrays needed for the DPD functionality, and is dependent on KINDS_F90 only. However, it also develops an allocation method that is dependent on SETUP_MODULE.

- **electrostatic modules**:: EWALD_MODULE MPOLES_MODULE

  This modules define all variables and arrays needed for the SPME (i) refreshment k-space driven properties (ii) and multipola relectrostatics control variable and arrays in the DL_POLY_4 scope when. They depend on KINDS_F90 and but their allocation methods on SETUP_MODULE.

- ***inter*-molecular interactions modules**:: VDW_MODULE METAL_MODULE TERSOFF_MODULE THREE_BODY_MODULE FOUR_BODY_MODULE

  The intermolecular modules define all variables and potential arrays needed for the calculation of the particular interaction in the DL_POLY_4 scope. They depend on KINDS_F90. Their allocation methods depend on SETUP_MODULE.

- ***extra*-molecular interactions modules**:: KIM_MODULE PLUMED_MODULE

  These modules define all variables, arrays and functions needed OpenKIM and PLUMED integrable into DL_POLY_4 plugins. They depend on KINDS_F90. Their allocation methods depend on SETUP_MODULE.

- ***intra*-molecular interactions and site-related modules**:: RDF_MODULE Z_DENSITY_MODULE CORE_SHELL_MODULE CONSTRAINTS_MODULE PMF_MODULE RIGID_BODIES_MODULE TETHERS_MODULE BONDS_MODULE ANGLES_MODULE DIHEDRALS_MODULE INVERSIONS_MODULE

  These modules define all variables and potential or statistical grid arrays needed for the calculation of the particular interaction or distribution function in the DL_POLY_4 scope. They all depend on KINDS_F90 with allocation methods depending on SETUP_MODULE.

- **external field module**:: EXTERNAL_FIELD_MODULE

  This module defines all variables and potential arrays needed for the application of an external field in the DL_POLY_4 scope. It depends on KINDS_F90 and its allocation method on SETUP_MODULE.

- **langevin module**:: LANGEVIN_MODULE

  This module defines all variables and arrays needed for the application of NPT and N$\underline{\underline{\sigma}}$T Langevin routines in the DL_POLY_4 scope. It depends on KINDS_F90 and its allocation method on SETUP_MODULE.

- **minimise module**:: MINIMISE_MODULE

  This module defines all variables and arrays needed for the application of a Conjugate Gradient Method minimisation routine in the DL_POLY_4 scope. It depends on KINDS_F90 and its allocation method on SETUP_MODULE.

- **msd module**:: MSD_MODULE

  This module globalises a CONTROL variable.

- **statistics module**:: STATISTICS_MODULE

  This module defines all variables and arrays needed for the statistical accountancy of a simulation in DL_POLY_4. It depends on KINDS_F90 and its allocation methods on SETUP_MODULE and COMMS_MODULE.

- **greenkubo module**:: GREENKUBO_MODULE

  This module defines all variables and arrays needed for calculation of Green-Kubo relations during a simulation in DL_POLY_4. It depends on KINDS_F90 and its allocation methods on SETUP_MODULE.

- **kinetic module**:: KINETIC_MODULE

  The kinetic module contains a collection of routines for the calculation of various kinetic properties. It is dependent on KINDS_F90.

- **DaFT module**:: GPFA_MODULE PARALLEL_FFT

  These modules contain all necessary functionality for DL_POLY_4 DaFT and it GPFA 1D FFT dependence. They have dependencies on KINDS_F90, COMMS_MODULE.F90 and SETUP_MODULE.F90.

11.2.2 File Structure

Generally, the DL_POLY_4 file structure can be divided into four groups as follows:

- **general files** in the *source* directory
- **SERIAL specific** files in the *source/SERIAL* directory

The files in each group are listed in hierarchal order as closely as possible as examplified in the relevant DL_POLY_4 Makefies in the *build* subdirectory. The further down the category the file, the more dependent it is on the files listed above it.

11.2.3 Module Files

The DL_POLY_4 module files contain all global variables (scalars and arrays) and parameters as well as some general methods and generic functions intrinsically related to the purpose or/and contents of the specific module. The file-names and the methods or/and functions developed in them have self-explanatory names. More information of their purpose can be found in their headers.

The rest of files in DL_POLY_4 are dependent on the module files in various ways. The dependency relation to a module file is explicitly stated in the declaration part of the code.

11.2.4 General Files

The DL_POLY_4 general files are common to both MPI and SERIAL version of the code. In most cases, they have self-explanatory names as their order is matched as closely as possible to that occurring in the main segment of the code - DL_POLY. Only the first five files are exception of that rule; WARNING and ERROR are important reporting subroutines that have call points at various places in the code, and NUMERIC_CONTAINER, and SPME_CONTAINER are containers of simple functions and subroutines related in some way to their purpose in the code.

11.2.5 SERIAL Specific Files

These implement an emulation of some general MPI calls used in DL_POLY_4 source code when compiling in serial mode as well as some modified counterparts of the general files changed to allow for faster and/or better memory optimised serial execution. Names are self-explanatory.

11.2.6 Comments on MPI Handling

Only a few files make explicit calls to MPI routines.

11.2.7 Comments on SETUP_MODULE

The most important module, by far, is SETUP_MODULE, which holds the most important global parameters and variables (some of which serve as "parameters" for global array bounds, set in SET_BOUNDS). A brief account of these is given below:

parameter	value	function
`DLP_VERSION`	string	version string - number
`DLP_RELEASE`	string	release string - date

`pi`	3.14159265358979312	π constant
`twopi`	6.28318530717958623	2π constant
`fourpi`	12.56637061435917246	4π constant
`sqrpi`	1.772453850905588	$\sqrt[2]{\pi}$ constant
`rtwopi`	0.15915494309189535	$\frac{1}{2\pi}$ constant
`rt2`	1.41421356237309515	$\sqrt[2]{2}$ constant
`rt3`	1.73205080756887719	$\sqrt[2]{3}$ constant
`r4pie0`	138935.4835	electrostatics conversion factor to internal units, i.e. $\frac{1}{4\pi\epsilon_o}$
`boltz`	0.831451115	Boltzmann constant in internal units
		also used as Kelvin/Boltzmann energy unit (very rarely used)
`engunit`	*variable*	the system energy unit
`eu_ev`	9648.530821	eV energy unit (most used)
`eu_kcpm`	418.4	kcal/mol energy unit (often used)
`eu_kjpm`	100.0	kJouls/mol energy unit (rarely used)
`prsunt`	0.163882576	conversion factor for pressure from internal units to katms
`tenunt`	1.660540200	conversion factor for surface tension from internal units to dyn/cm
`del_max`	0.01	maximum bin sizes in Angstroms for distance grids
`delth_max`	0.20	maximum bin sizes in degrees for angle grids
`nread`	5	main input channel
`nconf`	11	configuration file input channel
`nfield`	12	force field input channel
`ntable`	13	tabulated potentials file input channel
`nrefdt`	14	reference configuration input channel
`nrite`	6	main output channel
`nstats`	21	statistical data file output channel
`nrest`	22	output channel accumulators restart dump file
`nhist`	23	trajectory history file channel
`ndefdt`	24	output channel for defects data file
`nrdfdt`	25	output channel for RDF data
`nzdfdt`	26	output channel for Z-density data file
`nrsddt`	27	output channel for displacements data files
`npdfdt`	28	output channel for raw PDF files
`ngdfdt`	29	output channel for normalised RDF data files
`nvafdt`	28	output channel for VAF files
`nmpldt`	29	output channel for the PLOLES data file
`seed(1:3)`	*variable*	pair of seeds for the random number generator
`lseed`	*variable*	logical swich on/off indicator for seeding
`mxsite`	*variable*	max number of molecular sites
`mxatyp`	*variable*	max number of unique atomic types
`mxtmls`	*variable*	max number of unique molecule types
`mxexcl`	*variable*	max number of excluded interactions per atom
`mxompl`	*variable*	max number of multipolar order specified
`mximpl`	*variable*	max number of multipolar total momenta for this order
`mxspl`	*variable*	SPME FFT B-spline order
`mxspl1`	*variable*	SPME FFT B-spline possible extension when $r_{\rm pad} > 0$
`kmaxa`	*variable*	SPME FFT amended array dimension (a direction)
`kmaxb`	*variable*	SPME FFT amended array dimension (b direction)

`kmaxc`	*variable*	SPME FFT amended array dimension (c direction)
`kmaxa1`	*variable*	SPME FFT original array dimension (a direction)
`kmaxb1`	*variable*	SPME FFT original array dimension (b direction)
`kmaxc1`	*variable*	SPME FFT original array dimension (c direction)
`mxtshl`	*variable*	max number of specified core-shell unit types in system
`mxshl`	*variable*	max number of core-shell units per node
`mxfshl`	*variable*	max number of related core-shell units (1+1)
`mxtcon`	*variable*	max number of specified bond constraints in system
`mxcons`	*variable*	max number of constraint bonds per a node
`mxfcon`	*variable*	max number of related constraint units (6+1)
`mxlshp`	*variable*	max number of shared particles per node $\mathtt{Max}(2\ \frac{\mathtt{mxshl}}{2}, 2\ \frac{\mathtt{mxcons}}{2}, \frac{\mathtt{mxlrgd\ *\ mxrgd}}{2})$
`mxproc`	*variable*	number of neighbour nodes in DD hypercube (26)
`mxtpmf(1:2)`	*variable*	max number of specified particles in a PMF unit (1:2)
`mxpmf`	*variable*	max number of PMF constraints per a node
`mxfpmf`	*variable*	max number of related PMF units (1+1)
`mxtrgd`	*variable*	max number of types RB units
`mxrgd`	*variable*	max number of RB units per node
`mxlrgd`	*variable*	max number of constituent particles of an RB unit
`mxfrgd`	*variable*	max number of related RB units (1+1)
`mxtteth`	*variable*	max number of specified tethered potentials in system
`mxteth`	*variable*	max number of tethered atoms per node
`mxftet`	*variable*	max number of related tether units (1+1)
`mxpteth`	*variable*	max number of parameters for tethered potentials (3)
`mxtbnd`	*variable*	max number of specified chemical bond potentials in system
`mxbond`	*variable*	max number of chemical bonds per node
`mxfbnd`	*variable*	max number of related chemical bonds (1+(6*(6+1))/2)
`mxpbnd`	*variable*	max number of parameters for chemical bond potentials (4)
`mxgbnd`	*variable*	max number of grid points in chemical bond pot. arrays (> 1004)
`mxtang`	*variable*	max number of specified bond angle potentials in system
`mxangl`	*variable*	max number of bond angles per node
`mxfang`	*variable*	max number of related bond angles (1+(6*(6+1))/2)
`mxpang`	*variable*	max number of parameters for bond angle potentials (6)
`mxgang`	*variable*	max number of grid points in bond angle pot. arrays (> 1004)
`mxtdih`	*variable*	max number of specified dihedral angle potentials in system
`mxdihd`	*variable*	max number of dihedral angles per node
`mxfdih`	*variable*	max number of related dihedral angles (1+((6-2)6*(6+1))/2)
`mxpdih`	*variable*	max number of parameters for dihedral angle potentials (7)
`mxgdih`	*variable*	max number of grid points in dihedral angle pot. arrays (> 1004)
`mxtinv`	*variable*	max number of specified inversion angle potentials in system
`mxinv`	*variable*	max number of inversion angles per node
`mxfinv`	*variable*	max number of related inversion angles (1+(6*(6+1))/4)
`mxpinv`	*variable*	max number of parameters for inversion angle potentials (3)
`mxginv`	*variable*	max number of grid points in inversion angle pot. arrays (> 1004)
`mxrdf`	*variable*	max number of pairwise RDF in system
`mxgrdf`	*variable*	number of grid points for RDF and Z-density arrays (> 1004)
`mxgele`	*variable*	max number of grid points for ewald exclusion potential arrays
`mxgusr`	*variable*	number of grid points for umbrella sampling restraint's RDF (> 1004)
`mxvdw`	*variable*	max number of van der Waals potentials in system
`mxpvdw`	*variable*	max number of van der Waals potential parameters (5)
`mxgvdw`	*variable*	max number of grid points in vdw potential arrays (> 1004)
`mxmet`	*variable*	max number of metal potentials in system

`mxmed`	*variable*	max number of metal density potentials in system
`mxmds`	*variable*	max number of metal extra density potentials in system
`mxpmet`	*variable*	max number of metal potential parameters (9)
`mxgmet`	*variable*	max number of grid points in metal potential arrays (> 1004)
`mxter`	*variable*	max number of Tersoff potentials in system
`mxpter`	*variable*	max number of Tersoff potential parameters (11)
`mxgter`	*variable*	max number of grid points in tersoff potential arrays (> 1004)
`mxgrid`	*variable*	max number of grid points in potential arrays (> 1004)
`mxtana`	*variable*	max number of PDFs per type
`mxgana`	*variable*	max number of grid points for PDFs arrays
`mxgbnd`	*variable*	max number of grid points for chemical bonds PDFs
`mxgang`	*variable*	max number of grid points for bond angles PDFs
`mxgdih`	*variable*	max number of grid points for dihedral angles PDFs
`mxginv`	*variable*	max number of grid points for inversion angles PDFs
`mxtbp`	*variable*	max number of three-body potentials in system
`mx2tbp`	*variable*	array dimension of three-body potential parameters
`mxptbp`	*variable*	max number of three-body potential parameters (5)
`mxfbp`	*variable*	max number of four-body potentials in system
`mx2fbp`	*variable*	array dimension of four-body potential parameters
`mxpfbp`	*variable*	max number of four-body potential parameters (3)
`mxpfld`	*variable*	max number of external field parameters (5)
`mxstak`	*variable*	dimension of stack arrays for rolling averages
`mxnstk`	*variable*	max number of stacked variables
`mxlist`	*variable*	max number of atoms in the Verlet list on a node
`mxcell`	*variable*	max number of link cells per node
`mxatms`	*variable*	max number of local+halo atoms per node
`mxatdm`	*variable*	max number of local atoms per node
`mxbfdp`	*variable*	max dimension of the transfer buffer for deport functions
`mxbfss`	*variable*	max dimension of the transfer buffer for statistics functions
`mxbfxp`	*variable*	max dimension of the transfer buffer for export functions
`mxbfsh`	*variable*	max dimension of the transfer buffer for shared units
`mxbuff`	*variable*	max dimension of the principle transfer buffer
`zero_plus`	*variable*	the machine representation of +0 at working precision
`half_plus`	*variable*	the machine representation of $+0.5\uparrow$ at working precision
`half_minus`	*variable*	the machine representation of $+0.5\downarrow$ at working precision

Chapter 12

Examples

Scope of Chapter

This chapter describes the example simulations and benchmark tests for DL_POLY_4, the input and output files for which are in the *data* sub-directory.

12.1 Example Simulations

Because of the size of the data files for the DL_POLY_4 example simulations, they are not shipped in the standard download of the DL_POLY_4 source. Instead users are requested to download them from the CCP5 FTP server as follows:

```
FTP site : ftp.dl.ac.uk
Username : anonymous
Password : your email address
Directory: ccp5/DL_POLY/DL_POLY_4.0/DATA
Files    : test_X.tar.xz
```

where '_X' stands for the example simulation number.

Remember to use the BINARY data option when transferring these files.

Unpack the files in the 'data' subdirectory using 'gunzip' and 'tar -xf' to create the 'TEST_X' directory.

These are provided to give examples of DL_POLY_4 simulations and demonstrate a limited set of relevant functionality over a limited extent of molecular systems' complexity only. **Without modification, they are not necessarily appropriate for serious simulation of the given systems.** In other words, the examples are not warranted to have well-defined force fields in terms of applicability, transferability and fullness, nor are they likely to have a well-defined state point (i.e. initial configurations may be away from equilibrium, if physical at all).

The README.txt file supplied both in the *data* directory and in the directory on the CCP5 FTP server provides a list of all example simulations used as test cases to check that DL_POLY_4 is working correctly, including those described in more detail below. All the jobs are of a size suitable to test the code in parallel execution. They may not be suitable for a single processor computer. The files are stored in compressed format. The examples can be run by typing

select n

from the *execute* directory, where n is the number of the test case. The *select* macro will copy the appropriate input files (at least CONTROL, CONFIG, and FIELD in all cases) to the *execute* directory ready for execution. The output file OUTPUT may be compared with the file supplied in the *data* directory.

12.1.1 Example 1: Sodium Chloride

This is a 27,000 ion system with unit electric charges on sodium and chlorine. Simulation at 500 K with a NVT Berendsen ensemble. The SPME method is used to calculate the Coulombic interactions.

12.1.2 Example 2: DPMC in Water

The system consists of 200 DMPC molecules in 9379 and water molecules. Simulation at 300 K using NVE ensemble with SPME and RATTLE algorithm for the constrained motion. The total system size is 51,737 atoms.

12.1.3 Example 3: $KNaSi_2O_5$ - Potassium/Sodium Disilicate Glass

Potassium Sodium disilicate glass ($NaKSi_2O_5$) using two and three-body potentials. Some of the two-body potentials are read from the TABLE file. Simulation at 1000 K using NVT Nosé-Hoover ensemble with SPME. Cubic periodic boundaries are in use. The total system size is 69,120 ions.

12.1.4 Example 4: Gramicidin A Molecules in Water

The system consists of 8 gramicidin A molecules in aqueous solution (32,096 water molecules) with total of 99,120 of atoms. Simulation at 300 K using NPT Berendsen ensemble with SPME and SHAKE/RATTLE algorithm for the constrained motion.

12.1.5 Example 5: SiC with Tersoff Potentials

The system consists of 74,088 atoms. Simulation at 300 K using NPT Nosé-Hoover ensemble with Tersoff forces and no electrostatics.

12.1.6 Example 6: Cu_3Au alloy with Sutton-Chen (metal) Potentials

The systems consists of 32,000 atoms. Simulation at 300 K using NVT Nosé-Hoover ensemble with Sutton-Chen forces and no electrostatics.

12.1.7 Example 7: Lipid Bilayer in Water

The systems consists of 12,428 atoms. Simulation at 300 K using NVT Berendsen ensemble with SPME and SHAKE/RATTLE algorithm for the constrained motion.

12.1.8 Examples 8 and 9: MgO with Adiabatic and with Relaxed Shell Models

These system consist of 8,000 (4,000 shells) charged points. Simulation at 3000 K using NPT Berendsen ensemble with SPME.

12.1.9 Example 10: Potential of Mean Force on K+ in Water

The system consists of 13,500 (500 PMFs) atoms. Simulation at 300 K using NPT Berendsen ensemble with SPME and SHAKE/RATTLE algorithm for the constrained motion.

12.1.10 Example 11: Cu_3Au Alloy with Gupta (metal) Potentials

The system consists of 32,000 atoms. Simulation at 300 K using NVT Nosé-Hoover ensemble with Gupta forces and no electrostatics.

12.1.11 Example 12: Cu with EAM (metal) Potential

The system consists of 32,000 atoms. Simulation at 300 K using NPT Berendsen ensemble with EAM tabulated forces and no electrostatics.

12.1.12 Examples 13 and 14: Al with Analytic and with EAM Tabulated Sutton-Chen (metal) Potentials

The system consists of 32,000 atoms. Simulation at 300 K using NVT Evans ensemble with Sutton-Chen forces and no electrostatics.

12.1.13 Examples 15: NiAl Alloy with EAM (metal) Potentials

The system consists of 27,648 atoms. Simulation at 300 K using NVT Evans ensemble with EAM tabulated forces and no electrostatics.

12.1.14 Examples 16: Fe with Finnis-Sincair (metal) Potential

The system consists of 31,250 atoms. Simulation at 300 K using NPT Berendsen ensemble with Finnis-Sinclair forces and no electrostatics.

12.1.15 Examples 17: Ni with EAM (metal) Potential

The system consists of 32,000 atoms. Simulation at 300 K using NPT Berendsen ensemble with EAM tabulated forces and no electrostatics.

12.1.16 Examples 18 and 19: SPC IceVII Water with CBs and with RBs

The system consists of 11,664 (34,992 atoms) water molecules. Simulation at 25 K using NVE ensemble with CGM force minimisation and SPME electrostatics.

12.1.17 Example 20: NaCl Molecules in SPC Water Represented as CBs+RBs

The system consists of 64 NaCl ion pairs with 4,480 water molecules represented by constraint bonds and 4,416 water molecules represented by ridig bodies. Totalling 26,816 atoms. Simulation at 295 K using NPT Berendsen ensemble with CGM energy minimisation and SPME electrostatics.

12.1.18 Example 21: TIP4P Water: RBs with a Massless Charged Site

The system consists of 7,263 TIP4P rigid body water molecules totaling 29,052 particles. Simulation at 295 K using NPT Berendsen ensemble with CGM energy minimisation and SPME electrostatics.

12.1.19 Example 22: Ionic Liquid Dimethylimidazolium Chloride as RBs

The system consists of 44,352 ions. Simulation at 400 K using NPT Berendsen ensemble, using both particle and rigid body dynamics with SPME electrostatics.

12.1.20 Example 23: Calcite Nano-Particles in TIP3P Water

In this case 600 molecules of calcium carbonate in the calcite structure form 8 nano-particles which are suspended in 6,904 water molecules, represented by a flexible 3-centre TIP3P model. Simulation with SPME electrostatics at 310 K and 1 atmosphere maintained in a Hoover NPT ensemble. The system consists of 23,712 ions.

12.1.21 Example 24: Iron/Carbon Alloy with 2BEAM (metal) Potentials

In this case a steel alloy of iron and carbon in ratio 35132 to 1651 is modelled using an EEAM potential forcefield. Simulation at 1000 K and 0 atmosphere is maintained in a Berendsen NPT ensemble. The system consists of 36,803 particles.

12.1.22 Example 25: Iron/Chromium Alloy with 2BEAM (metal) Potentials

In this case a steel alloy of iron and chromium in ratio 27635 to 4365 is modelled using an 2BEAM potential forcefield. Simulation at 300 K and 0 atmosphere is maintained in an Evans NVT isokinetic ensemble. The system consists of 32,000 particles.

12.1.23 Examples 26 and 27: Hexane and Methanol Melts, with Full Atomistic and Coarse-Grained Force-Fields

These two examples contain a Hexane and a Methanol melt respectively, (1000 molecules each) modelled by the OPLSAA force-field (FF). Each system is also supplied in a CG-mapped representation as converted by VOTCA, http://www.votca.org/, or DL_CGMAP http://www.ccp5.ac.uk/projects/ccp5_cg.shtml.

These test cases are to exemplify the Coarse-Graining (CG) procedure (see Chapter 4), including FA-to-CG mapping and obtaining the PMF data by means of Boltzmann Inversion [116]. As a result, DL_POLY_4 could be used for simulating a CG system with numerically defined, tabulated FFs, see TABBND, TABANG, TABDIH and TABINV files for intra-molecular potentials, and TABLE for inter-molecular (short-range, VDW) potentials.

Both tests are also available as parts of the tutorial cases from the VOTCA package [117]. Therefore, the CONFIG, CONTROL and FIELD input files are fully consistent with the corresponding setup files found in the VOTCA tutorial directories "csg-tutorials/hexane" and "csg-tutorials/methanol'.

12.1.24 Example 28: Butane in CCl_4 Solution with Umbrella Sampling via PLUMED

Free Energy calculation for Buthane with respect to the dihedral angle as collective variable. We use umbrella sampling as implemented in PLUMED.

PLUMED enabling in CONTROL:

```
plumed input umbrella.dat
```

Contents of umbrella.dat:

```
phi: TORSION ATOMS=1,2,3,4
restraint-phi: RESTRAINT ARG=phi KAPPA=500 AT=1.20
PRINT STRIDE=10 ARG=phi,restraint-phi.bias FILE=COLVAR
```

Two extra output files are generated in this case: OUTPUT.PLUMED and COLVAR.

Note, a DL_POLY_4 version with PLUMED enabled is used for this.

12.1.25 Example 29: Iron with tabulated EAM (metal) Potential, TTM and Cascade

In this example 54,000 atoms of iron are modelled with a tabulated embedded-atom potential optimised to produce correct energetics of point defects and clusters (M07 in [118]). An energy impact of 10 keV is applied to an atom and the resulting radiation damage is evolved using the Two-Temperature Model (TTM) to represent energy transfers due to electron-phonon coupling and electronic stopping between atoms and a continuum electronic gas [99].

This test case produces additional output files: DUMP_E, LATS_E, LATS_I, PEAK_E and PEAK_I. It also requires an additional input file (Ce.dat) to supply tabulated heat capacity data required for evolving the electronic system.

12.1.26 Example 30: Silicon with original Tersoff Potential, TTM and Swift heavy ion irradiation

This system consists of 200,000 atoms of silicon modelled using an original Tersoff (T3) potential. The Two-Temperature Model (TTM) is in use and an energy deposition is applied to the electronic system using a Gaussian spatial function, an exponentially decaying temporal function and an electronic stopping power of 50,000 eV/nm. This simulation represents Swift heavy ion irradiation in silicon, including the resulting creation of ion tracks [101].

12.1.27 Example 31: Tungsten with extended Finnis-Sinclair Potential, TTM and laser irradiation

This system consists of 722,672 atoms of tungsten modelled using an extended Finnis-Sinclair potential. The Two-Temperature Model (TTM) is in use and an energy deposition is applied to the electronic system using a spatial function that is homogeneous in x and y directions and exponentially decaying in the z direction, as well as a Gaussian temporal function. This energy deposition represents a laser applied to the surface of a thin film of tungsten [119] with a surface fluence of 36 mJ/cm^2 and penetration depth of 12.5 nm, causing the film to expand outwards in the z direction.

Additional input files (Ce.dat and g.dat) are required to supply tabulated heat capacity and electron-phonon coupling values.

12.2 Benchmark Cases

DL_POLY_4 benchmark test cases are available to download them from the CCP5 FTP server as follows:

```
FTP site : ftp.dl.ac.uk
Username : anonymous
Password : your email address
Directory: ccp5/DL_POLY/DL_POLY_4.0/BENCH
```

The DL_POLY_4 authors provide these on an "AS IS" terms. For more information refer to the README.txt file within.

Appendix A

DL_POLY_4 Dissipative Particle Dynamics

A.1 Introduction

Although in a Molecular Dynamics sense Dissipative Particle Dynamics (DPD) is regarded as a type of thermostat, on its own it is an off-lattice, discrete particle method for modelling mesoscopic systems in a fluid state. It has little in common with Lattice Gas Automata and Lattice Boltzmann dynamics methods [92], except in its application to systems of similar length and time scales, and last but not least that it captures hydrodynamics behaviour.

The DPD method inherits its methodology from Brownian/Langevin Dynamics (BD). However, it differs from BD in an important way: it is *Galilean invariant* and for this reason conserves hydrodynamic behaviour, while the BD method does not (its microscopic behaviour is only diffusive. Many systems in their fluid state are crucially dependent on hydrodynamic interactions and it is essential to retain this feature in their models. DPD is particularly useful for simulating coarse-grained systems on the near-molecular scale, such as polymers, biopolymers, lipids, emulsions and surfactants – systems in which large scale structure evolves on a time scale that is too long to be modelled effectively by traditional MD. The particles in DPD [93] are not regarded as molecules in a fluid but as lumps of molecules grouped to form a *fluid particle* in much the same spirit as the renormalisation group has been applied in polymer physics where lumps of monomers are grouped to form a *bead*. Hence, the beads are regarded as carriers of momentum.

It is worth noting that DPD may also be used when such systems experience shear and flow gradients.

A.2 Outline of Method

Following [50] the DPD algorithm can be summarised by the following:

- A condensed phase system may be modelled as a system of 'free' particles interacting directly through *soft* forces. Note that DL_POLY_4 allows for the application of the DPD thermostat beyond systems of free particles only. Thus it will be valid on systems with any inratamolecular like interactions.

- The system is coupled to a heat bath via stochastic forces, which act on the particles in a pairwise manner.

- The particles also experience a damping or drag force, which also acts in a pairwise manner.

- Thermodynamic equilibrium is maintained through the balance of the stochastic and drag forces, i.e. the method satisfies the fluctuation-dissipation theorem.

- At equilibrium (or steady state) the properties of the system are calculated as averages over the individual particles, as in traditional Molecular Dynamics.

Therefore, the equation of motion are the same as these for the microcanonical ensemble (NVE) but force, f_i, on particle i is now a sum of pair forces:

$$\underline{f}_i = \sum_{j \neq i}^{N} \left(\underline{f}_{ij}^C + \underline{f}_{ij}^D + \underline{f}_{ij}^R \right) \quad , \tag{A.1}$$

in which $\underline{f}_{ij}^C$, $\underline{f}_{ij}^D$ and $\underline{f}_{ij}^R$ are the *conservative*, *drag* and *random* (or *stochastic*) pair forces respectively. Each represents the force exerted on particle i due to the presence of particle j.

The conservative interactions are usually *soft* (i.e. weakly interacting) so that the particles can pass by each other (or even through each other) relatively easily so that equilibrium is achieved quickly. A common form of interaction potential is an inverse parabola (part of VDW types of potentials, see Section 2.3.1)

$$V(r_{ij}) = \begin{cases} \frac{A_{ij}}{2} \; r_c \; \left(1 - \frac{r_{ij}}{r_c}\right)^2 & : \; r_{ij} < r_c \\ 0 & : \; r_{ij} \geq r_c \end{cases} \quad , \tag{A.2}$$

where $r_{ij} = |\underline{r}_j - \underline{r}_i|$, r_c is a cutoff radius and A_{ij} is the interaction strength (that may be the same for all particle pairs or may be different for different particle types).

Equation (A.1) gives rise to a repulsive force of the form:

$$\underline{f}_{ij}^C = A_{ij} \; w^C(r_{ij}) \frac{\underline{r}_{ij}}{r_{ij}} = A_{ij} \left(1 - \frac{r_{ij}}{r_c}\right) \frac{\underline{r}_{ij}}{r_{ij}} \quad . \tag{A.3}$$

This is the deterministic or *conservative* force $\underline{f}_{ij}^C$ exerted on particle i by particle j. Note the switching function:

$$w^C(r_{ij}) = \begin{cases} \left(1 - \frac{r_{ij}}{r_c}\right) & : \; r_{ij} < r_c \\ 0 & : \; r_{ij} \geq r_c \end{cases} \quad , \tag{A.4}$$

and the force are zero when $r_{ij} \geq r_c$ and thus the particles have an effective diameter of 1 in units of the cutoff radius r_c. In the DL_POLY_4 context all inter- and intra-molecular forces will fall into this category of force!

The stochastic forces experienced by the particles is again pairwise in nature and takes the form:

$$\underline{f}_{ij}^R = \sigma_{ij} w^R(r_{ij}) \zeta_{ij} \Delta t^{-\frac{1}{2}} \frac{\underline{r}_{ij}}{r_{ij}} \quad , \tag{A.5}$$

in which Δt is the time step and $w^R(r_{ij})$ is a switching function which imposes a finite limit on the range of the stochastic force. ζ_{ij} is a random number with zero mean and unit variance. The constant σ_{ij} is related to the temperature, as is understood from the role of the stochastic force in representing a heat bath.

Finally, the particles are subject to a drag force, which depends on the relative velocity between interacting pairs of particles:

$$\underline{f}_{ij}^D = -\gamma_{ij} w^D(r_{ij}) \left(\underline{r}_{ij} \cdot \underline{v}_{ij}\right) \frac{\underline{r}_{ij}}{r_{ij}^2} \quad , \tag{A.6}$$

where $w^D(r_{ij})$ is once again a switching function and $\underline{v}_{ij} = \underline{v}_j - \underline{v}_i$ is the inter-particle relative velocity. The constant γ_{ij} is the drag coefficient. It follows from the fluctuation-dissipation theorem that for thermodynamic equilibrium to result from this method the following relations must hold:

$$\sigma_{ij}^2 = 2 \; \gamma_{ij} k_B T \tag{A.7}$$

$$w^D(r_{ij}) = \left[w^R(r_{ij})\right]^2 \quad . \tag{A.8}$$

In practice, the switching functions are defined through:

$$w^R(r_{ij}) = \left[w^C(r_{ij})\right]^2 \quad , \tag{A.9}$$

which ensures that all interactions are switched off at the range $r_{ij} = r_c$.

In many DPD simulations, the stochastic and drag coefficients are often constant for all interactions, i.e. $\sigma_{ij} \equiv \sigma$ and $\gamma_{ij} \equiv \gamma$, although this assumption does not have to apply. In DL_POLY_4 the γ_{ij} coefficients may be supplied at the end of each specified vdw interaction potential as a parameter further to the last one for the particular vdw potential form. For a DPD thermostat to work correctly all possible two body interactions must be defined and all $\gamma_{ij} \neq 0$. What DL_POLY_4 will attempt first, if a two body interaction is missing, is to derive it using mixing rules (default may be overridden by user specification). However, if any of $\gamma_{ij} = 0$ then DL_POLY_4 will check for the existence of a global γ that may be optionally supplied by the user on the **ensemble nvt dpd*INT*** line and if it is non-zero a global override will occur. Otherwise, when the requirements for a DPD thermostat are not satisfied, everything else will result in a controlled termination.

A.3 Equation of state and dynamic properties

The form of the conservative force determines the equation of state for a DPD fluid, which can be derived using the virial theorem to express system pressure as follows:

$$\begin{aligned} \mathcal{P} &= \rho k_B T + \frac{1}{3V}\left\langle \sum_{j>i} (\underline{r}_i - \underline{r}_j)\cdot \underline{f}_{ij}^C \right\rangle && \text{(A.10)} \\ &= \rho k_B T + \frac{2\pi}{3}\rho^2 \int_0^{r_c} A\left(1-\frac{r}{r_c}\right) r^3 g(r)\, dr \quad , && \text{(A.11)} \end{aligned}$$

where $g(r)$ is a radial distribution function for the soft sphere model [50] and ρ is the DPD particle density. For sufficiently large densities ($\rho > 2$), $g(r)$ takes the same form and the equation of state can be well-approximated by:

$$\mathcal{P} = \rho k_B T + \alpha A \rho^2 \quad , \tag{A.12}$$

where the parameter $\alpha \approx 0.101 \pm 0.001$ has units equivalent to r_c^4. This expression permits the use of fluid compressibilities to obtain conservative force parameters for bulk fluids, e.g. for water $A \approx 75 k_B T/\rho$. Alternative equations of state may be obtained by modifying the functional form of conservative interactions to include localized densities (i.e. many-body DPD) [120, 121].

Transport coefficients for a DPD fluid without conservative forces can be derived using the expressions for the drag and stochastic forces[50, 122, 123]. The kinematic viscosity can be found to be

$$\nu \approx \frac{45 k_B T}{4\pi\gamma\rho r_c^3} + \frac{2\pi\gamma\rho r_c^5}{1575} \quad , \tag{A.13}$$

while the self-diffusion coefficient is given as

$$D \approx \frac{45 k_B T}{2\pi\gamma\rho r_c^3}. \tag{A.14}$$

The ratio of these two properties, the Schmidt number (Sc $= \nu/D$), is therefore:

$$\mathrm{Sc} \approx \frac{1}{2} + \frac{(2\pi\gamma\rho r_c^4)^2}{70875 k_B T} \tag{A.15}$$

and for values of the drag coefficient and density frequently used in DPD simulations, this value is of the order of unity, which is an appropriate magnitude for gases but three orders of magnitude too small for liquids.

This property of standard DPD does *not* rule it out for simulations of liquid phases except when hydrodynamics are important. It may also be argued that the self-diffusion of DPD particles might not correspond to that of individual molecules and thus a Schmidt number of the order 10^3 is unnecessary for modelling liquids [124]. Alternative thermostats are available in the DL_MESO [103] - http://www.ccp5.ac.uk/DL_MESO/ package, which can model systems with higher Schmidt numbers [125, 126].

A.4 Derivation of Equilibrium

The derivation of the DPD algorithm is based on the Fokker-Planck equation

$$\frac{\partial \rho}{\partial t} = \mathcal{L}\rho \tag{A.16}$$

where ρ is the equilibrium distribution function and $\mathcal{L}$ is the evolution operator, which may be split into *conservative* and *stochastic+dissipative* parts:

$$\mathcal{L} = \mathcal{L}^C + \mathcal{L}^{R+D} \tag{A.17}$$

with

$$\mathcal{L}^C = -\sum_{i=1}^{N} \frac{\underline{p}_i}{m_i} \frac{\partial}{\partial \underline{r}_i} - \sum_{i \neq j}^{N} \underline{f}_{ij}^C \frac{\partial}{\partial \underline{p}_i} \tag{A.18}$$

$$\mathcal{L}^{R+D} = \sum_{i=1}^{N} \hat{e}_{ij} \cdot \frac{\partial}{\partial \underline{p}_i} \left[\frac{\sigma^2}{2} \left\{ w^R(r_{ij}) \right\}^2 \hat{e}_{ij} \cdot \left\{ \frac{\partial}{\partial \underline{p}_i} - \frac{\partial}{\partial \underline{p}_j} \right\} + \gamma w^D \left(\hat{e}_{ij} \cdot \underline{v}_{ij} \right) \right] \quad , \tag{A.19}$$

where $\hat{e}_{ij} = \frac{\underline{r}_{ij}}{\underline{r}_{ij}}$.

When $\sigma = \gamma = 0$ then equation (A.16) becomes

$$\frac{\partial \rho}{\partial t} = \mathcal{L}^C \rho \quad , \tag{A.20}$$

for which the equilibrium solution is evidently

$$\rho^{eq} = \frac{1}{Z} \exp \left(\frac{1}{k_B T} \left[\sum_{i=1}^{N} \frac{p_i^2}{2m_i} + \frac{1}{2} \sum_{j \neq i}^{N} \phi(r_{ij}) \right] \right) \tag{A.21}$$

which is, of course, the Boltzmann distribution function for an equilibrium system. Thus it is apparent that for the simulation based on equation (A.16) to maintain the same distribution function, the terms in the operator $\mathcal{L}^{R+D}$ of equation (A.19) must sum to zero. It follows that the conditions given in equations (A.7) and (A.8) must apply.

A.5 Summary of Dissipative Particle Dynamics

DPD is a simple method that can be viewed as a novel thermostatting method for molecular dynamics. All that is required is a system of spherical particles enclosed in a periodic box undergoing time evolution as a result of the above forces. It should be noted that all computed interactions are pairwise, which means that the principle of the conservation of momentum in the system, or *Galilean invariance*, is preserved. The conservation of momentum is required for the preservation of hydrodynamic forces. Therefore, the DPD method is an NVT method that *preserves hydrodynamics*. The presence of hydrodynamics is important in annealing defects in ordered mesophases [127]. Thus DPD has an intrinsic advantage over other methods such as traditional molecular dynamics, dynamic density functional theory (which are purely *diffusive*!) or Monte Carlo methods, in trying to evolve a system towards an ordered thermodynamic equilibrium state.

Appendix B

DL_POLY_4 Periodic Boundary Conditions

B.1 Introduction

DL_POLY_4 is designed to accommodate a number of different periodic boundary conditions, which are defined by the shape and size of the simulation cell. Briefly, these are as follows (which also indicates the IMCON flag defining the simulation cell type in the CONFIG file - see Section 10.1.2):

1. None e.g. isolated polymer in space (`imcon` = 0)
2. Cubic periodic boundaries (`imcon` = 1)
3. Orthorhombic periodic boundaries (`imcon` = 2)
4. Parallelepiped periodic boundaries (`imcon` = 3)
5. Slab (X,Y periodic; Z non-periodic) (`imcon` = 6)

We shall now look at each of these in more detail. Note that in all cases the cell vectors and the positions of the atoms in the cell are to be specified in Angstroms (Å).

B.2 No periodic boundary (`imcon` = 0)

Simulations requiring no periodic boundaries are best suited to *in vacuuo* simulations, such as the conformational study of an isolated polymer molecule. This boundary condition is not recommended for studies in a solvent, since evaporation is likely to be a problem.

Note this boundary condition have to be used with caution. DL_POLY_4 is not naturally suited to carry out efficient calculations on systems with great fluctuation of the local density in space, as is the case for clusters in vacuum. The parallelisation and domain decomposition is therefore limited to eight domains (maximum of two in each direction in space).

This boundary condition should not used with the SPM Ewald summation method.

B.3 Cubic periodic boundaries (`imcon` = 1)

The cubic MD cell is perhaps the most commonly used in simulation and has the advantage of great simplicity. In DL_POLY_4 the cell is defined with the principle axes passing through the centres of the faces. Thus for a cube with sidelength D, the cell vectors appearing in the CONFIG file should be: (D,0,0); (0,D,0); (0,0,D). Note the origin of the atomic coordinates is the centre of the cell.

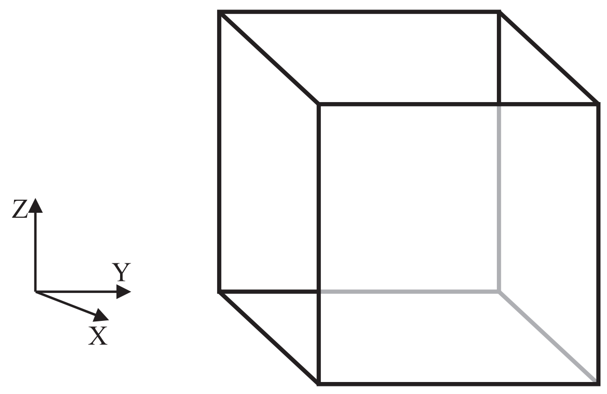


Figure B.1: The cubic MD cell

B.4 Orthorhombic periodic boundaries (`imcon` = 2)

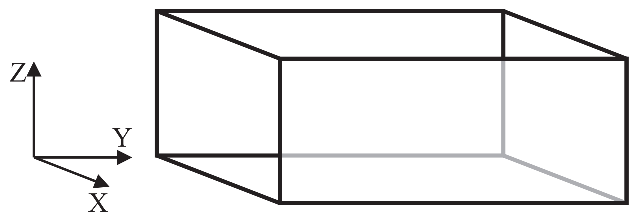


Figure B.2: The orthorhomic MD cell

The orthorhombic cell is also a common periodic boundary, which closely resembles the cubic cell in use. In DL_POLY_4 the cell is defined with principle axes passing through the centres of the faces. For an orthorhombic cell with sidelengths D (in X-direction), E (in Y-direction) and F (in Z-direction), the cell vectors appearing in the CONFIG file should be: (D,0,0); (0,E,0); (0,0,F). Note the origin of the atomic coordinates is the centre of the cell.

B.5 Parallelepiped periodic boundaries (`imcon` = 3)

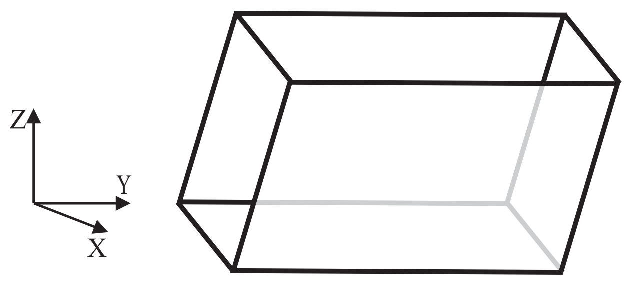


Figure B.3: The parallelepiped MD cell

The parallelepiped (e.g. monoclinic or triclinic) cell is generally used in simulations of crystalline materials, where its shape and dimension is commensurate with the unit cell of the crystal. Thus for a unit cell specified by three principal vectors $\underline{a}$, $\underline{b}$, $\underline{c}$, the MD cell is defined in the DL_POLY_4 CONFIG file by the vectors (La_1,La_2,La_3), (Mb_1,Mb_2,Mb_3), (Nc_1,Nc_2,Nc_3), in which L,M,N are integers, reflecting the multiplication of the unit cell in each principal direction. Note that the atomic coordinate origin is the centre of the MD cell.

B.6 Slab boundary conditions (`imcon` = 6)

Slab boundaries are periodic in the X- and Y-directions, but not in the Z-direction. They are particularly useful for simulating surfaces. The periodic cell in the XY plane can be any parallelogram. The origin of the X,Y atomic coordinates lies on an axis perpendicular to the centre of the parallelogram. The origin of the Z coordinate is where the user specifies it. However, it is recommended that it is in the middle of the slab. Domain decomposition division across Z axis is limited to 2.

If the XY parallelogram is defined by vectors $\underline{A}$ and $\underline{B}$, the vectors required in the CONFIG file are: (A_1,A_2,0), (B_1,B_2,0), (0,0,D), where D is any real number (including zero). If D is nonzero, it will be used by DL_POLY to help determine a 'working volume' for the system. This is needed to help calculate RDFs etc. (The working value of D is in fact taken as one of: 3×cutoff; or 2×max abs(Z coordinate)+cutoff; or the user specified D, whichever is the larger.)

The surface in a system with charges can also be modelled with DL_POLY_4 if periodicity is allowed in the Z-direction. In this case slabs of ions well-separated by vacuum zones in the Z-direction can be handled with `imcon` = 1, 2 or 3.

Appendix C

DL_POLY_4 Macros

Introduction

Macros are simple executable files containing standard UNIX commands. A number of the are supplied with DL_POLY_4 and are found in the *execute* sub-directory. These are not guaranteed to be immaculate but with little adaptation they can become a useful tool to a researcher. The available macros are as follows:

- *cleanup*
- *copy*
- *gopoly*
- *gui*
- *select*
- *store*

The function of each of these is described below. It is worth noting that most of these functions could be performed by the DL_POLY Java GUI [21].

cleanup

cleanup removes several standard data files from the *execute* sub-directory. It contains the UNIX commands:

```
rm -vf OUTPUT STATIS REVCON RVEIVE CFGMIN DEFECTS *DAT* *PMF *TAB MSDTMP
```

It is useful for cleaning the sub-directory up after a run. (Useful data should be stored elsewhere however!)

copy

copy invokes the UNIX commands:

```
mv -v CONFIG CONFIG.OLD
mv -v REVCON CONFIG
mv -v REVIVE REVOLD
```

which collectively prepare the DL_POLY_4 files in the *execute* sub-directory for the continuation of a simulation. It is always a good idea to store these files elsewhere in addition to using this macro.

gopoly

gopoly is used to submit a DL_POLY_4 job to the HPCx, which operates a LOAD-LEVELER job queuing system. It invokes the following script:

```
#@ shell = /usr/bin/tcsh
#
#@ job_type = parallel
#@ job_name = gopoly
#
#@ cpus = 32
#
#@ node_usage = not_shared
#@ network.MPI = csss,shared,US
#
#@ wall_clock_limit = 00:30:00
#@ account_no = mine
#
#@ output = $(job_name).$(schedd_host).$(jobid).out
#@ error  = $(job_name).$(schedd_host).$(jobid).err
#@ notification = never
#
#@ bulkxfer = yes
#@ data_limit = 850000000
#@ stack_limit = 10000000
#
#@ queue
#
# ENVIRONMENT SETTINGS
#
setenv MP_EAGER_LIMIT 65536
setenv MP_SHARED_MEMORY yes
setenv MEMORY_AFFINITY MCM
setenv MP_TASK_AFFINITY MCM
setenv MP_SINGLE_THREAD yes
#
poe  ./DLPOLY.Z
```

Using LOADLEVELLER, the job is submitted by the UNIX command:

llsubmit gopoly

where *llsubmit* is a local command for submission to the IBM SP4 cluster. The number of required nodes and the job time are indicated in the above script.

gui

gui is a macro that starts up the DL_POLY_4 Java GUI. It invokes the following UNIX commands:

```
java -jar ../java/GUI.jar $1 &
```

In other words the macro invokes the Java Virtual Machine which executes the instructions in the Java archive file GUI.jar, which is stored in the *java* subdirectory of DL_POLY_4. (Note: Java 1.3.0 or a higher version is required to run the GUI.)

select

select is a macro enabling easy selection of one of the test cases. It invokes the UNIX commands:

```
cp -vpLH ../data/TEST$1/CONTROL   .
cp -vpLH ../data/TEST$1/CONFIG    .
cp -vpLH ../data/TEST$1/HISTORY   .
cp -vpLH ../data/TEST$1/FIELD     .
cp -vpLH ../data/TEST$1/MPOLES    .
cp -vpLH ../data/TEST$1/TAB*      .
cp -vpLH ../data/TEST$1/REFERENCE .
cp -vpLH ../data/TEST$1/Ce.dat    .
cp -vpLH ../data/TEST$1/g.dat     .
```

select requires one argument (an integer) to be specified:

select n

where *n* is test case number, which ranges from 1 to 18.

This macro sets up the required input files in the *execute* sub-directory to run the *n*-th test case. The last three copy commands may not be necessary in most cases.

store

The *store* macro provides a convenient way of moving data back from the *execute* sub-directory to the *data* sub-directory. It invokes the UNIX commands:

```
mkdir -pv          ../data/TEST$1
cp -vpLH CONTROL   ../data/TEST$1
cp -vpLH CONFIG    ../data/TEST$1
cp -vpLH FIELD     ../data/TEST$1
cp -vpLH MPOLES    ../data/TEST$1
cp -vpLH TAB*      ../data/TEST$1
cp -vpLH REFERENCE ../data/TEST$1
cp -vpLH HISTORY   ../data/TEST$1
cp -vpLH Ce.dat    ../data/TEST$1
cp -vpLH g.dat     ../data/TEST$1
mv -v     OUTPUT   ../data/TEST$1
mv -v     STATIS   ../data/TEST$1
mv -v     REV*     ../data/TEST$1
mv -v     CFGMIN   ../data/TEST$1
mv -v     HISTORF  ../data/TEST$1
mv -v     DEFECTS  ../data/TEST$1
mv -v     *DAT*    ../data/TEST$1
mv -v     *PMF     ../data/TEST$1
mv -v     *TAB     ../data/TEST$1
mv -v     MSDTMP   ../data/TEST$1
mv -v     DUMP_E   ../data/TEST$1
mv -v     LATS_*   ../data/TEST$1
mv -v     PEAK_*   ../data/TEST$1
chmod -R a-w       ../data/TEST$1
```

which first creates a new DL_POLY *data/TEST..* sub-directory and then moves the standard DL_POLY_4 output data files into it.

store requires one argument:

store n

where *n* is a unique string or number to label the output data in the *data/TESTn* sub-directory.

Note that *store* sets the file access to read-only. This is to prevent the *store* macro overwriting existing data without your knowledge.

Appendix D

DL_POLY_4 Error Messages & User Action

Introduction

In this appendix we document the error messages encoded in DL_POLY_4 and the recommended user action. The correct response is described as the **standard user response** in the appropriate sections below, to which the user should refer before acting on the error encountered.

The reader should also be aware that some of the error messages listed below may be either disabled in, or absent from, the public version of DL_POLY_4. Note that the wording of some of the messages may have changed over time, usually to provide more specific information. The most recent wording appears below.

The Standard User Response

DL_POLY_4 uses FORTRAN90 dynamic array allocation to set the array sizes at run time. This means that a single executable may be compiled to over all the likely uses of the code. It is not foolproof however. Sometimes an estimate of the required array sizes is difficult to obtain and the calculated value may be too small. For this reason DL_POLY_4 retains array dimension checks and will terminate when an array bound error occurs.

When a dimension error occurs, the **standard user response** is to edit the DL_POLY_4 subroutine SET_BOUNDS. Locate where the variable defining the array dimension is fixed and increase accordingly. To do this you should make use of the dimension information that DL_POLY_4 prints in the OUTPUT file prior to termination. If no information is supplied, simply doubling the size of the variable will usually do the trick. If the variable concerned is defined in one of the support subroutines SCAN_CONFIG, SCAN_FIELD, SCAN_CONTROL you will need to insert a new line in SET_BOUNDS to redefine it - after the relevant subroutine has been called! Finally the code must be recompiled, as in this case it will only be necessary to recompile SET_BOUNDS and not the whole code.

The DL_POLY_4 Error Messages

Message 1: error - word_2_real failure

The semantics in some of the INPUT files is wrong. DL_POLY_4 has tried to read a number but the has found a word in non-number format.

Action:

Look into your INPUT files and correct the semantics where appropriate and resubmit. DL_POLY_4 will have printed out in the OUTPUT file what the found non-uniform word is.

Message 2: error - too many atom types in FIELD (scan_field)

This error arises when DL_POLY_4 scans the FIELD file and discovers that there are too many different types of atoms in the system (i.e. the number of unique atom types exceeds the 1000).

Action:

Increase the number of allowed atom types (mmk) in SCAN_FIELD, recompile and resubmit.

Message 3: error - unknown directive found in CONTROL file

This error most likely arises when a directive is misspelt in the CONTROL file.

Action:

Locate the erroneous directive in the CONTROL file and correct error and resubmit.

Message 4: error - unknown directive found in FIELD file

This error most likely arises when a directive is misspelt or is encountered in an incorrect location in the FIELD file, which can happen if too few or too many data records are included.

Action:

Locate the erroneous directive in the FIELD file and correct error and resubmit.

Message 5: error - unknown energy unit requested

The DL_POLY_4 FIELD file permits a choice of units for input of energy parameters. These may be: electron-Volts (**eV**); k-calories per mol (**kcal**/mol); k-Joules per mol (**kJ**/mol); Kelvin per Boltzmann (**K**elvin/Boltzmann); or the DL_POLY_4 internal units, 10 Joules per mol (**internal**). There is no default value. Failure to specify any of these correctly, or reference to other energy units, will result in this error message. See documentation of the FIELD file.

Action:

Correct energy keyword on **units** directive in FIELD file and resubmit.

Message 6: error - energy unit not specified

A **units** directive is mandatory in the FIELD file. This error indicates that DL_POLY_4 has failed to find the required record.

Action:

Add **units** directive to FIELD file and resubmit.

Message 7: error - selected external field incompatible with selected ensemble (NVE only!!!)

Action:

Change the external field directive in FIELD file and or the type of ensemble in CONTROL and resubmit.

Message 8: error - ewald precision must be a POSITIVE real number

Ewald precision must be a positive non-zero real number. For example 10e-5 is accepted as a standard.

Action:

Put a correct number at the "ewald precision" directive in the CONTROL file and resubmit.

Message 9: error - ewald sum parameters must be well defined

Ewald sum parameters must be well defined.

Action:

Referer to the manual and references within for understanding the meaning of the parameters and how to chose them. Alternatively, try using the "ewald precision" CONTROL directive with a sensible precision value, of say 10_{-5}.

Message 10: error - too many molecular types specified

This should never happen! This indicates an erroneous FIELD file or corrupted DL_POLY_4 executable. Unlike DL_POLY_Classic, DL_POLY_4 does not have a set limit on the number of kinds of molecules it can handle in any simulation (this is not the same as the number of molecules).

Action:

Examine FIELD for erroneous directives, correct and resubmit.

Message 11: error - duplicate molecule directive in FIELD file

The number of different types of molecules in a simulation should only be specified once. If DL_POLY_4 encounters more than one **molecules** directive, it will terminate execution.

Action:

Locate the extra **molecule** directive in the FIELD file and remove and resubmit.

Message 12: error - unknown molecule directive in FIELD file

Once DL_POLY_4 encounters the **molecules** directive in the FIELD file, it assumes the following records will supply data describing the intra-molecular force field. It does not then expect to encounter directives not related to these data. This error message results if it encounters a unrelated directive. The most probable cause is incomplete specification of the data (e.g. when the **finish** directive has been omitted.)

Action:

Check the molecular data entries in the FIELD file, correct and resubmit.

Message 13: error - molecule species not specified

This error arises when DL_POLY_4 encounters non-bonded force data in the FIELD file, *before* the molecular species have been specified. Under these circumstances it cannot assign the data correctly, and therefore terminates.

Action:

Make sure the molecular data appears before the non-bonded forces data in the FIELD file and resubmit.

Message 14: error - too many unique atom types specified

This should never happen! This error most likely arises when the FIELD file or/and DL_POLY_4 executable are corrupted.

Action:

Recompile the program and/or recreate the FIELD file afresh. If no combination of these works, send the problem to us.

Message 15: error - duplicate vdw potential specified

In processing the FIELD file, DL_POLY_4 keeps a record of the specified short range pair potentials as they are read in. If it detects that a given pair potential has been specified before, no attempt at a resolution of the ambiguity is made and this error message results. See specification of FIELD file.

Action:

Locate the duplication in the FIELD file, rectify and resubmit.

Message 16: error - strange exit from FIELD file processing

This should never happen! It simply means that DL_POLY_4 has ceased processing the FIELD data, but has not reached the end of the file or encountered a **close** directive. Probable cause: corruption of the DL_POLY_4 executable or of the FIELD file. We would be interested to hear of other reasons!

Action:

See action notes on message 14 above.

Message 17: error - strange exit from CONTROL file processing

This should never happen! It simply means that DL_POLY_4 has ceased processing the CONTROL data, but has not reached the end of the file or encountered a **close** directive. Probable cause: corruption of the DL_POLY_4 executable or of the FIELD file. We would be interested to hear of other reasons!

Action:

Recompile the program and/or recreate the CONTROL file afresh. If no combination of these works, send the problem to us.

Message 18: error - duplicate three-body potential specified

DL_POLY_4 has encountered a repeat specification of a three-body potential in the FIELD file.

Action:

Locate the duplicate entry, remove and resubmit job.

Message 19: error - duplicate four-body potential specified

A 4-body potential has been duplicated in the FIELD file.

Action:

Locate the duplicated four-body potential, remove and resubmit job.

Message 20: error - too many molecule sites specified

This should never happen! This error most likely arises when the FIELD file or/and DL_POLY_4 executable are corrupted.

Action:

See action notes on message 14 above.

Message 21: error - molecule contains more atoms/sites than declared

The molecule contains more atom/site entries that it declares in the beginning.

Action:

Recreate or correct the erroneous entries in the FIELD file and try again.

Message 22: error - unsuitable radial increment in TABLE||TABBND||TABANG||TABDIH||TABIN file

This arises when the tabulated van der Waals potentials presented in the TABLE file have an increment that is greater than that used to define the other potentials in the simulation. Ideally, the increment should be $r_{\rm cut}/(\texttt{mxgrid}-4)$, where $r_{\rm cut}$ is the largest potential cutoff of all supplied ,for the short range potentials and the domain decomposition link cell size, and `mxgrid` is the parameter defining the length of the interpolation arrays. An increment less than this is permissible however. The same argument holds for the tabulated intra-molecular interactions that are possibly supplied via the TABBND, TABANG, TABDIH and TABINV files. All should have grids sized less than the generic `mxgrid`−4.

Action:

The tables must be recalculated with an appropriate increment.

Message 23: error - incompatible FIELD and TABLE file potentials

This error arises when the specification of the short range potentials is different in the FIELD and TABLE files. This usually means that the order of specification of the potentials is different. When DL_POLY_4 finds a change in the order of specification, it assumes that the user has forgotten to enter one.

Action:

Check the FIELD and TABLE files. Make sure that you correctly specify the pair potentials in the FIELD file, indicating which ones are to be presented in the TABLE file. Then check the TABLE file to make sure all the tabulated potentials are present in the order the FIELD file indicates.

Message 24: error - end of file encountered in TABLE||TABBND||TABANG||TABDIH||TABINV file

This means the TABLE||TABBND||TABANG||TABDIH||TABINV file is incomplete in some way: either by having too few potentials included, or the number of data points is incorrect.

Action:

Examine the TABLE file contents and regenerate it if it appears to be incomplete. If it look intact, check that the number of data points specified is what DL_POLY_4 is expecting.

Message 25: error - wrong atom type found in CONFIG file

On reading the input file CONFIG, DL_POLY_4 performs a check to ensure that the atoms specified in the configuration provided are compatible with the corresponding FIELD file. This message results if they are not *or the parallel reading wrongly assumed that CONFIG complies with the DL_POLY_3/4 style.*

Action:

The possibility exists that one or both of the CONFIG or FIELD files has incorrectly specified the atoms in the system. The user must locate the ambiguity, using the data printed in the OUTPUT file as a guide, and make the appropriate alteration. If the reason is in the parallel reading then produce a new CONFIG using a serial reading and continue working with it.

Message 26: error - neutral group option now redundant

DL_POLY_4 does not have the neutral group option.

Action:

Use the Ewald sum option. (It's better anyway.)

Message 27: error - unit's member indexed outside molecule's site range

An intra-molecular or intra-molecular alike interaction (topological) unit has member/site which is given a number outside the scope of the molecule it is part of.

Action:

Find the erroneous entry in FIELD, correct it and try running DL_POLY_4 again.

Message 28: error - wrongly indexed atom entries found in CONFIG file

DL_POLY_4 has detected that the atom indices in the CONFIG file do not form a contnual and/or non-repeating group of indices.

Action:

Make sure the CONFIG file is complies with the DL_POLY_4 standards. You may use the **no index** option in the CONTROL file to override the crystalographic sites' reading from the CONFIG file from reading by index to reading by order of the atom entries with consecutive incremental indexing. Using this option assumes that the FIELD topology description matches the crystalographic sites (atoms entries) in the CONFIG file by order (consecutively).

Message 30: error - too many chemical bonds specified

This should never happen! This error most likely arises when the FIELD file or/and DL_POLY_4 executable are corrupted.

Action:

See action notes on message 14 above.

Message 31: error - too many chemical bonds per domain

DL_POLY_4 limits the number of chemical bond units in the system to be simulated (actually, the number to be processed by each node) and checks for the violation of this. Termination will result if the condition is violated.

Action:

Use **densvar** option in CONTROL to increase `mxbond` (alternatively, increase it by hand in SET_BOUNDS and recompile) and resubmit.

Message 32: error - coincidence of particles in core-shell unit

DL_POLY_4 has found a fault in the definition of a core-shell unit in the FIELD file. The same particle has been assigned to the core and shell sites.

Action:

Correct the erroneous entry in FIELD and resubmit.

Message 33: error - coincidence of particles in constraint bond unit

DL_POLY_4 has found a fault in the definition of a constraint bond unit in the FIELD file. The same particle has been assigned to the both sites.

Action:

Correct the erroneous entry in FIELD and resubmit.

Message 34: error - length of constraint bond unit >= real space cutoff (rcut)

DL_POLY_4 has found a constraint bond unit length (FIELD) larger than the real space cutoff (`rcut`) (CONTROL).

Action:

Increase cutoff in CONTROL or decrease the constraint bondlength in FIELD and resubmit. For small system consider using DL_POLY_Classic.

Message 35: error - coincidence of particles in chemical bond unit

DL_POLY_4 has found a faulty chemical bond in FIELD (defined between the same particle).

Action:

Correct the erroneous entry in FIELD and resubmit.

Message 36: error - only one *bonds* directive per molecule is allowed

DL_POLY_4 has found more than one bonds entry per molecule in FIELD.

Action:

Correct the erroneous part in FIELD and resubmit.

Message 38: error - outgoing transfer buffer size exceeded in metal_ld_export

This should not usually happen!

Action:

Consider using `densvar` option in CONTROL for extremely non-equilibrium simulations. Alternatively, increase `mxbfxp` parameter in SET_BOUNDS recompile and resubmit. Send the problem to us if this is persistent.

Message 39: error - incoming data transfer size exceeds limit in metal_ld_export

See notes on message 38 above.

Action:

See action notes on message 38 above.

Message 40: error - too many bond constraints specified

This should never happen!

Action:

See action notes on message 14 above.

Message 41: error - too many bond constraints per domain

DL_POLY_4 limits the number of bond constraint units in the system to be simulated (actually, the number to be processed by each node) and checks for the violation of this. Termination will result if the condition is violated.

Action:

Use **densvar** option in CONTROL to increase `mxcons` (alternatively, increase it by hand in SET_BOUNDS and recompile) and resubmit.

Message 42: error - undefined direction passed to deport_atomic_data

This should never happen!

Action:

Send the problem to us.

Message 43: error - outgoing transfer buffer size exceeded in deport_atomic_data

This may happen in extremely non-equilibrium simulations or usually when the potentials in use do not hold the system stable.

Action:

Consider using `densvar` option in CONTROL for extremely non-equilibrium simulations. Alternatively, increase `mxbfdp` parameter in SET_BOUNDS recompile and resubmit.

Message 44: error - incoming data transfer size exceeds limit in deport_atomic_data

Action:

See action notes on message 43 above.

Message 45: error - too many atoms in CONFIG file or per domain

This can happen in circumstances when indeed the CONFIG file has more atoms listed than defined in FIELD, or when one of the domains (managed by an MPI process) has higher particle density than the system average and contains more particles than allowed by the default based on the system.

Action:

Check if CONFIG and FIELD numbers of particles match. Try executing on various number of processors. Try using the **densvar** option in CONTROL to increase `mxatms` (alternatively, increase it by hand in SET_BOUNDS and recompile) and resubmit. Send the problem to us if this is persistent.

Message 46: error - undefined direction passed to export_atomic_data

This should never happen!

Action:

Send the problem to us.

Message 47: error - undefined direction passed to metal_ld_export

This should never happen!

Action:

Send the problem to us.

Message 48: error - transfer buffer too small in *_table_read

Action:

Standard user response. Increase `mxgrid` parameter in SET_BOUNDS recompile and resubmit.

Message 49: error - frozen shell (core-shell) unit specified

The DL_POLY_4 option to freeze the location of an atom (i.e. hold it permanently in one position) is not permitted for the shells in core-shell units.

Action:

Remove the frozen atom option from the FIELD file. Consider using a non-polarisable atom instead.

Message 50: error - too many bond angles specified

This should never happen! This error most likely arises when the FIELD file or/and DL_POLY_4 executable are corrupted.

Action:

See action notes on message 14 above.

Message 51: error - too many bond angles per domain

DL_POLY_4 limits the number of valence angle units in the system to be simulated (actually, the number to be processed by each node) and checks for the violation of this. Termination will result if the condition is violated.

Action:

Use **densvar** option in CONTROL to increase `mxangl` (alternatively, increase it by hand in SET_BOUNDS and recompile) and resubmit.

Message 52: error - end of FIELD file encountered

This message results when DL_POLY_4 reaches the end of the FIELD file, without having read all the data it expects. Probable causes: missing data or incorrect specification of integers on the various directives.

Action:

Check FIELD file for missing or incorrect data, correct and resubmit.

Message 53: error - end of CONTROL file encountered

This message results when DL_POLY_4 reaches the end of the CONTROL file, without having read all the data it expects. Probable cause: missing **finish** directive.

Action:

Check CONTROL file, correct and resubmit.

Message 54: error - outgoing transfer buffer size exceeded in export_atomic_data

See notes on message 38 above.

Action:

See naction otes on message 38 above.

Message 55: error - end of CONFIG file encountered

This error arises when DL_POLY_4 attempts to read more data from the CONFIG file than is actually present. The probable cause is an incorrect or absent CONFIG file, but it may be due to the FIELD file being incompatible in some way with the CONFIG file.

Action:

Check contents of CONFIG file. If you are convinced it is correct, check the FIELD file for inconsistencies.

Message 56: error - incoming data transfer size exceeds limit in export_atomic_data

See notes on message 38 above.

Action:

See action notes on message 38 above.

Message 57: error - too many core-shell units specified

This should never happen!

Action:

See action notes on message 14 above.

Message 58: error - number of atoms in system not conserved

Either and an atom has been lost in transfer between nodes/domains or your FIELD is ill defined with respect to what is supplied in CONFIG/HISTORY.

Action:

If this error is issued at start before timestep zero in a simulation then it is either your FIELD file is ill defined or that your CONFIG file (or the first frame of your HISTRORY being replayed). Check out for mistyped number or identities of molecules, atoms, etc. in FIELD and for mangled/blank lines in CONFIG/HISTORY, or a blank line(s) at the end of CONFIG or missing FOF (End Of File) character in CONFIG. If this error is issued after timestep zero in a simulation that is not replaying HISTORY then it is big trouble and you should report that to the authors. If it is during replaying HISTORY then your HISTORY file has corrupted frames and you must correct it before trying again.

Message 59: error - too many core-shell units per domain

DL_POLY_4 limits the number of core-shell units in the system to be simulated (actually, the number to be processed by each node) and checks for the violation of this. Termination will result if the condition is violated.

Action:

Use **densvar** option in CONTROL to increase `mxshl` (alternatively, increase it by hand in SET_BOUNDS and recompile) and resubmit.

Message 60: error - too many dihedral angles specified

This should never happen!

Action:

See action notes on message 14 above.

Message 61: error - too many dihedral angles per domain

DL_POLY_4 limits the number of dihedral angle units in the system to be simulated (actually, the number to be processed by each node) and checks for the violation of this. Termination will result if the condition is violated.

Action:

Use **densvar** option in CONTROL to increase `mxdihd` (alternatively, increase it by hand in SET_BOUNDS and recompile) and resubmit.

Message 62: error - too many tethered atoms specified

This should never happen!

Action:

See action notes on message 14 above.

Message 63: error - too many tethered atoms per domain

DL_POLY_4 limits the number of tethered atoms in the system to be simulated (actually, the number to be processed by each node) and checks for the violation of this. Termination will result if the condition is violated.

Action:

Use **densvar** option in CONTROL to increase `mxteth` (alternatively, increase it by hand in SET_BOUNDS and recompile) and resubmit.

Message 64: error - incomplete core-shell unit found in build_book_intra

This should never happen!

Action:

Report problem to authors.

Message 65: error - too many excluded pairs specified

This should never happen! This error arises when DL_POLY_4 is identifying the atom pairs that cannot have a pair potential between them, by virtue of being chemically bonded for example (see subroutine BUILD_EXCL_INTRA). Some of the working arrays used in this operation may be exceeded, resulting in termination of the program.

Action:

Contact authors.

Message 66: error - coincidence of particles in bond angle unit

DL_POLY_4 has found a fault in the definition of a bond angle in the FIELD file.

Action:

Correct the erroneous entry in FIELD and resubmit.

Message 67: error - coincidence of particles in dihedral unit

DL_POLY_4 has found a fault in the definition of a dihedral unit in the FIELD file.

Action:

Correct the erroneous entry in FIELD and resubmit.

Message 68: error - coincidence of particles in inversion unit

DL_POLY_4 has found a fault in the definition of a inversion unit in the FIELD file.

Action:

Correct the erroneous entry in FIELD and resubmit.

Message 69: error - too many link cells required in three_body_forces

The number of link cells required for the build up of the Verlet neighbour list (as in link_cell_pairs) or the calculation of three- & four-body as well tersoff forces (as in three_body_forces, four_body_forces, tersoff_body_forces) in the given model exceeds the number allowed for by the DL_POLY_4 arrays. Probable cause: your system has expanded unacceptably much to DL_POLY_4. This may not be physically sensible!

Action:

Consider using `densvar` option in CONTROL for extremely non-equilibrium simulations.

Message 70: error - constraint_quench failure

When a simulation with bond constraints is started, DL_POLY_4 attempts to extract the kinetic energy of the constrained atom-atom bonds arising from the assignment of initial random velocities. If this procedure fails, the program will terminate. The likely cause is a badly generated initial configuration.

Action:

Some help may be gained from increasing the cycle limit, by using the directive **mxshak** in the CONTROL file. You may also consider reducing the tolerance of the SHAKE iteration using the directive **shake** in the CONTROL file. However it is probably better to take a good look at the starting conditions!

Message 71: error - too many metal potentials specified

This should never happen!

Action:

Report to authors.

Message 72: error - too many tersoff potentials specified

This should never happen!

Action:

Report to authors.

Message 73: error - too many inversion potentials specified

This should never happen!

Action:

Report to authors.

Message 74: error - unidentified atom in tersoff potential list

This shows that DL_POLY_4 has encountered and erroneous entry for Tersoff potentials in FIELD.

Action:

Correct FIELD and resubmit.

Message 76: error - duplicate tersoff potential specified

This shows that DL_POLY_4 has encountered and erroneous entry for Tersoff potentials in FIELD.

Action:

Correct FIELD and resubmit.

Message 77: error - too many inversion angles per domain

DL_POLY_4 limits the number of inversion units in the system to be simulated (actually, the number to be processed by each node) and checks for the violation of this. Termination will result if the condition is violated.

Action:

Use **densvar** option in CONTROL to increase `mxinv` (alternatively, increase it by hand in SET_BOUNDS and recompile) and resubmit.

Message 79: error - tersoff potential cutoff undefined

This shows that DL_POLY_4 has encountered and erroneous entry for Tersoff potentials in FIELD.

Action:

Correct FIELD and resubmit.

Message 80: error - too many pair potentials specified

This should never happen!

Action:

Report to authors.

Message 81: error - unidentified atom in pair potential list

This shows that DL_POLY_4 has encountered and erroneous entry for vdw or metal potentials in FIELD or cited TABle file.

Action:

Correct FIELD and/or cited TABle file.

Message 82: error - calculated pair potential index too large

This should never happen! In checking the vdw and metal potentials specified in the FIELD file DL_POLY_4 calculates a unique integer indices that henceforth identify every specific potential within the program. If this index becomes too large, termination of the program results.

Action:

Report to authors.

Message 83: error - too many three-body/angles potentials specified

This should never happen!

Action:

Report to authors.

Message 84: error - unidentified atom in three-body/angles potential list

This shows that DL_POLY_4 has encountered and erroneous entry at three-body or angles definitions in FIELD.

Action:

Correct FIELD and resubmit.

Message 85: error - required velocities not in CONFIG file

If the user attempts to start up a DL_POLY_4 simulation with any type of **restart** directive (see description of CONTROL file,) the program will expect the CONFIG file to contain atomic velocities as well as positions. Termination results if these are not present.

Action:

Either replace the CONFIG file with one containing the velocities, or if not available, remove the **restart ...** directive altogether and let DL_POLY_4 create the velocities for itself.

Message 86: error - calculated three-body potential index too large

This should never happen! DL_POLY_4 has a permitted maximum for the calculated index for any three-body potential in the system (i.e. as defined in the FIELD file). If there are m distinct types of atom in the system, the index can possibly range from 1 to $(m^2 * (m-1))/2$. If the internally calculated index exceeds this number, this error reports results.

Action:

Report to authors.

Message 88: error - legend array exceeded in build_book_intra

The second dimension of a legend array has been exceeded.

Action:

If you have an intra-molecular (like) interaction present in abundance in your model that you suspect is driving this out of bound error increase its legend bound value, `mxf`interaction, at the end of SCAN_FIELD, recompile and resubmit. If the error persists contact authors.

Message 89: error - too many four-body/dihedrals/inversions potentials specified

This should never happen!

Action:

Report to authors.

Message 90: error - specified tersoff potentials have different types'

This is not allowed! Only one general type of tersoff potential is allowed in FIELD as there are no mixing rules between different tersoff potentials!

Action:

Correct your model representation in FIELD and try again.

Message 91: error - unidentified atom in four-body/dihedrals/inversions potential list

The specification of a four-body or dihedrals or inversions potential in the FIELD file has referenced an atom type that is unknown.

Action:

Locate the errant atom type in the four-body/dihedrals/inversions potential definition in the FIELD file and correct. Make sure this atom type is specified by an `atoms` directive earlier in the file.

Message 92: error - specified metal potentials have different types

The specified metal interactions in the FIELD file are referencing more than one generic type of metal potentials. Only one such type is allowed in the system.

Action:

Locate the errant metal type in the metal potential definition in the FIELD file and correct. Make sure only one metal type is specified for all relevan atom interactions in the file.

Message 93: error - PMFs mixing with rigid bodies not allowed

Action:

Correct FIELD and resubmit.

Message 95: error - error - rcut or (rcut+rpad) > minimum of all half-cell widths

In order for the minimum image convention to work correctly within DL_POLY_4, it is necessary to ensure that the major cutoff, plus its possible padding distance, applied to the pair interactions does not exceed half the perpendicular width of the simulation cell. (The perpendicular width is the shortest distance between opposing cell faces.) Termination results if this is detected. In NVE and NVT simulations this can only happen at the start of a simulation, but in NPT and N$\underline{\underline{\sigma}}$T, it may occur at any time.

Action:

Supply a cutoff that is less than half the cell width. If running constant pressure calculations, use a cutoff that will accommodate the fluctuations in the simulation cell. Study the fluctuations in the OUTPUT file to help you with this.

Message 96: error - incorrect atom totals assignments in metal_ld_set_halo

This should never happen!

Action:

Big trouble. Report to authors.

Message 97: error - constraints mixing with rigid bodies not allowed

Action:

Correct FIELD and resubmit.

Message 99: error - cannot have shells as part of a constraint, rigid body or tether

Action:

Correct FIELD and resubmit.

Message 100: error - core-shell unit separation > rcut (the system cutoff)

This could only happen if FIELD and CONFIG do not match each other or CONFIG is damaged.

Action:

Regenerate CONFIG (and FIELD) and resubmit.

Message 101: error - calculated four-body potential index too large

This should never happen! DL_POLY_4 has a permitted maximum for the calculated index for any four-body potential in the system (i.e. as defined in the FIELD file). If there are m distinct types of atom in the system, the index can possibly range from 1 to $(m^2 * (m+1) * (m+2))/6$. If the internally calculated index exceeds this number, this error report results.

Action:

Report to authors.

Message 102: error - rcut < 2*rcter (maximum cutoff for tersoff potentials)

The nature of the Tersoff interaction requires they have at least twice shorter cutoff than the standard pair interctions (or the major system cutoff).

Action:

Decrease Tersoff cutoffs in FIELD or increase cutoff in CONTROL and resubmit.

Message 103: error - parameter mxlshp exceeded in pass_shared_units

Various algorithms (constraint and core-shell ones) require that information about 'shared' atoms be passed between nodes. If there are too many such atoms, the arrays holding the information will be exceeded and DL_POLY_4 will terminate execution.

Action:

Use **densvar** option in CONTROL to increase `mxlshp` (alternatively, increase it by hand in SET_BOUNDS and recompile) and resubmit.

Message 104: error - arrays listme and lstout exceeded in pass_shared_units

This should not happen! Dimensions of indicated arrays have been exceeded.

Action:

Consider using `densvar` option in CONTROL for extremely non-equilibrium simulations.

Message 105: error - shake algorithm (constraints_shake) failed to converge

The SHAKE algorithm for bond constraints is iterative. If the maximum number of permitted iterations is exceeded, the program terminates. Possible causes include: a bad starting configuration; too large a time step used; incorrect force field specification; too high a temperature; inconsistent constraints (over-constraint) etc..

Action:

You may try to increase the limit of iteration cycles in the constraint subroutines by using the directive **mxshak** and/or decrease the constraint precision by using the directive **shake** in CONTROL. But the trouble may be much more likely to be cured by careful consideration of the physical system being simulated. For example, is the system stressed in some way? Too far from equilibrium?

Message 106: error - neighbour list array too small in link_cell_pairs

Construction of the Verlet neighbour list in subroutine LINK_CELL_PAIRS non-bonded (pair) force has exceeded the neighbour list array dimensions.

Action:

Consider using `densvar` option in CONTROL for extremely non-equilibrium simulations or increase by hand `mxlist` in SET_BOUNDS.

Message 107: error - too many pairs for rdf look up specified

This should never happen! A possible reason is corruption in FIELD or/and DL_POLY_4 executable.

Action:

See action notes on message 14 above.

Message 108: error - unidentified atom in rdf look up list

During reading of RDF look up pairs in FIELD DL_POLY_4 has found an unlisted previously atom type.

Action:

Correct FIELD by either defining the new atom type or changing it to an already defined one in the erroneous line. Resubmit.

Message 109: error - calculated pair rdf index too large

This should never happen! In checking the RDF pairs specified in the FIELD file DL_POLY_4 calculates a unique integer index that henceforth identify every RDF pair within the program. If this index becomes too large, termination of the program results.

Action:

Report to authors.

Message 108: error - duplicate rdf look up pair specified

During reading of RDF look up pairs in FIELD DL_POLY_4 has found a duplicate entry in the list.

Action:

Delete the duplicate line and resubmit.

Message 111: error - bond constraint unit separation > rcut (the system cutoff)

This should never happen! DL_POLY_4 has not been able to find an atom in a processor domain or its bordering neighbours.

Action:

Probable cause: link cells too small. Use larger potential cutoff. Contact DL_POLY_4 authors.

Message 112: error - only one *constraints* directive per molecule is allowed

DL_POLY_4 has found more than one constraints entry per molecule in FIELD.

Action:

Correct the erroneous part in FIELD and resubmit.

Message 113: error - intra-molecular bookkeeping arrays exceeded in deport_atomic_data

One or more bookkeeping arrays for site-related interactions have been exceeded.

Action:

Consider using `densvar` option in CONTROL for extremely non-equilibrium simulations. Alternatively, you will need to print extra diagnostic data from the DEPORT_ATOMIC_DATA subroutine to find which boded-like contribution has exceeded its assumed limit and then correct for it in SET_BOUNDS, recompile and resubmit.

Message 114: error - legend array exceeded in deport_atomic_data

The array `legend` has been exceeded.

Action:

Try increasing parameter `mxfix` in SET_BOUNDS, recompile and resubmit. Contact DL_POLY_4 authors if the problem persists.

Message 115: error - transfer buffer exceeded in update_shared_units

The transfer buffer has been exceeded.

Action:

Consider increasing parameter `mxbfsh` in SET_BOUNDS, recompile and resubmit. Contact DL_POLY_4 authors if the problem persists.

Message 116: error - incorrect atom transfer in update_shared_units

An atom has become misplaced during transfer between nodes.

Action:

This happens when the simulation is very numerically unstable. Consider carefully the physical grounds of your simulation, i.e. are you using the adiabatic shell model for accounting polarisation with too big a timestep or too large control distances for the variable timestep, is the ensemble type NPT or N$\underline{\underline{\sigma}}$T and the system target temperature too close to the melting temperature?

Message 118: error - construction error in pass_shared_units

This should not happen.

Action:

Report to authors.

Message 120: error - invalid determinant in matrix inversion

DL_POLY_4 occasionally needs to calculate matrix inverses (usually the inverse of the matrix of cell vectors, which is of size 3 × 3). For safety's sake a check on the determinant is made, to prevent inadvertent use of a singular matrix.

Action:

Locate the incorrect matrix and fix it - e.g. are cell vectors correct?

Message 122: error - FIELD file not found

DL_POLY_4 failed to find a FIELD file in your directory.

Action:

Supply a valid FIELD file before you start a simulation

Message 124: error - CONFIG file not found

DL_POLY_4 failed to find a CONFIG file in your directory.

Action:

Supply a valid CONFIG file before you start a simulation

Message 126: error - CONTROL file not found

DL_POLY_4 failed to find a CONTROL file in your directory.

Action:

Supply a valid CONTROL file before you start a simulation

Message 128: error - chemical bond unit separation > rcut (the system cutoff)

This could only happen if FIELD and CONFIG do not match each other or if the instantaneous configuration is ill defined because of generation of large forces on bonded particles. This may be due to having a badly defined force-field and/or starting form a configuration which is too much away from equilibrium.

Action:

Regenerate CONFIG (and FIELD) and resubmit. Try topology verification by using `nfold 1 1 1` in CONTROL. Try using options as `scale`, `cap`, `zero` and `optimise`. Try using smaller SHAKE tolerance if constraints are present in the system. You may as well try using the `variable timestep` option.

Message 130: error - bond angle unit diameter > rcut (the system cutoff)

See action notes on message 128 above.

Action:

See action notes on message 128 above.

Message 132: error - dihedral angle unit diameter > rcut (the system cutoff)

See notes on message 128 above.

Action:

See action notes on message 128 above.

Message 134: error - inversion angle unit diameter > rcut (the system cutoff)

See notes on message 128 above.

Action:

See action notes on message 128 above.

Message 138: error - incorrect atom totals assignments in refresh_halo_positions

This should never happen although, sometimes, it could due to ill defined force field and/or and/or starting form a configuration which is too much away from equilibrium.

Action:

Try using the `variable timestep` option and/or running in serial to determine if particles gain too much speed and leave domains.

Message 141: error - duplicate metal potential specified

During reading of metal potentials (pairs of atom types) in FIELD DL_POLY_4 has found a duplicate pair of atoms in the list.

Action:

Delete one of the duplicate entries and resubmit.

Message 150: error - unknown van der waals potential selected

DL_POLY_4 checks when constructing the interpolation tables for the short ranged potentials that the potential function requested is one which is of a form known to the program. If the requested potential form is unknown, termination of the program results. The most probable cause of this is the incorrect choice of the potential keyword in the FIELD file.

Action:

Read the DL_POLY_4 documentation and find the potential keyword for the potential desired.

Message 151: error - unknown EAM keyword in TABEAM

DL_POLY_4 checks when constructing the interpolation tables for the EAM metal potentials that the potential function requested is one which is of a form known to the program. If the requested potential form is unknown, termination of the program results. The most probable cause of this is the incorrect choice of the potential keyword in the FIELD file.

Action:

Read the DL_POLY_4 documentation and find the potential keyword for the potential desired.

Message 152: error - undefined direction passed to dpd_v_export

This should never happen!

Action:

Report to authors.

Message 154: error - outgoing transfer buffer size exceeded in dpd_v_export

See notes on message 38 above.

Action:

See action notes on message 38 above.

Message 156: error - incoming data transfer size exceeds limit in dpd_v_export

See notes on message 38 above.

Action:

See action notes on message 38 above.

Message 158: error - incorrect atom totals assignments in dpd_v_set_halo

This should never happen!

Action:

Big trouble. Report to authors.

Message 160: error - undefined direction passed to statistics_connect_spread

This should never happen!

Message 163: error - outgoing transfer buffer size exceeded in statistics_connect_spread

The transfer buffer has been exceeded.

Action:

Consider using `densvar` option in CONTROL for extremely non-equilibrium simulations. Alternatively, increase `mxbfss` parameters in SET_BOUNDS recompile and resubmit.

Message 164: error - incoming data transfer size exceeds limit in statistics_connect_spread

See notes on message 163 above.

Action:

See action notes on message 163 above.

Message 170: error - too many variables for statistics array

This error means the statistics arrays appearing in subroutine STATISTICS_COLLECT are too small. This should never happen!

Action:

Contact DL_POLY_4 authors.

Message 172: error - duplicate intra-molecular entries specified in TABBND||TABANG||TABDIH||TABINV

A duplicate entry has been encountered in the intra-molecular table file.

Action:

Contact DL_POLY_4 authors.

Message 200: error - rdf/z-density buffer array too small in system_revive

This error indicates that a global summation buffer array in subroutine SYSTEM_REVIVE is too small, i.e `mxbuff` < `mxgrdf`. This should never happen!

Action:

Contact DL_POLY_4 authors.

Message 210: error - only one *angles* directive per molecule is allowed

DL_POLY_4 has found more than one angles entry per molecule in FIELD.

Action:

Correct the erroneous part in FIELD and resubmit.

Message 220: error - only one *dihedrals* directive per molecule is allowed

DL_POLY_4 has found more than one dihedrals entry per molecule in FIELD.

Action:

Correct the erroneous part in FIELD and resubmit.

Message 230: error - only one *inversions* directive per molecule is allowed

DL_POLY_4 has found more than one inversions entry per molecule in FIELD.

Action:

Correct the erroneous part in FIELD and resubmit.

Message 240: error - only one *tethers* directive per molecule is allowed

DL_POLY_4 has found more than one tethers entry per molecule in FIELD.

Action:

Correct the erroneous part in FIELD and resubmit.

Message 300: error - incorrect boundary condition for link-cell algorithms

The use of link cells in DL_POLY_4 implies the use of appropriate boundary conditions. This error results if the user specifies octahedral or dodecahedral boundary conditions, which are only available in DL_POLY_Classic.

Action:

Correct your boundary condition or consider using DL_POLY_Classic.

Message 305: error - too few link cells per dimension for many-body and tersoff forces subroutines.

The link cells algorithms for many-body and tersoff forces in DL_POLY_4 cannot work with less than 3 (secondary) link cells per dimension. This depends on the cell size widths (as supplied in CONFIG) and the largest system cut-off (as specified in CONTROL although it may be drawn or overridden by cutoffs specified as part of some potentials' parameter sets in FIELD).

Action:

Decrease many-body and tersoff potentials cutoffs or/and number of nodes or/and increase system size.

Message 307: error - link cell algorithm violation

DL_POLY_4 does not like what you are asking it to do. Probable cause: the cutoff is too large to use link cells in this case.

Action:

Rethink the simulation model; reduce the cutoff or/and number of nodes or/and increase system size.

Message 308: error - link cell algorithm in contention with SPME sum precision

DL_POLY_4 does not like what you are asking it to do. Probable cause: you ask for SPME precision that is not achievable by the current settings of the link cell algorithm.

Action:

Rethink the simulation model; reduce number of nodes or/and SPME sum precision or/and increase cutoff.

Message 340: error - invalid integration option requested

DL_POLY_4 has detected an incompatibility in the simulation instructions, namely that the requested integration algorithm is not compatible with the physical model. It *may* be possible to override this error trap, but it is up to the user to establish if this is sensible.

Action:

This is a non-recoverable error, unless the user chooses to override the restriction.

Message 350: error - too few degrees of freedom

This error can arise if a small system is being simulated and the number of constraints applied is too large.

Action:

Simulate a larger system or reduce the number of constraints.

Message 360: error - degrees of freedom distribution problem

This should never happen for a dynamically sensical system. This error arises if a model system contains one or more free, zero mass particles. Zero mass (mass-less) particles/sites are only allowed for shells in core-shell units and as part of rigid bodies (mass-less but charged RB sites).

Action:

Inspect your FIELD to find and correct the erroneous entries, and try again.

Message 380: error - simulation temperature not specified or < 1 K

DL_POLY_4 has failed to find a **temp** directive in the CONTROL file.

Action:

Place a **temp** directive in the CONTROL file, with the required temperature specified.

Message 381: error - simulation timestep not specified

DL_POLY_4 has failed to find a **timestep** directive in the CONTROL file.

Action:

Place a **timestep** directive in the CONTROL file, with the required timestep specified.

Message 382: error - simulation cutoff not specified

DL_POLY_4 has failed to find a **cutoff** directive in the CONTROL file.

Action:

Place a **cutoff** directive in the CONTROL file, with the required forces cutoff specified.

Message 387: error - system pressure not specified

The target system pressure has not been specified in the CONTROL file. Applies to NPT simulations only.

Action:

Insert a **press** directive in the CONTROL file specifying the required system pressure.

Message 390: error - npt/nst ensemble requested in non-periodic system

A non-periodic system has no defined volume, hence the NPT algorithm cannot be applied.

Action:

Either simulate the system with a periodic boundary, or use another ensemble.

Message 402: error - van der waals not specified

The user has not set any cutoff in CONTROL, (`rvdw`) - the van der Waals potentials cutoff is needed in order for DL_POLY_4 to proceed.

Action:

Supply a cutoff value for the van der Waals terms in the CONTROL file using the directive `rvdw`, and resubmit job.

Message 410: error - cell not consistent with image convention

The simulation cell vectors appearing in the CONFIG file are not consistent with the specified image convention.

Action:

Locate the variable `imcon` in the CONFIG file and correct to suit the cell vectors.

Message 414: error - conflicting ensemble options in CONTROL file

DL_POLY_4 has found more than one **ensemble** directive in the CONTROL file.

Action:

Locate extra **ensemble** directives in CONTROL file and remove.

Message 416: error - conflicting force options in CONTROL file

DL_POLY_4 has found incompatible directives in the CONTROL file specifying the electrostatic interactions options.

Action:

Locate the conflicting directives in the CONTROL file and correct.

Message 430: error - integration routine not available

A request for a non-existent ensemble has been made or a request with conflicting options that DL_POLY_4 cannot deal with.

Action:

Examine the CONTROL and FIELD files and remove inappropriate specifications.

Message 432: error - undefined tersoff potential

This shows that DL_POLY_4 has encountered an unfamiliar entry for Tersoff potentials in FIELD.

Action:

Correct FIELD and resubmit.

Message 433: error - rcut must be specified for the Ewald sum precision

When specifying the desired precision for the Ewald sum in the CONTROL file, it is also necessary to specify the real space cutoff `rcut`.

Action:

Place the **cut** directive *before* the **ewald precision** directive in the CONTROL file and rerun.

Message 436: error - unrecognised ensemble

An unknown ensemble option has been specified in the CONTROL file.

Action:

Locate **ensemble** directive in the CONTROL file and amend appropriately.

Message 440: error - undefined angular potential

A form of angular potential has been requested which DL_POLY_4 does not recognise.

Action:

Locate the offending potential in the FIELD file and remove. Replace with one acceptable to DL_POLY_4 if this is possible. Alternatively, you may consider defining the required potential in the code yourself. Amendments to subroutines READ_FIELD and ANGLES_FORCES will be required.

Message 442: error - undefined three-body potential

A form of three-body potential has been requested which DL_POLY_4 does not recognise.

Action:

Locate the offending potential in the FIELD file and remove. Replace with one acceptable to DL_POLY_4 if this is reasonable. Alternatively, you may consider defining the required potential in the code yourself. Amendments to subroutines READ_FIELD and THREE_BODY_FORCES will be required.

Message 443: error - undefined four-body potential

DL_POLY_4 has been requested to process a four-body potential it does not recognise.

Action:

Check the FIELD file and make sure the keyword is correctly defined. Make sure that subroutine THREE_BODY_FORCES contains the code necessary to deal with the requested potential. Add the code required if necessary, by amending subroutines READ_FIELD and THREE_BODY_FORCES.

Message 444: error - undefined bond potential

DL_POLY_4 has been requested to process a bond potential it does not recognise.

Action:

Check the FIELD file and make sure the keyword is correctly defined. Make sure that subroutine BONDS_FORCES contains the code necessary to deal with the requested potential. Add the code required if necessary, by amending subroutines READ_FIELD and BONDS_FORCES.

Message 445: error - r_14 > rcut in dihedrals_forces

The 1-4 coulombic scaling for a dihedral angle bonding cannot be performed since the 1-4 distance has exceeded the system short range interaction cutoff, `rcut`, in subroutine DIHEDRAL_FORCES.

Action:

To prevent this error occurring again increase `rcut`.

Message 446: error - undefined electrostatic key in dihedral_forces

The subroutine DIHEDRAL_FORCES has been requested to process a form of electrostatic potential it does not recognise.

Action:

The error arises because the integer key `keyfrc` has an inappropriate value (which should not happen in the standard version of DL_POLY_4). Check that the FIELD file correctly specifies the potential. Make sure the version of DIHEDRAL_FORCES does contain the potential you are specifying. Report the error to the authors if these checks are correct.

Action:

To prevent this error occurring again increase `rvdw`.

Message 447: error - only one *shells* directive per molecule is allowed

DL_POLY_4 has found more than one shells entry per molecule in FIELD.

Action:

Correct the erroneous part in FIELD and resubmit.

Message 448: error - undefined dihedral potential

A form of dihedral potential has been requested which DL_POLY_4 does not recognise.

Action:

Locate the offending potential in the FIELD file and remove. Replace with one acceptable to DL_POLY_4 if this is reasonable. Alternatively, you may consider defining the required potential in the code yourself. Amendments to subroutines READ_FIELD and DIHEDRAL_FORCES (and its variants) will be required.

Message 449: error - undefined inversion potential

A form of inversion potential has been encountered which DL_POLY_4 does not recognise.

Action:

Locate the offending potential in the FIELD file and remove. Replace with one acceptable to DL_POLY_4 if this is reasonable. Alternatively, you may consider defining the required potential in the code yourself. Amendments to subroutines READ_FIELD and INVERSIONS_FORCES will be required.

Message 450: error - undefined tethering potential

A form of tethering potential has been requested which DL_POLY_4 does not recognise.

Action:

Locate the offending potential in the FIELD file and remove. Replace with one acceptable to DL_POLY_4 if this is reasonable. Alternatively, you may consider defining the required potential in the code yourself. Amendments to subroutines READ_FIELD and TETHERS_FORCES will be required.

Message 451: error - three-body potential cutoff undefined

The cutoff radius for a three-body potential has not been defined in the FIELD file.

Action:

Locate the offending three-body force potential in the FIELD file and add the required cutoff. Resubmit the job.

Message 452: error - undefined vdw potential

A form of vdw potential has been requested which DL_POLY_4 does not recognise.

Action:

Locate the offending potential in the FIELD file and remove. Replace with one acceptable to DL_POLY_4 if this is reasonable. Alternatively, you may consider defining the required potential in the code yourself. Amendments to subroutines READ_FIELD, VDW_GENERATE* and DIHEDRALS_14_VDW will be required.

Message 453: error - four-body potential cutoff undefined

The cutoff radius for a four-body potential has not been defined in the FIELD file.

Action:

Locate the offending four-body force potential in the FIELD file and add the required cutoff. Resubmit the job.

Message 454: error - unknown external field

A form of external field potential has been requested which DL_POLY_4 does not recognise.

Action:

Locate the offending potential in the FIELD file and remove. Replace with one acceptable to DL_POLY_4 if this is reasonable. Alternatively, you may consider defining the required potential in the code yourself. Amendments to subroutines READ_FIELD and EXTERNAL_FIELD_APPLY will be required.

Message 456: error - external field xpis-ton is applied to a layer with at least one frozen particle

For a layer to emulate a piston no particle constituting it must be frozen.

Action:

Locate the offending site(s) in the FIELD file and unfreeze the particles.

Message 461: error - undefined metal potential

A form of metal potential has been requested which DL_POLY_4 does not recognise.

Action:

Locate erroneous entry in the FIELD file and correct the potental interaction to one of the allowed ones for metals in DL_POLY_4.

Message 462: error - thermostat friction constant must be > 0

A zero or negative value for the thermostat friction constant has been encountered in the CONTROL file.

Action:

Locate the **ensemble** directive in the CONTROL file and assign a positive value to the time constant.

Message 463: error - barostat friction constant must be > 0

A zero or negative value for the barostat friction constant has been encountered in the CONTROL file.

Action:

Locate the **ensemble** directive in the CONTROL file and assign a positive value to the time constant.

Message 464: error - thermostat relaxation time constant must be > 0

A zero or negative value for the thermostat relaxation time constant has been encountered in the CONTROL file.

Action:

Locate the **ensemble** directive in the CONTROL file and assign a positive value to the time constant.

Message 466: error - barostat relaxation time constant must be > 0

A zero or negative value for the barostat relaxation time constant has been encountered in the CONTROL file.

Action:

Locate the **ensemble** directive in the CONTROL file and assign a positive value to the time constant.

Message 467: error - rho must not be zero in valid buckingham potential

User specified vdw type buckingham potential has a non-zero force and zero rho constants. Only both zero or both non-zero are allowed.

Action:

Inspect the FIELD file and change the values in question appropriately.

Message 468: error - r0 too large for snm potential with current cutoff

The specified location (r0) of the potential minimum for a shifted n-m potential exceeds the specified potential cutoff. A potential with the desired minimum cannot be created.

Action:

To obtain a potential with the desired minimum it is necessary to increase the van der Waals cutoff. Locate the `rvdw` directive in the CONTROL file and reset to a magnitude greater than r0. Alternatively adjust the value of r0 in the FIELD file. Check that the FIELD file is correctly formatted.

Message 470: error - n < m in definition of n-m potential

The specification of a n-m potential in the FIELD file implies that the exponent m is larger than exponent n. (Not all versions of DL_POLY_4 are affected by this.)

Action:

Locate the n-m potential in the FIELD file and reverse the order of the exponents. Resubmit the job.

Message 471: error - rcut < 2*rctbp (maximum cutoff for three-body potentials)

The cutoff for the pair interactions is smaller than twice that for the three-body interactions. This is a bookkeeping requirement for DL_POLY_4.

Action:

Either use a smaller three-body cutoff, or a larger pair potential cutoff.

Message 472: error - rcut < 2*rcfbp (maximum cutoff for four-body potentials)

The cutoff for the pair interactions is smaller than twice that for the four-body interactions. This is a bookkeeping requirement for DL_POLY_4.

Action:

Either use a smaller four-body cutoff, or a larger pair potential cutoff.

Message 474: error - conjugate gradient mimimiser cycle limit exceeded

The conjugate gradient minimiser exceeded the iteration limit (100 for the relaxed shell model, 1000 for the configuration minimiser).

Action:

Decrease the respective convergence criterion. Alternatively, you may try to increase the limit by hand in CORE_SHELL_RELAX or in MINIMISE_RELAX respectively and recompile. However, it is unlikely that such measures will cure the problem as it is more likely to lay in the physical description of the system being simulated. For example, are the core-shell spring constants well defined? Is the system being too far from equilibrium?

Message 476: error - shells MUST all HAVE either zero or non-zero masses

The polarisation of ions is accounted via a core-shell model as the shell dynamics is either relaxed - shells have no mass, or adiabatic - all shells have non-zero mass.

Action:

Choose which model you would like to use in the simulated system and adapt the shell masses in FIELD to comply with your choice.

Message 478: error - shake algorithms (constraints & pmf) failed to converge

Your system has both bond and PMF constraints. SHAKE (RATTLE_VV1) is done by combined application of both bond and PMF constraints SHAKE (RATTLE_VV1) in an iterative manner until the PMF constraint virial converges to a constant. No such convergence is achieved.

Action:

See action notes on message 515 below.

Message 480: error - PMF constraint length > minimum of all half-cell widths

The specified PMF length has exceeded the minimum of all half-cell widths.

Action:

Specify shorter PMF length or increase MD cell dimensions.

Message 484: error - only one potential of mean force permitted

Only one potential of mean force is permitted in FIELD.

Action:

Correct the erroneous entries in FIELD.

Message 486: error - only one of the PMF units is permitted to have frozen atoms

Only one of the PMF units is permitted to have frozen atoms.

Action:

Correct the erroneous entries in FIELD.

Message 488: error - too many PMF constraints per domain

This should not happen.

Action:

Is the use of PMF constraints in your system physically sound?

Message 490: error - local PMF constraint not found locally

This should not happen.

Action:

Is your system physically sound, is your system equilibrated?

Message 492: error - a diameter of a PMF unit > minimum of all half cell widths

The diameter of a PMF unit has exceeded the minimum of all half-cell widths.

Action:

Consider the physical concept you are trying to imply in the simulation. Increase MD cell dimensions.

Message 494: error - overconstrained PMF units

PMF units are oveconstrained.

Action:

DL_POLY_4 algorithms cannot handle overconstrained PMF units. Decrease the number of constraints on the PMFs.

Message 497: error - pmf_quench failure

Action:

See notes on message 515 below.

Message 498: error - shake algorithm (pmf_shake) failed to converge

Action:

See action notes on message 515 below.

Message 499: error - rattle algorithm (pmf_rattle) failed to converge

See notes on message 515 below.

Action:

See action notes on message 515 below.

Message 500: error - PMF unit of zero length is not permitted

PMF unit of zero length is found in FIELD. PMF units are either a single atom or a group of atoms usually forming a chemical molecule.

Action:

Correct the erroneous entries in FIELD.

Message 501: error - coincidence of particles in PMF unit

A PMF unit must be constituted of non-repeating particles!

Action:

Correct the erroneous entries in FIELD.

Message 502: error - PMF unit member found to be present more than once

A PMF unit is a group of unique (distingushed) atoms/sites. No repetition of a site is allowed in a PMF unit.

Action:

Correct the erroneous entries in FIELD.

Message 504: error - cutoff too large for TABLE||TABBND file

The requested cutoff exceeds the information in the TABLE file or the TABBND cutoff is larger than half the system cutoff `rcut`.

Action:

In the case when this is received while reading TABLE, reduce the value of the vdw cutoff (`rvdw`) in the CONTROL file or reconstruct the TABLE file. In the case when this is received while reading TABBND then specify a larger `rcut` in CONTROL.

Message 505: error - EAM metal densities or pair crossfunctions out of range

The resulting densities or pair crossfunctions are not defined in the TABEAM file.

Action:

Recreate a TABEAM file with wider interval of defined densities and pair cross functions.

Message 506: error - EAM or MBPC metal densities out of range

The resulting densities are not defined in the TABEAM file if EAM is used or ill defined due to atoms nearly overlapping when MBPC metal potential is in use..

Action:

Recreate a TABEAM file with wider range of densities.

Message 507: error - metal density embedding out of range

In the case of EAM type of metal interactions this indicates that the electron density of a particle in the system has exceeded the limits for which the embedding function for this particle's type is defined (as supplied in TABEAM. In the case of Finnis-Sinclair type of metal interactions, this indicates that the density has become negative.

Action:

Reconsider the physical sanity and validity of the metal interactions in your system and this type of simulation. You MUST change the interactions' parameters and/or the way the physical base of your investigation is handled in MD terms.

Message 508: error - EAM metal interaction entry in TABEAM unspecified in FIELD

The specified EAM metal interaction entry found in TABEAM is not specified in FIELD.

Action:

For N metal atom types there are $(5N + N^2)/2$ EAM functions in the TABEAM file. One density (N) and one embedding (N) function for each atom type and $(N + N^2)/2$ cross-interaction functions. Fix the table entries and resubmit.

Message 509: error - duplicate entry for a pair interaction detected in TABEAM

A duplicate cross-interaction function entry is detected in the TABEAM file.

Action:

Remove all duplicate entries in the TABEAM file and resubmit.

Message 510: error - duplicate entry for a density function detected in TABEAM

A duplicate density function entry is detected in the TABEAM file.

Action:

Remove all duplicate entries in the TABEAM file and resubmit.

Message 511: error - duplicate entry for an embedding function detected in TABEAM

A duplicate embedding function entry is detected in the TABEAM file.

Action:

Remove all duplicate entries in the TABEAM file and resubmit.

Message 512: error - non-definable vdw/dpd interactions detected in FIELD

A VDW corss-interaction was uncpecified and recovering it by using a mixing rule proved impossible due to type difference of the single species potentials.

Action:

Rethink your FIELD file interactions before restarting the job with a new compatible FIELD and possibly CONTROL file.

Message 513: error - particle assigned to non-existent domain in read_config

This can only happen if particle coordinates do not match the cell parameters in CONFIG. Probably, due to negligence or numerical inaccuracy inaccuracy in generation of big supercell from a small one.

Action:

Make sure lattice parameters and particle coordinates marry each other. Increase accuracy when generating a supercell.

Message 514: error - allowed image conventions are: 0, 1, 2, 3 and 6

DL_POLY_4 has found unsupported boundary condition specified in CONFIG.

Action:

Correct your boundary condition or consider using DL_POLY_Classic.

Message 515: error - rattle algorithm (constraints_rattle) failed to converge

The RATTLE algorithm for bond constraints is iterative. If the maximum number of permitted iterations is exceeded, the program terminates. Possible causes include: incorrect force field specification; too high a temperature; inconsistent constraints (over-constraint) etc..

Action:

You may try to increase the limit of iteration cycles in the constraint subroutines by using the directive **mxshak** and/or decrease the constraint precision by using the directive **shake** in CONTROL. But the trouble may be much more likely to be cured by careful consideration of the physical system being simulated. For example, is the system stressed in some way? Too far from equilibrium?

Message 517: error - allowed configuration information levels are: 0, 1 and 2

DL_POLY_4 has found an erroneous configuration information level, l : 0.$le.l.le$.2, (i) for the trajectory option in CONTROL or (ii) in the header of CONFIG.

Action:

Correct the error in CONFIG and rerun.

Message 518: error - control distances for variable timestep not intact

DL_POLY_4 has found the control distances for the variable timestep algorithm to be in contention with each other.

Action:

`mxdis` MUST BE $>$ 2.5$\times$ `mndis`. Correct in CONTROL and rerun.

Message 519: error - REVOLD is incompatible or does not exist

Either REVOLD does not exist or its formatting is incompatible.

Action:

Change the `restart` option in CONTROL and rerun.

Message 520: error - domain decomposition failed

A DL_POLY_4 check during the domain decomposition mapping has been violated. The number of nodes allowed for imcon = 0 is only 1,2,4 and 8! The number of nodes allowed for imcon = 6 is restricted to 2 along the z direction! The number of nodes should not be a prime number since these are not factorisable/decomposable!

Action:

You must ensure DL_POLY_4 execution on a number of processors that complies with the advise above.

Message 530: error - pseudo thermostat thickness MUST comply with: 2 Angs <= thickness < a quarter of the minimum MD cell width

DL_POLY_4 has found a check violated while reading CONTROL.

Action:

Correct accordingly in CONTROL and resubmit.

Message 540: error - pseudo thermostat MUST only be used in bulk simulations, i.e. imcon MUST be 1, 2 or 3

DL_POLY_4 has found a check violated while reading CONTROL.

Action:

Correct accordingly in CONTROL nve or in CONFIG (`imcon`) and resubmit.

Message 551: error - REFERENCE not found !!!

The `defect` detection option is used in conjunction with `restart` but no REFERENCE file is found.

Action:

Supply a REFERENCE configuration.

Message 552: error - REFERENCE must contain cell parameters !!!

REFERENCE MUST contain cell parameters i.e. image convention MUST be **imcon** = 1, 2, 3 or 6.

Action:

Supply a properly formatted REFERENCE configuration.

Message 553: error - REFERENCE is inconsistent !!!

An atom has been lost in transfer between nodes. This should never happen!

Action:

Big trouble. Report problem to authors immediately.

Message 554: error - REFERENCE's format different from CONFIG's !!!

REFERENCE complies to the same rules as CONFIG with the exception that image convention MUST be **imcon** = 1, 2, 3 or 6.

Action:

Supply a properly formatted REFERENCE configuartion.

Message 555: error - particle assigned to non-existent domain in defects_read_reference

See notes on message 513 above.

Action:

See action notes on message 513 above.

Message 556: error - too many atoms in REFERENCE file

See notes on message 45 above.

Action:

See action notes on message 45 above.

Message 557: error - undefined direction passed to defects_reference_export

See notes on message 42 above.

Action:

See action notes on message 42 above.

Message 558: error - outgoing transfer buffer exceeded in defects_reference_export

See notes on message 38 above.

Action:

See action notes on message 38 above.

Message 559: error - incoming data transfer size exceeds limit in defects_reference_export

See notes on message 38 above.

Action:

See action notes on message 38 above.

Message 560: error - rdef found to be > half the shortest interatomic distance in REFERENCE

The defect detection option relies on a cutoff, rdef, to define the vicinity around a site (defined in REFERENCES) in which a particle can claim to occupy the site. Evidently, rdef MUST be < half the shortest interatomic distance in REFERENCE.

Action:

Decrease the value of rdef at directive **defect** in CONTROL.

Message 570: error - unsupported image convention (0) for system expansion option nfold

System expansion is possible only for system with periodicity on their boundaries.

Action:

Change the image convention in CONFIG to any other suitable periodic boundary condition.

Message 580: error - replay (HISTORY) option can only be used for structural property recalculation

No structural property has been specified for this option to activate itself.

Action:

In CONTROL specify properties for recalculation (RDFs,z-density profiles, defect detection) or alternatively remove the option.

Message 585: error - end of file encountered in HISTORY file

This means that the HISTORY file is incomplete in some way: Either should you abort the replay (HISTORY) option or provide a fresh HISTORY file before restart.

Action:

In CONTROL specify properties for recalculation (RDFs,z-density profiles, defect detection) or alternatively remove the option.

Message 590: error - uknown minimisation type, only ”force”, ”energy” and ”distance” are recognised

Configuration minimisation can take only these three criteria.

Action:

In CONTROL specify the criterion you like followed by the needed arguments.

Message 600: error - ”impact” option specified more than once in CONTROL

Only one instance of the ”impact” option is allowed in CONTROL.

Action:

Remove any extra instances of the ”impact” option in CONTROL.

Message 610: error - ”impact” applied on particle that is either frozen, or the shell of a core-shell unit or part of a RB

It is the user’s responsibility to ensure that impact is initiated on a ”valid” particle.

Action:

In CONTROL remove the ”impact” directive or correct the particle identity in it so that it complies with the requirements.

Message 620: error - duplicate or mixed intra-molecular entries specified in FIELD

The FIELD parser has detected an inconsistency in the description of bonding interactions. It is the user’s responsibility to ensure that no duplicate or mixed-up intra-molecular entries are specified in FIELD.

Action:

Look at the preceding warning message in OUTPUT and find out which entry of what intra-molecular-like interaction is at fault. Correct the bonding description and try running again.

Message 625: error - only one *rigid* directive per molecule is allowed

DL_POLY_4 has found more than one rigids entry per molecule in FIELD.

Action:

Correct the erroneous part in FIELD and resubmit.

Message 630: error - too many rigid body units specified

This should never happen! This indicates an erroneous FIELD file or corrupted DL_POLY_4 executable. Unlike DL_POLY_Classic, DL_POLY_4 does not have a set limit on the number of rigid body types it can handle in any simulation (this is not the same as the total number of RBs in the system or per domain).

Action:

Examine FIELD for erroneous directives, correct and resubmit.

Message 632: error - rigid body unit MUST have at least 2 sites

This is likely to be a corrupted FIELD file.

Action:

Examine FIELD for erroneous directives, correct and resubmit.

Message 634: error - rigid body unit MUST have at least one non-massless site

No RB dynamics is possible if all sites of a body are massless as no rotational inertia can be defined!

Action:

Examine FIELD for erroneous directives, correct and resubmit.

Message 638: error - coincidence of particles in rigid body unit

This indicates a corrupted FIELD file as all members of a RB unit must be destinguishable from one another.

Action:

Examine FIELD for erroneous directives, correct and resubmit.

Message 640: error - too many rigid body units per domain

DL_POLY_4 limits the number of rigid body units in the system to be simulated (actually, the number to be processed by each node) and checks for the violation of this. Termination will result if the condition is violated.

Action:

Use **densvar** option in CONTROL to increase `mxrgd` (alternatively, increase it by hand in SET_BOUNDS and recompile) and resubmit.

Message 642: error - rigid body unit diameter > rcut (the system cutoff)

DL_POLY_4 domain decomposition limits the size of a RB to a largest diagonal < system cutoff. I.e. the largest RB type is still within a linked cell volume.

Action:

Increase cutoff.

Message 644: error - overconstrained rigid body unit

This is a very unlikely message which usually indicates a corrupted FIELD file or unphysically overconstrained system.

Action:

Decrease constraint on the system. Examine FIELD for erroneous directives, if any, correct and resubmit.

Message 646: error - overconstrained constraint unit

This is a very unlikely message which usually indicates a corrupted FIELD file or unphysically overconstrained system.

Action:

Decrease constraint on the system. Examine FIELD for erroneous directives, if any, correct and resubmit.

Message 648: error - quaternion setup failed

This error indicates that the routine Q_SETUP has failed in reproducing all the atomic positions in rigid units from the centre of mass and quaternion vectors it has calculated.

Action:

Check the contents of the CONFIG file. DL_POLY_4 builds its local body description of a rigid unit type from the *first* occurrence of such a unit in the CONFIG file. The error most likely occurs because subsequent occurrences were not sufficiently similar to this reference structure. If the problem persists increase the value of `tol` in Q_SETUP and recompile. If problems still persist double the value of `dettest` in RIGID_BODIES_SETUP and recompile. If you still encounter problems contact the authors.

Message 650: error - failed to find principal axis system

This error indicates that the routine RIGID_BODIES_SETUP has failed to find the principal axis for a rigid unit.

Action:

This is an unlikely error. DL_POLY_4 should correctly handle linear, planar and 3-dimensional rigid units. There is the remote possibility that the unit has all of its mass-bearing particles frozen while some of the massless are not or the unit has just one mass-bearing particle. Another, more likely, possibility, in case of linear molecules is that the precision of the coordinates of these linear molecules' constituentsi, as produced by the user, is not good enough, which leads DL_POLY_4 to accepting it as non-linear while, in fact, it is and then failing at the current point. It is quite possible, despite considered as wrong practice, that the user defined system of linear RBs is, in fact, generated from a system of CBs (3 per RB) which has not been run in a high enough SHAKE/RATTLE tolerance accuracy (10ˆ8 and higher may be needed). Check the definition of the rigid unit in the CONFIG file, if sensible report the error to the authors.

Message 655: error - FENE bond breaking failure

A FENE type bond was broken.

Action:

Examine FIELD for erroneous directives, if any, correct and resubmit.

Message 660: error - bond-length > cutoff in TABBND or cutoff for PDF collection

A bond has reached a length beyond the cutoff over which (i) its interactions are defined in TABBND or (ii) its potential distribution function is sampled.

Action:

If there is a TABBND present, reconstruct TABBND with the original interaction potentials defined over a larger cutoff and try to run the system with new TABBND. If bonds PDFs are collected then increase the PDF bond cutoff value in CONTROL and try to run the system.

Message 670: error - insufficient electronic temperature cells for TTM heat diffusion

The number of coarse-grained electronic temperature (CET) cells for the heat diffusion calculations of the two-temperature model (TTM) in any direction (x, y or z) is less than the number of coarse-grained ionic temperature (CIT) cells.

Action:

Examine the OUTPUT for the number of ionic temperature cells and modify the **ttm ncet** directive in the CONTROL file to ensure there are at least as many electronic temperature cells.

Message 680: error - rpad too large for calculation of ionic temperatures

The padding distance applied to pair interactions is the maximum distance a particle may exist beyond a periodic boundary. In the case of calculating ionic temperatures for TTM, this distance extends beyond the extent (in any direction) of the ionic temperature cells and thus this property cannot be reliably calculated.

Action:

Reduce the padding distance for pair interactions with the **rpad** directive in the CONTROL file.

Message 681: error - electronic specific heat not fully specified

Not enough information is given for electronic specific heat capacity functions (other than a constant value) used in two-temperature model calculations.

Action:

Ensure that two positive parameters are given with either the **ttm cetanh** or **ttm celin** directives in the CONTROL file.

Message 682: error - thermal conductivity of metal not specified

No information given for thermal conductivity of metal used in two-temperature model calculations.

Action:

Ensure that a positive parameter is given with either the **ttm keconst** or **ttm kedrude** directives in the CONTROL file.

Message 683: error - thermal diffusivity of non-metal not specified

No information given for thermal diffusivity of non-metal used in two-temperature model calculations.

Action:

Ensure that a positive parameter is given with the **ttm diff** directive in the CONTROL file.

Message 684: error - cannot find or open thermal conductivity table file (Ke.dat)

No readable file (Ke.dat) is available to read tabulated thermal conductivities.

Action:

Ensure that a readable text file named Ke.dat is available in the directory where DL_POLY_4 is being run. (This must be supplied if the **ttm ketab** directive is given in the CONTROL file.)

Message 685: error - no data found in thermal conductivity table file (Ke.dat)

No tabulated thermal conductivities can be read from the Ke.dat file.

Action:

Ensure that the Ke.dat file is formatted correctly in two columns: temperature and thermal conductivity. Temperatures should be greater than or equal to 0 kelvin.

Message 686: error - cannot find or open volumetric heat capacity table file (Ce.dat)

No readable file (Ce.dat) is available to read tabulated volumetric heat capacities.

Action:

Ensure that a readable text file named Ce.dat is available in the directory where DL_POLY_4 is being run. (This must be supplied if the **ttm cetab** directive is given in the CONTROL file.)

Message 687: error - no data found in volumetric heat capacity table file (Ce.dat)

No tabulated volumetric heat capacities can be read from the Ce.dat file.

Action:

Ensure that the Ce.dat file is formatted correctly in two columns: temperature and volumetric heat capacity (equal to product of specific heat capacity and density). Temperatures should be greater than or equal to 0 kelvin.

Message 688: error - cannot find or open thermal diffusivity table file (De.dat)

No readable file (Ce.dat) is available to read tabulated volumetric heat capacities.

Action:

Ensure that a readable text file named De.dat is available in the directory where DL_POLY_4 is being run. (This must be supplied if the **ttm detab** directive is given in the CONTROL file.)

Message 689: error - no data found in thermal diffusivity table file (De.dat)

No tabulated thermal diffusivities can be read from the De.dat file.

Action:

Ensure that the De.dat file is formatted correctly in two columns: temperature and thermal diffusivity. Temperatures should be greater than or equal to 0 kelvin.

Message 690: error - cannot find or open coupling constant table file (g.dat)

No readable file (g.dat) is available to read tabulated electron-phonon coupling constants.

Action:

Ensure that a readable text file named g.dat is available in the directory where DL_POLY_4 is being run. (This must be supplied if the **ttm gvar** directive is given in the CONTROL file.)

Message 691: error - no data found in coupling constant table file (g.dat)

No tabulated thermal conductivities can be read from the g.dat file.

Action:

Ensure that the g.dat file is formatted correctly in two columns: temperature and electron-phonon coupling constant. Temperatures should be greater than or equal to 0 kelvin.

Message 692: error - end of file encountered in table file (Ke.dat, Ce.dat, De.dat or g.dat)

DL_POLY_4 encountered the end of one of the tabulated files (Ke.dat, Ce.dat, De.dat or g.dat) for two-temperature model calculations prematurely.

Action:

Check that the tabulated files are not corrupted or incomplete in some way.

Message 693: error - negative electronic temperature: instability in electronic heat diffusion equation

A negative (non-physical) electronic temperature has been obtained during solution of the thermal diffusion equation used in the two-temperature model. This is an indication of instability in the numerical solution of this partial differential equation.

Action:

This error should not happen due to careful selection of the timestep used for the explicit difference solver. In some limited circumstances, it may be possible to improve the solver stability by decreasing the value of `fopttstep` in TTM_THERMAL_DIFFUSION, which is used to scale the timestep.

Message 694: error - electronic temperature restart file (DUMP_E) does not exist

The DUMP_E file (containing electronic temperatures for restarting simulations with the two-temperature model) does not exist.

Action:

A DUMP_E file should be supplied if the **restart** directive is included in the CONTROL file.

Message 695: error - mismatch in electronic temperature lattice sizes between restart (DUMP_E) and CONTROL files

The electronic temperature lattice size (number of CET cells) given in the DUMP_E restart file for two-temperature model simulations does not correspond to the size given in the CONTROL file.

Action:

Ensure that the **ttm ncet** directive in the CONTROL file matches up with the three numbers in the first line of the DUMP_E file.

Message 696: error - cannot read electronic temperature restart (DUMP_E) file

The electronic temperature lattice size (number of CET cells) given in the DUMP_E restart file for two-temperature model simulations does not correspond to the size given in the CONTROL file.

Action:

Check that the DUMP_E file is not corrupted or incomplete in some way.

Message 1000: error - working precision mismatch between FORTRAN90 and MPI implementation

DL_POLY_4 has failed to match the available modes of MPI precision for real numbers to the defined in sc kinds_f90 FORTRAN90 working precision `wp` for real numbers. `wp` is a precompile parameter.

Action:

This simply mean that `wp` must have been changed from its original value to something else and the new value is not matched by the `mpi_wp` variable in COMMS_MODULE. It is the user's responsibility to ensure that `wp` and `mpi_wp` are compliant. Make the necessary corrections to sc kinds_f90 and/or COMMS_MODULE.

Message 1001: error - allocation failure in comms_module $->$ gcheck_vector

DL_POLY_4 has failed to find available memory to allocate an array or arrays, i.e. there is lack of sufficient memory (per node) on the execution machine.

Action:

This may simply mean that your simulation is too large for the machine you are running on. Consider this before wasting time trying a fix. Try using more processing nodes if they are available. If this is not an option investigate the possibility of increasing the heap size for your application. Talk to your systems support people for advice on how to do this.

Message 1002: error - deallocation failure in comms_module − > gcheck_vector

DL_POLY_4 has failed to deallocate an array or arrays, i.e. to free memory that is no longer in use.
Action:
Talk to your systems support people for advice on how to manage this.

Message 1003: error - allocation failure in comms_module − > gisum_vector

See notes on message 1001 above.
Action:
See action notes on message 1001 above.

Message 1004: error - deallocation failure in comms_module − > gisum_vector

See notes on message 1002 above.
Action:
See action notes on message 1002 above.

Message 1005: error - allocation failure in comms_module − > grsum_vector

See notes on message 1001 above.
Action:
See action notes on message 1001 above.

Message 1006: error - deallocation failure in comms_module − > grsum_vector

See notes on message 1002 above.
Action:
See action notes on message 1002 above.

Message 1007: error - allocation failure in comms_module − > gimax_vector

See notes on message 1001 above.
Action:
See action notes on message 1001 above.

Message 1008: error - deallocation failure in comms_module − > gimax_vector

See notes on message 1002 above.
Action:
See action notes on message 1002 above.

Message 1009: error - allocation failure in comms_module − > grmax_vector

See notes on message 1001 above.

Action:

See action notes on message 1001 above.

Message 1010: error - deallocation failure in comms_module − > grmax_vector

See notes on message 1002 above.

Action:

See action notes on message 1002 above.

Message 1011: error - allocation failure in parse_module − > get_record

See notes on message 1001 above.

Action:

See action notes on message 1001 above.

Message 1012: error - deallocation failure in parse_module − > get_record

See notes on message 1002 above.

Action:

See action notes on message 1002 above.

Message 1013: error - allocation failure in angles_module − > allocate_angles_arrays

See notes on message 1001 above.

Action:

See action notes on message 1001 above.

Message 1014: error - allocation failure in bonds_module − > allocate_bonds_arrays

See notes on message 1001 above.

Action:

See action notes on message 1001 above.

Message 1015: error - allocation failure in core_shell_module − > allocate_core_shell_arrays

See notes on message 1001 above.

Action:

See action notes on message 1001 above.

Message 1016: error - allocation failure in statistics_module − > allocate_statitics_arrays

See notes on message 1001 above.

Action:

See action notes on message 1001 above.

Message 1017: error - allocation failure in tethers_module − > allocate_tethers_arrays

See notes on message 1001 above.

Action:

See action notes on message 1001 above.

Message 1018: error - allocation failure in constraints_module − > allocate_constraints_arrays

See notes on message 1001 above.

Action:

See action notes on message 1001 above.

Message 1019: error - allocation failure in external_field_module − > allocate_external_field_arrays

See notes on message 1001 above.

Action:

See action notes on message 1001 above.

Message 1020: error - allocation failure in dihedrals_module − > allocate_dihedrals_arrays

See notes on message 1001 above.

Action:

See action notes on message 1001 above.

Message 1021: error - allocation failure in inversions_module − > allocate_inversion_arrays

See notes on message 1001 above.

Action:

See action notes on message 1001 above.

Message 1022: error - allocation failure in vdw_module − > allocate_vdw_arrays

See notes on message 1001 above.

Action:

See action notes on message 1001 above.

Message 1023: error - allocation failure in metal_module − > allocate_metal_arrays

See notes on message 1001 above.

Action:

See action notes on message 1001 above.

Message 1024: error - allocation failure in three_body_module − > allocate_three_body_arrays

See notes on message 1001 above.

Action:

See action notes on message 1001 above.

Message 1025: error - allocation failure in config_module − > allocate_config_arrays

See notes on message 1001 above.

Action:

See action notes on message 1001 above.

Message 1026: error - allocation failure in site_module − > allocate_site_arrays

See notes on message 1001 above.

Action:

See action notes on message 1001 above.

Message 1027: error - allocation failure in tersoff_module − > alocate_tersoff_arrays

See notes on message 1001 above.

Action:

See action notes on message 1001 above.

Message 1028: error - deallocation failure in angles_module − > deallocate_angles_arrays

See notes on message 1002 above.

Action:

See action notes on message 1002 above.

Message 1029: error - deallocation failure in bonds_module − > deallocate_bonds_arrays

See notes on message 1002 above.

Action:

See action notes on message 1002 above.

Message 1030: error - deallocation failure in core_shell_module − > deallocate_core_shell_arrays

See notes on message 1002 above.

Action:

See action notes on message 1002 above.

Message 1031: error - deallocation failure in tethers_module − > deallocate_tethers_arrays

See notes on message 1002 above.

Action:

See action notes on message 1002 above.

Message 1032: error - deallocation failure in constraints_module − > deallocate_constraints_arrays

See notes on message 1002 above.

Action:

See action notes on message 1002 above.

Message 1033: error - deallocation failure in dihedrals_module − > deallocate_dihedrals_arrays

See notes on message 1002 above.

Action:

See action notes on message 1002 above.

Message 1034: error - deallocation failure in inversions_module − > deallocate_inversions_arrays

See notes on message 1002 above.

Action:

See action notes on message 1002 above.

Message 1035: error - allocation failure in defects_module − > allocate_defects_arrays

See notes on message 1001 above.

Action:

See action notes on message 1001 above.

Message 1036: error - allocation failure in pmf_module − > allocate_pmf_arrays

See notes on message 1001 above.

Action:

See action notes on message 1001 above.

Message 1037: error - deallocation failure in pmf_module − > deallocate_pmf_arrays

See notes on message 1002 above.

Action:

See action notes on message 1002 above.

Message 1038: error - allocation failure in minimise_module − > allocate_minimise_arrays

See notes on message 1001 above.

Action:

See action notes on message 1001 above.

Message 1039: error - deallocation failure in minimise_module − > deallocate_minimise_arrays

See notes on message 1002 above.

Action:

See action notes on message 1002 above.

Message 1040: error - allocation failure in ewald_module − > ewald_allocate_kall_arrays

See notes on message 1001 above.

Action:

See action notes on message 1001 above.

Message 1041: error - allocation failure in langevin_module − > langevin_allocate_arrays

See notes on message 1001 above.

Action:

See action notes on message 1001 above.

Message 1042: error - allocation failure in rigid_bodies_module − > allocate_rigid_bodies_arrays

See notes on message 1001 above.

Action:

See action notes on message 1001 above.

Message 1043: error - deallocation failure in rigid_bodies_module − > deallocate_rigid_bodies_arrays

See notes on message 1002 above.

Action:

See action notes on message 1002 above.

Message 1044: error - allocation failure in comms_module − > gimin_vector

See notes on message 1001 above.

Action:

See action notes on message 1001 above.

Message 1045: error - deallocation failure in comms_module − > gimin_vector

See notes on message 1002 above.

Action:

See action notes on message 1002 above.

Message 1046: error - allocation failure in comms_module − > grmin_vector

See notes on message 1001 above.

Action:

See action notes on message 1001 above.

Message 1047: error - deallocation failure in comms_module − > grmin_vector

See notes on message 1002 above.

Action:

See action notes on message 1002 above.

Message 1048: error - error - allocation failure in comms_module − > grsum_matrix

See notes on message 1001 above.

Action:

See action notes on message 1001 above.

Message 1049: error - deallocation failure in comms_module − > grsum_matrix

See notes on message 1002 above.

Action:

See action notes on message 1002 above.

Message 1050: error - sorted I/O base communicator not set

Possible corruption if IO_MODULE. This should never happen!

Action:

Make sure you have a clean copy of DL_POLY_4, compiled without any suspicious warning messages. Contact authors if the problem persists.

Message 1053: error - sorted I/O allocation error

Your I/O buffer (and possibly batch) size is too big.

Action:

Decrease the value of the I/O buffer (and possibly batch) size in CONTROL and restart your job.

Message 1056: error - unkown write option given to sorted I/O

This should never happen!

Action:

Contact authors if the problem persists.

Message 1059: error - unknown write level given to sorted I/O

This should never happen!

Action:

Contact authors if the problem persists.

Message 1060: error - allocation failure in statistics_module − > allocate_statitics_connect

See notes on message 1001 above.

Action:

See action notes on message 1001 above.

Message 1061: error - allocation failure in statistics_module − > deallocate_statitics_connect

See notes on message 1001 above.

Action:

See action notes on message 1001 above.

Message 1063: error - allocation failure in vdw_module − > allocate_vdw_table_arrays

See notes on message 1001 above.

Action:

See action notes on message 1001 above.

Message 1066: error - allocation failure in vdw_module − > allocate_vdw_direct_fs_arrays

See notes on message 1001 above.

Action:

See action notes on message 1001 above.

Message 1069: error - allocation failure in metal_module − > allocate_metal_table_arrays

See notes on message 1001 above.

Action:

See action notes on message 1001 above.

Message 1070: error - allocation failure in ewald_module − > ewald_allocate_kfrz_arrays

See notes on message 1001 above.

Action:

See action notes on message 1001 above.

Message 1072: error - allocation failure in bonds_module − > allocate_bond_pot_arrays

See notes on message 1001 above.

Action:

See action notes on message 1001 above.

Message 1073: error - allocation failure in bonds_module − > allocate_bond_dst_arrays

See notes on message 1001 above.

Action:

See action notes on message 1001 above.

Message 1074: error - allocation failure in angles_module − > allocate_angl_pot_arrays

See notes on message 1001 above.

Action:

See action notes on message 1001 above.

Message 1075: error - allocation failure in angles_module − > allocate_angl_dst_arrays

See notes on message 1001 above.

Action:

See action notes on message 1001 above.

Message 1076: error - allocation failure in dihedrals_module − > allocate_dihd_pot_arrays

See notes on message 1001 above.

Action:

See action notes on message 1001 above.

Message 1077: error - allocation failure in dihedrals_module − > allocate_dihd_dst_arrays

See notes on message 1001 above.

Action:

See action notes on message 1001 above.

Message 1078: error - allocation failure in inversions_module − > allocate_invr_pot_arrays

See notes on message 1001 above.

Action:

See action notes on message 1001 above.

Message 1079: error - allocation failure in inversions_module − > allocate_invr_dst_arrays

See notes on message 1001 above.

Action:

See action notes on message 1001 above.

Message 1080: error - allocation failure in greenkubo_module − > allocate_greenkubo_arrays

See notes on message 1001 above.

Action:

See action notes on message 1001 above.

Message 1081: error - allocation failure in dpd_module − > allocate_dpd_arrays

See notes on message 1001 above.

Action:

See action notes on message 1001 above.

Message 1083: error - allocation failure in ttm_module − > allocate_ttm_arrays

See notes on message 1001 above.

Action:

See action notes on message 1001 above.

Message 1084: error - deallocation failure in ttm_module − > deallocate_ttm_arrays

See notes on message 1002 above.

Action:

See action notes on message 1002 above.

Message 1085: error - allocation failure in ttm_ion_temperature

See notes on message 1001 above.

Action:

See action notes on message 1001 above.

Message 1086: error - deallocation failure in ttm_ion_temperature

See notes on message 1002 above.

Action:

See action notes on message 1002 above.

Message 1087: error - allocation failure in ttm_thermal_diffusion

See notes on message 1001 above.

Action:

See action notes on message 1001 above.

Message 1088: error - deallocation failure in ttm_thermal_diffusion

See notes on message 1002 above.

Action:

See action notes on message 1002 above.

Message 1089: error - allocation failure in ttm_track_module − > depoinit

See notes on message 1001 above.

Action:

See action notes on message 1001 above.

Message 1090: error - deallocation failure in ttm_track_module − > depoevolve

See notes on message 1002 above.

Action:

See action notes on message 1002 above.

Appendix E

DL_POLY_4 INSTALL Notes & README Wisdom

E.1 INSTALL

```
# Building notes
* these notes are for building with [**cmake**](https://cmake.org)
* you can pass options to cmake using **-DOPTION=value**. For a complete list of options
   inspect [cmake/DLPOLYBuildOptions.cmake](cmake/DLPOLYBuildOptions.cmake)
* cmake -L <path to CMakeLists.txt> will show you a list of all available options.
* explicit compiler specification can be achieved by using environment variable **FC** (
   eg. using Intel ifort *FC=ifort*)
* compiler flags can be altered via **FFLAGS**, (eg *FFLAGS="-O3 -xHost"*)
* one also can use **cmake-gui** or **ccmake** to setup the build options
* to change the install path use **-DCMAKE_INSTALL_PREFIX=<path>** (*-
   DCMAKE_INSTALL_PREFIX=$HOME/101/DL_POLY*)
* automatic testing can be done after **DL_POLY_4** is built, using **make test**
* to see all the tests available use **ctest -N**
* to run one specific test use **ctest -R <TESTNAME>**
* for a list of all supported targets **make help**
* TODO check it works on Windows...

## Standard MPI
‘‘‘sh
mkdir build-mpi-pure
pushd build-mpi-pure
FFLAGS="-O3" cmake ../
make -j10
make install
‘‘‘
* will use whatever default MPI is found

*Intel Compilers - Intel MPI*
‘‘‘sh
FC=ifort FFLAGS="-O3" cmake ../ -DMPI_Fortran_COMPILER=mpiifort
‘‘‘
*Intel Compilers - Some default mpi library, other than Intel MPI*
‘‘‘sh
```

```
FC=ifort FFLAGS="-O3" cmake ../
‘‘‘

## Hybrid MPI and OpenMP
‘‘‘sh
mkdir build-mpi-openmp
pushd build-mpi-openmp
FFLAGS="-O3" cmake ../ -DWITH_OPENMP=ON
make -j10
make install
‘‘‘
*Intel Compilers - Intel MPI*
‘‘‘sh
FC=ifort FFLAGS="-O3" cmake ../ -DWITH_OPENMP=ON -DMPI_Fortran_COMPILER=mpiifort
‘‘‘

## Serial
‘‘‘sh
mkdir build-serial
pushd build-serial
FFLAGS="-O3" cmake ../ -DWITH_MPI=OFF
‘‘‘
*Intel Compilers*
‘‘‘sh
FC=ifort FFLAGS="-O3" cmake ../ -DWITH_MPI=OFF
‘‘‘

## Serial with OpenMP threads
‘‘‘sh
mkdir build-openmp
pushd build-openmp
FFLAGS="-O3" cmake ../ -DWITH_OPENMP=ON -DWITH_MPI=OFF
‘‘‘
*Intel Compilers*
‘‘‘sh
FC=ifort FFLAGS="-O3" cmake ../ -DWITH_OPENMP=ON -DWITH_MPI=OFF
‘‘‘
## Xeon Phi
‘‘‘sh
mkdir build-xeonphi
pushd build-xeonphi
FC=ifort FFLAGS="-O3 " cmake ../ -DWITH_PHI=ON -DWITH_MPI=ON
‘‘‘

## Optimisation flags
* gfortran

‘‘‘sh
FFLAGS="-O3 -mtune=native"
‘‘‘

* Intel
```

```
‘‘‘sh
FFLAGS="-fpp -O3 -xHost -fimf-domain-exclusion=15"
‘‘‘

* If you plan to run the binary on a different type of a machine than you build it, check
    the manual of your compiler
for the flags matching the _running machine_

## Debugging, or when things go unexpected
* gfortran/ifort

‘‘‘sh
cmake ../ -DCMAKE_BUILD_TYPE=Debug
‘‘‘
* other compilers

‘‘‘sh
FFLAGS="desired flags" cmake ../
‘‘‘

## Building with NETCDF support
‘‘‘sh
mkdir build-mpi-netcdf
pushd build-mpi-netcdf
FFLAGS="-O3" cmake ../ -DWITH_NETCDF=ON
make -j10
make install
‘‘‘

## Building with KIM support
‘‘‘
mkdir build-mpi-kim
pushd build-mpi-kim
FFLAGS="-O3" cmake ../ -DWITH_KIM=ON
make -j10
make install
‘‘‘

## Building with PLUMED support
‘‘‘sh
mkdir build-mpi-plumed
pushd build-mpi-plumed
FFLAGS="-O3" cmake ../ -DWITH_PLUMED=ON
make -j10
make install
‘‘‘

## building with DOXYGEN API Documentation

‘‘‘sh
mkdir build-mpi
```

````
cd build mpi
cmake -DDOCS_DOXYGEN=On ..
make doxygen
‘‘‘

## building with FORD API Documentation

‘‘‘sh
mkdir build-mpi
cd build mpi
cmake -DDOCS_FORD=On ..
make ford
‘‘‘

# FAQ

## On Ubuntu machines

It was noticed that for some mpi implementations the linking stage fails. You will see a
   lot of errors claiming undefined references to MPI_*
**solution**

‘‘‘sh
FC=mpif90 FFLAGS="-O3" cmake ../
‘‘‘

## Intel MPI

Intel MPI Fortran wrapper breaks ifort preprocessing
you will get an error on the lines Len_trim(xxx) not supported or similar.
**solution**
do not use FC=mpiifort
````

E.2 coding style

````
# Coding Style

The programming style of DL\_POLY is intended to ensure the code is a
consistent, readable and maintainable as possible. The following rules apply
throughout the code. Contributors to the code are urged to read and apply this
style before submitting code for inclusion consideration.

# Units

 All routines employ DL_POLY internal units. Output contains, when relevant, the units.

    SIMULATION CONTROL PARAMETERS

    simulation temperature (K)          1.0000E+01

    simulation pressure (katms)         0.0000E+00
````

~~~~
    Integration : Leapfrog Verlet
    Ensemble : NVT Nose-Hoover
    thermostat relaxation time (ps)     1.0000E-01

    selected number of timesteps         5000

# General Style

-   Use modern Fortran free form syntax.
-   Code should be written in Fortran2003/2008 dialect if possible.
-   Check the deprecated features from Fortran 2003/2008 and prior.
    Avoid using these features.
-   Fortran2008 should be treated with care, since not all compilers
    implement it.
-   Do not use **Common** blocks and **Block Data**, use **Modules**
    with public data if constant or a user defined type to group data which
    changes during runtime.
-   Do not use **go to** statements
-   Do not use **format** statements
-   File extension shall be .F90 for any new written code.
-   Indent code blocks by two space characters. Do not use
    tabs as they are not part of the Fortran standard.

‘‘‘fortran
Subroutine init(T,k,traj)
  Class(current_type) :: T
  Type(trajectory_type),Intent(in)  :: traj
  Real(Kind=dp),Intent(in)          :: k(3)

  Allocate(T%jk(traj%nFrames,3))
  T%k = k
  T%n = traj%nFrames
End Subroutine init
‘‘‘

-   Indent **Contains** statements to the same level as the sorrounding
    **Module**, or procedure

‘‘‘fortran
Module my_module
  Implicit None

  Integer( Kind = wi ), Parameter :: pi = 3.14...

  ...

Contains

  Subroutine my_subroutine(a,b)

    ...
~~~~

```
End Module my_module
‘‘‘

-   Do not use more than one blank line to separate blocks of code.
-   Code lines shall not exceed 132 characters (the modern Fortran standard
    limit).
-   Do not write Fortran keywords in ALL capitals. Capitalize the first
    letter of the statement to make them stand out for the reader.

‘‘‘fortran
    Program dl_poly
      Use kinds, Only : wi,wp
      Implicit None
    End Program dl_poly
‘‘‘
-   While Fortran supports multiple statements on a line separated by *;*
    try to use them sparingly and wisely.
-   Variables, constants and program units should be named in English or using
    common physics notation. Do not use cryptic names.
-   Try to group variables names based on physical meaning or usage.

‘‘‘fortran
  ! Verlet neighbour list data
  !> Update cells flag
  Logical, Public :: update = .true.
  !> Unconditional update flag
  Logical, Public :: unconditional_update = .false.


  !> Tracking points for Verlet neighbour list
  Real( Kind = wp ), Allocatable, Public :: xbg(:),ybg(:),zbg(:)

  !> Largest vdw cutoff, defines Verlet neighbour list radius
  Real( Kind = wp ), Public :: cutoff
  !> Padding around cutoff
  Real( Kind = wp ), Public :: padding
  !> Actual Verlet neighbour list cutoff (cutoff+padding)
  Real( Kind = wp ), Public :: cutoff_extended

  !> Linked cell list
  Integer( Kind = wi ), Allocatable, Public :: list(:,:)
  !> Maximum rank of linked cell list
  Integer( Kind = wi ), Public :: max_list
  !> Maximum number of cells per domain
  Integer( Kind = wi ), Public :: max_cell
‘‘‘

-   Avoid naming program units, variables, constants using
    intrinsincs routines or keywords from fortran standard.
-   While Fortran is case insensitive, use for lower case letters for program
    units, variables and constants.
-   Where more than a word is used in a name use _ as separator (snake
```

~~~~
    case notation).
-   Use the mordern syntax for logical expressions,
    -   == not .eq.
    -   /= not .ne.
    -   &gt; not .gt.
    -   &lt; not .lt.
    -   &gt;= not .ge.
    -   &lt;= not .le. .
-   Prefer positive testing in logical blocks.

‘‘‘fortran
  If (isw == 0) Then
     ! do some fancy code
  Else
    ! do some not so fancy code
  End If
! to
  If (isw /= 0) Then
    ! do some not so fancy code
  Else
    ! do some fancy code
  End If
‘‘‘
-   One line **If** shall be avoided.
-   Always use the optional separation space for **End** constructs, *e.g.*
    **End If** not **Endif** and **End Do** not **Enddo**
-   For program units use the full **End**: **End Subroutine name**.
-   Never use **Go To** statements
-   Use **Do** ... **End Do** instead of **Do** ... Label **Continue**
-   Do not use **Format**.

‘‘‘fortran

  Write(*,20) reclen

20 Format('(',i0,'a)')

! replace with

  Write(*,'(a,i0,a)')"(",reclen,"a)"
‘‘‘

-   Do not use interactive commands, like **Read** from keyboard or
    **Pause**.
-   Use **Implicit None** to avoid unwanted side effects. This is only nessecary once per
     module.
-   Avoid implicit casting of data: use 2.0_wp instead of 2.0 in an
    expression.

‘‘‘fortran
 If (sw == 1) Then
   factor = 2.0_wp*engke*rf
~~~~

```
 End If
! and not
If (sw == 1) factor = 2.0_wp*engke*rf
‘‘‘

-   Floating point comparisons shall be done with care. Consider the following
    examples using a tolerance of **Epsilon(a)** or **Tiny(a)**,
    -   a > tolerance instead of a > 0.0
    -   a - b > tolerance instead of a > b
    -   abs(a - b) < tolerance instead of a == b .
-   Avoid use of magic numbers, declare constants at the top level of a module and use
   those instead.

‘‘‘fortran
If (integrator == 1 ) Then
! instead use

Integer( Kind = wi ), Parameter :: VELOCITY_VERLET = 1

...

If (integrator == VELOCITY_VERLET ) Then
‘‘‘

-   Any new feature shall be turnable on/off from CONTROL file.
-   Any new feature shall be documented in the manual and will cite relevant
    literature.

## Modular structure

-   All subroutines/functions shall be enclosed by a module.
-   Modules may contain the following:
  -  Type declarations
  -  Subroutines and functions
  -  Paramter definitions (using the **Parameter** attribute)
  -  Interfaces for overloaded procedures
  -  Declaration of **Public** data
-  Modules may NOT contain the following:
  -  Variables (_i.e._ specifications without the **Parameter** attribute)
-   By default everything in a module shall be made private, explicitly
    make public what is needed outside the module or type using the **Public** statement.
-   Data which is used only in the defining module should be declared
    private.
-   Each module should be in a separate file.
-   Module names shall match their file names.
-   When using a module with the **Use** statement, **Only** must also be used
-   While overloading operators may be tempting, it is best avoided if
    you prefer performance to aesthetical beauty.

‘‘‘fortran
Use domains_module, Only : map
‘‘‘
```

~~~~
-   User derived types should be created to contain data relevant to the module,
   preferably one per module.
-   Types may match the module name but append them with **_type**.
-   Types shall provide init and **Final** procedures, optionally a summary
    procedure.
-   If provided a summary procedure shall produce valid YAML 1.2 output.

## Specification Statements

-   Align declaration statements clearly separating types and attributes
    from names.
-   Separate variable declaration block from code block of a subroutine
    by a blank line.
-   Use separate declaration blocks for routine arguments and local
    variables.

‘‘‘fortran
!> Calculate the factorial of n
Function factorial(n) Result(res)
  !> The integer to calculate the factorial of
  Integer( Kind = wi ), Intent( In    ) :: n
  !> The factorial of n
  Integer( Kind = wi ) :: res

  !> Running total
  Integer( Kind = wi ) :: total
  !> Loop iterator
  Integer( Kind = wi ) :: i

  ...
‘‘‘

-   Always use :: to separate types and attributes from names.
-   Use :: as an alignment guide if the list of names is too long.
-   Separate logical declaration blocks can be aligned differently to
    save screen real estate, *e.g.* parameters vs internal variables of a
    routine.

‘‘‘fortran
Integer,             Intent( In    ) :: imcon,mxshak
Real( Kind = wp ), Intent( InOut ) :: xxx(1:mxatms),yyy(1:mxatms),zzz(1:mxatms)
Real( Kind = wp ), Intent(   Out ) :: strcon(1:9)
Real( Kind = wp ), Intent(   Out ) :: vircon

Logical              :: safe
Integer              :: fail(1:2),i,j,k,icyc
Real( Kind = wp ) :: amti,amtj,dli,dlj, &
                        gamma,gammi,gammj,tstep2
‘‘‘

## Comments
~~~~

```
-   Comments shall be indented to align with the code
-   Comments shall be written in lower case except for proper nouns and
    standard abreviations.
-   Comments shall explain what a code does and why, not how it does
    it. Let the code explain how it is done.

## Ford and Doxygen

-  By conforming to the following style useful developer documentation may be created
   automatically using either FORD or Doxygen.
-  Comments attached to program units, variables and derived types may be automatically
   documented.
-  Documentation must precede the declaration of the unit, variable or derived type.
-  Comments to be documented must use the tag "!>" (This is default tag in FORD for a
   comment preceding the content. In Doxygen, "!>" is the only tag which can both start
   and continue a comment. So this seems to be the best compromise to make the source
   compatible with both systems. FORD does not like inline comments).
-  To insert a line break in a multiline comment use a blank comment line.
-  Comments use markdown syntax for formatting.
-  LaTeX style maths may be used to include equations in the documentation.
-  See the example structure for more comprehensive examples of documentation comments.

‘‘‘fortran
!># Example
!>
!> Author - John Smith
!>
!> An example program
Program example
  Use kinds, Only : wi
  Implicit None

  !> Integer counter
  Integer( Kind = wi ) :: i

  ...

  !> Calculate the factorial of n
  Function fact(n)

  ...
‘‘‘

## Procedures

-   A **Function** shall be pure (with no side-effects). If side-effect are
    needed use a **Subroutine**.
-   All arguments of a **Function/Subroutine** shall have an **Intent**
    attribute (with the exception of the **Class** in type bound procedures).
-   Avoid using **Recursive** procedures.
-   When you are passing an array argument to a **Subroutine/Function**,
    and the **Subroutine/Function** does not change the size of the
```

~~~~
    array, you should pass it as an assumed shape array. Memory
    management of such an array is automatically handled by the
    **Subroutine/Function**, and you do not have to worry about having
    to **Allocate** or **Deallocate** your array. It also helps the
    compiler to optimise the code.

## Allocatable Data and Pointers

-   If possible **Allocate** and **Deallocate** for an array or pointer
    shall appear in the same scope.
-   For **Allocatable** members of a user defined type, allocation shall
    happen in the **init** and deallocation in the **final** subroutine.
-   In all cases, **Deallocate** in the opposite order you did **Allocate**.

```fortran

  Allocate(xxx(n),Stat = stat)
  Allocate(yyy(n),Stat = stat)
  Allocate(zzz(n),Stat = stat)

  Deallocate(zzz)
  Deallocate(yyy)
  Deallocate(xxx)
```

-   If using **Pointer**, define it before usage by pointing to **Null**
    or using **Nullify**.
-   Similarly, when a pointer is not used anymore nullify it using the
    same techniques as above.
~~~~

E.3 contributing

~~~~
# Contribution workflow and the review process

This document outlines the best practice, using git and CI, which **must** be
followed for all contributions to DL_POLY. Also contained are instructions and
tips for managing your fork of the project which will help keep merges clean
and avoid many headaches.

## Golden rules

In brief the rules for contribution are as follows,

 * Follow the branch, fix, merge model, from your own fork
 * An issue must be created for every piece of work (bug, feature, *etc.*)
 * Merge requests will not be accepted without a review
 * New features must have a test
 * All tests must pass, no regressions may be merged

## Issues

### Using issues
~~~~

```
 * Open an issue for each piece of work done.
 * Open issues for design discussions. For example the questions Aidan had
   about link cells/domain decomposition. The answers may be important for
   newer members.
 * New features, _e.g._ task parallelism by Aidan, shaped particles by Vlad,
   shall have an issue too, the comments shall be used to provide succint
   reports on progress.

### Labels

Labels may be assigned to issues to help classify them. Examples include,

 * Testing: as in testing the water. This label shall be attached to things one
   may think to make one day.
 * Development: this is what I am working now on.
 * Production: anything in this is critical and shall be fixed asap.
 * Design: queries or suggestions about the structure of the program.
 * Leader: for issues relating to new features.

## Review

All merge requests will be reviewed to ensure the integrity of the code.

The reviewer/s have the following responsibilities,
 * Ensuring all contribution rules have been followed
 * Ensuring the [coding style](./coding_style.md) is adhered to
 * Only accepting a merge if all tests have passed
 * Using the comments system to request changes for the submittor to make

## Using the git for development

The Gitlab instance hosts a _devel_ repository, which we will refer to as
_devel_. Contributors will work on their own personal copies of the
repository by creating _forks_. This allows us to keep _devel_ clean (one
commit per merge request, if possible, all commits passing tests) while users may work on
their own _fork_, creating commits and changing the code as they see fit.

The _devel_ repository may be cloned as follows,

‘‘‘sh
git clone git@gitlab.com:ccp5/dl-poly.git dl-poly-devel
‘‘‘
A _fork_ is created using the web UI. It may then be cloned for a user called 'username'
   as follows,

‘‘‘sh
git clone git@gitlab.com:username/dl-poly.git dl-poly-fork
‘‘‘

### Branch, fix, merge model:
```

```
All work should follow the workflow of branch, fix, merge. Let us assume you
have an issue with your code which needs to be fixed.

#### Step 1: Branch from your fork

Create a new branch for the issue on the dashboard of your fork, we will assume
the branch is called 'issueXYZ'. Clone the branch,

‘‘‘sh
$ git clone -b issueXYZ --single-branch git@gitlab.com:username/dl-poly.git dl-poly-
   issueXYZ
‘‘‘

Alternatively you can create the branch in the cli using

‘‘‘sh
# clone the repository, if you already have a local repository this is not nessecary
$ git clone git@gitlab.com:username/dl-poly.git dl-poly-issueXYZ
$ pushd dl-poly-issueXYZ
# create and checkout a new branch (this is equivilent to git branch followed by git
   checkout)
$ git checkout -b issueXYZ
# create a remote tracking branch for you to push your changes to
$ git push -u origin issueXYZ
‘‘‘

#### Step 2: Fix the issue and commit your changes

Fix whatever is wrong.
Use git status to see which files you have changed and prepare a commit.

‘‘‘sh
# stage changes
$ git add <filename|folder> to add the new things
# commit the changes with a clear and brief message
$ git commit -m "<commit message>"
# push the commit to origin
$ git push
‘‘‘

#### Step 3a: Merge your branch into devel

On the web interface navigate to _devel_ and create a merge request for your
branch on your _fork_. Add any relevant labels or milestones and assign a
reviewer. Compare the code and if you are happy click Submit Merge Request.

After the merge request has been submitted tests will be run and your reviewer
will be notified.

#### Step 3b: Finalising the merge

If all is OK with the commit your reviewer may set the request to be merged
```

```text
once all tests pass. Otherwise the reviewer may open discussions using the
Gitlab comment system to point out issues that may need to be addressed before
the commit can be merged.

If changes need to be made you may make more commits onto your branch. When you
push your branch to your _fork_ the merge request will be automatically updated
to use the latest commit. Reply to the discussions to indicate when and how
they have been addressed.

If your branch has become out of sync with _devel_ then conflicts may arise.
Sometimes these cannot be automatically resolved and you will need to resolve
them by hand. Gitlab provides instructions for this, or you can follow this
routine,

‘‘‘sh
# add devel as a remote if you have not already
$ git remote add devel git@gitlab.com:ccp5/dl-poly.git
# get the changes to devel since you started working on your issue
$ git fetch devel
# merge these changes into your branch (assuming you want to merge into the master branch
    on devel)
$ git merge devel/devel
# resolve any conflicts
# push to your fork
$ git push
‘‘‘

Alternatively you may use rebase which will replay the changes you made in your
branch on top of _devel/devel_ however be sure you understand the differences between
   merge and rebase

‘‘‘sh
# add devel as a remote if you have not already
$ git remote add devel git@gitlab.com:ccp5/dl-poly.git
# get the changes to devel since you started working on your issue
$ git fetch devel
# merge these changes into your branch (assuming you want to merge into the master branch
    on devel)
$ git rebase devel/devel
# resolve any conflicts
# push to your fork
$ git push
‘‘‘

### Advanced git

#### Keeping your fork in sync with project

By adding two remotes, one for _devel_ and one for your _fork_ it is possible
to keep your _fork_ in sync with _devel_. This will greatly simplify merge
requests.
```

```
‘‘‘sh
# clone your fork
$ git clone git@gitlab.com:username/dl-poly.git dl-poly-fork
pushd dl-poly-fork
# add a remote for devel
$ git remote add devel git@gitlab.com:ccp5/dl-poly.git
‘‘‘

These commands need to be done only once. ‘git remote -v‘ shall show you
the origin and project fetch and push links

‘‘‘sh
$ git remote -v
origin  git@gitlab.com:username/dl-poly.git (fetch)
origin  git@gitlab.com:username/dl-poly.git (push)
devel git@gitlab.com:ccp5/dl-poly.git (fetch)
devel git@gitlab.com:ccp5/dl-poly.git (push)
‘‘‘

When you need to sync your _fork_ with _devel_, do the following,

‘‘‘sh
# get the latest commits from devel
$ git fetch devel
# ensure you are in the master branch of your fork
$ git checkout master
# merge your master branch into the master branch of devel
$ git merge devel/devel
# push these changes back to the remote of your fork (origin)
$ git push
‘‘‘

of course one can use a similar process to merge any other branch or available
projects.

#### Rebasing commits

When working on an issue you may use multiple commits. When you are ready to
create a merge request, you should squash your changes into one commit in order
to keep _devel_ clean. This is most easily achieved with an interactive
rebase.

Assuming you have made five commits,

‘‘‘sh
# rebase your branch five commits before HEAD i.e. where your branch originally diverged
$ git rebase -i HEAD~5
# follow the instructions. ’pick’ the first commit then ’sqaush’ or ’fixup’ the rest.
# You should now be left with a single commit containing all your changes
# Push your commmit to the remote, use --force if you have already pushed this branch to
# ’rewrite history’
$ git push origin branchname --force
```

```
‘‘‘
using force is a powerful and dangerous option. use it only if you know 150% nobody else
   touched that branch.

#### Cleaning stale branches

Deleting branches from the web interface will get rid of the remotes and not of
your local copies. The local branches left behind are called stale branches. To
get rid of them

‘‘‘sh
$ git remote prune origin
‘‘‘

To delete a local branch

‘‘‘sh
$ git branch -d localBranch
‘‘‘

if unmerged commits exists but you still want to delete use

‘‘‘sh
$ git branch -D localBranch
‘‘‘

To delete a remote branch on the remote _origin_ use

‘‘‘sh
$ git push -d origin remoteBranch
‘‘‘

## Code Coverage
If one builds DL_POLY_4 with **-DWITH_COVERAGE=ON** two targets will be
available *make coverage* and *make runcoverage*.  First will run the code
coverage on all tests from *make test*.

*make runcoverage* will run on the inputs which are put by user in
**CodeAnalysis**. If one uses MPI **-DMPI_NPROCS**, default 4, controls on how
many processes the job will run.
```

Appendix F

DL_POLY_4 Academic Licence Agreement

F.1 The full version

reproducing the Combined Work from the Application, but excluding the System Libraries of the Combined Work.

1. Exception to Section 3 of the GNU GPL.
You may convey a covered work under sections 3 and 4 of this License without being bound by section 3 of the GNU GPL.

2. Conveying Modified Versions.
If you modify a copy of the Library, and, in your modifications, a facility refers to a function or data to be supplied by an Application that uses the facility (other than as an argument passed when the facility is invoked), then you may convey a copy of the modified version:

a) under this License, provided that you make a good faith effort to ensure that, in the event an Application does not supply the function or data, the facility still operates, and performs whatever part of its purpose remains meaningful, or
b) under the GNU GPL, with none of the additional permissions of this License applicable to that copy.
3. Object Code Incorporating Material from Library Header Files.
The object code form of an Application may incorporate material from a header file that is part of the Library. You may convey such object code under terms of your choice, provided that, if the incorporated material is not limited to numerical parameters, data structure layouts and accessors, or small macros, inline functions and templates (ten or fewer lines in length), you do both of the following:

a) Give prominent notice with each copy of the object code that the Library is used in it and that the Library and its use are covered by this License.
b) Accompany the object code with a copy of the GNU GPL and this license document.
4. Combined Works.
You may convey a Combined Work under terms of your choice that, taken together, effectively do not restrict modification of the portions of the Library contained in the Combined Work and reverse engineering for debugging such modifications, if you also do each of the following:

a) Give prominent notice with each copy of the Combined Work that the Library is used in it and that the Library and its use are covered by this License.
b) Accompany the Combined Work with a copy of the GNU GPL and this license document.
c) For a Combined Work that displays copyright notices during execution, include the copyright notice for the Library among these notices, as well as a reference directing the user to the copies of the GNU GPL and this license document.
d) Do one of the following:
0) Convey the Minimal Corresponding Source under the terms of this License, and the Corresponding Application Code in a form suitable for, and under terms that permit, the user to recombine or relink the Application with a modified version of the Linked Version to produce a modified Combined Work, in the manner specified by section 6 of the GNU GPL for conveying Corresponding Source.
1) Use a suitable shared library mechanism for linking with the Library. A suitable mechanism is one that (a) uses at run time a copy of the Library already present on the user's computer system, and (b) will operate properly with a modified version of the Library that is interface-compatible with the Linked Version.
e) Provide Installation Information, but only if you would otherwise be required to provide such information under section 6 of the GNU GPL, and only to the extent that

Bibliography

[1] Smith, W., and Forester, T. R., 1996, *J. Molec. Graphics*, **14**, 136. 3

[2] Todorov, I. T., and Smith, W., 2004, *Phil. Trans. R. Soc. Lond. A*, **362**, 1835. 3, 233

[3] Todorov, I. T., Smith, W., Trachenko, K., and Dove, M. T., 2006, *J. Mater. Chem.*, **16**, 1611–1618. 3, 233

[4] Smith, W., 1987, *Molecular Graphics*, **5**, 71. 3

[5] Smith, W., 1991, *Comput. Phys. Commun.*, **62**, 229. 3, 5, 233

[6] Smith, W., 1993, *Theoretica. Chim. Acta.*, **84**, 385. 3, 5, 233

[7] Smith, W., and Forester, T. R., 1994, *Comput. Phys. Commun.*, **79**, 52. 3

[8] Smith, W., and Forester, T. R., 1994, *Comput. Phys. Commun.*, **79**, 63. 3, 5, 76

[9] Pinches, M. R. S., Tildesley, D., and Smith, W., 1991, *Molecular Simulation*, **6**, 51. 3, 5, 233

[10] Rapaport, D. C., 1991, *Comput. Phys. Commun.*, **62**, 217. 3, 5, 233

[11] Daw, M. S., and Baskes, M. I., 1984, *Phys. Rev. B*, **29**, 6443. 4, 33

[12] Foiles, S. M., Baskes, M. I., and Daw, M. S., 1986, *Chem. Phys. Lett.*, **33**, 7983. 4, 33

[13] Finnis, M. W., and Sinclair, J. E., 1984, *Philos. Mag. A*, **50**, 45. 4, 33, 35

[14] Sutton, A. P., and Chen, J., 1990, *Philos. Mag. Lett.*, **61**, 139. 4, 35

[15] Rafii-Tabar, H., and Sutton, A. P., 1991, *Philos. Mag. Lett.*, **63**, 217. 4, 35, 43

[16] Todd, B. D., and Lynden-Bell, R. M., 1993, *Surf. Science*, **281**, 191. 4, 35

[17] Tersoff, J., 1989, *Phys. Rev. B*, **39**, 5566. 4, 44, 234

[18] van Gunsteren, W. F., and Berendsen, H. J. C. 1987, *Groningen Molecular Simulation (GROMOS) Library Manual.* BIOMOS, Nijenborgh, 9747 Ag Groningen, The Netherlands. Standard GROMOS reference. 4, 13

[19] Mayo, S. L., Olafson, B. D., and Goddard, W. A., 1990, *J. Phys. Chem.*, **94**, 8897. 4, 13, 47, 48, 203

[20] Weiner, S. J., Kollman, P. A., Nguyen, D. T., and Case, D. A., 1986, *J. Comp. Chem.*, **7**, 230. 4, 13

[21] Smith, W., 2003, *Daresbury Laboratory.* 4, 10, 138, 149, 150, 151, 186, 258

[22] Allen, M. P., and Tildesley, D. J. 1989, *Computer Simulation of Liquids.* Oxford: Clarendon Press. 5, 53, 72, 74, 77, 233, 235

[23] Andersen, H. C., 1983, *J. Comput. Phys.*, **52**, 24. 5, 74, 233

[24] Miller, T. F., Eleftheriou, M., Pattnaik, P., Ndirango, A., Newns, D., and Martyna, G. M., 2002, *J. Chem. Phys.*, **116**, 8649. 5, 101, 102

[25] Evans, D. J., and Morriss, G. P., 1984, *Computer Physics Reports*, **1**, 297. 5, 73, 77

[26] Adelman, S. A., and Doll, J. D., 1976, *J. Chem. Phys.*, **64**, 2375. 5, 73, 77

[27] Andersen, H. C., 1979, *J. Chem. Phys.*, **72**, 2384. 5, 73, 77

[28] Berendsen, H. J. C., Postma, J. P. M., van Gunsteren, W. F., DiNola, A., and Haak, J. R., 1984, *J. Chem. Phys.*, **81**, 3684. 5, 73, 77, 85

[29] Hoover, W. G., 1985, *Phys. Rev.*, **A31**, 1695. 5, 73, 77, 81, 83, 85

[30] Quigley, D., and Probert, M. I. J., 2004, *J. Chem Phys.*, **120**, 11432. 5, 73, 85, 86

[31] Martyna, G. M., Tuckerman, M. E., Tobias, D. J., and Klein, M. L., 1996, *Molec. Phys.*, **87**, 1117. 5, 73, 85, 96, 102

[32] Warner, H. R. J., 1972, *ind. Eng. Chem. Fundam.*, **11**, 379. 15, 193

[33] Bird, R. B. e. a. 1977, *Dynamics of Polymeric Liquids*, volume 1 and 2. Wiley, New York. 15, 193

[34] Grest, G. S., and Kremer, K., 1986, *Phys. Rev. A*, **33**, 3628. 15, 193

[35] Allinger, N. L., Yuh, Y. H., and Lii, J.-H., 1998, *J. Am. Chem. Soc.*, **111**, 8551. 15, 16, 18, 19, 20, 193, 194

[36] Vessal, B., 1994, *J. Non-Cryst. Solids*, **177**, 103. 17, 19, 48, 194, 203

[37] Smith, W., Greaves, G. N., and Gillan, M. J., 1995, *J. Chem. Phys.*, **103**, 3091. 17, 19, 48, 194, 203

[38] Sun, H., 1998, *J. Phys. Chem. B*, **102(38)**, 7338–7364. 18, 20, 194

[39] Kumagai, N., Kawamura, K., and Yokokawa, T., 1994, *Mol. Simul.*, **12(3-9)**, 177. 18, 20

[40] Smith, W., 1993, *CCP5 Information Quarterly*, **39**, 14. 19, 22, 26

[41] Ryckaert, J. P., and Bellemans, A., 1975, *Chem. Phys. Lett.*, **30**, 123. 20, 23, 195

[42] Schmidt, M. E., Shin, S., and Rice, S. A., 1996, *J. Chem. Phys.*, **104**, 2101. 21, 23, 195

[43] Rohl, A. L., Wright, K., and Gale, J. D., 2003, *Amer. Mineralogist*, **88**, 921. 26

[44] Raiteri, P., and Gale, J. D., 2010, *J. Am. Chem. Soc.*, **132**, 17623–17634. 26

[45] Barrat, J.-L., and Bocquet, L., 1999, *Phys. Rev. Lett.*, **82**(23), 4671–4674. 29, 198

[46] Wang, X., Ramirez-Hinestrosa, S., Dobnikar, J., and Frenkel, D., 2020, *Phys. Chem. Chem. Phys.*, **22**, 10624–10633. 29, 198

[47] Mie, G., 1903, *Annalen der Physik*, **11**, 657–697. 29, 198

[48] Clarke, J. H. R., Smith, W., and Woodcock, L. V., 1986, *J. Chem. Phys.*, **84**, 2290. 29, 198

[49] Weeks, J. D., Chandler, D., and Andersen, H. C., 1971, *J. Chem. Phys.*, **54**, 5237. 30, 198

[50] Groot, R. D., and Warren, P. B., 1997, *J. Chem. Phys.*, **107(11)**, 4423–4435. 30, 77, 198, 251, 253

[51] Ponder, J. W., Wu, C., Ren, P., Pande, V. S., Chodera, J. D., Schnieders, M. J., Haque, I., Mobley, D. L., Lambrecht, D. S., DiStasio Jr., R. A., Head-Gordon, M., Clark, G. N. I., Johnson, M. E., and Head-Gordon, T., 2010, *J. Phys. Chem. B*, **114**, 2549–2564. 30, 194, 198

[52] Pedone, A., Malavasi, G., Menziani, M. C., Cormack, A. N., and Segre, U., 2006, *The Journal of Physical Chemistry B*, **110**(24), 11780–11795. 30

[53] Ziegler, J. F., Biersack, J. P., and Littmark, U. 1985, *The Stopping and Range of Ions in Matter.* Pergamon, New York. 30

[54] Trachenko, K., Dove, M. T., and Salje, E. K. H., 2003, *Journal of Physics: Condensed Matter*, **15**(37), 6457. 31

[55] Raiteri, P., Gale, J. D., Quigley, D., and Rodger, P. M., 2010, *The Journal of Physical Chemistry C*, **114**(13), 5997–6010. 31

[56] Al-Matar, A. K., and Rockstraw, D. A., 2004, *J. Comput. Chem.*, **25**, 660–668. 33

[57] Hepburn, D. J., and Ackland, G. J., 2008, *Phys. Rev. B*, **78**(16), 165115. 33, 43

[58] Lau, T. T., Först, C. J., Lin, X., Gale, J. D., Yip, S., and Vliet, K. J. V., 2007, *Phys. Rev. Lett.*, **98**(21), 215501. 33, 43

[59] Cooper, M. W. D., Rushton, M. J. D., and Grimes, R. W., 2014, *J. Phys.: Condens. Matter*, **26**, 105401. 33, 35

[60] J., F., 1952, *Philos. Mag.*, **43**, 153. 34

[61] Ackland, G. J., and Reed, S. K., 2003, *Phys. Rev. B*, **67**, 1741081–1741089. 34

[62] Olsson, P., Wallenius, J., Domain, C., Nordlund, K., and Malerba, L., 2005, *Phys. Rev. B*, **72**, 2141191–2141196. 34

[63] Dai, X. D., Kong, Y., Li, J. H., and Liu, B. X., 2006, *J. Phys.: Condens. Matter*, **18**, 4527–4542. 35

[64] Cleri, F., and Rosato, F., 1993, *Phys. Rev. B*, **48**, 22. 35

[65] Johnson, R. A., 1989, *Phys. Rev. B*, **39**, 12556. 43

[66] Kumagai, T., Izumi, S., Hara, S., and Sakai, S., 2007, *Comput. Mat. Sci.*, **39**, 457. 44

[67] Eastwood, J. W., Hockney, R. W., and Lawrence, D. N., 1980, *Comput. Phys. Commun.*, **19**, 215. 47, 48, 49

[68] Fennell, C. J., and Gezelter, D. J., 2006, *J. Chem. Phys.*, **124**, 234104. 51, 53

[69] Neumann, M., 1985, *J. Chem. Phys.*, **82**, 5663. 52

[70] Fuchs, K., 1935, *Proc. R. Soc., A*, **151**, 585. 54, 56

[71] Essmann, U., Perera, L., Berkowitz, M. L., Darden, T., Lee, H., and Pedersen, L. G., 1995, *J. Chem. Phys.*, **103**, 8577. 54, 55, 131, 236

[72] Boateng, H. A., and Todorov, I. T., 2015, *J. Chem. Phys.*, **142**, 034117. 56

[73] Sagui, C., Pedersen, L., and Darden, T., 2004, *J. Chem. Phys.*, **120**, 73. 57

[74] Drude, P., 1900, *Annalen der Physik*, **306**(3), 566. 63

[75] Drude, P., 1900, *Annalen der Physik*, **308**(11), 369. 63

[76] Lemkul, J. A., Huang, J., Roux, B., and MacKerell Jr., A. D., 2016, *ACS Chem. Rev.*, **116**, 4983–5013. 63, 64

[77] Thole, B. T., 1981, *J. Chem. Phys.*, **59**, 341–350. 64, 208

[78] Fincham, D., and Mitchell, P. J., 1993, *J. Phys. Condens. Matter*, **5**, 1031. 64

[79] Lindan, P. J. D., and Gillan, M. J., 1993, *J. Phys. Condens. Matter*, **5**, 1019. 65

[80] Shewchuk, J. R. August 4, 1994, *An Introduction to the Conjugate Gradient Method Without the Agonizing Pain, Edition 1 1/4*. School of Computer Science, Carnegie Mellon University, Pittsburgh, PA 15213. 65, 146

[81] Schroder, U., 1966, *Solid State Commun.*, **4**, 347–349. 65

[82] Torrie, G. M., and Valleau, J. P., 1977, *Journal of Computational Physics*, **23**, 187–199. 67, 69, 205

[83] Kastner, J., 2011, *Wiley Interdisciplinary Reviews: Computational Molecular Science*, **1**, 932–942. 67, 69, 205

[84] Shardlow, T., 2003, *SIAM Journal on Scientific Computing*, **24**(4), 1267–1282. 73, 77, 84

[85] Duffy, D. M., and Rutherford, A. M., 2007, *J. Phys.: Condens. Matter*, **19**(1), 016207. 73, 77, 79, 112

[86] Leimkuhler, B., Noorizadeh, E., and Theil, F., 2009, *J. Stat. Phys.*, **135**, 261–277. 73, 77, 83

[87] Ikeguchi, M., 2004, *J. Comp. Chemi.*, **25**, 529–541. 74, 89, 91, 95, 96, 98

[88] Ryckaert, J. P., Ciccotti, G., and Berendsen, H. J. C., 1977, *J. Comput. Phys.*, **23**, 327. 74, 233

[89] McCammon, J. A., and Harvey, S. C. 1987, *Dynamics of Proteins and Nucleic Acids*. Cambridge: University Press. 76

[90] Izaguirre, J. A. Langevin stabilisation of multiscale mollified dynamics. In Brandt A., Binder K., B. J., editor, *Multiscale Computational Methods in Chemistry and Physics*, volume 117 of *NATO Science Series: Series III - Computer and System Sciences*, pages 34–47. IOS Press, Amsterdam, 2001. 77, 79

[91] Samoletov, A., Chaplain, M. A. J., and Dettmann, C. P., 2007, *J. Stat. Phys.*, **128**, 1321–1336. 77, 83

[92] Hoogerbrugge, P. J., and Koelman, J. M. V. A., 1992, *EPL (Europhysics Letters)*, **19**(3), 155–160. 77, 251

[93] Español, P., and Warren, P., 1995, *EPL (Europhysics Letters)*, **30**(4), 191–196. 77, 251

[94] Melchionna, S., Ciccotti, G., and Holian, B. L., 1993, *Molec. Phys.*, **78**, 533. 92

[95] Martyna, G. M., Tobias, D. J., and Klein, M. L., 1994, *J. Chem. Phys.*, **101**, 4177. 92, 99

[96] Fincham, D., 1992, *Molecular Simulation*, **8**, 165. 101

[97] Lifshits, I., Kaganov, M., and Tanatarov, L., 1960, *J. Nucl. Energy A*, **12**(1), 69–78. 112

[98] Duffy, D., Khakshouri, S., and Rutherford, A., 2009, *Nucl. Instrum. Meth. B*, **267**(18), 3050. 112

[99] Zarkadoula, E., Daraszewicz, S. L., Duffy, D. M., Seaton, M. A., Todorov, I. T., Nordlund, K., Dove, M. T., and Trachenko, K., 2014, *J. Phys.: Condens. Matter*, **26**(8), 085401. 112, 249

[100] Daraszewicz, S. L., Giret, Y., Naruse, N., Murooka, Y., Yang, J., Duffy, D. M., Shluger, A. L., and Tanimura, K., 2013, *Phys. Rev. B*, **88**(18), 184101. 112

[101] Khara, G. S., Murphy, S. T., Daraszewicz, S. L., and Duffy, D. M., 2016, *J. Phys.: Condens. Matter*, **28**(39), 395201. 112, 250

[102] Zarkadoula, E., Daraszewicz, S. L., Duffy, D. M., Seaton, M. A., Todorov, I. T., Nordlund, K., Dove, M. T., and Trachenko, K., 2013, *J. Phys.: Condens. Matter*, **252**(12), 125402. 117

[103] Seaton, M., Anderson, R., Metz, S., and Smith, W., 2013, *Molecular Simulation*, **39**(10), 796–821. 119, 254

[104] Duarte, F., and Kamerlin, S. C. L. *Theory and Applications of the Empirical Valence Bond Approach*, chapter 2, pages 27–61. 2017. 129

[105] Scivetti, I., Sen, K., Elena, A. M., and Todorov, I., 2020, *The Journal of Physical Chemistry A*, **124**(37), 7585–7597. 129, 131, 132, 133, 135, 136

[106] Mones, L., Kulhnek, P., Simon, I., Laio, A., and Fuxreiter, M., 2009, *The Journal of Physical Chemistry B*, **113**(22), 7867–7873. 130

[107] Todorov, I. T., Bush, I. J., and Porter, A. R., 2009, *Parallel Scientific Computing and Optimization. Springer Optimization and Its Applications*, **27**. 145

[108] Todorov, I. T., Bush, I. J., and Smith, W., 2008, *Cray User Group 2008*. 145

[109] Bush, I. J., Todorov, I. T., and Smith, W., 2010, *Cray User Group 2010*. 145

[110] Diver, A., Dicks, O., Elena, A. M., Todorov, I. T., and Trachenko, K., 2020, *Journal of Physics: Condensed Matter*, **32**(41), 415703. 231

[111] Hockney, R. W., and Eastwood, J. W. 1981, *Computer Simulation Using Particles.* McGraw-Hill International. 233, 235, 236

[112] Smith, W., 1992, *Comput. Phys. Commun.*, **67**, 392. 233

[113] Smith, W., and Fincham, D., 1993, *Molecular Simulation*, **10**, 67. 233

[114] Bush, I. J., Todorov, I. T., and Smith, W., 2006, *Computer Physics Communications*, **175**, 323. 236

[115] Bush, I. J., 2000, *Daresbury Laboratory*. 236

[116] Reith, D., Pütz, M., and Mü, ller-Plathe, F., 2003, *J. Comp. Chem.*, **24**, 1624. 249

[117] Rühle, V., Junghans, C., Lukyanov, A., Kremer, K., and Andrienko, D., 2009, *J. Chem. Theory Comput.*, **5**, 3211. 249

[118] Malerba, L., Marinica, M. C., Anento, N., Björkas, C., Nguyen, H., Domain, C., Djurabekova, F., Olsson, P., Nordlund, K., Serra, A., Terentyev, D., Willaime, F., and Becquart, C. S., 2010, *Journal of Nuclear Materials*, **406**(1), 19–38. 249

[119] Murphy, S. T., Daraszewicz, S. L., Giret, Y., Watkins, M., Shluger, A. L., Tanimura, K., and Duffy, D. M., 2015, *Physical Review B*, **92**, 134110. 250

[120] Pagonabarraga, I., and Frenkel, D., 2001, *Journal of Chemical Physics*, **115**(11), 5015–5026. 253

[121] Trofimov, S. Y., Nies, E. L. F., and Michels, M. A. J., 2002, *Journal of Chemical Physics*, **117**(20), 9383–9394. 253

[122] Koelman, J. M. V. A., and Hoogerbrugge, P. J., 1993, *EPL (Europhysics Letters)*, **21**(3), 363–368. 253

[123] Mar, C. A., Backx, G., and Ernst, M. H., 1997, *Physical Review E*, **56**(2), 1676–1691. 253

[124] Peters, E. A. J. F., 2004, *EPL (Europhysics Letters)*, **66**(3), 311–317. 254

[125] Lowe, C. P., 1999, *EPL (Europhysics Letters)*, **47**(2), 145–151. 254

[126] Stoyanov, S. D., and Groot, R. D., 2005, *Journal of Chemical Physics*, **122**(11), 114112. 254

[127] Gonella, G., Orlandini, E., and Yeomans, J. M., 1997, *Phys. Rev. Lett.*, **78**, 1695. 254

Index