DL_MESO USER MANUAL

M. A. Seaton and W. Smith

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

Version 2.7 rev 08, March 2020

Contents

Acknowledgements

DL_MESO 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 members of the CCP5 DL_MESO consortium were:

David M. Heyes, University of Surrey

Chris M. Care, Sheffield Hallam University

Peter V. Coveney, University College London

David Emerson, UKRI STFC Daresbury Laboratory

Rob English, North East Wales Institute

Andrea Ferrante, Novidec

Ian Halliday, Sheffield Hallam University

John Harding, University of Sheffield

Sebastian Reich, Imperial College

Bill Smith, UKRI STFC Daresbury Laboratory

Patrick B. Warren, Unilever Port Sunlight

Julia Yeomans, Oxford University

Many other people have given advice and encouragement in the development of DL_MESO. We gratefully acknowledge the support of the following people: Maurice Leslie, Richard Wain, Alexandre Dupuis, Jonathan Chin, Michael Dupin, Weiming Liu, John Purton, Ilian Todorov, David Bray, Annalaura Del Regno, Olga Lobanova, Antoine Schlijper, Andrea Ferrante, Massimo Noro, Ian Stott, Neil George, John Hone, Peter Dowding, Kai Luo and the UK Consortium for Mesoscale Engineering Sciences (UKCOMES), Luke Mason, Sergi Siso and Terry Hewitt.

Particular thanks go to Rongshan Qin at The Open University as the original author of the Lattice Boltzmann Equation source code (DL_MESO_LBE) and the DL_MESO graphical user interface, Jianping Meng at UKRI STFC Daresbury Laboratory for his contributions to DL_MESO_LBE, Richard Anderson at UKRI STFC Daresbury Laboratory, Ard van Bergen at Novidec and Bill Swope at IBM for their contributions to the Dissipative Particle Dynamics source code (DL_MESO_DPD), and Michael Johnston and Leopold Grinberg at IBM for their extensive optimisation work on both codes.

Chapter 1

DL_MESO General Information

1.1 Description

DL_MESO is a general purpose mesoscopic simulation package developed at Daresbury Laboratory by Dr Michael Seaton 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 package is the property of the UKRI Science and Technology Facilities Council (STFC).

DL_MESO is issued free under licence to academic institutions pursuing scientific research of a non-commercial nature. All recipients of the code must first agree to the terms and conditions of the licence and register with us to be kept aware of new developments and discovered bugs. Commercial organisations interested in acquiring the package should approach the Scientific Computing Department, UKRI STFC Daresbury Laboratory in the first instance. Daresbury Laboratory is the sole centre for distribution of the package. Under no account is it to be redistributed to third parties without consent of the owners.

DL_MESO contains two mesoscale simulation methods:

- Lattice Boltzmann Equation (included with version 1.0 and later)
- Dissipative Particle Dynamics (included with version 2.0 and later)

1.2 Functionality

The following is a list of the features that DL_MESO currently supports. 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.

1.2.1 Lattice Boltzmann Equation

DL_MESO_LBE can simulate lattice-gas systems using the Lattice Boltzmann Equation (LBE). The following properties and features are currently available:

- Multiple fluid components, solutes and coupled heat transfers[144]
- Collisions: Bhatnagar-Gross-Krook (BGK) single-relaxation-time[7], Two-Relaxation-Time (TRT)[39], Multiple-Relaxation-Time (MRT)[68, 23, 124] or cascaded LBE (CLBE)[32, 30]
- Forcing methods: Martys/Chen[84], Equal Difference Method (EDM)[65], Guo[46], He[50]
- Rheological models: Newtonian, power law, Bingham plastic[8], Herschel-Bulkley plastic[51], Casson[16], Carreau-Yasuda[14, 142]

- Boundary conditions: Periodic, bounce-back (including stationary objects), constant pressure/velocity at planar surfaces[148, 59, 69, 3]
- Mesoscale interactions: Shan-Chen pseudopotential method[108, 109], Lishchuk continuum-based method[77], Swift free-energy method[126, 125]
- Initial conditions can either be determined by DL_MESO_LBE or specified by the user

1.2.2 Dissipative Particle Dynamics

DL_MESO_DPD can model DPD particles ('beads') with soft or hard potential fields, along with thermostatting dissipative and random forces. The following properties and features are currently available:

- Choice of integrators/thermostats: standard Velocity Verlet, DPD Velocity Verlet[37], Lowe-Andersen[79], Peters[92] and Stoyanov-Groot[121]
- Constant volume (NVT) or constant pressure (NPT) simulations with Berendsen[6] or Langevin[60] barostats
- User selection of interaction lengths, conservative and dissipative force parameters for each species and between unlike species
- Bond stretching, angles and dihedrals between beads in user-defined 'molecules'
- Potentials: standard Groot-Warren DPD[43], density-dependent (many-body) DPD[89, 130], Lennard-Jones[62], Weeks-Chandler-Andersen[137]
- Electrostatic potentials between charged beads using modified Ewald summations[41, 136], optionally using Smooth Particle Mesh Ewald[28]
- Boundaries: Periodic, hard reflecting walls with optional short-range repulsions (DPD[97] or Weeks-Chandler-Andersen), frozen particle walls, Lees-Edwards periodic shearing boundaries[71]
- Initial conditions can either be determined by DL_MESO_DPD or specified by the user

1.2.3 Backwards compatibility with previous versions of DL_MESO

Some of the features in the LBE and DPD codes are additions to those from previous versions of DL_MESO, and minor changes have thus been made to input file formats to accommodate them. If any pre-existing input files are intended to be used with this version, some small modifications may be required: please consult Appendix A for more details.

The main output file formats for DL_MESO_LBE have not substantially changed, although changes have been made in how these files are written in parallel running to reduce the number produced per time frame. If a single file per frame is requested, MPI-IO is used to share the writing of these files among processor cores.

The binary file formats for DL_MESO_DPD have significantly changed to allow the use of stream I/O instead of Fortran record-based I/O, as well as the application of MPI-IO to write single output files regardless of the number of processor cores used for calculations. As such, previously generated binary output files cannot be used with this version of DL_MESO_DPD and all post-processing utilities that directly read and manipulate these files have had to be changed, but the new formats allow simulation restarts on different numbers of processor cores and typically require less time to read for analysis and visualisation.

1.3 Requirements

1.3.1 Software requirements

- Standard C++ Compiler for LBE source code, DL_MESO_LBE

- Standard Fortran (2003 or later) Compiler for DPD source code, DL_MESO_DPD
- GNU Make (included in standard Unix/Linux distributions; can be installed for Windows)
- Message Passing Interface version 2 (MPI-2) or higher (if parallel execution required)
- JAVA 2 Version 1.4 or higher (if GUI is to be used)

Versions of the codes exist that use Open Multi-Processing (OpenMP) to divide up calculations on each processor core among threads, which require compilers that can link in OpenMP libraries: the majority of recent standard C++ and Fortran compilers are able to do this. For Smooth Particle Mesh Ewald calculations in the DPD code, either the FFTW 3.x or IBM ESSL Fast Fourier Transform (FFT) libraries may be used in place of the internal FFT solver.

1.3.2 System requirements

DL_MESO is designed to work in both serial and parallel running; it can be run on standalone machines, clusters and supercomputers. The code has been tested on Solaris, Windows XP/7, IBM p690+ HPCx, PowerPC 450 Blue Gene/P, PowerPC A2 Blue Gene/Q, Cray XT4/XT6 HECToR, Cray XC30 ARCHER and Intel Xeon E5-2670 (Sandy Bridge) machines.

1.4 The DL_MESO Directory Structure

The supplied version of DL_MESO is a zip file **dl_meso_2.x**, where x is a generation number: this unpacks as a directory **dl_meso**. Beneath the top level of this directory are a number of subdirectories:

- **LBE** - containing the LBE source code
- **DPD** - containing the DPD source code
- **JAVA** - containing the GUI source code
- **MAN** - containing the DL_MESO user manual
- **DEMO** - containing test cases for DL_MESO
- **WORK** - an example 'working directory'

1.5 Disclaimer

Neither UKRI STFC, CCP5 nor any of the authors of the DL_MESO package guarantee that the package is free from error. Neither do they accept responsibility for any loss or damage that results from its use.

1.6 Copyright

1.7 Authors

Dr Michael Seaton and Prof. William Smith
Scientific Computing Department
UKRI STFC Daresbury Laboratory
Sci-Tech Daresbury
Warrington
WA4 4AD
United Kingdom

1.8 Suggestions and Bug Reports

We encourage users to send suggestions for improvements and new features for DL_MESO, including bug reports and subroutines, as well as any additional test cases that demonstrate its features. All of these should be sent to `michael.seaton@stfc.ac.uk`

Chapter 2

Quickstart guide to compile and run DL_MESO

This chapter is a brief guide on how to get started with DL_MESO: how to download, compile and run the codes supplied in this package. More experienced users can jump to Appendix B for further details on how to compile DL_MESO_LBE, DL_MESO_DPD and their associated utilities, but the following will allow first-time users to get started.

2.1 Obtaining DL_MESO

DL_MESO is exclusively distributed by UKRI STFC Daresbury Laboratory and users of DL_MESO must obtain a licence. Academic users can obtain one for free by registering: to do so, visit the DL_MESO website at `www.ccp5.ac.uk/DL_MESO` and follow the link 'Registering for the DL_MESO Package', which has a link to a Registration Form that needs to be filled out and submitted – a correct email address needs to be supplied, as the user will receive instructions on how to download the package by automated email. If the user intends to use DL_MESO for non-academic commercial purposes, a commercial licence will be required: please contact Dr M A Seaton at `michael.seaton@stfc.ac.uk` in the first instance.

2.2 DL_MESO_LBE

To compile DL_MESO_LBE with OpenMP and a reasonable level of optimization, enter the directory **dl_meso/WORK** in a terminal window and type one of the following two commands:

- `c++ -O3 -openmp ../LBE/slbe.cpp -o lbe.exe`
- `mpicxx -O3 -openmp ../LBE/plbe.cpp -o lbe.exe`

where `c++` should be replaced with the installed C++ compiler, `mpicxx` with the C++ compiler wrapped by the installed MPI implementation, `-openmp` with the compiler flag required to invoke OpenMP and `lbe.exe` with the user's choice of executable name. The first command will compile the serial (single core) version of DL_MESO_LBE, while the second will produce the parallel (multiple core) version.

Running DL_MESO_LBE requires at least two input files: `lbin.sys` for system/simulation properties and `lbin.spa` for boundary conditions. (See Chapter 6 for more details of the contents of these files, and the various test cases in **dl_meso/DEMO/LBE** for examples of these.) These files should be in the same directory as the DL_MESO_LBE executable. To run the serial version of the code, either type `lbe.exe` (if using Windows) or `./lbe.exe` (if using Unix, Linux or macOS). The parallel version of the code will need a command (e.g. `mpirun`, `mpiexec`) to specify the number of processor cores required to run the calculation, which will depend on the available MPI installation. For example, to use 8 cores the command might be:

- `mpirun -np 8 ./lbe.exe`

For supercomputers, this command will need to be included in a job batch script: consult the system administrator or the machine's documentation for more details.

2.3 DL_MESO_DPD

DL_MESO_DPD consists of various modules that need to be compiled and linked together to create an executable. To do this, makefiles are available in the **dl_meso/DPD/makefiles** directory for versions of DL_MESO_DPD with and without MPI, with and without OpenMP: these require GNU Make to be installed, which is automatically available in Linux, Unix and macOS. Enter the directory **dl_meso/WORK** in a terminal window and copy one of the makefiles, renaming it as `Makefile`, using one of the following commands (substituting `Makefile-*`, the first command for Windows, the second for Unix, Linux or macOS):

- `copy ..\DPD\makefiles\Makefile-* Makefile`
- `cp ../DPD/makefiles/Makefile-* ./Makefile`

This makefile can be edited to ensure the correct command is used for the installed Fortran compiler (with or without MPI wrapping) in the line beginning `FC=`, the right compiler flags (`FFLAGS=`) and the required executable name (`EXE=`). It can then be invoked with the command `make`. If you do not intend to recompile the code, you can delete the various module and object files afterwards using either `del *.o *.mod` (Windows) or `rm *.o *.mod` (Unix, Linux, macOS).

Running DL_MESO_DPD requires at least two input files: `CONTROL` for the simulation properties and `FIELD` for types of particles and interactions between them. (See Chapter 10 for more details of the contents of these files, and the various test cases in **dl_meso/DEMO/DPD** for examples of these.) These files should be in the same directory as the DL_MESO_DPD executable. Running the DPD code can be carried out in a similar fashion to the LBE code above, substituting `lbe.exe` with DL_MESO_DPD's executable `dpd.exe`.

2.4 Utilities

DL_MESO is supplied with a number of utilities to help create some input files (especially non-standard system configurations) and to analyse or visualise outputs. Some are written in C++ and others are written in Fortran: a makefile is available in the working directory **dl_meso/WORK** to compile them all. Enter the directory **dl_meso/WORK** in a terminal window and type the command `makefile -f Makefile-utils`. (If the Fortran or C++ compilers given in this makefile are not available, these can be modified at the `FC=` and `CC=` lines respectively.)

Running the resulting executable files with the command-line option `-h` will reveal what each of them can do and the various other command-line options available. More details can also be found in Appendix C.

2.5 Further information

- Consult this manual;
- Visit the DL_MESO website at `www.ccp5.ac.uk/DL_MESO`, especially 'DL_MESO Useful Links';
- Attend a DL_Software training workshop (look out for INFOMAIL mailshots);
- Email Dr Michael Seaton at `michael.seaton@stfc.ac.uk` for one-on-one training, scientific collaboration and to contribute code/scripts for future releases.

Best of luck!

Chapter 3

The DL_MESO GUI

3.1 Getting Started with the DL_MESO GUI

The DL_MESO GUI offers a convenient way of using the DL_MESO package, although it is not an essential tool for those who prefer command line operation: Appendix B provides details on compiling the DL_MESO program codes manually. Working with the GUI requires the availability of Java tools, particularly the `javac` compiler and the `java` runner for Java 2 version 1.4 or later. These may be obtained from the `java.sun.com` website.

To build the GUI, proceed as follows:

- Enter the `DL_MESO/JAVA` directory.
- Type `javac *.java` to compile the source code.
- Type `jar -cfm GUI.jar manifest.mf *.class` to create the `GUI.jar` executable `JAR` file.
- Move to your working directory.
- Launch the GUI.

A Unix/Linux script called `makegui` that performs the build of the GUI can be found in the `JAVA` subdirectory.

Your working directory is the directory from which you wish to work when running DL_MESO. Working there will keep any files you generate separate from the DL_MESO source files. **Note** in the current version of DL_MESO the working directory should be at the same directory level as the `JAVA` direction, i.e. within the `DL_MESO` top directory, and contain the executables of any external utilities required to set up input files and gather or process output files from simulations. An example of such a working directory (called `WORK`) is present under the `DL_MESO` top directory; this includes a makefile to compile all of the external utilties which can be invoked by the command `make -f Makefile-utils`.

In your working directory you can start the GUI with the command

- `java -jar ../JAVA/GUI.jar`

You may consider saving this command in a script for simple execution. An example script for Unix/Linux/-macOS called `rungui` is present in the `WORK` subdirectory.

Figure 3.1 shows the DL_MESO GUI when it is started. The main window displays the names of the authors, the copyright message and the detected operating system (Windows, macOS, Unix/Linux). Clicking the **LBE** and **DPD** buttons will produce the Lattice Boltzmann and Dissipative Particle Dynamics panels respectively, which will guide you through setting up input files, modifying and compiling the program code, running the simulation and gathering the results files for plotting and visualization. The **SPH** button is for Smoothed

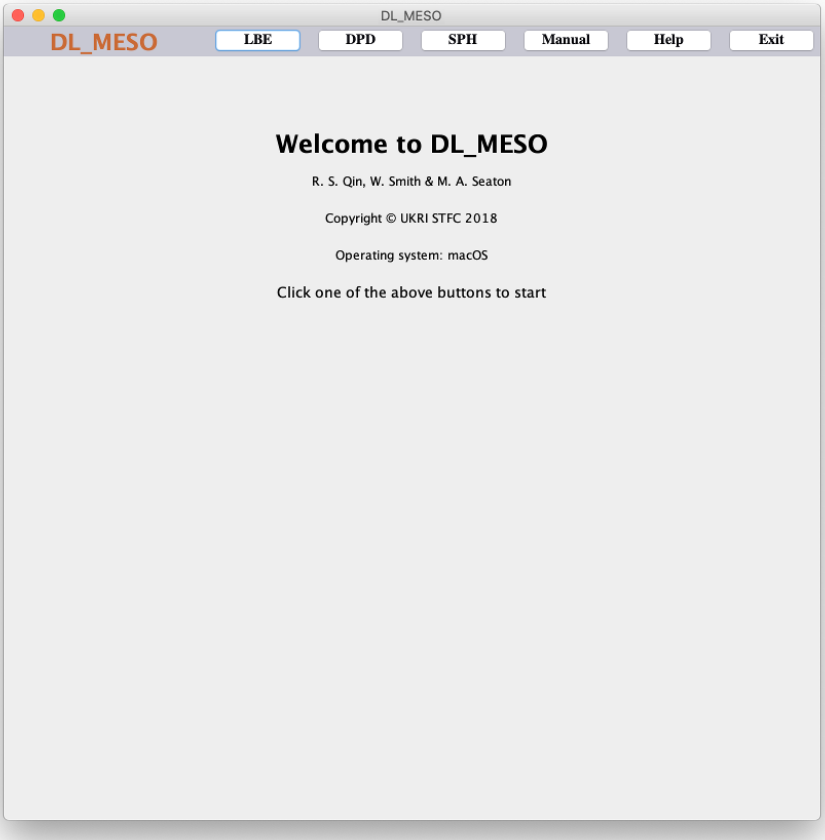


Figure 3.1: DL_MESO GUI on startup

Particle Hydrodynamic simulations, which will be included in future versions of DL_MESO: clicking on this button will currently produce a warning message. This user manual can be read in Adobe Acrobat Reader (if installed) by clicking the **Manual** button, while **Help** will advise you to visit the DL_MESO website at `www.ccp5.ac.uk/DL_MESO`.

3.2 Lattice Boltzmann and the DL_MESO GUI

To access the LBE facilities in the DL_MESO GUI, proceed as follows:

- Click the **LBE** button to get the LBE panel.
- Click the **Define LBE System** button and supply the required information. The file `lbin.sys` will be created by the step.
- Click the **Set LBE Space** button to define the simulation space. The file `lbin.spa` will be created by this step.

3.2.1 Defining the System

Figure 3.2 shows the Define LBE System panel. The required data are as follows:

1. The required **LBE model** can be selected from the pull-down list: the D2Q9, D3Q15, D3Q19 and D3Q27 square lattice schemes are available. The tickbox can be selected to specify that the fluids in the system should be treated as **incompressible**.

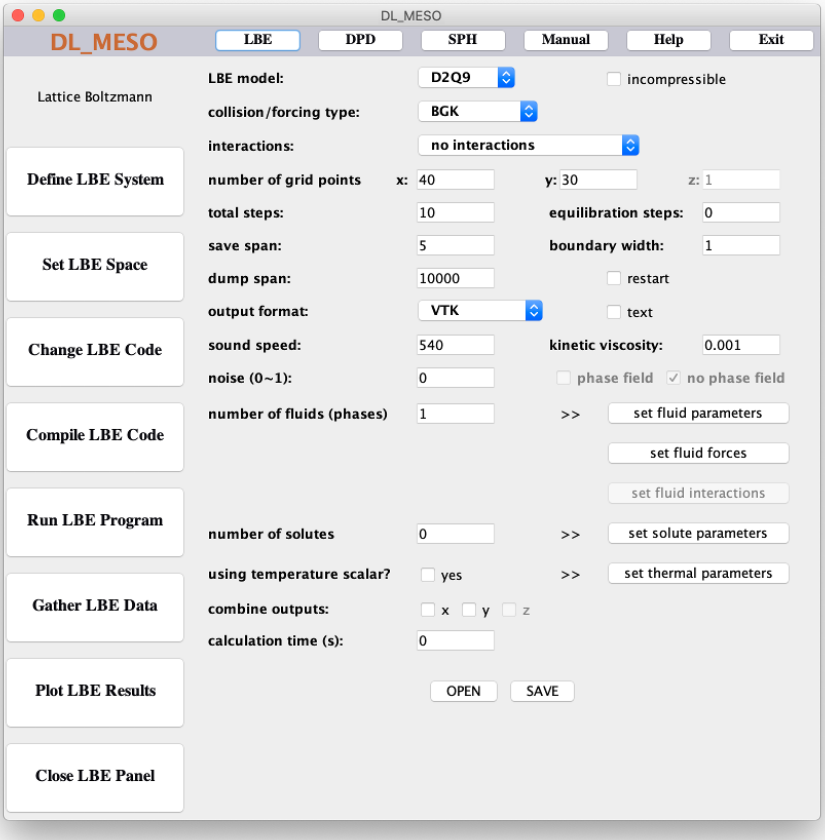


Figure 3.2: Define LBE System

2. The **collision/forcing type** for the system can be selected out of BGK, TRT, MRT and CLBE, each with standard, EDM, Guo or He forcing.

3. The required mesophase **interactions** can be selected from the pull-down list: currently available options include no interactions, Shan/Chen pseudopotential interactions (standard or with quadratic terms), Lishchuk continuum-based interactions (original, Lishchuk-Spencer forces, Lishchuk-Spencer tensors and calculated locally) and Swift free-energy interactions.

4. The **number of grid points** sets the size of the system. For 2D systems, the number of grid points in the z direction must equal 1; selecting a two-dimensional lattice model greys out this box.

5. The **total steps** and the **equilibration steps** for the simulation.

6. The **save span** (number of timesteps between system outputs) and the **boundary width** for running the parallel version of DL_MESO_LBE are given in this row. (The serial version by default automatically resets the boundary width to zero.)

7. The **dump span** (number of timesteps between writing simulation restart files) and the switch for using a **restart** file (`lbout.dump`) to resume a simulation are given in this row. (If the dump span is set to zero, the restart file will only be written at the end of the simulaiton.)

8. The **output format** for system snapshots is set using this pull-down list: VTK, Legacy VTK and Plot3D. By default the output files are written in big endian binary, but ticking the **text** box will make DL_MESO write the output files in text (ANSI) format.

9. The **sound speed** (c) and **kinetic viscosity** (ν) are real-life quantities for the first (main) fluid. These do not influence calculations at all but allow conversions between lattice and real units: the time step and lattice spacing are given by $\Delta t = \frac{\nu}{c^2\left(\tau_f - \frac{1}{2}\right)}$ and $\Delta x = \frac{\sqrt{3}\nu}{c\left(\tau_f - \frac{1}{2}\right)}$ respectively.

10. The **noise** magnitude only has an effect for initializing multiple phase simulations. DL_MESO_LBE may include either a **phase field** parameter or **no phase field** parameter for systems with multiple phases; no mesophase algorithm requires it and thus this option is currently disabled.

11. The **number of fluids (phases)** can be increased if a multiple fluid system is to be studied: up to 6 fluids may be modelled in DL_MESO_LBE. The parameters and boundary conditions for the fluid(s) must then be set by clicking the **set fluid parameters** button – see below for more details. 'item The **set fluid forces** button can be clicked to specify constant, oscillating and Boussinesq forces acting on the fluids – see below for more details.

12. If mesophase interactions are selected, the **set fluid interactions** button can be clicked to specify interaction strengths, equations of state etc. – see below for more details.

13. The **number of solutes** needs changing if solute parameters are required: if the number of solutes is greater than zero (and up to 6), the number of fluids in the above row must be set to 0 or 1. If used, the parameters and boundary conditions for the solutes must be set by clicking the **set solute parameters** button – see below for more details.

14. The **using temperature scalar** box may be clicked **yes** if thermal systems are to be studied. If checked, the thermal parameters must be set by clicking the **set thermal parameters** button – see below for more details.

15. If running the simulation in parallel, the user can ask DL_MESO to **combine outputs** along x, y and/or z dimensions to reduce the number of files written per timestep.

16. The **calculation time** can be set to limit how long DL_MESO will run before terminating: this is particularly useful when running on a system with a job queuing system that limits the total runtime available per calculation, although the user is advised to allow some time to allow a restart file to be written if the simulation terminates before the last specified timestep is reached. (If this value is set to zero, the simulation will continue until the last timestep is reached.)

If a valid `lbin.sys` file already exists in the (current) working directory, the **OPEN** button can be clicked to load its information into the GUI, which can then be viewed and edited. Once all the data in this window and any pop-up windows for fluid, solute and thermal parameters are filled in, the **SAVE** button should be clicked to write the `lbin.sys` file: this button must be pressed if any previously specified values are changed.

3.2.1.1 Fluid, solute and thermal parameters

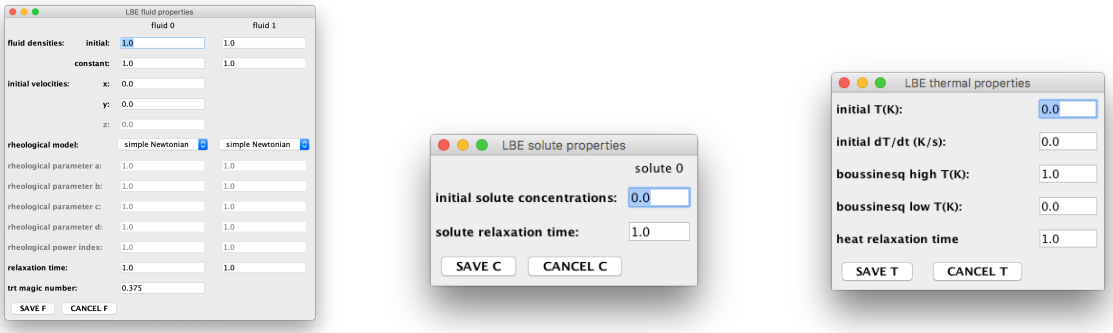
Examples of the pop-up windows for fluid, solute and thermal parameters can be seen in Figure 3.3: multiple columns of dialogue boxes are made available for systems with multiple fluids and/or solutes.

For fluids, the required data are as follows:

1. The **initial** fluid densities are applied throughout the system and used to initialize LBE calculations.

2. The **constant** fluid density (ρ_0) for incompressible systems: this property can also be used to define the reference densities for Shan/Chen pseudopotentials and for initialising systems with fluid drops.

3. The **initial velocities** for all fluids in the system: the z-component will be greyed out if a two-dimensional system is being set up.



(a) Fluid parameters (b) Solute parameters (c) Thermal parameters

Figure 3.3: Fluid, solute and thermal parameter pop-up windows

4. The **rheological model** for each fluid can be specified using the pull-down boxes. These are followed by boxes for each **rheological parameter** (up to four per fluid) and, if required by the rheological model, a **rheological power index**. Details of the available rheological models are given in Section 5.1.2, while Chapter 6 indicates how the parameters for those models are identified in DL_MESO_LBE inputs.

5. The **relaxation time** (τ_f) for each fluid: these values should be greater than 0.5 to give non-zero kinetic viscosities. For rheological models other than simple Newtonian fluids, these will be used as initial values for all grid points before being corrected from measurements of shear rates.

6. For collisions other than BGK, other parameters are required:

   - Two Relaxation Time (TRT) collisions require the **trt magic number** to be specified, which is used to calculate antisymmetric relaxation times.
   - Multiple Relaxation Time (MRT) collisions require the **bulk relaxation time** ($\tau_{f,bulk}$) for each fluid, as well as additional **mrt frequencies** to control higher order moments for all fluids in the system: the number of these frequencies will vary according to the selected lattice scheme.
   - Cascaded Lattice Boltzmann Equation (CLBE) collisions require the **bulk relaxation time**, a **third order relaxation time** ($\tau_{f,3}$) and a **fourth order relaxation time** ($\tau_{f,4}$) for each fluid.

Solutes require the following data in the following rows:

1. The **initial** concentrations of the solutes throughout the system, as used for initialization.
2. The **relaxation time** (τ_s) for each solute, representing diffusivities.

If selected for inclusion, the required thermal properties are:

1. The **initial T** (temperature) for the system.
2. The **initial dT/dt** (rate of change of temperature: related to heat transfers in or out) for the entire system.
3. The **Boussinesq high** reference temperature (T_h) for heat convection in the system.
4. The **Boussinesq low** reference temperature (T_l) for heat convection in the system.
5. The **heat relaxation time** (τ_t) for the system, which represents the thermal diffusivity.

After filling in all the required values, clicking the relevant save button (**SAVE F**, **SAVE C** or **SAVE T**) will store the data in preparation for writing to the `lbin.sys` input file. The cancel buttons (**CANCEL F**, **CANCEL C** and **CANCEL T**) will close the pop-ups without saving any values.

3.2.1.2 Fluid forces

An example of the pop-up window for fluid forces can be seen in Figure 3.4: multiple columns of dialogue boxes are made available for systems with multiple fluids.

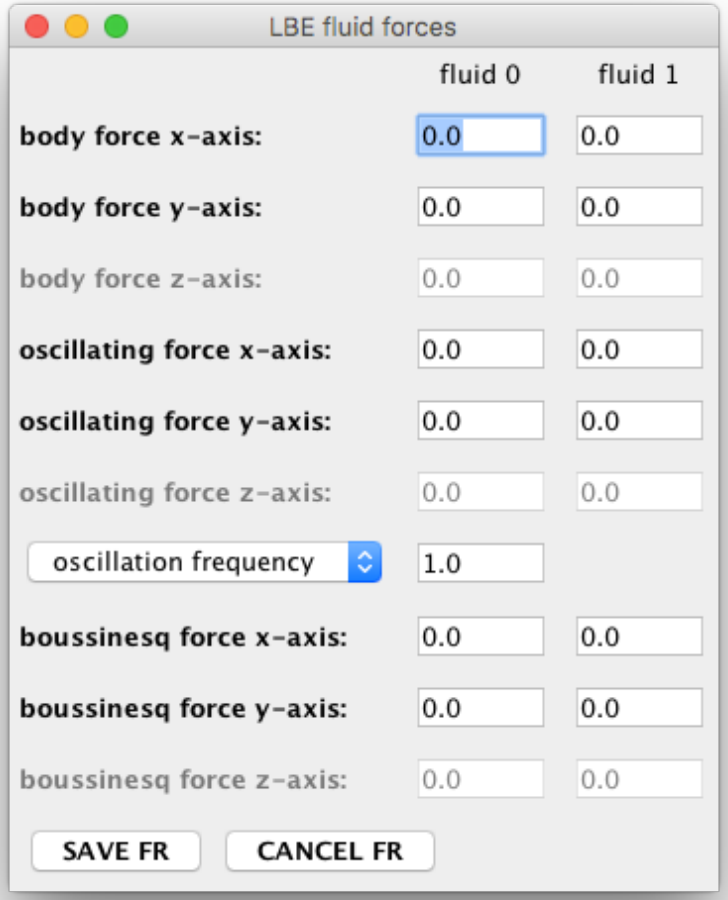


Figure 3.4: Fluid forces pop-up window

The required data are as follows:

1. The **body force x-axis** gives the x-components of constant body forces on each fluid.
2. The **body force y-axis** gives the y-components of constant body forces on each fluid.
3. The **body force z-axis** gives the z-components of constant body forces on each fluid. (Greyed out for two-dimensional systems.)
4. The **oscillating force x-axis** gives the x-components of sinusoidally oscillating forces on each fluid.
5. The **oscillating force y-axis** gives the y-components of sinusoidally oscillating forces on each fluid.
6. The **oscillating force z-axis** gives the z-components of sinusoidally oscillating forces on each fluid. (Greyed out for two-dimensional systems.)
7. Either the **oscillation frequency** or **oscillation period** for sinusoidally oscillating forces can be specified for all fluids: the pull-down box can be used to specify which property the text box includes.

8. The **boussinesq force x-axis** gives the x-component of the Boussinesq force parameter ($\vec{g}\beta$) on each fluid.

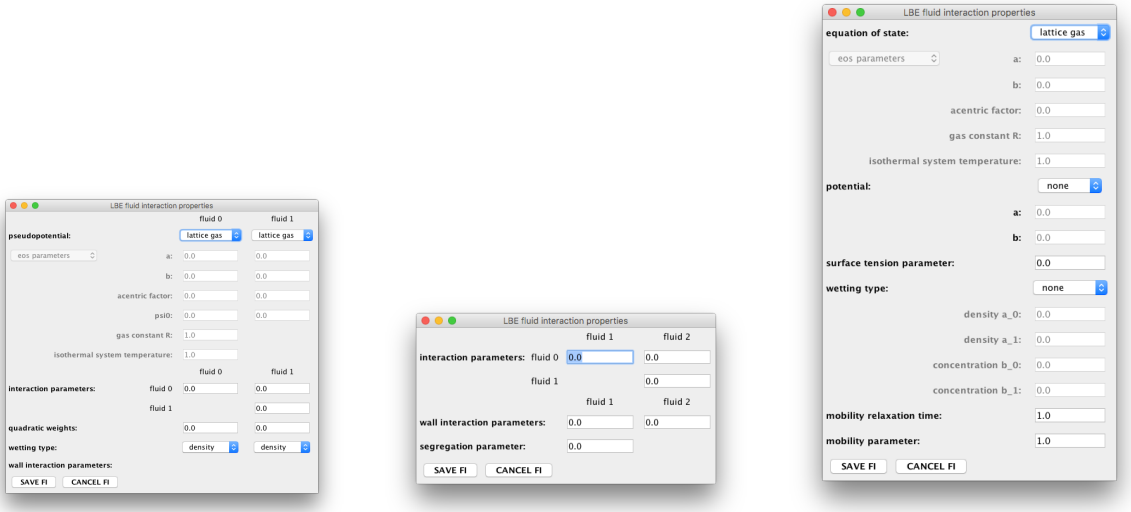
9. The **boussinesq force y-axis** gives the y-component of the Boussinesq force parameter ($\vec{g}\beta$) on each fluid.

10. The **boussinesq force z-axis** gives the z-component of the Boussinesq force parameter ($\vec{g}\beta$) on each fluid. (Greyed out for two-dimensional systems.)

After filling in all the required values, clicking the save button (**SAVE FR**) will store the data in preparation for writing to the `lbin.sys` input file. The cancel button **CANCEL FR** will close the pop-up without saving any values.

3.2.1.3 Fluid interactions

Examples of the pop-up windows for fluid interactions can be seen in Figure 3.5: the form that appears will depend on the type of mesoscale interactions chosen in the main LBE system window. Multiple columns of dialogue boxes are made available for systems with multiple fluids.



(a) Shan/Chen pseudopotential interactions

(b) Lishchuk continuum-based interactions

(c) Swift free-energy interactions

Figure 3.5: Fluid interaction pop-up windows

For Shan/Chen pseudopotential interactions, the required data are as follows:

1. The **pseudopotential** type for each fluid, which provides its equation of state, must be chosen using the pull-down boxes at the top of the window. The available forms include the standard *lattice gas*, the original Shan/Chen pseudopotential[108] (*SC 1993*), the thermodynamically consistent Shan/Chen pseudopotential[109] (*SC 1994*), the Qian pseudopotential[98] (*Qian 1995*), *density* as the pseudopotential, an *ideal gas*, van der Waals (*vdW*), Redlich-Kwong (*RK*), Soave-Redlich-Kwong (*SRK*), Peng-Robinson (*PR*), Carnahan-Starling-van der Waals (*CS-vdW*) and Carnahan-Starling-Redlich-Kwong (*CS-RK*). Each equation of state requires different parameters, and the dialogue boxes below are enabled or disabled accordingly.

2. Either **eos parameters** or **critical properties** can be chosen using the pull-down box to parameterise the pseudopotential (equation of state) for each fluid. If the former, interaction parameters **a** and **b** need to be specified for each fluid; if the latter, the critical temperature (**Tc**) and critical pressure (**Pc**) are needed.

3. Certain equations of state require a value for the **acentric factor** (ω).

4. The thermodynamically consistent Shan/Chen pseudopotential requires a maximum value for the pseudopotential (**psi0**).

5. Most equations of state require values for the **gas constant R**.

6. If no temperature scalar is to be used for simulations, the **isothermal system temperature** needs to be specified for certain equations of state.

7. The **interaction parameters** between fluids (g_{ab}) need to be specified for both $a = b$ and $a \neq b$. If using pseudopotentials for ideal gas, cubic equations of state or Carnahan-Starling equations of state, the values of g_{aa} should be equal to 1.

8. If using Shan/Chen interactions with quadratic terms, the **quadratic weights** can be specified for individual fluids (but not between different fluid species).

9. The **wetting type** for each fluid (*density*, *pseudopotential* or *screened* pseudopotential) can be specified using pull-down boxes and the **wall interaction parameters** for each fluid ($g_{a,wall}$) can be given in the dialogue boxes below. (Note that if no wall interaction is needed, the latter can be set to zero.)

If Lishchuk continuum-based interactions are used, the following data are required:

1. The **interaction parameters** between fluids (g_{ab}) need to be specified for $a \neq b$ (i.e. unlike fluid species).

2. The **wall interaction parameters** with fluids ($g_{wall,a}$) need to be specified for all fluids other than the continuous species (assumed to be fluid 0).

3. The **segregation parameter** (β) is required to ensure immiscible fluids can separate from each other.

Note that at least two fluids are required for Lishchuk interactions to be used: this pop-up window will not open if only one fluid is specified in the main LBE system window.

For Swift free-energy based interactions, the required data are as follows:

1. The **equation of state** for all fluids must be chosen using the pull-down box at the top of the window. The available equations of state include the standard *lattice gas*, the equation of state based on the original Shan/Chen pseudopotential (*SC 1993*), the equationf of state based on the thermodynamically consistent Shan/Chen pseudopotential (*SC 1994*), the equation of state based on Qian's pseudopotential (*Qian 1995*), a *quadratic* equation of state, an *ideal gas*, van der Waals (*vdW*), Redlich-Kwong (*RK*), Soave-Redlich-Kwong (*SRK*), Peng-Robinson (*PR*), Carnahan-Starling-van der Waals (*CS-vdW*) and Carnahan-Starling-Redlich-Kwong (*CS-RK*). Each equation of state requires different parameters, and the dialogue boxes below are enabled or disabled accordingly.

2. If a Shan/Chen or quadratic equation of state is specified, the interaction parameter **g** must be given in the top dialogue box, while the maximum pseudopotential value (**psi0**) should also be given if the thermodynamically consistent Shan/Chen equation of state is used. If a cubic or Carnahan-Starling equation of state is specified, it is possible to either specify the **eos parameters** (**a** and **b**) or the **critical properties** (critical temperature **Tc** and critical pressure **Pc**) based on the pull-down box.

3. Certain equations of state require a value for the **acentric factor** (ω).

4. Most equations of state require values for the **gas constant R**.

5. If no temperature scalar is to be used for simulations, the **isothermal system temperature** needs to be specified for certain equations of state.

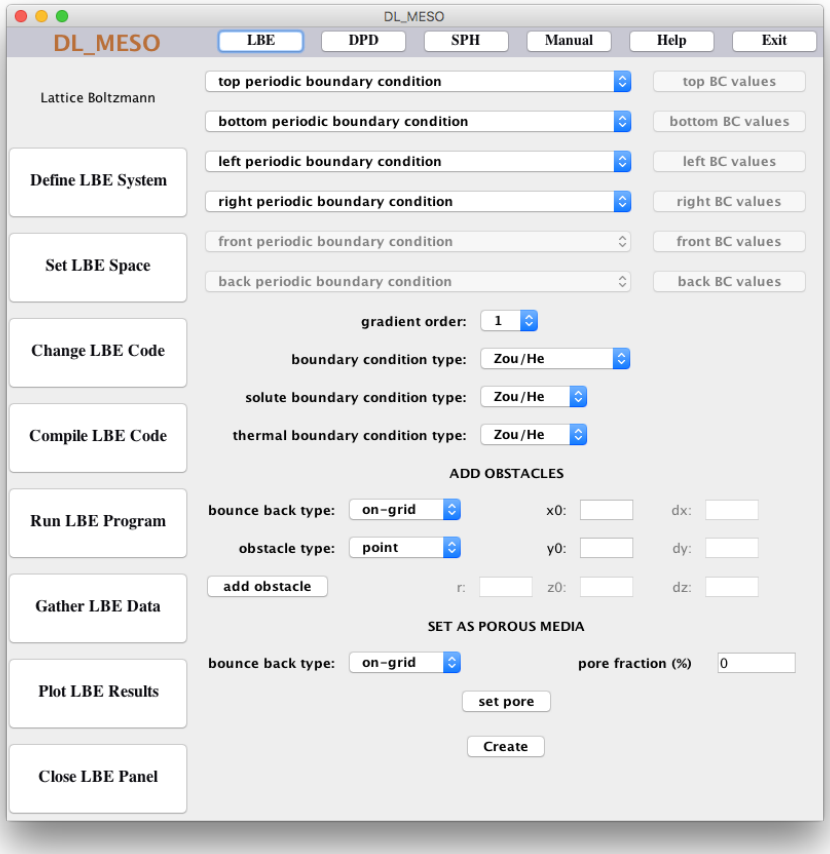


Figure 3.6: Set LBE Space

6. If two fluids are to be used, the **potential** form (*none* or *quartic*) should also be specified using the pull-down box, along with the parameters **a** and **b** as appropriate.

7. The **surface tension parameter** (κ) needs to be defined for both one and two fluid systems.

8. The **wetting type** (*none* or *quadratic*) can be selected using the drop-down box, and the parameters for **density** (**a_0** and **a_1**) can be specified, along with parameters for **concentration** (**b_0** and **b_1**) if two fluids are in use.

9. If the system includes two fluids, both the **mobility relaxation time** (τ_ϕ) and the **mobility parameter** (Γ) need to be specified.

Note that the Swift free-energy schemes are only available for one or two fluids: this pop-up window will not open if more than two fluids are specified in the main LBE system window.

After filling in all the required values, clicking the save button (**SAVE FI**) will store the data in preparation for writing to the `lbin.sys` input file. The cancel button (**CANCEL FI**) will close the pop-ups without saving any values.

3.2.2 Defining the Space Properties

If this option is selected before saving the LBE system data, a warning message advising that the system should be re-defined will appear.

Figure 3.6 shows the Set LBE Space panel. The following data are required:

1. The **top boundary condition** can be selected using the pull-down list from:

- periodic
- on-grid bounce back
- mid-grid bounce back
- outflow
- fixed V (velocity), C (concentration) and T (temperature)
- fixed V and C, Neumann[1] T
- fixed V and T, Neumann C
- fixed V, Neumann C and T
- fixed P (pressure or density), C and T
- fixed P and T, Neumann C
- fixed P and C, Neumann T
- fixed P, Neumann C and T

and values of required properties for this boundary can be specified by clicking **top BC values** to open the pop-up box – see below for more details.

2. The **bottom boundary condition** can be selected using the pull-down list and any relevant values can be specified in the pop-up box obtained by clicking **bottom BC values**.

3. The **left boundary condition** can be selected using the pull-down list and any relevant values can be specified in the pop-up box obtained by clicking **left BC values**.

4. The **right boundary condition** can be selected using the pull-down list and any relevant values can be specified in the pop-up box obtained by clicking **right BC values**.

5. The **front boundary condition** can be selected using the pull-down list and any relevant values can be specified in the pop-up box obtained by clicking **front BC values**. Both the pull-down list and the button will be greyed out for two-dimensional systems.

6. The **back boundary condition** can be selected using the pull-down list and any relevant values can be specified in the pop-up box obtained by clicking **back BC values**. Both the pull-down list and the buttone will be greyed out for two-dimensional systems.

7. The **gradient order** for determining spatial gradients at boundary points can be selected (1 or 2) using the pull-down list.

8. The **boundary condition type** for constant velocity/density boundaries can be specified from Zou/He, Simple Zou/He, Inamuro or regularized using the pull-down list.

9. The **solute boundary condition type** for constant solute concentrations can be specified from Zou/He or Inamuro using the pull-down list.

10. The **thermal boundary condition type** for constant temperatures can be specified from Zou/He or Inamuro using the pull-down list.

11. Solid obstacles can be added to the calculation space by selecting the bounce back (on-grid or mid-grid) and obstacle types in the pull-down lists, entering its location on the grid and, if necessary, entering its size, and clicking **add obstacle**.

    - A single **point** will be located at (**x0**, **y0**, **z0**); **z0** can be omitted for two-dimensional systems.
    - A **sphere** is centred at (**x0**, **y0**, **z0**) and has radius **r**.

[1] For a property ϕ, DL_MESO currently only calculates $\nabla\phi = 0$ by using on-grid bounce back on the related distribution function.

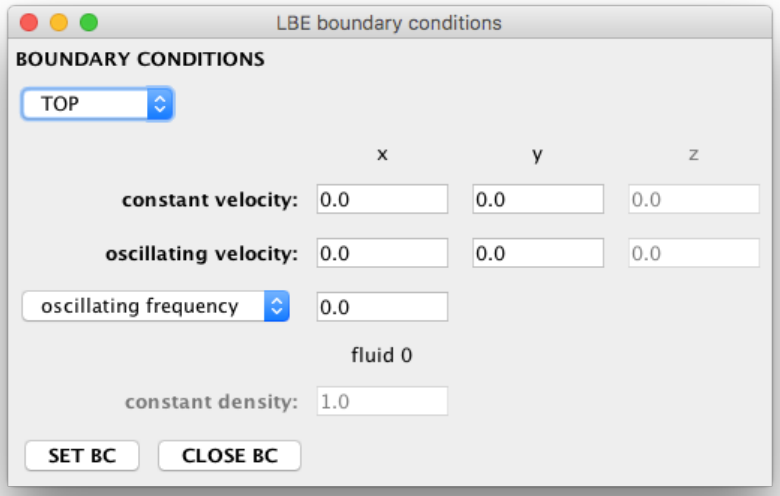


Figure 3.7: Set LBE Boundary Conditions

- A two-dimensional **circle** is centred at (**x0**, **y0**) and has radius **r**.
- A **block** has a vertex at (**x0**, **y0**, **z0**) and has size (**dx**, **dy**, **dz**): both **z0** and **dz** can be omitted for two-dimensional blocks.

Note that lattice points well within an obstacle are set as blank sites, i.e. they will be ignored in LBE calculations.

12. The entire system can be set up as a porous solid by selecting the bounce back type, specifying a **pore fraction** and clicking **set pore** to randomly select an appropriate number of solid lattice sites.

Clicking the **Create** button will write all the lattice space data to a `lbin.spa` file; any lattice point defined more than once will hold its *latest* definition. Values of properties for constant velocity/density/solute concentration/temperature boundaries, the gradient order and the boundary condition types will be appended to the pre-existing `lbin.sys` file at the same time.

3.2.2.1 Boundary condition values

If any of the **BC values** buttons are clicked, the pop-up window shown in Figure 3.7 will appear. Multiple columns will appear based on the number of fluids and/or solutes in the system.

1. The user can switch between **TOP**, **BOTTOM**, **LEFT**. **RIGHT**, **FRONT** and **BACK** boundary conditions using this pull-down box: note that these options will only be available if the user has specified fixed velocity/density boundaries for those system planes in the Set LBE Space panel (see above).

2. If the user has specified a velocity boundary for the given plane, the **constant velocity** at that boundary can be set by specifying x-, y- and z-components. (The z-component will be greyed out if the system is two-dimensional.)

3. If the user has specified a velocity boundary, the amplitude of a sinusoidal **oscillating velocity** at that boundary can be set by specifying x-, y- and z-components. (The z-component will be greyed out if the system is two-dimensional.)

4. Either the **oscillating frequency** or **oscillating period** can be specified for the given boundary: the pull-down box can be used to specify which of these is used.

5. If the user has specified a density boundary, the **constant density** at that boundary can be set for each fluid.

6. If the user has specified a fixed solute boundary and there are solutes in the system, the **constant solute concentration** at that boundary can be set for each solute.

7. If the user has specified a fixed temperature boundary and heat transfers are activated for the system, the **constant temperature** and **rate of temperature change** can be set for that boundary.

Clicking the **SET BC** button will fix the velocities, densities, solute concentrations, temperature and heat transfer rate for the boundary selected in the top pull-down box in memory ready for writing to the `lbin.sys` file. The **CLOSE BC** button will close the pop-up box: this can be used before or after the property values at boundary conditions are set.

3.3 Dissipative Particle Dynamics and the DL_MESO GUI

To access the DPD facilities in the DL_MESO GUI, proceed as follows:

- Click the **DPD** button to get the DPD panel.
- Click the **Define DPD System** button and supply the required information. The `CONTROL` file will be created by this step.
- Note that currently no simulation space settings or molecular structure data can be entered using the GUI.
- Click **EXIT** to finish the settings.

3.3.1 Defining the System

Figure 3.8 shows the Define DPD System panel.

The required data are as follows:

1. The **job header**: a line of text up to 80 characters long describing the simulation.

2. The system **volume**: the pull-down list can be used to specify whether this is cubic or orthogonal, or whether replication of a `CONFIG` file is required (nfold). If specifying a cubic volume, the total volume should be specified, while orthogonal volumes require the sizes for all three dimensions and the nfold setting requires integer values specifying the number of replications in each dimension.

3. The target **temperature** (k_BT) and **pressure** (P_0) for the system. (The latter is greyed out if no barostat is to be used.)

4. The maximum **interaction cutoff** (r_c) for pairwise particle interactions and the **many-body cutoff** (r_d) for determining localized particle densities as used for many-body DPD. If the interaction cutoff is not specified (i.e. if it is set to zero), DL_MESO will use the largest interaction length available from the interaction data specified in the Set DPD Interactions panel.

5. If required, the **electrostatic cutoff** (r_e) for short-range electrostatic interactions and the **surface cutoff** (z_c) for interactions between particles and solid walls. (These are greyed out if not required.)

6. The size of the **boundary halo** for copying particle data from neighbouring subdomains or across periodic boundaries and the size of each **time step** (Δt) for integrating the equations of motion.

7. The **total steps** required for the DPD simulation and the number of time steps required to equilibrate the system (**equilibration steps**).

8. The numbers of time steps to store system variables for rolling averages (**stack interval**) and between rescaling of particle velocities to the desired system temperature during equilibration (**temp scale interval**). The latter can be set to zero if no temperature rescaling is required.

9. The starting time step (**save start**) and the number of time steps between saves (**save interval**) of trajectory data to `HISTORY` files for later visualization. The latter can be set to zero if no trajectory data are required.

10. The **save level** for trajectory data can be specified using the pull-down list: either positions only, positions and velocities, or positions, velocities and forces can be selected. The **random seed** for initializing random number generators can be set with this dialogue box.

11. The numbers of time steps between printing summaries in the `OUTPUT` file (**print interval**) and outputs of statistical data (system energy, potential energies, pressure, temperature etc.) to a plottable `CORREL` file (**plot interval**). The latter value can be set to zero if no plot file is required.

12. The number of time steps between dumps of system configurations to `export` files for simulation restarts (**dump interval**) and the percentage variation in particle density (**density var**) to allow for unevenly distributed systems.

13. The **job time** is the maximum (real) time that can be spent carrying out the DPD simulation: the **close time** gives the time needed to write restart files and shut down the calculation in a controlled manner.

14. The **restart key** for the simulation: this can either be set to **none** for a new simulation, a **full restart** to continue a previous run using `export*` files, a **new run** which takes a starting state (particle positions and velocities) for a new simulation from `export*` files, and **rescaled** does the same as a new run but additionally rescales the particle velocities to give the specified system temperature.

15. The system **thermostat**: the dissipative and random forces as defined for DPD with the standard (molecular dynamics) form of the Velocity Verlet integrator (DPD/MD-VV) is the default, but recalculation of dissipative forces at the end of each step (DPD/DPD-VV)[37], the Lowe-Andersen[79], Peters[92] and Stoyanov-Groot[121] thermostats can also be selected. Values of γ for the DPD and Peters thermostats and Γ for the Lowe-Andersen and Stoyanov-Groot thermostats can be specified elsewhere for each pair of species, but an additional parameter for the Stoyanov-Groot thermostat should be set by clicking on **set thermostat** – see below for more details.

16. The system **barostat**: no barostat is used by default, but Langevin[60] and Berendsen[6] barostats with constant pressure (NPT), constant surface area (NPAT) and constant surface tension (NsT) ensembles are available in combination with all five thermostats. If either barostat is selected, its parameters can be set by clicking on **set barostat** and the target system pressure can be specified.

17. The **electrostatics** scheme for the simulation: the Ewald sum method with Slater-type (exponential)[41] or Gaussian[21] charge smearing schemes are available in DL_MESO_DPD. If selected, the short-range electrostatic cutoff can be edited and the parameters for the Ewald sum and charge smearing can be specified by clicking on **set electrostatics**.

18. The **surfaces** to be applied to the system: by default periodic boundary conditions are used, but alternative boundary conditions include Lees-Edwards shearing periodic boundaries, hard walls with specular reflections[133] or bounce back reflections. The boundaries with the specified condition, the position of hard walls relative to the simulation boundary and vacuum gaps for electrostatics can be selected by clicking on **set surfaces**.

19. The inclusion of at least one **frozen bead wall** can be specified using the tick boxes in this row: this will also active the **set surfaces** button to allow vacuum gaps for electrostatics to be specified.

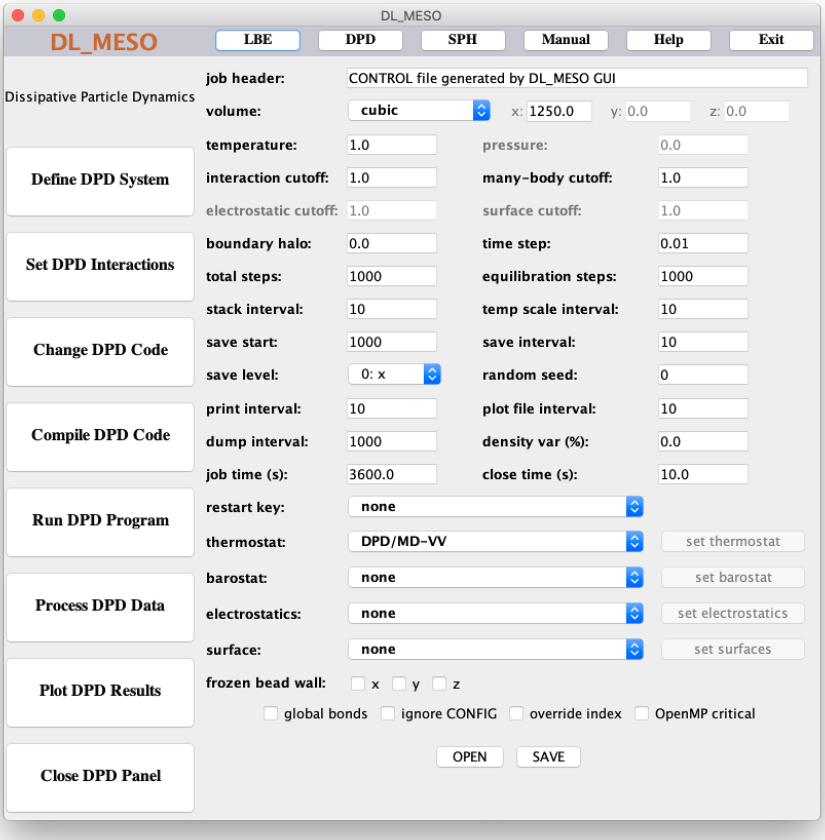


Figure 3.8: Define DPD System

20. Switches to use global storage of bonds (**global bonds**), to **ignore CONFIG** files, to **override index** numbers in a `CONFIG` file and to **OpenMP critical** code sections to assign particle forces can be set using these tickboxes.

Note that the DPD code uses reduced units in which the unit of length is the particle size, the unit of mass is the particle mass and the unit of energy is the primary energy parameter of the potential energy function. From these the time unit may be derived. The temperature is defined to be $\frac{2}{3}$ of the system kinetic energy.

If a valid `CONTROL` file already exists in the (current) working directory, the **OPEN** button can be clicked to load its information into the GUI, which can then be viewed and edited. The `CONTROL` file for input into DL_MESO_DPD is created by clicking the **SAVE** button.

3.3.1.1 Thermostat, barostat, electrostatic and surface parameters

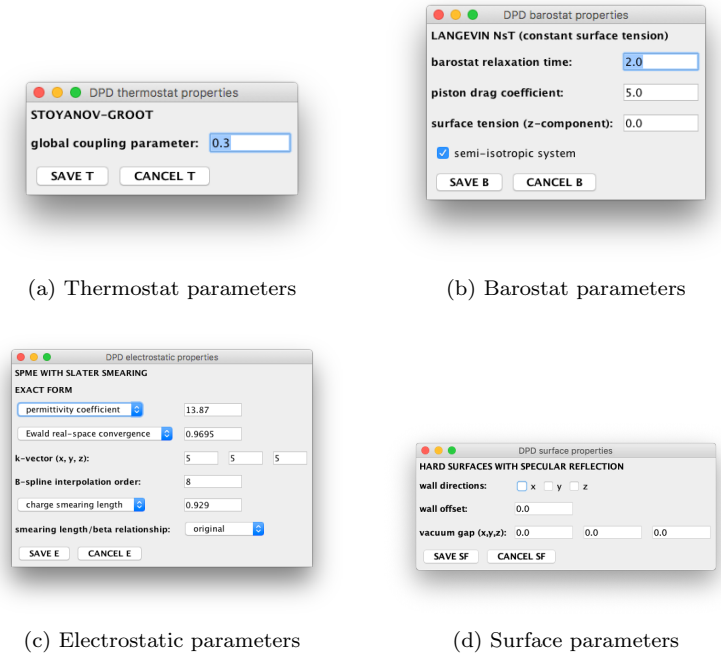
Examples of the pop-up windows for thermostat, barostat, electrostatic and surface parameters can be seen in Figure 3.9: multiple columns of dialogue boxes are made available for systems with multiple species.

The thermostat pop-up window is formatted as in Figure 3.9(a):

1. The type of thermostat to be used in the simulation.

2. Thermostat parameters: for the Stoyanov-Groot thermostat (currently the only type that requires an additional parameter), a **global coupling parameter** for the Nosé-Hoover part (α) is required.

Figure 3.9(b) gives the layout for the barostat pop-up window:



(a) Thermostat parameters

(b) Barostat parameters

(c) Electrostatic parameters

(d) Surface parameters

Figure 3.9: Thermostat, barostat, electrostatic and surface pop-up windows

1. The type of barostat and ensemble to be used in the simulation.
2. Barostat parameters: for the Langevin barostat, a **barostat relaxation time** (τ_p) and **piston drag coefficient** (γ_p) are required, while the Berendsen barostat requires the **compressibility/relaxation** ratio ($\frac{\beta}{\tau_p}$).
3. If a constant surface tension ensemble is used, the **surface tension** parameter is needed: note that it acts only in the z-dimension of the system.
4. This check box determines whether or not an **isotropic system**, i.e. one where pressure acts uniformly in all dimensions, should be modelled. If unchecked, the barostat will act differently in each dimension and the shape of the system will change over time. The check box will only appear for constant surface tension (NsT) ensembles.

The parameters for electrostatics can be given in the pop-up window shown in Figure 3.9(c):

1. The type of electrostatics to be used in the simulation.
2. Electrostatic parameters: either the **permittivity coefficient** (Γ) or **bjerrum length** (l_B) needs to be specified.
3. Either the **Ewald real-space convergence** (α) coefficient and maximum **k-vector** should be specified, or alternatively the **relative error in Ewald sum** can be used to obtain the former two properties for the system.
4. Charge smearing schemes either require the **charge smearing length** or **charge smearing beta** to be specified: the **smearing length/beta relationship** can also be supplied to indicate the connection between the two properties.

If non-periodic boundaries are to be used, the parameters for surfaces can be entered in the appropriate pop-up window (Figure 3.9(d)):

1. The type of surface interactions or boundary conditions to be applied.

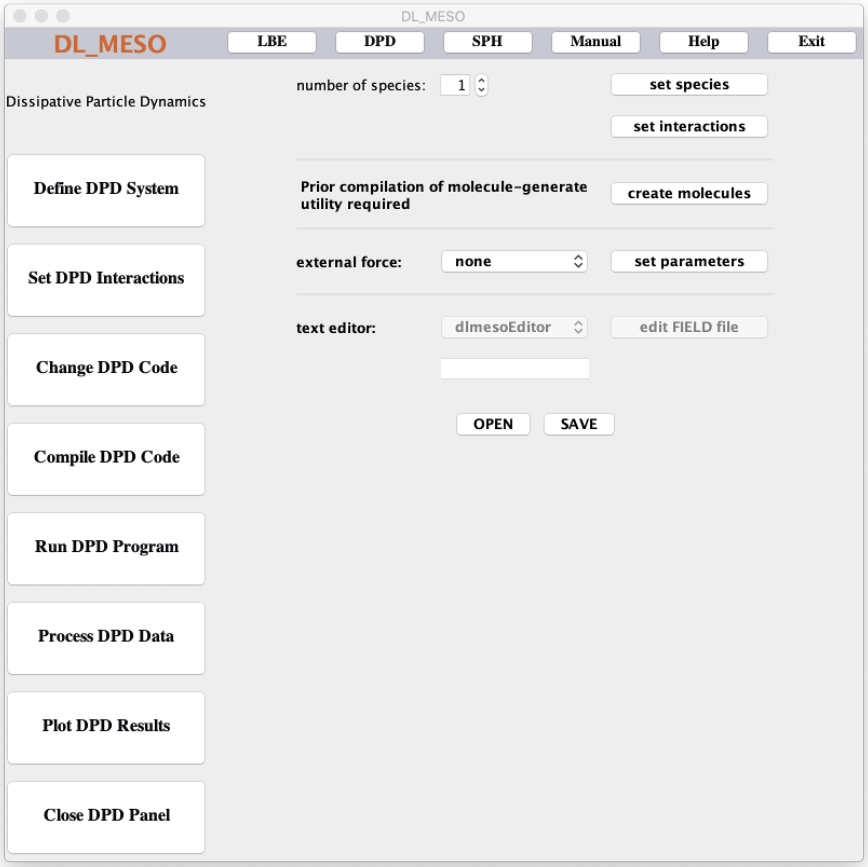


Figure 3.10: Set DPD Interactions

2. **Wall directions**: if the checkbox for a particular dimension is ticked, the boundary condition will be applied to the surfaces orthogonal to the specified axis.

3. **Wall offset**: this indicates the position of the boundaries relative to the system boundaries – this value can be set particularly to apply a reflective boundary ahead of frozen bead walls to prevent any particles from moving through them.

4. For systems with electrostatics, the **vacuum gap** in the reciprocal space part of an Ewald sum can be specified to help reduce its periodicity orthogonally to any non-periodic boundaries.

After filling in all the required values, clicking the relevant save button (**SAVE T**, **SAVE B**, **SAVE E** or **SAVE SF**) will store the data in preparation for writing to the `CONTROL` file. The cancel buttons (**CANCEL T**, **CANCEL B**, **CANCEL E** and **CANCEL SF**) will close the pop-ups without saving any values.

3.3.2 Defining DPD Interactions

Figure 3.10 shows the Set DPD Interactions panel. The following data are required:

1. The **number of species** is required to specify all interactions between particles in a DPD simulation. The spinner box allows the user to define up to 10 particle species, while the button **set species** opens a pop-up window for the user to enter the properties for each species – see below for more details.

2. Once the particles species have been defined, the button **set interactions** opens a pop-up window to allow the user to define non-bonded interactions between particle species (including surface interactions) and, if applicable, which particle species is to be used for frozen bead walls.

3. The molecule generation utility `molecule-generate.cpp` can be run to **create molecules** for the DPD simulation, which can be included in the `FIELD` file. The utility should be compiled beforehand to give the executable `molecule.exe` (refer to Appendix C for more details).

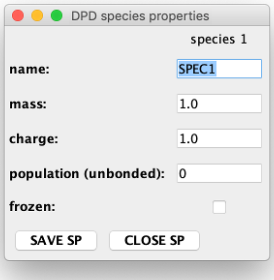
4. It is possible to define an **external force** field on all particles in the system, using the pull-down box to define the type. Constant gravitational fields, linear shear boundaries and constant electric fields can be defined. Clicking on the button **set parameters** opens a pop-up window to define the parameters for the external force field.

5. Once the FIELD file has been created, a **text editor**, including the built-in `dlmesoEditor`, may be selected using the pull-down box to view and edit the `FIELD` file. An alternative editor can be used by selecting 'other' and typing its name in the text box before clicking on the **edit FIELD file** button.

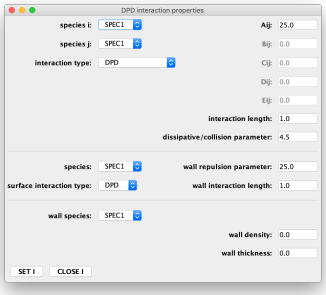
A pre-existing `FIELD` file can be read into the GUI by clicking **OPEN**: any species beyond the 10th contained therein will be ignored, but the user will be able to subsequently edit the species and interaction properties. Note that this does not preserve any included molecular data, which may be lost if the the `FIELD` file is subsequently saved unless it is included in a separate `molecule` file (as generated by the `molecule-generate.cpp` utility). Clicking **SAVE** write all interaction data, including any created molecule data in the `molecule` file, to a `FIELD` file, which can still be viewed and edited afterwards using the text editor option described above.

3.3.2.1 Species, interactions and external field parameters

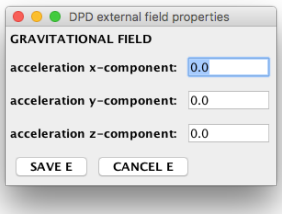
Examples of the pop-up windows for species, non-bonded interactions and external field parameters can be seen in Figure 3.11.



(a) Species parameters



(b) Non-bonded interaction parameters



(c) External field parameters

Figure 3.11: Species, interactions and external field pop-up windows

The species pop-up window is formatted as in Figure 3.11(a), with individual columns for each species:

1. The **name** of each species, which can be up to 8 characters long.

2. The **mass** of a particle for the species (m_i).

3. The **charge** of a particle for the species (q_i).
4. The number of unbonded particles of the species (**population (unbonded)**) in the system.
5. The tickbox indicates whether or not the particles for the species should be **frozen**.

Figure 3.11(b) gives the layout for the interaction pop-up window:

1. The pair of species can be selected using the first two pull-down boxes: the interaction parameters and type currently set for the selected species pair will be displayed.
2. The **interaction type** for the species pair: the standard DPD model by Groot and Warren[43] is the default, but many-body (density dependent) DPD[89, 130], Lennard-Jones[62] and Weeks-Chandler-Andersen 'hard sphere' models can also be selected. Note that while the Lennard-Jones and Weeks-Chandler-Andersen (WCA)[137] models are *not DPD* models, the DPD thermostat can be used with them to maintain system temperature.
3. The energy parameters for the species pair can be typed into the boxes labelled **Aij** (**Bij**, **Cij**, **Dij**, **Eij**) and set using the button **SET I**. Only one energy parameter is required for standard DPD (A_{ij}), Lennard-Jones and WCA (ϵ_{ij}), while many-body DPD can use up to five: the exact number required will depend upon the model selected by the user. Note that values for these and other interaction parameters for *all species pairs* will be written to the `FIELD` file: if many-body DPD interactions are not included and mixing rules are to be used between unlike species, the file can subsequently be edited to remove extraneous definitions.
4. The maximum **interaction length** between the two species ($r_{c,ij}$ or σ_{ij}) can be typed into this box and set using the button **SET I**.
5. The dissipative factor (γ_{ij}) or collision frequency (Γ_{ij}) for the species pair (i.e. the parameter for the selected thermostat) can be typed into this box and set using the **SET I** button.
6. If non-periodic hard walls are in use, surface interaction parameters can be set for each **species**: the available **surface interaction type**s include DPD and WCA.
7. The **wall repulsion parameter** ($A_{i,wall}$ or $\epsilon_{i,wall}$ and **wall interaction length** ($z_{c,i}$ or σ_i) can be typed in and set using the button **SET I**.
8. If a non-periodic frozen bead surface is defined, the species of beads making up the walls can be selected using a pull-down box.
9. The **wall density** of frozen beads can be typed in a box, as can the **wall thickness**.

Note that if hard walls or frozen bead walls are not in use, the relevant pull-down boxes and text fields will not be displayed in this window.

The parameters for external force fields can be given in the pop-up window shown in Figure 3.11(c):

1. The type of external field to be used in the simulation.
2. External field parameters: for constant gravitational fields (or similar constant external force fields), the x-, y- and z-components of gravitational **acceleration** ($\vec{G}$) need to be specified. For linear shear boundaries, the x-, y- and z-components of the **boundary velocity** ($\vec{V}_w$) need to be defined, although the component orthogonal to the wall will be ignored in simulations. For constant electric fields, the x-, y- and z-components of the electric field ($\vec{E}$) need to be specified.

After filling in all the required values, clicking the relevant set or save button (**SAVE SP**, **SET I** or **SAVE E**) will ensure the data will be writen to the `FIELD` file. The close/cancel buttons (**CLOSE SP**, **CLOSE I** and **CANCEL E**) will close the pop-ups without changing any previously-saved values.

3.4 Compiling and running DL_MESO

- Compiling the LBE/DPD code may be accomplished through the compiler panel which is activated from either of the **Compile LBE Code** or **Compile DPD Code** buttons.
- The **Compile LBE Code** panel allows you to select the operating system, a C++ compiler, compiler flags (both standard and for OpenMP) and the version (serial or parallel, with or without OpenMP) of the code you wish to build. If you require a C++ compiler that is not included in the pull-down list, select other and type the command for the required compiler in the neighbouring box. Clicking the **COMPILE** button will start the compilation and a message box will signal its completion.
- The **Compile DPD Code** panel allows you to select the operating system, a Fortran compiler, compiler flags and the version (serial or parallel, with or without OpenMP) of the code you wish to build. If you require a Fortran compiler that is not included in the pull-down list, select other and type the command for the required compiler in the neighbouring box. The **Create Makefile** button needs to be clicked first to create a makefile in the working directory, which automates compilation and may be edited by the user[2]. Clicking the **COMPILE** button will invoke the makefile to compile the code and a message box will signal its completion.
- If the compilation fails, you may need to edit the code. An editing panel is available for this purpose using either the **Change LBE Code** or **Change DPD code** buttons. Its function is similar to the compilation panel in operation with a choice of text editors, including one packaged with the DL_MESO GUI.
  - The files in the LBE code that can be edited include the parallel and serial main files (standard and custom), the core loops (parallel and serial), the lattice model file, the boundary condition file, the core routines for LBE calculations, the collision routines (BGK, TRT and MRT), the force calculation routines, the file for user-defined routines, the main head file and the head file for user-defined routines. Others can be edited by selecting other and typing the name of the file in the neighbouring box.
  - The files in the DPD code that may be edited include the main program, constants, global variables and the modules configuration, start (for system initialization), field (for force calculations, both with and without OpenMP), bond interactions (with and without OpenMP), many-body DPD (with and without OpenMP), Ewald sums (with and without OpenMP), surfaces and statistics. Other code files can be edited by selecting other and typing the name in the neighbouring box.
- Running the LBE/DPD code is made possible through the **Run LBE Program** or **Run DPD Program** button, which activates a panel that allows you to select the required submission command and then submit the job. You may need to create a suitable run script in your working directory beforehand if running the job in parallel.
- Collecting data from multiple processors and processing it for visualization is possible using the **Gather LBE Data** and **Process DPD Data** buttons. For DPD simulations, several dialogue and pull-down boxes are available for various post-processing utilities: one of these can specify a particular species and the `FIELD` file is needed to read species names for the simulation. Note that the utilities need to be compiled in the working directory prior to use: details on this and their functions can be found in Appendix C.
- The results of LBE and DPD calculations may be plotted using the **Plot LBE Results** and **Plot DPD Results** buttons, which allows the user to select plotting and visualization applications, including those not available in the pull-down lists. Note that these need to be already installed on the workstation in use before being invoked by the GUI: if they require running from a command-line, tick the **run in terminal** box before launching the application.

[2]If using Cygwin in Windows, editing the makefile will be necessary as the Windows formatting for directories cannot ordinarily be used.

3.5 Notes

- There are some inactive buttons reserved for later use.
- The GUI does not produce initial state files (`lbin.init` for LBE, `CONFIG` for DPD) prior to simulations, although there are utilities available to do this: see Appendix C for further details.
- Click **EXIT** to close down the GUI.

Part I

Lattice Boltzmann Equation (LBE)

Chapter 4

The Lattice Boltzmann Equation: Basic Theory

4.1 Introduction

The Lattice Boltzmann Equation (LBE) method is based on modelling a fluid consisting of fictional particles, which collide and move over a discrete lattice grid. This method is similar to its ancestor, Lattice Gas Cellular Automata (LGCA), but the main difference is that while LGCA represents the existence or otherwise for each particle at a grid point, LBE describes the physical state of an ensemble of particles by a single distribution function. This difference allows LBE to simulate both dilute fluids (i.e. those in which the mean free path of component particles is much larger than the lattice spacing) and condensed matter such as liquids.

The Lattice Boltzmann method uses fully discretized space, time and velocity to describe the evolution of fluid. Space is represented by a regularly distributed grid, time flow is obtained by integrating over discrete time steps and discrete velocity vectors (lattice links) are defined to ensure that a particle moves from one grid point to another without falling between them.

The Lattice Boltzmann algorithm can be summarized by the following:

- Fluid properties are mapped onto a discrete lattice.
- The physical state of the fluid at each lattice point is described by a set of particle distribution functions.
- The system evolves towards an equilibrium (or steady state) by means of a two-step process:
    1. Collision (relaxation) of the distribution function towards its local equilibrium form;
    2. Propagation of collided distribution functions along lattice links to neighbouring points.
- Macroscopic fluid variables (e.g. density, momenta) can be calculated from moments of the distribution functions.

Major benefits of the Lattice Boltzmann Equation method include the local nature of its most computationally intensive process (collision), making the method inherently and massively parallelizable, and its ability to model complex system geometries and/or fluid interactions with comparatively little additional computational cost.

4.2 Basic Definitions

Triangular and rectangular lattices are two of the most popular grid forms used in Lattice Boltzmann simulations. Triangular lattices have sixth-order rotation isotropy and have been widely applied in two-dimensional systems, e.g. the D2Q7 and D2Q13 models. Rectangular lattices have only fourth-order rotation isotropy but can more easily handle the simulation of three-dimensional systems with complex boundary conditions. The local

equilibria for D2Q9 and D3Q27 lattice models can be derived *a priori* from the Maxwell equilibrium distribution. D3Q15 and D3Q19 models appear to be more popular than D3Q27 because the latter is much more expensive in terms of computing cost.

It is required that the equilibrium state should be able to reproduce elementary macroscopic fluid variables:

$$\rho = \sum_{i=0}^{q} f_i \tag{4.1}$$

$$\rho u_\alpha = \sum_{i=0}^{q} f_i e_{i\alpha} \tag{4.2}$$

where ρ is the density, f_i the ith particle distribution function, $\hat{e}_i$ the ith lattice link vector and u_α the macroscopic velocity along the α-axis.

In the Lattice Boltzmann method, the lattice link vectors $\hat{e}_i$ do not represent the thermal velocities of a particle and therefore

$$E \neq \frac{1}{2} \sum f_i \left(\hat{e}_i - \vec{u}\right)^2 \tag{4.3}$$

Equation (4.3) implies that the temperature cannot ordinarily be derived from the lattice particle distribution function and that the fluid modelled using LBE is generally *athermal*. It is possible, however, to alleviate this problem by defining a temperature at each grid point and either using a thermal lattice scheme with additional link vectors or modelling either the temperature or internal energy on an additional lattice grid.

4.3 Derivation of Equilibrium

There are two methods by which the local equilibria for the Lattice Boltzmann Equation can be constructed. The *bottom-up* method obtains the equilibrium from the Maxwell-Boltzmann equilibrium distribution. The *top-down* method constructs the equilibrium so that the required macroscopic properties can be reproduced. Only the bottom-up method is shown here; the top-down method can be found in [20].

The Maxwell-Boltzmann single particle equilibrium distribution function is

$$f^{eq} = \frac{\rho}{(2\pi\theta)^{\frac{D}{2}}} \exp\left[-\frac{\left(\vec{\xi} - \vec{u}\right)^2}{2\theta}\right] \tag{4.4}$$

where $\theta = k_B T/m$, k_B is the Boltzmann constant, T is temperature, m is molar mass, D is the space dimension, $\vec{\xi}$ is the thermal velocity and $\vec{u}$ the macroscopic velocity.

When $|\vec{\xi} - \vec{u}| \ll \sqrt{\theta}$, Equation (4.4) can be expanded into

$$f^{eq} = \frac{\rho}{(2\pi\theta)^{\frac{D}{2}}} \exp\left(-\frac{\xi^2}{2\theta}\right)\left[1 + \frac{\vec{\xi}\cdot\vec{u}}{\theta} + \frac{\left(\vec{\xi}\cdot\vec{u}\right)^2}{2\theta^2} - \frac{u^2}{2\theta}\right] \tag{4.5}$$

For a microscopic quantity $\psi(\xi)$, the associated macroscopic quantity Ψ is calculated by

$$\Psi = \int \psi(\xi) f^{eq} d\xi \tag{4.6}$$

Let $\vec{\xi} = \sqrt{2\theta}\vec{c}$, where $\vec{c}$ is a rescaled thermal velocity; the macroscopic velocity $\vec{u}$ can be similarly rescaled to $\sqrt{2\theta}\vec{u}$. Equations (4.5) and (4.6) can thus be combined to give

$$\Psi = \int e^{-c^2} \psi(c) \frac{\sqrt{2\theta}\rho}{(2\pi\theta)^{\frac{D}{2}}} \left[1 + 2\left(\vec{c}\cdot\vec{u}\right) + 2\left(\vec{c}\cdot\vec{u}\right)^2 - u^2\right] dc \tag{4.7}$$

Using Gaussian quadrature, Equation (4.7) changes into

$$\Psi = \sum_i \psi(c_i) \frac{\sqrt{2\theta}\rho}{(2\pi\theta)^{\frac{D}{2}}} w(c_i) \left[1 + 2(\vec{c}\cdot\vec{u}) + 2(\vec{c}\cdot\vec{u})^2 - u^2\right] \tag{4.8}$$

Let

$$w_i = \frac{\sqrt{2\theta}\rho}{(2\pi\theta)^{\frac{D}{2}}} w(c_i) \tag{4.9}$$

and

$$f_i^{eq}(\rho,\vec{u}) = w_i\rho\left[1 + 2(\vec{c}\cdot\vec{u}) + 2(\vec{c}\cdot\vec{u})^2 - u^2\right] \tag{4.10}$$

The value of $w(c_i)$ can be obtained from Gauss-Hermite integration. Equation (4.10) is the equilibrium particle distribution function in the discrete regime. w_i is called the weight factor for speed vector c_i. Equation (4.10) can also be written as

$$f_i^{eq}(\rho,\vec{u}) = w_i\rho\left[1 + \frac{3(\hat{e}_i\cdot\vec{u})}{c^2} + \frac{9(\hat{e}_i\cdot\vec{u})^2}{2c^4} - \frac{3u^2}{2c^2}\right] \tag{4.11}$$

where $c = \sqrt{3\theta} = \sqrt{\frac{3k_BT}{m}}$ is the modulus of the basic lattice vector and equivalent to the fluid speed of sound (e.g. for water at 20℃, $c = 367.8$ m/s).

4.4 Structural Relaxation and Macroscopic Equations

The Lattice Boltzmann method often uses the BGK (Bhatnagar, Gross and Krook) approximation[7] to describe the structural relaxation. The single particle distribution function evolves to the equilibrium state via

$$f_i(\vec{x}+\hat{e}_i\Delta t, t+\Delta t) - f_i(\vec{x},t) = -\frac{\Delta t}{\tau_f}\left[f_i(\vec{x},t) - f_i^{eq}(\rho(\vec{x},t),\vec{u}(\vec{x},t)\right] \tag{4.12}$$

where τ_f is called the relaxation time and is related to the kinetic viscosity of fluid. This evolution equation can be divided into two separate processes of *collision* (where t^+ denotes a time after collision has taken place)

$$f_i(\vec{x},t^+) = f_i(\vec{x},t) - \frac{\Delta t}{\tau_f}\left[f_i(\vec{x},t) - f_i^{eq}(\rho(\vec{x},t),\vec{u}(\vec{x},t)\right] \tag{4.13}$$

and *propagation* (or free-streaming)

$$f_i(\vec{x}+\hat{e}_i\Delta t, t+\Delta t) = f_i(\vec{x},t^+). \tag{4.14}$$

To derive the macroscopic equations, the left hand side of Equation (4.12) can be expanded as

$$f_i(\vec{x}+\hat{e}_i\Delta t, t+\Delta t) - f_i(\vec{x},t) = \sum_{m=1}^{\infty}\frac{\Delta t^m}{m!}\left(\partial_t + e_{i\alpha}\partial_\alpha\right)^m f_i(\vec{x},t) \tag{4.15}$$

Expanding the instantaneous particle distribution function around its equilibrium and retaining only the first order gives

$$f_i(\vec{x},t) = f_i^{eq}(\vec{x},t) - \tau_f\left(\partial_t + e_{i\alpha}\partial_\alpha\right) f_i^{eq}(\vec{x},t) + O\left(\partial^2\right) \tag{4.16}$$

Substituting Equations (4.15) and (4.16) into the left hand side of Equation (4.12) gives the second order differential equation for the equilibrium distribution

$$\frac{f_i^{eq} - f_i}{\tau_f} = \left(\partial_t + e_{i\alpha}\partial_\alpha\right) f_i^{eq} - w_f\left(\partial_t + e_{i\alpha}\partial_\alpha\right)^2 f_i^{eq} + O\left(\partial^3\right) \tag{4.17}$$

where $w_f = \tau_f - \frac{\Delta t}{2}$.

Summing Equation (4.17) over i and ignoring the second order deriviative we obtain

$$0 = \partial_t \rho + \partial_\alpha \rho u_\alpha - w_f \partial_\beta \left(\partial \rho u_\beta + \partial_\alpha \sum_i f_i^{eq} e_{i\alpha} e_{i\beta} \right) + O\left(\partial^3\right) \tag{4.18}$$

Summing Equation (4.18) times e_i over i we obtain

$$0 = \partial_t \rho u_\alpha + \partial_\beta \sum_i f_i^{eq} e_{i\alpha} e_{i\beta} - w_f \partial_\gamma \left(\partial_t \sum_i f_i^{eq} e_{i\alpha} e_{i\gamma} + \partial_\beta \sum_i f_i^{eq} e_{i\alpha} e_{i\beta} e_{i\gamma} \right) + O\left(\delta^3\right) \tag{4.19}$$

Equation (4.19) shows that the second term in Equation (4.18) is of the third order in the derivative. Therefore we have the continuity equation to the second order of the derivative

$$\partial_t \rho + \nabla \cdot \rho \vec{u} = 0 \tag{4.20}$$

Defining the third and fourth order moments

$$\sum_i f_i^{eq} e_{i\alpha} e_{i\beta} = P_{\alpha\beta} + \rho u_\alpha u_\beta \tag{4.21}$$

$$\sum_i f_i^{eq} e_{i\alpha} e_{i\beta} e_{i\gamma} = P_{\alpha\beta} u_\gamma + P_{\alpha\gamma} u_\beta + P_{\beta\gamma} u_\alpha + \rho u_\alpha u_\beta u_\gamma \tag{4.22}$$

With these definitions, Equation (4.19) leads to the weakly compressible Navier-Stokes equation

$$\partial_t(\rho u_\alpha) + \partial_\beta(\rho u_\alpha u_\beta) = -\partial_\beta P_{\alpha\beta} + \frac{w_f D}{3} \partial_\beta \left(\frac{\delta_{\alpha\beta} - 3\partial_\alpha P_{\alpha\beta}}{D} \partial_\gamma(\rho u_\gamma) + \partial_\alpha(\rho u_\beta) + \partial_\beta(\rho u_\alpha) \right) + O\left(\partial^3\right) \tag{4.23}$$

where the kinetic viscosity is given by $\nu = \frac{w_f}{3} = \frac{1}{3}\left(\tau_f - \frac{1}{2}\right)$.

Similarly, the convection diffusion equation governing the evolution of solute transfer and the convection/conduction equation governing the evolution of thermal transfer can be obtained.

4.5 Mesoscale Interaction

The rate of change of momentum is proportional to the force

$$\rho u_\alpha (t + \Delta t) = \rho u_\alpha (t) + F_\alpha \Delta t \tag{4.24}$$

The Lattice Boltzmann Equation takes into account the effect of relaxation and it is possible to express Equation (4.24) in a new format[84]:

$$\rho u_\alpha (t + \Delta t) = \rho u_\alpha (t) + F_\alpha \tau_f \tag{4.25}$$

$\vec{F}$, with F_α as the component in the α direction, can be a long-ranged body force or any local interactions.

In the Shan-Chen pseudopotential model for multiple phases and components[108], the force on component a is defined as

$$\vec{F}^a = -\psi^a (\vec{x}) \sum_b g_{ab} \sum_i w_i \psi^b (\vec{x} + \hat{e}_i) \hat{e}_i \tag{4.26}$$

where g_{ab} is the interaction coefficient between elements a and b. ψ^a is related to the density of element a and can take many different forms, e.g.

$$\psi^a (\vec{x}) = \rho_0^a \left[1 - \exp\left(-\frac{\rho^a}{\rho_0^a} \right) \right] \tag{4.27}$$

In the Lishchuk continuum-based model for single phases and multiple components[77], the force acting between components a and b is expressed as

$$\vec{F}_{ab} = \frac{1}{2} g_{ab} K_{ab} \nabla \rho_{ab}^N \tag{4.28}$$

where $\rho_{ab}^{N} = \frac{\rho^a - \rho^b}{\rho^a + \rho^b}$ is a phase index between the two components and K_{ab} is the local curvature from the interface model, which can be determined from spatial gradients of the phase index. This algorithm requires a modification to the collision step: at each lattice point all fluids are collided as a single fluid, which is then segregated back out into the individual fluids.

In the Swift free-energy-based model for two phases or two fluids[125, 126], modifications are made to the local equilibrium distribution function to incorporate the fluid equation of state (in the form of bulk pressure) and the interface surface tension between the phases or fluids, e.g. for a single fluid with two phases

$$\begin{aligned} f_i^{eq} = w_i^{00}\rho + w_i\left[\rho\left\{(\hat{e}_i\cdot\vec{u}) + \frac{3}{2}(\hat{e}_i\cdot\vec{u})^2 - \frac{1}{2}u^2\right\} + \lambda\left\{3(\hat{e}_i\cdot\vec{u})(\hat{e}_i\cdot\nabla\rho) + [\gamma_i(\hat{e}_i\cdot\hat{e}_i) + \delta_i](\vec{u}\cdot\nabla\rho)\right\}\right] \\ + w_i^p P_0 - w_i^t\kappa\rho\nabla^2\rho + w_i^{xx}\kappa(\partial_x\rho)^2 + w_i^{yy}\kappa(\partial_y\rho)^2 + w_i^{zz}\kappa(\partial_z\rho)^2 \\ + w_i^{xy}\kappa\partial_x\rho\partial_y\rho + w_i^{xz}\kappa\partial_x\rho\partial_z\rho + w_i^{yz}\kappa\partial_y\rho\partial_z\rho \end{aligned} \tag{4.29}$$

where P_0 is the bulk pressure of the fluid, κ the interfacial surface tension and λ is a correction term to ensure Galilean invariance. The above form of local equilibrium requires first and second order gradients of density, which can be calculated for each grid point based on density values for neighbouring points, and can correctly obtain the free energy density for the fluid by means of thermodynamic consistency at the continuum. This method can be extended to two-fluid systems by modelling the total fluid density and additionally modelling a fluid concentration ($\phi = \frac{\rho_a - \rho_b}{\rho}$), whose local equilibrium distribution function includes the chemical potential between the two fluids (given as the derivative of bulk free energy density with concentration).

4.6 Summary of Lattice Boltzmann Equation

Lattice Boltzmann is an established numerical methodology for handling hydrodynamics in fluid. It is particularly suited to simulating systems with complex boundary conditions and is also suitable for systems with phase transitions since mesoscale interactions can be merged into the method easily.

Chapter 5

DL_MESO_LBE Features

5.1 Collision and Propagation Algorithms

The collision and propagation routines are a fundamental part of Lattice Boltzmann Equation calculations. Implementation involves the calculation of post-collisional values for the distribution functions at each lattice point (at time t^+). A generalised form of the collision operator can be given as a product of a square matrix of size equal to the number of lattice links, $\mathbf{\Lambda}$ and the deviation of the distribution function from the local equilibrium value, which can be reduced to a collision operator for link i by summation over all links, i.e.

$$C_i = \sum_j \Lambda_{ij} \left(f_j - f_j^{eq} \right).$$

The generalised form of the Lattice Boltzmann Equation can thus be given with the collision operator C_i and forcing term F_i:

$$f_i \left(\vec{x}, t^+ \right) = f_i \left(\vec{x}, t \right) + C_i + F_i \Delta t$$

and movement of these distribution functions to neighbouring lattice nodes:

$$f_i \left(\vec{x} + \hat{e}_i \Delta t, t + \Delta t \right) = f_i \left(\vec{x}, t^+ \right)$$

which combine to give the governing equation for calculations.

5.1.1 Collision algorithms

The forms of collision currently available in DL_MESO_LBE are:

- the Bhatnagar-Gross-Krook (BGK) single relaxation time[7] scheme;
- the Two Relaxation Time (TRT)[39] scheme;
- Multiple Relaxation Time (MRT) schemes[68, 23, 124];
- Cascaded Lattice Boltzmann Equation (CLBE) schemes[35, 32, 30];

and each collision method can be coupled to four different methods to apply interfacial and external forces to the fluids: the default Martys-Chen method, the Exact Difference Method and the second-order accurate Guo and He methods.

The Martys-Chen method[84] (used as the default in DL_MESO_LBE) simply modifies the velocity of the fluid used to calculate the equilibrium distribution function for collisions. If the measured macroscopic velocity is defined as $\vec{u} = \frac{1}{\rho} \sum_i f_i \hat{e}_i$, the equilibrium fluid velocity is given as $\vec{v} = \vec{u} + \frac{\tau_f \Delta t \vec{F}}{\rho}$. This gives an equivalent forcing term[74] of

$$F_i = \tau_f w_i \left[\frac{\hat{e}_i - \vec{u}}{c_s^2} + \frac{(\hat{e}_i \cdot \vec{u})}{c_s^4} \hat{e}_i \right] \cdot \vec{F} + \frac{w_i \tau_f^2 \Delta t}{\rho} \left[\frac{\left(\hat{e}_i \cdot \vec{F} \right)^2}{2 c_s^4} - \frac{F^2}{2 c_s^2} \right] \tag{5.1}$$

if the measured macroscopic velocity ($\vec{u}$) were used as the equilibrium fluid velocity.

The Exact Difference Method (EDM)[65] uses the equilibrium fluid velocity $\vec{v} = \vec{u}$ and applies an additional forcing term defined as:

$$F_i \Delta t = f_i^{eq}\left(\rho, \vec{u} + \frac{\vec{F}\Delta t}{\rho}\right) - f_i^{eq}(\rho, \vec{u}). \tag{5.2}$$

This expression can be written explicitly if the form of the equilibrium distribution function is known. If the second-order accurate equilibrium distribution function (Equation 4.11) is to be used, the forcing term can be expressed as

$$F_i = w_i \left[\frac{\hat{e}_i - \vec{u}}{c_s^2} + \frac{(\hat{e}_i \cdot \vec{u})}{c_s^4}\hat{e}_i\right] \cdot \vec{F} + \frac{w_i \Delta t}{\rho}\left[\frac{\left(\hat{e}_i \cdot \vec{F}\right)^2}{2c_s^4} - \frac{F^2}{2c_s^2}\right]. \tag{5.3}$$

Guo's method[46] uses an adjusted fluid velocity for both equilibrium distribution functions and additional forcing terms. The equilibrium fluid velocity is defined as $\vec{v} = \vec{u} + \frac{\vec{F}\Delta t}{2\rho}$ and the forcing term is related to the following source term:

$$S_i = w_i \left[\frac{\hat{e}_i - \vec{v}}{c_s^2} + \frac{(\hat{e}_i \cdot \vec{v})}{c_s^4}\hat{e}_i\right] \cdot \vec{F}, \tag{5.4}$$

derived to ensure correctness to the Navier-Stokes equations as determined by Chapman-Enskog analysis to second order in Knudsen number.

The He method[50] is derived from second-order integration of the continuous Boltzmann equation with time and a transformation of the distribution function to remove implicitness: like Guo's method, this also uses the adjusted fluid velocity $\vec{v}$ for equilibrium distribution functions and forcing terms. The forcing term for this method is determined from the derivative of the Maxwell-Boltzmann equilibrium distribution function – Equation 4.4 – with respect to the lattice link vector $\hat{e}_i$ as an approximation for the same derivative of the main distribution function. This results in the source term:

$$S_i = \frac{f_i^{eq}}{\rho c_s^2}(\hat{e}_i - \vec{v}) \cdot \vec{F}, \tag{5.5}$$

which can use the (usual) expanded local equilibrium distribution function for efficient calculation and often be truncated to second-order velocity terms in line with the general accuracy of LBE calculations, i.e.

$$S_i = w_i \left[\frac{\hat{e}_i - \vec{v}}{c_s^2} + \frac{(\hat{e}_i \cdot \vec{v})(\hat{e}_i - \vec{v})}{c_s^4} + \left(\frac{(\hat{e}_i \cdot \vec{v})^2}{c_s^2} - v^2\right)\frac{\hat{e}_i}{2c_s^4}\right] \cdot \vec{F}. \tag{5.6}$$

The forcing term for both Guo and He methods is generically expressed as

$$F_i = \sum_j \left(\delta_{ij} - \frac{\Lambda_{ij}}{2}\right) S_j, \tag{5.7}$$

where δ_{ij} is the Dirac delta (equal to 1 when $i = j$ and 0 when $i \neq j$) and Λ_{ij} is a relevant matrix operator for the collision scheme in use.

5.1.1.1 BGK single relaxation time

The BGK collision operator is defined by

$$C_i = -\frac{\Delta t}{\tau_f}\left(f_i(\vec{x}, t) - f_i^{eq}(\rho, \vec{v})\right) \tag{5.8}$$

using the macroscopic density (ρ) and equilibrium velocity[1] ($\vec{v}$) at the grid point $\vec{x}$ and time t. τ_f is the relaxation time, related to the kinetic (shear) viscosity of fluid by

$$\nu = \frac{1}{3}\left(\tau_f - \frac{1}{2}\right)\frac{(\Delta x)^2}{\Delta t}$$

[1] This quantity – particularly how it is related to the macroscopic velocity $\vec{u}$ – will depend upon the forcing algorithm in use.

with the kinematic bulk viscosity of the fluid, ν', set equal to $\frac{2}{3}\nu$[22]. This reduces the equation for post-collisional distribution functions to

$$f_i\left(\vec{x}, t^+\right) = f_i\left(\vec{x}, t\right) - \frac{\Delta t}{\tau_f}\left(f_i\left(\vec{x}, t\right) - f_i^{eq}(\rho, \vec{v})\right),$$

the same as Equation 4.12.

The BGK collision operator with the single relaxation time is equivalent to setting the generalised collision matrix to

$$\mathbf{\Lambda} = -\frac{\Delta t}{\tau_f}\mathbf{I}$$

where $\mathbf{I}$ is the identity matrix (a square matrix with the diagonal values set to 1 and all other values set to 0).

All four forcing schemes (Martys-Chen, EDM, Guo and He) can be used in conjunction with the BGK collision operator: the Martys-Chen scheme is applied by default. The Guo forcing term for BGK collisions is defined as:

$$\begin{aligned} F_i &= \left(1 - \frac{1}{2\tau_f}\right) S_i \\ &= \left(1 - \frac{1}{2\tau_f}\right) w_i \left[\frac{\hat{e}_i - \vec{v}}{c_s^2} + \frac{(\hat{e}_i \cdot \vec{v})}{c_s^4}\hat{e}_i\right] \cdot \vec{F}, \end{aligned}$$

while He forcing term for BGK collisions is defined similarly as:

$$\begin{aligned} F_i &= \left(1 - \frac{1}{2\tau_f}\right) S_i \\ &= \left(1 - \frac{1}{2\tau_f}\right) w_i \left[\frac{\hat{e}_i - \vec{v}}{c_s^2} + \frac{(\hat{e}_i \cdot \vec{v})(\hat{e}_i - \vec{v})}{c_s^4} + \left(\frac{(\hat{e}_i \cdot \vec{v})^2}{c_s^2} - v^2\right)\frac{\hat{e}_i}{2c_s^4}\right] \cdot \vec{F}. \end{aligned}$$

5.1.1.2 Two Relaxation Time (TRT)

The TRT collision operator is based on defining symmetric (even) and antisymmetric (odd) distribution functions, which can be determined for a particular link from its own distribution function and the value from its conjugate opposite link. For link i, if $\hat{e}_j = -\hat{e}_i$, then the symmetric distribution function is defined as $f_i^+ = \frac{1}{2}(f_i + f_j)$ and the antisymmetric distribution function is $f_i^- = \frac{1}{2}(f_i - f_j)$. The equilibrium distribution functions can be similarly divided into symmetric and antisymmetric parts, $f_i^{eq,+} = \frac{1}{2}(f_i^{eq} + f_j^{eq})$ and $f_i^{eq,-} = \frac{1}{2}(f_i^{eq} - f_j^{eq})$ respectively. The collision operator is thus defined as

$$C_i = -\frac{\Delta t}{\tau_f^+}\left(f_i^+\left(\vec{x}, t\right) - f_i^{eq,+}(\rho, \vec{v})\right) - \frac{\Delta t}{\tau_f^-}\left(f_i^-\left(\vec{x}, t\right) - f_i^{eq,-}(\rho, \vec{v})\right) \tag{5.9}$$

where τ_f^+ and τ_f^- are respectively the symmetric and antisymmetric relaxation times.

The symmetric relaxation time defines the kinematic (shear) and bulk viscosities as

$$\nu = \frac{1}{3}\left(\tau_f^+ - \frac{1}{2}\right)\frac{(\Delta x)^2}{\Delta t}$$

$$\nu' = \left(\frac{2}{3} - c_s^2\right)\left(\tau_f^+ - \frac{1}{2}\right)\frac{(\Delta x)^2}{\Delta t},$$

and is functionally identical to the single relaxation time used for BGK collisions (i.e. $\tau_f^+ \equiv \tau_f$). The antisymmetric relaxation time is essentially a free 'kinetic' parameter that can be used to ensure the accuracy of simulations for all fluid viscosities when using particular boundary conditions, usually by fixing the product of even and odd eigenvalues to some 'magic number', $\Lambda_{eo} = \Lambda_e\Lambda_o = (\tau_f^+ - \frac{1}{2})(\tau_f^- - \frac{1}{2})$. For instance, for planar walls using bounce-back conditions with the wall placed halfway between lattice points, Λ_{eo} should be equal to $\frac{3}{8}$ to give zero wall slip[38]. The symmetric relaxation time can be specified for each fluid in the same manner as the BGK single relaxation time, while a value of Λ_{eo} can be given by the user and used to determine τ_f^-.

To reduce computational time, the above collision operator can be re-expressed in terms of the full distribution functions:

$$C_i = -\frac{\Delta t}{\tau_{f,p}}\left(f_i\left(\vec{x},t\right) - f_i^{eq}(\rho,\vec{v})\right) - \frac{\Delta t}{\tau_{f,n}}\left(f_j\left(\vec{x},t\right) - f_j^{eq}(\rho,\vec{v})\right), \tag{5.10}$$

where $\tau_{f,p} = \frac{2\tau_f^+\tau_f^-}{\tau_f^+ + \tau_f^-}$ and $\tau_{f,n} = \frac{2\tau_f^+\tau_f^-}{\tau_f^- - \tau_f^+}$.

All four forcing schemes can be used in conjunction with the TRT collision operator, with the Martys-Chen scheme used as the default. The Guo and He forcing terms for TRT collisions can be applied using symmetric and antisymmetric source terms[105], i.e. $S_i^{\pm} = \frac{1}{2}(S_i \pm S_j)$ and is thus defined as:

$$\begin{aligned} F_i &= \left(1 - \frac{1}{2\tau_f^+}\right) S_i^+ + \left(1 - \frac{1}{2\tau_f^-}\right) S_i^- \\ &= \left(1 - \frac{1}{2\tau_{f,p}}\right) S_i - \frac{1}{2\tau_{f,n}} S_j. \end{aligned}$$

5.1.1.3 Multiple Relaxation Time (MRT)

The MRT collision operator operates on a similar principle to the quasilinear Lattice Boltzmann Equation[52], which expresses collisions in terms of a square matrix with dimensions equal to the number of lattice links per grid point ($m \equiv$ `lbsy.nq`). Unlike quasilinear LBE, however, MRT collision schemes are applied to the (raw) moments for each lattice point rather than the distribution functions[68, 23, 124], which are related to each other by

$$\vec{M} = \mathbf{T}\vec{f} \tag{5.11}$$

where $\vec{f}$ is a vector of all m distribution functions for the point, i.e. $(f_0, f_1 \ldots f_{m-1})^T$, $\vec{M}$ the vector of moments (also of size m and dependent on the lattice system) and $\mathbf{T}$ the transformation matrix that renders the moments in terms of the distribution functions. Equilibrium values for the moments, $\vec{M}^{eq}$, can be determined by transforming the standard local equilibrium distribution function into moment space by

$$\vec{M}^{eq} = \mathbf{T}\vec{f}^{eq} \tag{5.12}$$

where $\vec{f}^{eq}$ is the vector of local equilibrium distribution functions: the resulting equilibrium moments can alternatively be expressed directly as functions of fluid density and velocity. Certain moments, such as density (ρ) and momentum ($\vec{j}$), must be conserved and their equilibrium values are set so that no changes are made. The post-collisional moments are determined by relaxation of the non-equilibrium part, i.e.

$$\vec{M}(\vec{x},t^+) = \vec{M}(\vec{x},t) - \mathbf{\Lambda}(\vec{M}(\vec{x},t) - \vec{M}^{eq}(\vec{x},t)) \tag{5.13}$$

where $\mathbf{\Lambda}$ is the collision matrix, which takes the form of a diagonal matrix of m collision parameters (represented by $\vec{s}$):

$$\mathbf{\Lambda} = \mathrm{diag}(\vec{s}) \tag{5.14}$$

Some of the collision parameters can be specified by the user to set both kinematic and bulk viscosities, a few others can be tuned to improve simulation stability and the remainder (i.e. those for density and momentum) are fixed to conserve macroscopic hydrodynamics. If all values of s_i are set to $\frac{1}{\tau_f}$, the scheme reduces to BGK single relaxation time collisions. Since the collisional matrix is diagonal, equation 5.13 can be rewritten in terms of each moment, i.e.

$$M_i(\vec{x},t^+) = M_i(\vec{x},t) - s_i\left(M_i(\vec{x},t) - M_i^{eq}(\vec{x},t)\right) \tag{5.15}$$

Multiplying $\vec{M}(\vec{x},t^+)$ by the inverse of the transformation matrix, $\mathbf{T}^{-1}$, gives the post-collisional distribution functions.

Interfacial and external forces can be applied using each of the three forcing methods, noting that these require all collision parameters for conserved moments to be set to unity. The Martys-Chen method can be applied during the main collision step by the addition of $\tau_f\vec{F}$ to the fluid momentum. The source term derived for the Exact Difference Method can be applied after the collided moments are converted back to post-collisional

distribution functions. To apply forces using Guo's or the He method, a moment-transformed source term, $\vec{S}^m$, can be derived[96] (where $\vec{S}$ is a vector of Guo/He forcing terms, $(S_0, S_1 \dots S_{m-1})^T$):

$$\vec{S}^m = \mathbf{T}\vec{S} \tag{5.16}$$

and applied by the following to correct the post-collisional moments:

$$\Delta\vec{M} = \left(\mathbf{I} - \tfrac{1}{2}\mathbf{\Lambda}\right)\vec{S}^m\Delta t \tag{5.17}$$

where $\mathbf{I}$ is an identity matrix. The above equation can be re-expressed as

$$\Delta M_i = \left(1 - \tfrac{1}{2}s_i\right)S_i^m\Delta t \tag{5.18}$$

The above equations after inverse transformation reduce to equations 5.4 or 5.6 when the collision parameters are set to $\frac{1}{\tau_f}$.

An example can be given for the D2Q9 lattice system[68]; the moment vector is

$$\vec{M} = (\rho, e, \epsilon, j_x, q_x, j_y, q_y, p_{xx}, p_{xy})^T$$

with ρ as the density, e the energy, ϵ the square of energy, $\vec{j}$ momentum, $\vec{q}$ energy flux, p_{xx} the diagonal stress tensor component and p_{xy} the off-diagonal stress tensor component. The transformation matrix is

$$\mathbf{T} = \begin{bmatrix} 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 \\ -4 & 2 & -1 & 2 & -1 & 2 & -1 & 2 & -1 \\ 4 & 1 & -2 & 1 & -2 & 1 & -2 & 1 & -2 \\ 0 & -1 & -1 & -1 & 0 & 1 & 1 & 1 & 0 \\ 0 & -1 & 2 & -1 & 0 & 1 & -2 & 1 & 0 \\ 0 & 1 & 0 & -1 & -1 & -1 & 0 & 1 & 1 \\ 0 & 1 & 0 & -1 & 2 & -1 & 0 & 1 & -2 \\ 0 & 0 & 1 & 0 & -1 & 0 & 1 & 0 & -1 \\ 0 & -1 & 0 & 1 & 0 & -1 & 0 & 1 & 0 \end{bmatrix}$$

The equilibrium moment vector is

$$\vec{M}^{eq} = \left(\rho, -2\rho + \frac{3(j_x^2 + j_y^2)}{\rho}, w_\epsilon\rho + w_{\epsilon j}\frac{(j_x^2 + j_y^2)}{\rho}, j_x, -j_x, j_y, -j_y, \frac{j_x^2 - j_y^2}{\rho}, \frac{j_x j_y}{\rho}\right)^T$$

with w_ϵ and $w_{\epsilon j}$ as adjustable parameters: for convergence to the single relaxation time BGK scheme, these are set equal to 1 and -3 respectively. Of the 9 collision parameters available, s_0, s_3 and s_5 have no effect (except when applying external forces, when they should be set equal to one) as the associated moments are preserved and s_2, s_4 and s_6 are tuneable parameters for calculation stability with the condition that $s_4 = s_6$. The remaining parameters represent the viscosities of the fluid, i.e.

$$\begin{aligned} \nu &= \frac{1}{3}\left(\frac{1}{s_7} - \frac{1}{2}\right)\frac{(\Delta x)^2}{\Delta t} = \frac{1}{3}\left(\frac{1}{s_8} - \frac{1}{2}\right)\frac{(\Delta x)^2}{\Delta t} \equiv \frac{1}{3}\left(\tau_f - \frac{1}{2}\right)\frac{(\Delta x)^2}{\Delta t} \\ \nu' &= \frac{1}{6}\left(\frac{1}{s_1} - \frac{1}{2}\right)\frac{(\Delta x)^2}{\Delta t} \equiv \frac{1}{6}\left(\tau_{f,bulk} - \frac{1}{2}\right)\frac{(\Delta x)^2}{\Delta t} \end{aligned}$$

i.e. $\tau_f = \frac{1}{s_7} = \frac{1}{s_8}$ and $\tau_{f,bulk} = \frac{1}{s_1}$. If the moment-transformed source terms are to be used for Guo-like forcing, the vector $\vec{S}^m$ for this lattice scheme is defined as

$$\vec{S}^m = \begin{pmatrix} 0 \\ 6(u_xF_x + u_yF_y) \\ -6(u_xF_x + u_yF_y) \\ F_x \\ -F_x \\ F_y \\ -F_y \\ 2(u_xF_x - u_yF_y) \\ u_xF_y + u_yF_x \end{pmatrix},$$

while He forcing uses the following moment-transformed source terms:

$$\vec{S}^m = \begin{pmatrix} 0 \\ 6(u_x F_x + u_y F_y) \\ -6(u_x F_x + u_y F_y) \\ F_x \\ -F_x(1-3u_y^2) + 6u_x u_y F_y \\ F_y \\ -F_y(1-3u_x^2) + 6u_x u_y F_x \\ 2(u_x F_x - u_y F_y) \\ u_x F_y + u_y F_x \end{pmatrix},$$

Similar schemes are available for the D3Q15 [23], D3Q19 [23] and D3Q27 [124] lattices.

5.1.1.4 Cascaded Lattice Boltzmann Equation (CLBE)

The cascaded LBE (CLBE) collision operator, in a similar form to the Multiple Relaxation Time (MRT) schemes, operates by calculating and colliding moments of the distribution function. In this case, central moments of the distribution function are used[35], i.e.

$$\tilde{M}_{pqr} = \sum_i f_i \left(e_{i,x} - u_x\right)^p \left(e_{i,y} - u_y\right)^q \left(e_{i,z} - u_z\right)^r \tag{5.19}$$

where p, q and r can each take values of 0, 1 or 2, and $M_{pqr} = \sum_i f_i e_{i,x}^p e_{i,y}^q e_{i,z}^r$ is the equivalent raw moment. Some algebra can be used to find relationships between higher-order raw and central moments, e.g. for M_{110}

$$M_{110} = \tilde{M}_{110} + \rho u_x u_y.$$

It is possible to express distribution functions in terms of raw or central moments[80, 81], but a simpler implementation is to use transformation matrices in a similar fashion to moment-based MRT schemes [31, 30], i.e.

$$\vec{\tilde{M}} = \mathbf{NT}\vec{f} \tag{5.20}$$

where $\mathbf{N}$ is an additional (lower triangular) matrix used to transform raw moments into central moments. The central moments can be collided in a similar fashion to MRT:

$$\vec{\tilde{M}}(\vec{x}, t^+) = \vec{\tilde{M}}(\vec{x}, t) - \mathbf{\Lambda}(\vec{\tilde{M}}(\vec{x}, t) - \vec{\tilde{M}}^{eq}(\vec{x}, t)), \tag{5.21}$$

and inverses of the two transformation matrices can be used to convert post-collisional central moments back to distribution functions. While combinations of second-order raw moments can be used, it is possible to use these moments without combining them if the collision matrix $\mathbf{\Lambda}$ is modified from a purely diagonal matrix to a block diagonal one with additional contributions from other second-order central moments[5]. This simplifies the lower diagonal second transformation matrix $\mathbf{N}$, making it easier to determine its inverse $\mathbf{N}^{-1}$. The transformations from distribution functions to central moments (and back) can be implemented by either applying the two matrices in turn (with the raw moments as an intermediary vector) or applying the product of the two matrices directly to the distribution functions. The matrix products ($\mathbf{NT}$ and $\mathbf{T}^{-1}\mathbf{N}^{-1}$) can be found analytically for each lattice scheme: either these or the second transform matrix ($\mathbf{N}$) and its inverse have to be determined for each lattice point to correspond with its local velocity.

The collision parameters for zeroth and first order central moments (density and momentum) are fixed to conserve macroscopic hydrodynamics, while those for second order central moments (relaxation frequencies) can be specified by the user to set both kinematic and bulk viscosities. Collision parameters for most third and fourth order central moments will affect simulation stability and thus can be specified by the user, while others are generally fixed to relax higher order moments (including fifth and sixth orders) directly to equilibrium values[81].

In a similar fashion to MRT collisions, local equilibrium central moments can be obtained using the relationship

$$\vec{\tilde{M}}^{eq} = \mathbf{NT}\vec{f}^{eq}. \tag{5.22}$$

The Maxwell-Boltzmann (general) local equilibrium distribution function (Equation 4.4) can be used in the above expression[5], which gives the following values for even orders of central moments:

$$\tilde{M}^{eq}_{000} = \rho \tag{5.23}$$
$$\tilde{M}^{eq}_{200} = \tilde{M}^{eq}_{020} = \tilde{M}^{eq}_{002} = c_s^2\rho \tag{5.24}$$
$$\tilde{M}^{eq}_{220} = \tilde{M}^{eq}_{202} = \tilde{M}^{eq}_{022} = c_s^4\rho \tag{5.25}$$
$$\tilde{M}^{eq}_{222} = c_s^6\rho \tag{5.26}$$

and zero for all other (odd) orders. Inverting these equilibrium moments results in higher order approximations than those typically used for BGK, TRT and MRT simulations (fourth order in velocity for two dimensions, up to sixth order in three dimensions).

For instance, a D2Q9 lattice will use the following transformation matrix to convert distribution functions to raw moments [32]:

$$\mathbf{T} = \begin{bmatrix} 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 \\ 0 & -1 & -1 & -1 & 0 & 1 & 1 & 1 & 0 \\ 0 & 1 & 0 & -1 & -1 & -1 & 0 & 1 & 1 \\ 0 & 1 & 1 & 1 & 0 & 1 & 1 & 1 & 0 \\ 0 & 1 & 0 & 1 & 1 & 1 & 0 & 1 & 1 \\ 0 & -1 & 0 & 1 & 0 & -1 & 0 & 1 & 0 \\ 0 & 1 & 0 & -1 & 0 & -1 & 0 & 1 & 0 \\ 0 & 1 & 0 & -1 & 0 & 1 & 0 & 1 & 0 \\ 0 & -1 & 0 & 1 & 0 & 1 & 0 & 1 & 0 \end{bmatrix}$$

and the following transformation matrix to convert raw moments to central moments:

$$\mathbf{N} = \begin{bmatrix} 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ -u_x & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ -u_y & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ u_x^2 & -2u_x & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ u_y^2 & 0 & -2u_y & 0 & 1 & 0 & 0 & 0 & 0 \\ u_xu_y & -u_y & -u_x & 0 & 0 & 1 & 0 & 0 & 0 \\ -u_x^2u_y & .2u_xu_y & u_x^2 & -u_y & 0 & -2u_x & 1 & 0 & 0 \\ -u_xu_y^2 & u_y^2 & 2u_xu_y & 0 & -u_x & -2u_y & 0 & 1 & 0 \\ u_x^2u_y^2 & -2u_xu_y^2 & -2u_x^2u_y & u_y^2 & u_x^2 & 4u_xu_y & -2u_y & -2u_x & 1 \end{bmatrix}.$$

These produce nine raw and central moments

$$\vec{M} = (M_{00}, M_{10}, M_{01}, M_{20}, M_{02}, M_{11}, M_{21}, M_{12}, M_{22})^T$$
$$\vec{\tilde{M}} = \left(\tilde{M}_{00}, \tilde{M}_{10}, \tilde{M}_{01}, \tilde{M}_{20}, \tilde{M}_{02}, \tilde{M}_{11}, \tilde{M}_{21}, \tilde{M}_{12}, \tilde{M}_{22}\right)^T$$

where $M_{00} = \tilde{M}_{00} = \rho$, $M_{10} = j_x = \rho u_x$, $\tilde{M}_{10} = M_{10} - \rho u_x$, $M_{01} = j_y = \rho u_y$, $\tilde{M}_{01} = M_{01} - \rho u_y$ etc. The collision matrix takes the following block diagonal form:

$$\mathbf{\Lambda} = \text{diag}\left(1, 1, 1, \begin{bmatrix} s_+ & s_- \\ s_- & s_+ \end{bmatrix} \omega, \omega_3, \omega_3, \omega_4\right)$$

where $\omega = \tau_f^{-1}$, $\omega_b = \tau_{f,bulk}^{-1}$, ω_3 and ω_4 are third and fourth order relaxation frequencies, $s_+ = \frac{1}{2}(\omega_b + \omega)$ and $s_- = \frac{1}{2}(\omega_b - \omega)$. The effective local equilibrium distribution function for this lattice scheme (derived by inverse shifting and transformation of the equilibrium central moments) can be expressed as:

$$f_i^{eq}(\rho, \vec{u}) = w_i\rho\left[1 + \frac{3(\hat{e}_i \cdot \vec{u})}{c^2} + \frac{9(\hat{e}_i \cdot \vec{u})^2}{2c^4} - \frac{3u^2}{2c^2} + \frac{9(\hat{e}_i \cdot \vec{u})^3}{2c^6} - \frac{9(\hat{e}_i \cdot \vec{u})u^2}{2c^4} + \frac{9u_x^2u_y^2}{4c^8}\left(3e_{i,x}^2 - 1\right)\left(3e_{i,y}^2 - 1\right)\right].$$

Interfacial and external forces can be applied using each of the four forcing methods. The Martys-Chen method is applied by substituting the fluid velocity in the moment transformation matrix $\mathbf{N}$ with $\vec{v} = \vec{u} + \frac{\tau_f \Delta t \vec{F}}{\rho}$. The source term for the Exact Difference Method can be applied after calculating post-collisional distribution functions using the following local equilibrium distribution function with third-order velocity terms in Equation 5.2[80, 81]:

$$f_i^{eq}(\rho, \vec{u}) = w_i \rho \left[1 + \frac{3\left(\hat{e}_i \cdot \vec{u}\right)}{c^2} + \frac{9\left(\hat{e}_i \cdot \vec{u}\right)^2}{2c^4} - \frac{3u^2}{2c^2} + \frac{9\left(\hat{e}_i \cdot \vec{u}\right)^3}{2c^6} - \frac{9\left(\hat{e}_i \cdot \vec{u}\right) u^2}{2c^4} \right].$$

To apply forces using Guo's or He's method, central moments of the source terms:

$$\tilde{S}_{pqr} = \sum_i S_i \left(e_{i,x} - u_x\right)^p \left(e_{i,y} - u_y\right)^q \left(e_{i,z} - u_z\right)^r \tag{5.27}$$

can be calculated and added to the collided central moments in the following manner:

$$\Delta \tilde{M}_{pqr} = \left(1 - \tfrac{1}{2}\omega_{pqr}\right) \tilde{S}_{pqr} \Delta t \tag{5.28}$$

where ω_{pqr} is the relaxation frequency for the given central moment. Along with replacing the fluid velocity with $\vec{v} = \vec{u} + \frac{\vec{F}\Delta t}{2\rho}$, these modified collided central moments can be used in the same expressions for post-collisional distribution functions.

The central moment form of the Guo source terms can be calculated from matrix multiplications of the standard source terms, i.e. $\vec{\tilde{S}} = \mathbf{NT}\vec{S}$. The equivalent central moment source terms for He forcing can be derived using the same approach as local equilibria, i.e. using the general form of local equilibrium (Equation 4.4) in Equation 5.5 [32, 30]. Certain odd-numbered orders of central moment have non-zero values, i.e.

$$\begin{aligned}
\tilde{S}_{100} &= F_x \\
\tilde{S}_{010} &= F_y \\
\tilde{S}_{001} &= F_z \\
\tilde{S}_{120} = \tilde{S}_{102} = &= c_s^2 F_x \\
\tilde{S}_{210} = \tilde{S}_{012} = &= c_s^2 F_y \\
\tilde{S}_{201} = \tilde{S}_{021} = &= c_s^2 F_z \\
\tilde{S}_{122} &= c_s^4 F_x \\
\tilde{S}_{212} &= c_s^4 F_y \\
\tilde{S}_{221} &= c_s^4 F_z.
\end{aligned}$$

Similar schemes are available for the D3Q19 and D3Q27 [30] lattices: no scheme for D3Q15 is currently available.

5.1.2 Rheological models

Any of the above collision operators can be applied for fluids with either constant or variable relaxation times. Since the relaxation time τ_f can be related to dynamic viscosity μ as follows:

$$\tau_f = \frac{\nu}{c_s^2} + \frac{1}{2} = \frac{\mu}{\rho c_s^2} + \frac{1}{2}, \tag{5.29}$$

it is possible to determine a grid point's relaxation time from its viscosity, which can be described as a function of the local shear rate $\dot{\gamma}$:

$$\dot{\gamma} = \sqrt{2 \sum_{\alpha,\beta} S_{\alpha\beta} S_{\alpha\beta}}, \tag{5.30}$$

where $S_{\alpha\beta}$ is the rate-of-strain tensor for dimensions α and β and defined as sums of velocity gradients:

$$S_{\alpha\beta} = \frac{1}{2}\left(\frac{\partial u_\beta}{\partial x_\alpha} + \frac{\partial u_\alpha}{\partial x_\beta}\right). \tag{5.31}$$

To avoid calculating (non-local) velocity gradients, it is possible to obtain the rate-of-strain tensors from momentum flux tensors[4, 9]:

$$S_{\alpha\beta} = -\frac{3}{2\tau_f \rho \Delta t}\Pi_{\alpha\beta} = -\frac{3}{2\tau_f \rho \Delta t}\sum_i \left(f_i - f_i^{eq}\right) e_{i,\alpha} e_{i,\beta} \tag{5.32}$$

which can be calculated locally from distribution functions. While the above expression can be used directly for BGK collisions, it can be generalised for collision operators with more than one relaxation time:

$$S_{\alpha\beta} = \frac{3}{2\rho\Delta t}\sum_i e_{i,\alpha} e_{i,\beta} \sum_j \Lambda_{ij}\left(f_j - f_j^{eq}\right). \tag{5.33}$$

For TRT collisions, the above expression reduces to:

$$S_{\alpha\beta} = -\frac{3}{2\rho\Delta t}\sum_i e_{i,\alpha} e_{i,\beta}\left[\tau_{f,p}\left(f_i - f_i^{eq}\right) + \tau_{f,m}\left(f_j - f_j^{eq}\right)\right] \tag{5.34}$$

where in this case $\hat{e}_j = -\hat{e}_i$ (i.e. j is the conjugate link to i). The equivalent expression for MRT collisions[17] is given as:

$$S_{\alpha\beta} = -\frac{3}{2\rho\Delta t}\sum_i e_{i,\alpha} e_{i,\beta} \sum_j \left(\mathbf{T}^{-1}\mathbf{\Lambda}\mathbf{T}\right)_{ij}\left(f_j - f_j^{eq}\right), \tag{5.35}$$

and similarly, the expression for cascaded LBE collisions can be given as:

$$S_{\alpha\beta} = -\frac{3}{2\rho\Delta t}\sum_i e_{i,\alpha} e_{i,\beta} \sum_j \left(\mathbf{T}^{-1}\mathbf{N}^{-1}\mathbf{\Lambda}\mathbf{N}\mathbf{T}\right)_{ij}\left(f_j - f_j^{eq}\right). \tag{5.36}$$

All of the above expressions for $S_{\alpha\beta}$ are implicit with the relaxation time τ_f (the property being determined) included in some form. While it may be possible to obtain a direct expression for the relaxation time for the BGK case given a particular rheological expression for the viscosity [88], this cannot realistically be achieved for other collision forms due to their complexity. The previously obtained relaxation time is therefore used in the expressions for $S_{\alpha\beta}$ and the calculated value of shear rate $\dot{\gamma}$ is used to update τ_f. (This could be made self-consistent by subsequently repeating the rate-of-strain tensor calculations and iterating τ_f until it converges within a small tolerance, but this is generally not necessary as convergence is usually obtained in a small number of timesteps.)

Seven rheological models are included in DL_MESO:

1. Simple Newtonian fluid (constant kinematic viscosity/relaxation time):

$$\mu = \rho\nu_0$$

2. (Density-dependent) Newtonian fluid (constant dynamic viscosity):

$$\mu = \mu_0$$

3. Power-law fluid:

$$\mu = K\dot{\gamma}^{n-1}$$

4. Bingham plastic[8]:

$$\mu = \mu_0 + \frac{\sigma_y}{\dot{\gamma}} \quad (\sigma = \mu\dot{\gamma} > \sigma_y)$$

5. Herschel-Bulkley plastic[51]:

$$\mu = K\dot{\gamma}^{n-1} + \frac{\sigma_y}{\dot{\gamma}} \quad (\sigma = \mu\dot{\gamma} > \sigma_y)$$

6. Casson fluid[16]:

$$\mu = \left(\sqrt{\mu_0} + \sqrt{\frac{\sigma_y}{\dot{\gamma}}}\right)^2 \quad (\sigma = \mu\dot{\gamma} > \sigma_y)$$

7. Carreau-Yasuda fluid[12, 14, 142]:

$$\mu = \mu_\infty + (\mu_0 - \mu_\infty)\left(1 + (\lambda\dot{\gamma})^d\right)^{\frac{n-1}{d}}$$

where ν_0 is a constant kinematic viscosity, μ_0 is constant dynamic viscosity (at zero shear rate), μ_∞ is constant dynamic viscosity (at infinite shear rate), K is a consistency viscosity (when shear rate is equal to unity), σ_y is a yield stress (below which no flow may occur for Bingham-like plastics and Casson fluids), λ is a yield relaxation time for the Carreau-Yasuda fluid, and n and d are indices acting on the shear rate.

Each fluid can use any one of these seven rheological models, although the simple Newtonian fluid is used by default if the model is not specified. The relaxation time or frequency still needs to be set for fluids using models other than constant kinematic viscosities, as these will be used to initialise local relaxation times for each lattice point.

Setting the power n to 1 for power-law and Carreau-Yasuda fluids will revert to Newtonian (constant dynamic viscosity) behaviour, while the same for Herschel-Bulkley plastics will revert to Bingham plastic behaviour. If the yield stress σ_y in Bingham plastics or Casson fluids is set to zero, the fluid will revert to Newtonian behaviour; setting the same for Herschel-Bulkley plastics will revert to power-law behaviour (if $n \neq 1$) or Newtonian behaviour (if $n = 1$).

The discontinuities in the Bingham plastic, Herschel-Bulkley plastic and Casson fluid models can be overcome by multiplying the yield stress contribution by a growth exponential function[90], i.e.

- Bingham plastic:
$$\mu = \mu_0 + \frac{\sigma_y}{\dot{\gamma}}\left(1 - e^{-m\dot{\gamma}}\right)$$
- Herschel-Bulkley plastic:
$$\mu = K\dot{\gamma}^{n-1} + \frac{\sigma_y}{\dot{\gamma}}\left(1 - e^{-m\dot{\gamma}}\right)$$
- Casson fluid:
$$\mu = \left(\sqrt{\mu_0} + \sqrt{\frac{\sigma_y}{\dot{\gamma}}}\left(1 - e^{-m\dot{\gamma}}\right)\right)^2$$

where m is the stress growth exponent, with large values giving good approximations to the original functions.

5.1.3 Propagation algorithms

The simplest implementation, the *two-lattice algorithm*, involves the use of a temporary array to copy post-collisional distribution functions to their new positions, which are subsequently copied back to the main distribution function array. While this particular method is clear, easy to understand and can be applied throughout the system's lattice points in any order, its drawbacks include the use of two loops for propagation and array copying, two large arrays for the distribution functions at each lattice node and significant amounts of time expended in memory access.

An alternative, more efficient implementation of propagation is the *swap algorithm*[85], whereby this process is carried out by systematic swapping of pairs of collided distribution function values. To make this easier to implement, the lattice links are organised so that the conjugate link j to link i (i.e. $\hat{e}_j = -\hat{e}_i$) is equal to $i + \frac{m-1}{2}$ for $i = 1 \ldots \frac{m-1}{2}$. Looping i between 1 and $\frac{m-1}{2}$ the post-collisional distribution functions for each lattice point $f_i(\vec{x})$ are initially swapped with their conjugate values $f_j(\vec{x})$. then at each point the value $f_j(\vec{x})$ is then swapped with $f_i(\vec{x} + \hat{e}_i\Delta t)$.

These sets of swaps can be carried out either in two separate steps or in one go. The use of separate swap steps requires two sweeps through the domain, but the order in which the distribution functions are swapped does not matter and no boundary domain is necessary for serial calculations. Simultaneous swapping cannot make use of automatic periodic boundary conditions (thus a non-zero domain boundary is required) and requires lattice

links to be additionally ordered so that the first half are all directed to lattice points that have previously gone through at least the first swap stage, but only a single sweep through the domain is required.

By default the serial and multithreaded versions of DL_MESO_LBE use propagations based on two separate swap steps, while the parallel version (without multithreading) uses the combined (simultaneous) swap step. The user can use combined swap propagations in serial (without multithreading) using a form of the customised code supplied with DL_MESO that includes additional routines to deal with boundary halos: more details of this version can be found in the DL_MESO Developer Manual.

5.2 Boundary conditions

To apply boundary conditions to a Lattice Boltzmann Equation simulation, the distribution functions f_i at boundary lattice points have to be modified or replaced during each time step to give the required fluid velocity or pressure/concentration/temperature. This may take place either between the collision and propagation stages or at the end of each time step. A space property is used to define the boundary conditions for each lattice node in the system.

5.2.1 Periodic

Periodic boundaries are used to simulate bulk fluids sufficiently far away from the actual boundaries of a real physical system so that surface effects can be neglected. As the fluid moves out of one face of the system volume, it reappears on the opposite face with the same velocity, density etc.

DL_MESO_LBE applies periodic boundary conditions in two different ways depending on the size of the boundary domains. If there is no boundary domain (the default for serial running), periodic boundary conditions are automatically applied during the propagation step by using a modulo function to adjust the destination of distribution functions leaving the system so they are placed at the opposite side. No periodic boundary using this implementation needs to be defined by the space property, which can be left equal to zero as for the bulk fluid.

For systems that include a non-zero boundary domain size, the distribution functions at the edges of the actual system are copied into the buffer at the opposing sides before collisions and propagation take place. This requires the use of the space property to determine the location of the domain buffer and identify boundary halo grid points.

Seven different boundary types exist in DL_MESO_LBE: two variants of bounce-back can be applied to any lattice point in a system, while outflow boundaries can only be applied at planar surfaces (or edges in two-dimensional systems). Constant velocity or density boundary conditions can be applied at the extremes of a rectangular or cuboidal system, including concave edges and corners along with planar surfaces, using one of four schemes: Zou-He (non-equilibrium bounce-back), Inamuro, regularised and kinetic.

5.2.2 On-grid bounce-back

The on-grid bounce-back condition applies a no-slip condition (i.e. zero fluid velocity) at a boundary that lies halfway between grid points. This is applied after the propagation stage by reversing the distribution functions sitting on each wall node ($\vec{x}_w$), i.e.

$$f_i(\vec{x}_w, t) = f_j(\vec{x}_w, t) \tag{5.37}$$

where j is the opposite lattice link to i, i.e. $\hat{e}_j = -\hat{e}_i$. The reflection of distribution functions occurs *on-grid*. On-grid bounce-back is a first-order approximation of the boundary condition[87], i.e. the error is proportional to the lattice spacing Δx, but it is completely local (i.e. only uses distribution functions at the wall node).

5.2.3 Mid-grid bounce-back

Like the on-grid version, the mid-grid bounce-back condition applies a no-slip condition at a boundary halfway between lattice points[123]. This is applied by assigning post-collisional distribution functions to the wall node based on those values at neighbouring points, i.e.

$$f_i(\vec{x}_w, t^+) = f_j(\vec{x}_w + \hat{e}_i \Delta t, t^+). \tag{5.38}$$

This method essentially applies the actual reflection halfway between timesteps and is a spatially second-order method, although it is weakly non-local due to its use of distribution functions from neighbouring nodes.

5.2.4 Outflow

Free outflow conditions can be achieved by using a Neumann boundary condition (zero gradient) to determine the distribution functions for lattice links pointing back into the fluid[78]. A first-order approximation for the gradient uses neighbouring fluid points to determine relevant distribution functions for inward pointing lattice links, i.e.

$$f_i(\vec{x}_w, t^+) = f_i(\vec{x}_w + \Delta x, t^+)$$

while a second-order approximation uses the neighbour and next-neighbour grid points in the fluid, i.e.

$$f_i(\vec{x}_w, t^+) = 2f_i(\vec{x}_w + \Delta x, t^+) - f_i(\vec{x}_w + 2\Delta x, t^+).$$

Either of these conditions can be used for all distribution functions (density, solute concentration, temperature), regardless of mesoscopic interaction type or collision operator.

5.2.5 Zou-He

To specify velocities or densities (pressures), solute concentrations or temperatures at planar boundaries, the Zou-He method[148] is available in DL_MESO_LBE. This is based upon applying the bounce-back rule to the non-equilibrium distribution functions, i.e.

$$f_i^{(1)}(\vec{x}_w, t) = f_j^{(1)}(\vec{x}_w, t) \tag{5.39}$$

where $f_i^{(1)} = f_i - f_i^{eq}$, with the equilibrium distribution function f_i^{eq} as a function of density and velocity. This function can be used to determine the missing wall velocity or density along with the known distribution function values. For instance, for a top edge with a known velocity $\vec{u}_w$ using the D2Q9 lattice scheme, the wall density and missing distribution functions (all for the boundary node at $\vec{x}_w$) are given as:

$$\begin{aligned}
\rho_w &= \frac{f_0 + f_2 + f_6 + 2(f_1 + f_7 + f_8)}{1 + u_{w,y}} \\
f_4 &= f_8 - \frac{2\rho_w u_{w,y}}{3} \\
f_3 &= f_7 + \tfrac{1}{2}(f_6 - f_2) - \tfrac{1}{2}\rho_w u_{w,x} - \tfrac{1}{6}\rho_w u_{w,y} \\
f_5 &= f_1 - \tfrac{1}{2}(f_6 - f_2) + \tfrac{1}{2}\rho_w u_{w,x} - \tfrac{1}{6}\rho_w u_{w,y}
\end{aligned}$$

A simplified form of the Zou-He boundary for constant velocity conditions is also available, which simply applies the non-equilibrium bounce-back rule for all unknown distribution functions without considering tangential corrections, e.g. for the top boundary

$$\begin{aligned}
\rho_w &= \frac{f_0 + f_2 + f_6 + 2(f_1 + f_7 + f_8)}{1 + u_{w,y}} \\
f_4 &= f_8 - \frac{2\rho_w u_{w,y}}{3} \\
f_3 &= f_7 - \tfrac{1}{6}\rho_w u_{w,x} - \tfrac{1}{6}\rho_w u_{w,y} \\
f_5 &= f_1 + \tfrac{1}{6}\rho_w u_{w,x} - \tfrac{1}{6}\rho_w u_{w,y}
\end{aligned}$$

The other form, specifying the wall fluid density, requires the calculation of the wall velocity, which can be simplified by setting non-orthogonal velocity components to zero. For the analogous example of a constant density at the top wall for D2Q9, the same equations for f_4, f_3 and f_5 can be used together with

$$\rho_w u_{w,x} = 0, \rho_w u_{w,y} = f_0 + f_2 + f_6 + 2(f_1 + f_7 + f_8) - \rho_w.$$

Different local equilibrium distribution functions may result in slightly different expressions to those above, e.g. when using cascaded LBE collisions, some terms with higher order powers of velocity will result. A notable exception is Swift free-energy-based calculations: while the expressions for local equilibria include additional terms, these cancel out when differences are applied.

One complication for three-dimensional lattices is the requirement to apply the non-equilibrium bounce-back to all unknown distribution functions, which ordinarily over-specifies the system but can be counteracted using transverse momentum corrections for directions other than orthogonal to the boundary, which are non-zero for e.g. shearing flows. It should be noted that if the wall velocity is set to zero, the boundary condition reduces to on-grid bounce-back.

Concave corners (in two or three dimensions) and edges (in three-dimensions) can also be dealt with in a similar fashion, although this approach requires both density and velocity to be specified: in the case of constant velocity boundaries, the density can be taken from a nearby fluid lattice point, while the velocity is set to zero for constant density boundaries. For instance, in the case of a two-dimensional bottom-left corner, the non-equilibrium bounce-back rule can be applied to distribution functions along the neighbouring edges:

$$\begin{aligned} f_6 &= f_2 + \tfrac{2}{3}\rho_w u_{w,x} \\ f_8 &= f_4 + \tfrac{2}{3}\rho_w u_{w,y}, \end{aligned}$$

the diagonal link between these two re-entering the fluid can be found from the x- and y-components of momentum:

$$f_7 = f_3 + \tfrac{1}{6}\rho_w u_{w,x} + \tfrac{1}{6}\rho_w u_{w,y},$$

and the remaining diagonal links (considered 'buried' as they neither originate from nor re-enter the fluid) can be found to satisfy both the momentum and fluid density:

$$\begin{aligned} f_1 &= \tfrac{1}{2}(\rho_w - f_0) - f_2 - f_3 - f_4 - \tfrac{1}{2}\rho_w u_{w,x} - \tfrac{1}{3}\rho_w u_{w,y} \\ f_5 &= \tfrac{1}{2}(\rho_w - f_0) - f_2 - f_3 - f_4 - \tfrac{1}{3}\rho_w u_{w,x} - \tfrac{1}{2}\rho_w u_{w,y}. \end{aligned}$$

For three-dimensional systems with fewer governing equations than unknown distribution functions, each unknown distribution function beyond those along other boundaries (edges or planar surfaces) can be considered to consist of unknown contributions for wall density (Q_0) and components of momentum ($\vec{Q}$). Edges can use non-equilibrium bounce-back with a tangential correction in a similar fashion to planar surfaces: this tangential correction can also be used in the 'buried' links. All other contributions can be assumed to apply to each unknown link weighted by w_i [18], i.e. $f_i = w_i\left(Q_0 + \hat{e}_i \cdot \vec{Q}\right)$. These can be inserted into the equations for density and momentum, allowing the unknowns to be solved based on known properties.

For solute concentrations and temperatures, the wall velocity will already either be set or calculated from the fluid density calculation. In this situation, all links that are not directly orthogonal to the boundary use the non-equilibrium bounce-back rule, while the distribution functions for the orthogonal link are calculated by ensuring the sum of all distribution functions gives the specified concentration or temperature.

DL_MESO_LBE includes implementations of the Zou-He boundary condition for all four lattice schemes: D2Q9, D3Q15, D3Q19 and D3Q27. Planar surfaces, concave edges and corners can be dealt with using this method, whose expressions are extended from those above to incorporate fluid forces.

5.2.6 Inamuro

To specify velocities or densities (pressures), solute concentrations or temperatures at planar, edge or corner boundaries, the Inamuro method[59, 58] can be used in DL_MESO_LBE. This is based upon substituting the

unknown distribution functions for a boundary point with local equilibrium values, but using an adjusted density/concentration/temperature and (if applying for the fluids) an additional slip velocity in components tangential to the boundary. These properties are calculated by substituting the equilibrium distribution function for the unknown populations into the sum of distribution functions (equal to required density/concentration/temperature) and the sums of first moments (equal to the momentum).

For instance, for a bottom edge with a known velocity $\vec{u}_w$ using the D2Q9 lattice scheme, the wall density, adjusted density and slip velocity (all for the boundary node at $\vec{x}_w$) are given as:

$$\begin{aligned}
\rho_w &= \frac{f_0 + f_2 + f_6 + 2(f_3 + f_4 + f_5)}{1 - u_{w,y}} \\
\rho' &= \frac{6\left(\rho_w u_{w,y} + f_3 + f_4 + f_5\right)}{1 + 3u_{w,y} + 3u_{w,y}^2} \\
u_{s,x} &= \frac{1}{1 + 3u_{w,y}}\left[\frac{6\left(\rho_w u_{w,x} + f_2 + f_3 - f_5 - f_5\right)}{\rho'}\right] - u_{w,x}
\end{aligned}$$

with the missing populations ($i = 1, 7, 8$ in this case) given by

$$f_i = w_i \rho' \left[1 + \frac{3\left(\hat{e}_i \cdot \vec{u}_{ws}\right)}{c^2} + \frac{9\left(\hat{e}_i \cdot \vec{u}_{ws}\right)^2}{2c^4} - \frac{3u_{ws}^2}{2c^2}\right]$$

where $\vec{u}_{ws} = \vec{u}_w + \vec{u}_s$. A constant density boundary uses the same relationships to calculate the orthogonal velocity component, adjusted velocity and tangential velocity components, making the assumption that the tangential wall velocities (excluding slip terms) are zero, while solute concentration and temperature conditions are implemented by using the known wall velocity and calculating an adjusted concentration/temperature. Since Swift free-energy-based interactions and CLBE collisions make use of different local equilibrium distribution functions, different expressions for adjusted densities and slip velocities are required.

A similar approach can be used for corners and edges, applying the modified local equilibrium distribution function to all unknown links (including 'buried' ones). Three-dimensional edges for fluids apply a tangential velocity component, while all corners use the adjusted density as its only unknown quantity, which can be found to satisfy the required wall density and momentum.

DL_MESO_LBE includes implementations of the Inamuro boundary condition for all four lattice schemes. Planar surfaces, concave edges and corners can be dealt with using this method, whose expressions are extended from those above to incorporate fluid forces.

5.2.7 Regularised

To specify velocities or densities (pressures) at planar, edge and corner boundaries, the regularised boundary condition method[69] is available in DL_MESO_LBE. This is based upon calculating post-collisional values for non-equilibrium momentum flux tensors, which can then be used to calculate replacement values for distribution functions. The non-equilibrium momentum flux tensor can be calculated as follows:

$$\begin{aligned}
\Pi_{\alpha\beta}^{neq} &= \Pi_{\alpha\beta} - \Pi_{\alpha\beta}^{eq} \\
&= \sum_I e_{i,\alpha} e_{i,\beta} f_i - \sum_I e_{i,\alpha} e_{i,\beta} f_i^{eq} \\
&= \sum_I e_{i,\alpha} e_{i,\beta} f_i - \rho u_\alpha u_\beta - \rho c_s^2 \delta_{\alpha\beta} \\
&= -2c_s^2 \rho \tau_f S_{\alpha\beta}
\end{aligned}$$

where $S_{\alpha\beta} = \frac{1}{2}\left(\frac{\partial u_\beta}{\partial x_\alpha} + \frac{\partial u_\alpha}{\partial x_\beta}\right)$ is the strain rate tensor. The same non-equilibrium distribution function bounce-back condition as for simplified Zou-He boundaries ($f_i - f_i^{eq} = f_j - f_j^{eq}$) can be used to obtain the unknown distribution functions. For instance, for a bottom edge with known velocity $\vec{u}_w$ using the D2Q9 lattice scheme

(V??CETF), the various terms of the non-equilibrium momentum flux tensor are given as follows:

$$\begin{aligned}\Pi_{xx}^{neq} &= f_2 + f_6 + 2\left(f_3 + f_5\right) + \rho_w\left(\tfrac{1}{3}u_{w,y} - u_{w,x}^2 - \tfrac{1}{3}\right)\\ \Pi_{yy}^{neq} &= 2\left(f_3 + f_4 + f_5\right) + \rho_w\left(u_{w,y} - u_{w,y}^2 - \tfrac{1}{3}\right)\\ \Pi_{yx}^{neq} = \Pi_{xy}^{neq} &= 2\left(f_3 - f_5\right) + \rho_w\left(\tfrac{1}{3}u_{w,x} - u_{w,x}u_{w,y}\right)\end{aligned}$$

.

The replacement distribution functions are determined from local equilibrium distribution functions for the given density and velocity, plus an additional term dependent on the non-equilibrium momentum flux tensor or the strain rate tensor:

$$\begin{aligned}f_i - f_i^{eq}(\rho_w, \vec{u}_w) &\approx -\frac{\rho_w w_i \tau_f}{c_s^2}\left(\vec{e}_i\vec{e}_i - c_s^2\mathbf{I}\right) : \mathbf{S}\\ &= \frac{w_i}{2c_s^2}\left(\vec{e}_i\vec{e}_i - c_s^2\mathbf{I}\right) : \mathbf{\Pi}^{neq}.\end{aligned}$$

Combining the above expression for the regularised non-equilibrium distribution function with the expression for the non-equilibrium momentum flux tensor gives a local boundary scheme. For instance, for the above boundary condition for D2Q9, the replacement distribution functions are given as:

$$f_i = f_i^{(eq)}(\rho_w, \vec{u}_w) + \frac{w_i}{2c_s^4}\left[\left(e_{i,x}e_{i,x} - c_s^2\right)\Pi_{xx}^{neq} + \left(e_{i,y}e_{i,y} - c_s^2\right)\Pi_{yy}^{neq} + 2e_{i,x}e_{i,y}\Pi_{xy}^{neq}\right]$$

and are to be calculated for all link vectors. A constant density boundary uses the same relationships to calculate the non-equilibrium momentum flux tensor and distribution functions, albeit assuming that the tangential wall velocities are zero. Since incompressible, cascaded lattice Boltzmann and Swift free-energy models use different local equilibrium distribution functions, different expressions for the momentum flux tensor and distribution functions to those shown above are required.

DL_MESO_LBE includes implementations of regularised boundary conditions for all four lattice schemes. Planar surfaces, concave edges and corners can be dealt with using this method, whose expressions for momentum flux tensors are extended to incorporate fluid forces [104], adding $\frac{\Delta t}{2}\left(u_{w,\beta}F_\alpha + u_{w,\alpha}F_\beta\right)$ to the above expressions. This kind of boundary condition cannot be used for solute concentrations or temperatures as non-equilibrium momentum flux tensors cannot be defined from the relevant distribution functions.

5.2.8 Kinetic

To specify velocities or densities (pressures) at planar, edge and corner boundaries, the kinetic boundary condition method[3] is available in DL_MESO_LBE. This is based on the assumption that particles re-entering the system forget their history and their velocities are renormalised using a Maxwellian distribution. This scheme produces a non-zero, physically realistic slip velocity at a given wall boundary, which is especially suited to higher flow speeds and/or low density, rarefied gas flows[55].

The boundary condition is applied by replacing the incoming distribution functions with values determined from local equilibrium distribution functions with an adjusted density ρ', i.e.

$$f_i = f_i^{eq}\left(\rho', \vec{u}_w\right).$$

The adjusted density is obtained by ensuring mass conservation between distribution functions leaving and re-entering the system. For instance, for a bottom edge in the D2Q9 lattice scheme for mildly compressible fluids (either V??CETF or P??CETF), the adjusted density is given by:

$$\rho' = \frac{f_3 + f_4 + f_5}{\frac{1}{\rho_w}\left(f_1^{eq}\left(\rho_w, \vec{u}_w\right) + f_7^{eq}\left(\rho_w, \vec{u}_w\right) + f_8^{eq}\left(\rho_w, \vec{u}_w\right)\right)} = \frac{6\left(f_3 + f_4 + f_5\right)}{1 + 3u_{w,y} + 3u_{w,y}^2}.$$

For incompressible and Swift free-energy systems, only the parts of the distribution functions affected by the (variable) density are used in the expression for adjusted density, as other terms in the equilibrium distribution functions used in these cases do not depend upon this quantity.

DL_MESO_LBE includes implementations of the kinetic boundary condition for all four lattice schemes, with different variants for incompressible fluids, cascaded LBE collisions and Swift free-energy mesoscopic interactions due to changes in local equilibrium distribution functions. Planar surfaces, concave edges and corners can be dealt with using this method, whose expressions are extended from those above to incorporate fluid forces, primarily in calculations of density or velocity for planar surfaces and 'buried' links for edges and corners.

5.3 Mesoscale interactions

DL_MESO_LBE includes a number of algorithms that can be used to apply interactions between fluid components at the mesoscale, most commonly to model immiscible fluids. The user should take care to ensure the correct model is used for the type of system being modelled.

If `lbin.init` files are used to insert fluid drops into the simulation domain, DL_MESO_LBE includes the option of carrying out equilibration to allow the shapes of drops to settle by modelling the system without external body forces and boundaries imposing specific velocities, densities, solute concentrations or temperatures. This option can be selected using the `equilibration_step` keyword in the `lbin.sys` file.

5.3.1 Shan-Chen pseudopotential model

The Shan-Chen model[108, 109] models interactions between multiple phases and components by calculating pairwise interaction potentials. These potentials use an 'effective mass' or pseudopotential for each component, ψ_a, which is a function of density. Defining g_{ab} as the interaction coefficient between components a and b, the interfacial force on fluid component a (due to interactions with itself and other components) can be defined as

$$\vec{F}^a = -c_s^2 \psi_a \sum_b g_{ab} \nabla \psi_b. \tag{5.40}$$

If a single component (a) is considered, this force gives the following equation of state[106]

$$P = \rho c_s^2 + \frac{1}{2} c_s^2 g_{aa} \psi_a^2(\rho) \tag{5.41}$$

and interfacial tension[107]

$$\sigma_{ab} = -\frac{e_4 g_{aa} c_s^4 (\Delta x)^2}{2} \int_{-\infty}^{+\infty} \left(\frac{d\psi_a}{dz} \right)^2 dz \tag{5.42}$$

where e_4 is a lattice-dependent constant and z distance along the normal from the phase interface. Both the equation of state and the interfacial tension are governed by the form of pseudopotential used for interactions.

To carry out collisions for multiple fluid systems, the fluid velocity used to calculate local equilibrium distribution functions is weighted by the relaxation times, i.e.

$$\vec{u} = \frac{\sum_a \sum_i f_i^a \hat{e}_i \tau_{f,a}^{-1}}{\sum_a \sum_i f_i^a \tau_{f,a}^{-1}}. \tag{5.43}$$

Separate routines looping through the grid points, `fCollision*ShanChen`, are available to make use of this definition of fluid velocity and all variants of the Shan-Chen model employ these routines.

Original Shan-Chen model

An approximation of the spatial gradient of the pseudopotential can be made for a lattice-based system[70]:

$$c_s^2 \Delta t \nabla \psi_b \approx \sum_i w_i \psi_b \left(\vec{x} + \hat{e}_i \right) \hat{e}_i.$$

Its substitution into equation 5.40 gives an easily calculable expression for the overall force on fluid component a:

$$\vec{F}^a = -\psi_a \left(\vec{x} \right) \sum_b g_{ab} \sum_i w_i \psi_b \left(\vec{x} + \hat{e}_i \right) \hat{e}_i. \tag{5.44}$$

which can be applied in a Lattice Boltzmann Equation simulation with any suitable forcing algorithm. The additional fluid-solid interaction forces can be given in a similar manner, i.e.

$$\vec{F}^a_{wet} = -\psi_a(\vec{x})\, g_{a,wall} \sum_i w_i s(\vec{x}+\hat{e}_i)\,\hat{e}_i. \tag{5.45}$$

Shan-Chen model with quadratic terms

An alternative form of force was devised by Zhang and Chen[146]:

$$\vec{F} = -\nabla(p - \rho c_s^2)$$

where the quantity $p - \rho c_s^2$ can be seen to be related to the square of the pseudopotential. Re-expressing the force for fluid component a in terms of the pseudopotential for all components gives:

$$\vec{F}^a = -\frac{c_s^2}{2} \sum_b g_{ab} \nabla \psi_b^2$$

which is equivalent to the force applied by the original Shan-Chen model and gives the same equation of state and interfacial tension. For lattice-based calculations, this force can be approximated by

$$\vec{F}^a = -\frac{1}{2} \sum_b g_{ab} \sum_i w_i \psi_b^2(\vec{x}+\hat{e}_i)\,\hat{e}_i. \tag{5.46}$$

A weighted combination of the Shan-Chen and Zhang-Chen forces can be shown to give greater accuracy below the critical point for vapour/liquid systems[40, 66], i.e.

$$\vec{F}^a = -\psi_a(\vec{x}) \sum_b g_{ab}\beta_{ab} \sum_i w_i \psi_b(\vec{x}+\hat{e}_i)\,\hat{e}_i - \frac{1}{2} \sum_b g_{ab}(1-\beta_{ab}) \sum_i w_i \psi_b^2(\vec{x}+\hat{e}_i)\,\hat{e}_i \tag{5.47}$$

where β_{ab} is the weighting factor for components a and b, which can be tuned for various equations of state[2]. The original Shan-Chen and Zhang-Chen models can be recovered by setting β_{ab} equal to 1 and 0 respectively. Values of β_{aa} for all fluid species can be specified in `lbin.sys` by a single directive: by default the values of β_{ab} for $a \neq b$ are set to 1, in common with models involving multiple components and phases that use this approach[57].

Surface wetting

Additional forces on the fluids due to surface wetting can be included with either form of the Shan-Chen model. A switching function $s(\vec{x})$ can be defined to represent the existence ($s(\vec{x}) = 1$) or absence ($s(\vec{x}) = 0$) of surfaces at a lattice site $\vec{x}$. The gradient of the switching function can then be used to mimic an adhesive force between the solid and fluid. Three forms are available in DL_MESO_LBE according to the fluid property used at the lattice site:

- Density-based interactions[84]:

$$\vec{F}^a_{wet}(\vec{x}) = -g_{a,wall}\rho_a(\vec{x}) \sum_i w_i s(\vec{x}+\hat{e}_i)\hat{e}_i, \tag{5.48}$$

- Potential-based interactions[99, 100]:

$$\vec{F}^a_{wet}(\vec{x}) = -g_{a,wall}\psi_a(\vec{x}) \sum_i w_i s(\vec{x}+\hat{e}_i)\hat{e}_i, \tag{5.49}$$

- Screened potential interactions[75]:

$$\vec{F}^a_{wet}(\vec{x}) = -g_{a,wall}\psi_a(\vec{x}) \sum_i w_i \phi(\vec{x}) s(\vec{x}+\hat{e}_i)\hat{e}_i \tag{5.50}$$

where $\phi(\vec{x})$ is a screening function, currently set as $\phi(\vec{x}) = \psi_a(\vec{x})$ but can be changed by the user (see the DL_MESO Developer Manual). The interaction coefficient between component a and the wall, $g_{a,wall}$, can be given a positive (negative) value to reduce (increase) its wetting.

[2]The original parameter used in [66], A, is equivalent to $\frac{1}{2}(1-\beta)$.

Pseudopotentials

Pseudopotentials are calculated for each fluid species at each lattice point. While the same form of pseudopotential is normally used for every fluid species, it is possible to use different pseudopotential forms for different species.

There are two subsets of pseudopotentials available in DL_MESO_LBE. The first subset relies on the value of g_{ab} as an effective system 'temperature'. These include:

- The original Shan-Chen (1993) model[108]:

$$\psi_a = \rho_0^a \left[1 - \exp\left(-\frac{\rho^a}{\rho_0^a}\right)\right] \tag{5.51}$$

- The Shan-Chen thermodynamically consistent (1994) model[109]:

$$\psi_a = \psi_0^a \exp\left(-\frac{\rho_0^a}{\rho^a}\right) \tag{5.52}$$

- The Qian (1995) model[98]:

$$\psi_a = \frac{\rho_0^a \rho^a}{\rho_0^a + \rho^a} \tag{5.53}$$

- The density model:

$$\psi_a = \rho^a \tag{5.54}$$

where ρ^a, ρ_0^a and ψ_0^a are the local fluid density (for the lattice point), reference density and maximum value for the pseudopotential of component a respectively. A critical point can be determined for each of these pseudopotentials: for interaction parameters below this value, separation of the fluid into dense (liquid) and light (vapour) phases will occur.

The original Shan-Chen pseudopotential (1993) is a switch-like form that gives a sharp transition between liquid and vapour phases for $g_{aa} < g_{c,a} = -\frac{4}{\rho_0^a}$ (with a corresponding critical density $\rho_c^a = \rho_0^a \ln 2$). The second Shan-Chen pseudopotential (1994) guarantees that the pressure tensor normal to an interface is equal to the static bulk pressure and ensures thermodynamic consistency: the critical value of the interaction coefficient is $g_{c,a} = -\frac{\rho_0^a}{\psi_0^a} e^2$ and the critical density $\rho_c^a = \rho_0^a$. The Qian model gives a transition between liquid and vapour phases for $g_{aa} < g_{c,a} = -\frac{27}{4\rho_0^a}$ (with a corresponding critical density $\rho_c^a = \frac{1}{2}\rho_0^a$). The density model gives a quadratic equation of state and therefore does not give phase separation: this pseudopotential provides no critical point, yields zero pressure when $\rho = -\frac{1}{g_{aa}}$ and is only suitable for low density gases.

The other subset of pseudopotentials are those derived by rearranging equation 5.41 to apply particular equations of state for the fluids[145]:

$$\psi_a = \sqrt{\frac{2\left(P - \rho c_s^2\right)}{g_{aa} c_s^2}} \tag{5.55}$$

where the interaction strength g_{aa} is used to ensure the expression inside the square root is positive. To explicitly use interaction strengths between different fluid species (while setting $g_{aa} = 1$ for all a), the pseudopotential can be rewritten using a signum function as

$$\psi_a = \mathrm{sgn}\left(P - \rho c_s^2\right) \sqrt{2\left|\frac{P}{c_s^2} - \rho\right|}. \tag{5.56}$$

The function in this form can be used to apply phase separation for a single component, and can be changed by the user by modifying the subroutine `fCalcPotential_ShanChen`.

Seven equations of state (ideal, cubic and rigid-sphere) are provided in DL_MESO using this form of pseudopotential:

Table 5.1: Equation of state parameters

Equation of state	a	b	T_c	P_c	ρ_c
van der Waals	$\frac{27R^2T_c^2}{64p_c}$	$\frac{RT_c}{8p_c}$	$\frac{8a}{27bR}$	$\frac{a}{27b^2}$	$\frac{1}{3b}$
Redlich-Kwong	$0.42748\frac{R^2T_c^{2.5}}{p_c}$	$0.08664\frac{RT_c}{p_c}$	$0.34504\left(\frac{a}{bR}\right)^{\frac{2}{3}}$	$0.02989\left(\frac{a^2R}{b^5}\right)^{\frac{1}{3}}$	$\frac{0.25992}{b}$
Carnahan-Starling-van der Waals	$0.49639\frac{R^2T_c^2}{p_c}$	$0.18729\frac{RT_c}{p_c}$	$0.37731\frac{a}{bR}$	$0.07067\frac{a}{b^2}$	$\frac{0.52178}{b}$
Soave-Redlich-Kwong	$0.42748\frac{R^2T_c^2}{p_c}$	$0.08664\frac{RT_c}{p_c}$	$0.20268\frac{a}{bR}$	$0.01756\frac{a}{b^2}$	$\frac{0.25992}{b}$
Peng-Robinson	$0.45724\frac{R^2T_c^2}{p_c}$	$0.07780\frac{RT_c}{p_c}$	$0.17014\frac{a}{bR}$	$0.01324\frac{a}{b^2}$	$\frac{0.25308}{b}$
Carnahan-Starling-Redlich-Kwong	$0.46112\frac{R^2T_c^{2.5}}{p_c}$	$0.10483\frac{RT_c}{p_c}$	$0.37248\left(\frac{a}{bR}\right)^{\frac{2}{3}}$	$0.03905\left(\frac{a^2R}{b^5}\right)^{\frac{1}{3}}$	$\frac{0.33258}{b}$

- Ideal gas:

$$P = \rho RT$$

- van der Waals:

$$P = \frac{\rho RT}{1-b\rho} - a\rho^2$$

- Carnahan-Starling-van der Waals[13]:

$$P = \rho RT\left(\frac{1+\phi+\phi^2-\phi^3}{(1-\phi)^3}\right) - a\rho^2$$

- Redlich-Kwong[101]:

$$P = \frac{\rho RT}{1-b\rho} - \frac{a\rho^2}{\sqrt{T}\,(1+b\rho)}$$

- Soave-Redlich-Kwong[118]:

$$P = \frac{\rho RT}{1-b\rho} - \frac{a\alpha(T_r,\omega)\rho^2}{1+b\rho}$$

- Peng-Robinson[91]:

$$P = \frac{\rho RT}{1-b\rho} - \frac{a\alpha(T_r,\omega)\rho^2}{1+2b\rho-b^2\rho^2}$$

- Carnahan-Starling-Redlich-Kwong[13]:

$$P = \rho RT\left(\frac{1+\phi+\phi^2-\phi^3}{(1-\phi)^3}\right) - \frac{a\rho^2}{\sqrt{T}\,(1+b\rho)}$$

where R is the universal gas constant, a and b are species-dependent coefficients, α is a function of reduced temperature ($T_r = \frac{T}{T_c}$, where T_c is the critical temperature for the species) and acentric factor ω for the Soave-Redlich-Kwong and Peng-Robinson equations, and $\phi = \frac{b\rho}{4}$ for the Carnahan-Starling equations. Two versions of pseudopotentials for the Soave-Redlich-Kwong, Peng-Robinson and Carnahan-Starling-Redlich-Kwong equations exist: one set for systems with constant temperature (i.e. when thermal fields are not used) and another for variable temperature systems (i.e. when thermal fields are explicitly modelled).

The coefficients a and b can be derived from critical temperatures and pressures, since the pressure curve forms a saddle point at the critical point (i.e. both first and second derivatives of pressure by density are zero): approximate relationships between the coefficients and critical properties are given in Table 5.1.

The functions of reduced temperature and acentric factor for the Soave-Redlich-Kwong and Peng-Robinson equations are empirical equations fitted to experimental data. The original functions are given as:

- Soave-Redlich-Kwong:

$$\alpha(T_r,\omega) = \left[1+(0.480+1.574\omega-0.176\omega^2)(1-\sqrt{T_r})\right]^2$$

- Peng-Robinson:

$$\alpha(T_r,\omega) = \left[1 + (0.37464 + 1.54226\omega - 0.26992\omega^2)(1 - \sqrt{T_r})\right]^2$$

but alternatives can be substituted by modifying the subroutines that read in parameters and calculate pseudopotentials (see the DL_MESO Developer Manual).

5.3.2 Lishchuk continuum-based model

The Lishchuk model[77, 47] is a modified form of the 'chromodynamic' model devised by Gunstensen and Rothman[44], which models interactions between multiple components by applying forces based on the existence of those components. This continuum-based model is primarily suited for systems where hydrodynamic interactions dominate and no further fluid separation takes place.

A phase field is defined between components a and b as

$$\rho^N_{ab} = \frac{\rho^a - \rho^b}{\rho^a + \rho^b}. \tag{5.57}$$

noting that $\rho^N_{ba} = -\rho^N_{ab}$. The interfacial normal between the phases can be determined from a first-order spatial differential of this quantity, i.e.

$$\hat{n}_{ab} = \frac{\nabla\rho^N_{ab}}{|\nabla\rho^N_{ab}|}. \tag{5.58}$$

This differential can either be determined non-locally (for lattice points without neighbouring walls) using a fourth-order accurate isotropic scheme[70, 48]:

$$\nabla\rho^N_{ab} = \frac{1}{c_s^2\Delta t}\sum_i w_i\hat{e}_i\rho^N_{ab}\left(\vec{x} + \hat{e}_i\Delta t\right) \tag{5.59}$$

or it can be approximated locally using phase fields determined from individual links[120], i.e.

$$\nabla\rho^N_{ab} = -\frac{1}{c_s^2\Delta t}\sum_i w_i\hat{e}_i\left(\frac{f^a_i - f^b_i}{f^a_i + f^b_i}\right) \tag{5.60}$$

which may be sufficiently accurate for calculating the interfacial normal vector but not the phase field differential itself. This form is not as numerically robust as the non-local method and has to be used solely in interfacial regions (i.e. when $|\rho^N_{ab}| \leq 1-\epsilon$, where ϵ is a small value) to avoid numerical instabilities. The interfacial normals can subsequently be used to determine the interfacial tension forces acting between fluid components, which can be applied by either calculating an interfacial force and using any available forcing algorithm, or by directly calculating a forcing term to be applied during collisions.

Rather than colliding each fluid separately, a single 'achromatic' distribution function is defined as the sum of distribution functions for all fluids

$$f_i = \sum_a f^a_i \tag{5.61}$$

the sum of which is equal to the density of all fluids, ρ. This distribution function is collided using any valid method with all interaction forces combined together and collision operators interpolated according to local mass fraction, e.g.

$$\frac{1}{\tau_f} = \frac{1}{\rho}\sum_a \frac{\rho^a}{\tau^a_f}. \tag{5.62}$$

If the two-relaxation-time (TRT) scheme is used and a value for the 'magic number' (Λ_{eo}) is specified, this value will be used along with the calculated value of τ_f (τ^+) for the combined fluids to give the antisymmetric relaxation time (τ^-).

After the achromatic fluid is collided, the fluids are segregated to produce post-collisional distribution functions for each fluid. This is achieved using the D'Ortona algorithm[24], which gives a non-zero boundary thickness between the fluids and reduces non-physical effects such as pinning of drops to the lattice, spatial anisotropy

in interfacial tension and spurious microcurrents. The equation for the post-collisional segregated distribution function for fluid a[119] is given as

$$f_i^a\left(\vec{x},t^+\right)=\frac{\rho^a}{\rho}f_i\left(\vec{x},t^+\right)+\sum_{b\neq a}\beta^{ab}w_i\frac{\rho^a\rho^b}{\rho^2}\hat{e}_i\cdot\hat{n}_{ab} \tag{5.63}$$

where β^{ab} is the segregation parameter that controls the width of the diffuse boundary between phases and $\beta^{ba}=\beta^{ab}$.

This class of mesoscale interaction method can directly simulate a specified interfacial tension (σ_{ab}), which is related to the lattice-based parameter g_{ab} by

$$\sigma_{ab}=\frac{4g_{ab}\nu^2\rho_0}{c_s^4\left(2\tau_f-1\right)^2\Delta x} \tag{5.64}$$

using the physical mean density (ρ_0), kinematic viscosity (ν) and relaxation time (τ_f) of a reference fluid (e.g. the continuous fluid for the system).

5.3.2.1 Original Lishchuk algorithm

The original Lishchuk algorithm requires the calculation of local curvatures at interfaces between fluids (expressed between fluids a and b), which can be obtained from the relevant interfacial normals:

$$K_{ab}=-\nabla_S\cdot\hat{n}_{ab}. \tag{5.65}$$

This quantity is used to calculate the interfacial tension force acting between the two fluids

$$\vec{F}^{ab}=\frac{1}{2}g_{ab}K_{ab}\nabla\rho_{ab}^N, \tag{5.66}$$

where the first-order spatial differential of the phase field can be obtained from the interfacial normal, the densities of the two components and of all fluids (ρ) and the segregation parameter between the two fluids (β^{ab}):

$$\nabla\rho_{ab}^N=\frac{4\beta^{ab}\rho^a\rho^b}{\rho^3}\hat{n}_{ab}. \tag{5.67}$$

The interfacial forces can be applied using any available forcing algorithm, although the Guo forcing algorithm is generally recommended to reduce spurious microcurrents originating from the interface. This version of the algorithm is robust for interactions between pairs of fluids, but cannot readily be used at grid points where more than two fluids are in contact as the curvature cannot be defined at these points. (Note that this form of the Lishchuk algorithm will still work for multiple fluid systems, provided no more than two fluids are in contact at any given lattice point: this can be ensured by setting g_{ab} between drops of different fluids to significantly higher values than those between the background fluid and the drop fluids.)

A similar form of force can be used at solid boundaries to apply surface wetting effects[25], assuming that fluid 0 is the background fluid and wets the walls, applying an uncompensated Young stress on fluid a:

$$\vec{F}_{wet}^{0a}=-\frac{1}{2}g_{wall,a}\nabla_{S,wall}\rho_{0a}^N=\frac{2g_{wall,a}\beta^{0a}\rho^0\rho^a}{\rho^3}\left(\hat{n}_w\left(\hat{n}_{0a}\cdot\hat{n}_w\right)-\hat{n}_{0a}\right), \tag{5.68}$$

where $\hat{n}_w$ is the normal vector to the solid surface. This force is applied parallel to solid walls and the parameter $g_{wall,a}$ can be selected to vary the contact angle of drops of fluid a.

This form of the algorithm calculates interfacial normals from non-local spatial gradients of phase indices as per equation 5.59. As interfacial normals are needed in boundary points for collisions and curvature calculations, an additional communication per timestep is needed in parallel running. All interaction forces between pairs of fluid species are carried out using equation 5.66 with non-local calculations of interfacial curvatures, and surface wetting forces are calculated with equation 5.68. Modified collision routines are used to apply achromatic collisions, forces using the required forcing scheme and D'Ortona segregation of the fluids.

5.3.2.2 Lishchuk-Spencer algorithm

A variant of the Lishchuk algorithm exists to avoid calculating curvatures between pairs of fluids[119], which uses the following force acting between components a and b:

$$\vec{F}^{ab} = -2g_{ab}\beta^{ab}\nabla\cdot\left(\frac{\rho^a\rho^b}{\rho^3}\hat{n}_{ab}\hat{n}_{ab}\right). \tag{5.69}$$

This force produces the same interfacial tension between the components as the original force (equation 5.66), but also produces a reduction in density of up to $\frac{g_{ab}\beta^{ab}}{2c_s^2}$ in the interfacial region of mixed fluids and larger interfacial microcurrents than the original Lishchuk method. This method can be shown to give correct behaviour at points with more than two fluids, where it is difficult to accurately define the curvature.

Compared to the original Lishchuk algorithm, the interaction forces between pairs of fluid species are calculated differently, using equation 5.69 in this case with non-local calculations of gradients of fluid densities and interfacial normals. This algorithm is otherwise very similar: the interfacial normals are calculated in the same way with non-local spatial gradients of phase indices, the same surface wetting forces are calculated and the same form of collisions are used to apply forces, achromatic collisions and fluid segregation.

5.3.2.3 Lishchuk-Spencer-tensor algorithm

A further variant of the above Lishchuk-Spencer algorithm avoids calculating interfacial forces between fluid species (as well as interfacial curvatures) and instead uses a direct forcing term for the interaction between the components[119, 120], i.e.

$$F_i^{ab} = \frac{w_i\beta^{ab}g^{ab}\rho^a\rho^b}{c_s^4\rho^3\tau_f\Delta t}\left(\hat{n}_{ab}\hat{n}_{ab}-\mathbf{I}\right):\left(\hat{e}_i\hat{e}_i-c_s^2\mathbf{I}\right). \tag{5.70}$$

which requires no further forcing term nor adjustment of the equilibrium fluid velocity used for collision calculations.

Along with the standard non-local calculations of interfacial normals (equation 5.59), this forcing term makes up the so-called 'Spencer tensor' variant of the Lishchuk algorithm, which provides correct interactions at grid points with more than two fluids as well as avoiding the drop in total density in the interfacial region. In parallel calculations, the use of the forcing term also avoids the need to communicate interfacial forces between processor cores.

As before, this form of the algorithm calculates interfacial normals (non-locally) and surface wetting forces in the same way as the original Lishchuk algorithm. No interaction forces in bulk fluids are calculated: the forcing terms in equation 5.70 are used instead in this form of the algorithm during the collision stage alongside achromatic fluid collisions and D'Ortona segregation.

5.3.2.4 Local form of Lishchuk algorithm

Another variant of the Lishchuk-Spencer algorithm makes use of the direct forcing term (equation 5.70) and locally-calculated interfacial normals (equation 5.60). This provides a completely local form of the Lishchuk model, i.e. interactions at a lattice point do not require information from any neighbouring lattice points. Like the Lishchuk-Spencer and Spencer tensor methods, this form of the Lishchuk algorithm will give correct behaviour for more than two fluids in contact, albeit with larger microcurrents due to the local approximation of the phase index gradient used to calculate interfacial normals.

The approximate phase index gradients are used to calculate interfacial normals for all fluid lattice points in the main domain and in boundary halos: as such, no communication of interfacial normals is required for parallel running. Note that phase index gradients are not calculated at grid points with boundary conditions: surface wetting forces are calculated at these points using equation 5.68 and also do not need to be communicated between processor cores. As for the Spencer tensor method, collisions are applied achromatically to all fluids with the direct forcing term (equation 5.70) and D'Ortona segregation.

5.3.3 Free-energy models

Free-energy lattice Boltzmann models[126, 125] are available for two-phase and two-fluid systems that give thermodynamic consistency at the continuum. The Swift-Osborne-Yeomans approach exploits the van der Waals formulation of a two-phase isothermal fluid by defining a Landau free energy density function Ψ as:

$$\Psi = \int \left[\frac{\kappa}{2} \left| \nabla \rho \right|^2 + \psi \left(\rho \right) \right] dV, \tag{5.71}$$

where the first term is the energy penalty in building density gradients (with κ as an interfacial surface tension parameter) and the second term describes the bulk free energy density. The chemical potential is given by the derivative of the free energy density function with density ($\mu = \frac{d\Psi}{d\rho}$). The non-local pressure is related to the free energy density function by

$$P = \rho \frac{d\Psi}{d\rho} - \Psi = P_0 - \kappa \rho \nabla^2 \rho - \frac{\kappa}{2} \left| \nabla \rho \right|^2, \tag{5.72}$$

where

$$P_0 = \rho \frac{d\psi}{d\rho} - \psi \tag{5.73}$$

is the equation of state for the fluid. The full pressure tensor in a non-uniform fluid

$$P_{\alpha\beta} = P \delta_{\alpha\beta} + \kappa \partial_\alpha \rho \partial_\beta \rho \tag{5.74}$$

includes non-diagonal terms that are related to interfacial surface tension effects at equilibrium. Since the pressure tensor is equal to the second moment of the local equilibrium distribution function with lattice links ($P_{\alpha\beta} = \sum_i f_i e_{i\alpha} e_{i\beta}$), it can be incorporated into the lattice Boltzmann equation model by adding terms to the local equilibrium distribution function. As this type of free-energy model affects the definition of the distribution function, it is currently unavailable if cascaded LBE collision models are in use, but it is available for BGK, TRT and MRT collisions.

One fluid vapour/liquid systems

For the square lattices used in DL_MESO_LBE, the local equilibrium distribution function for a vapour/liquid system[94, 26, 93, 67] is given as:

$$\begin{aligned} f_i^{eq} = w_i^{00} \rho + w_i \left[\rho \left\{ \left(\hat{e}_i \cdot \vec{u} \right) + \frac{3}{2} \left(\hat{e}_i \cdot \vec{u} \right)^2 - \frac{1}{2} u^2 \right\} + \lambda \left\{ 3 \left(\hat{e}_i \cdot \vec{u} \right) \left(\hat{e}_i \cdot \nabla \rho \right) + \left[\gamma_i \left(\hat{e}_i \cdot \hat{e}_i \right) + \delta_i \right] \left(\vec{u} \cdot \nabla \rho \right) \right\} \right] \\ + w_i^p P_0 - w_i^t \kappa \rho \nabla^2 \rho + w_i^{xx} \kappa \left(\partial_x \rho \right)^2 + w_i^{yy} \kappa \left(\partial_y \rho \right)^2 + w_i^{zz} \kappa \left(\partial_z \rho \right)^2 \\ + w_i^{xy} \kappa \partial_x \rho \partial_y \rho + w_i^{xz} \kappa \partial_x \rho \partial_z \rho + w_i^{yz} \kappa \partial_y \rho \partial_z \rho \end{aligned} \tag{5.75}$$

where w_i, w_i^{00}, γ_i, δ_i, w_i^p, w_i^t, w_i^{xx}, w_i^{yy}, w_i^{zz}, w_i^{xy}, w_i^{xz} and w_i^{yz} are lattice-dependent weighting parameters and are selected to ensure that $\sum_i f_i^{eq} = \rho$: values for the two-dimensional lattice D2Q9 are given in Table 5.2. The parameter λ is used to ensure Galilean invariance[54] and is defined as

$$\lambda = \nu \left(1 - 3 \left(\frac{\Delta t}{\Delta x} \right)^2 \frac{\partial P_0}{\partial \rho} \right), \tag{5.76}$$

which reduces to zero if the standard equation of state for a lattice gas, $P_0 = c_s^2 \rho$, is used. The derivatives and Laplacian of density can be calculated using stencils to give contributions from neighbouring sites, which can be chosen to reduce spurious velocities[94]. For a two-dimensional problem, the best choice of stencils is

$$\partial_x = \frac{1}{12 \Delta x} \begin{bmatrix} -1 & 0 & 1 \\ -4 & 0 & 4 \\ -1 & 0 & 1 \end{bmatrix} \tag{5.77}$$

$$\nabla^2 = \frac{1}{6(\Delta x)^2} \begin{bmatrix} 1 & 4 & 1 \\ 4 & -20 & 4 \\ 1 & 4 & 1 \end{bmatrix} \tag{5.78}$$

Table 5.2: Weighting parameters for D2Q9 free-energy model

i	w_i	w_i^{00}	γ_i	δ_i	w_i^p	w_i^t	w_i^{xx}	w_i^{yy}	w_i^{zz}	w_i^{xy}	w_i^{xz}	w_i^{yz}
0	$\frac{4}{3}$	1	0	$-\frac{21}{8}$	$-\frac{5}{3}$	$-\frac{5}{3}$	$-\frac{1}{6}$	$-\frac{1}{6}$	0	0	0	0
1	$\frac{1}{12}$	0	$\frac{3}{2}$	$-\frac{3}{2}$	$\frac{1}{12}$	$\frac{1}{12}$	$-\frac{1}{24}$	$-\frac{1}{24}$	0	$-\frac{1}{4}$	0	0
2	$\frac{1}{3}$	0	$\frac{3}{2}$	$-\frac{3}{2}$	$\frac{1}{3}$	$\frac{1}{3}$	$\frac{1}{3}$	$-\frac{1}{6}$	0	0	0	0
3	$\frac{1}{12}$	0	$\frac{3}{2}$	$-\frac{3}{2}$	$\frac{1}{12}$	$\frac{1}{12}$	$-\frac{1}{24}$	$-\frac{1}{24}$	0	$\frac{1}{4}$	0	0
4	$\frac{1}{3}$	0	$\frac{3}{2}$	$-\frac{3}{2}$	$\frac{1}{3}$	$\frac{1}{3}$	$-\frac{1}{6}$	$\frac{1}{3}$	0	0	0	0
5	$\frac{1}{12}$	0	$\frac{3}{2}$	$-\frac{3}{2}$	$\frac{1}{12}$	$\frac{1}{12}$	$-\frac{1}{24}$	$-\frac{1}{24}$	0	$-\frac{1}{4}$	0	0
6	$\frac{1}{3}$	0	$\frac{3}{2}$	$-\frac{3}{2}$	$\frac{1}{3}$	$\frac{1}{3}$	$\frac{1}{3}$	$-\frac{1}{6}$	0	0	0	0
7	$\frac{1}{12}$	0	$\frac{3}{2}$	$-\frac{3}{2}$	$\frac{1}{12}$	$\frac{1}{12}$	$-\frac{1}{24}$	$-\frac{1}{24}$	0	$\frac{1}{4}$	0	0
8	$\frac{1}{3}$	0	$\frac{3}{2}$	$-\frac{3}{2}$	$\frac{1}{3}$	$\frac{1}{3}$	$-\frac{1}{6}$	$\frac{1}{3}$	0	0	0	0

with the matrices centred on the grid point of interest. Collisions can be applied using any available algorithm, although modifications are required for some equilibrium moments (energy, squared energy, stress tensor and fourth-order moments) in multiple-relaxation-time (MRT) schemes to add terms for the bulk pressure (P_0), interfacial tension (κ) and Galilean invariance (λ).

Beyond the default equation of state for an ideal lattice gas ($P_0 = \rho c_s^2$), the following equations of state can be used in DL_MESO_LBE with Swift free-energy interactions:

- Shan-Chen (1993) model[108]:
$$P_0 = \rho c_s^2 + \tfrac{1}{2} c_s^2 g \rho_0^2 \left(1 - e^{-\frac{\rho}{\rho_0}}\right)^2$$
- Shan-Chen (1994) model[109]:
$$P_0 = \rho c_s^2 + \tfrac{1}{2} c_s^2 g \psi_0^2 e^{-\frac{2\rho_0}{\rho}}$$
- Qian (1995) model[98]:
$$P_0 = \rho c_s^2 + \tfrac{1}{2} c_s^2 g \frac{\rho_0^2 \rho^2}{(\rho_0 + \rho)^2}$$
- Density (rho) model:
$$P_0 = \rho c_s^2 + \tfrac{1}{2} c_s^2 g \rho^2$$
- Ideal gas:
$$P_0 = \rho R T$$
- van der Waals:
$$P_0 = \frac{\rho R T}{1 - b\rho} - a\rho^2$$
- Carnahan-Starling-van der Waals[13]:
$$P_0 = \rho R T \left(\frac{1 + \phi + \phi^2 - \phi^3}{(1-\phi)^3}\right) - a\rho^2$$
- Redlich-Kwong[101]:
$$P_0 = \frac{\rho R T}{1 - b\rho} - \frac{a\rho^2}{\sqrt{T}\,(1 + b\rho)}$$
- Soave-Redlich-Kwong[118]:
$$P_0 = \frac{\rho R T}{1 - b\rho} - \frac{a\alpha(T_r, \omega)\rho^2}{1 + b\rho}$$
- Peng-Robinson[91]:
$$P_0 = \frac{\rho R T}{1 - b\rho} - \frac{a\alpha(T_r, \omega)\rho^2}{1 + 2b\rho - b^2\rho^2}$$

- Carnahan-Starling-Redlich-Kwong[13]:

$$P_0 = \rho RT\left(\frac{1+\phi+\phi^2-\phi^3}{(1-\phi)^3}\right) - \frac{a\rho^2}{\sqrt{T}\,(1+b\rho)}$$

As for Shan-Chen pseudopotentials based on cubic and Carnahan-Starling equations of state, either a heat transfer lattice (see below) can be used to find local temperature, or a constant temperature can be chosen for the entire system if no temperature field is used.

The first and second order gradients of fluid density are calculated for all grid points, while the bulk pressure P_0 and Galilean invariance corrector λ can also be calculated at each grid point for the given equation of state. These values can then be used to calculate local equilibrium distribution functions used for BGK and TRT collisions: alternatively, equilibrium distribution functions transformed to moment space for MRT collisions can be obtained.

Two fluid systems

To extend the Swift free-energy approach to two-fluid systems, a second distribution function can be defined to give a concentration field, i.e.

$$\sum_i g_i = \phi = \frac{\rho_a - \rho_b}{\rho_a + \rho_b} \tag{5.79}$$

where ϕ represents the density difference between fluids a and b and can have values between -1 and 1. The standard distribution function becomes the total density of both fluids, i.e. $\sum_i f_i = \rho = \rho_a + \rho_b$. As with density, the Landau free energy can consist of an energy penalty due to concentration gradients and a bulk free energy density that can additionally be a function of concentration, i.e.

$$\Psi = \int \left[\frac{\kappa}{2}\left(|\nabla\rho|^2 + |\nabla\phi|^2\right) + \psi\left(\rho,\phi\right)\right] dV. \tag{5.80}$$

The non-local pressure now includes derivatives of the concentration, making the equation of state for the fluids

$$P_0 = \rho\frac{\partial\psi}{\partial\rho} + \phi\frac{\partial\psi}{\partial\phi} - \psi. \tag{5.81}$$

It should be noted that if the equation of state is of the form $P_0 = c_s^2\rho + f(\phi)$ (i.e. ideal lattice gases with a concentration dependence), all spatial derivatives of ρ in the bulk free energy density and density local equilibrium distribution function get set equal to zero[95]. The derivative of the Landau free energy with respect to concentration

$$\frac{\partial\Psi}{\partial\phi} = \frac{\partial\psi}{\partial\phi} - \kappa\nabla^2\phi = \mu \tag{5.82}$$

gives the chemical potential between the two fluids.

For two-fluid systems, the local equilibrium distribution function for the density field should include concentration gradients for all terms involving κ. The most general form is given by substituting the derivatives of density in equation 5.75 with sums of derivatives of density and concentration[125], i.e.

$$\begin{aligned} f_i^{eq} = w_i^{00}\rho + w_i\left[\rho\left\{\left(\hat{e}_i\cdot\vec{u}\right) + \frac{3}{2}\left(\hat{e}_i\cdot\vec{u}\right)^2 - \frac{1}{2}u^2\right\} + \lambda\left\{3\left(\hat{e}_i\cdot\vec{u}\right)\left(\hat{e}_i\cdot\nabla\rho\right) + \left[\gamma_i\left(\hat{e}_i\cdot\hat{e}_i\right) + \delta_i\right]\left(\vec{u}\cdot\nabla\rho\right)\right\}\right] \\ + w_i^p P_0 - w_i^t\kappa(\rho\nabla^2\rho + \phi\nabla^2\phi) + w_i^{xx}\kappa((\partial_x\rho)^2 + (\partial_x\phi)^2) + w_i^{yy}\kappa((\partial_y\rho)^2 + (\partial_y\phi)^2) + w_i^{zz}\kappa((\partial_z\rho)^2 + (\partial_z\phi)^2) \\ + w_i^{xy}\kappa(\partial_x\rho\partial_y\rho + \partial_x\phi\partial_y\phi) + w_i^{xz}\kappa(\partial_x\rho\partial_z\rho + \partial_x\phi\partial_z\phi) + w_i^{yz}\kappa(\partial_y\rho\partial_z\rho + \partial_y\phi\partial_z\phi). \end{aligned} \tag{5.83}$$

If the equation of state is based on ideal lattice gases, the derivatives of density for all terms ($\partial_\alpha\rho$ and $\nabla^2\rho$) involving κ can be set to zero[95]. Combined with the elimination of the λ term (since $\frac{\partial P_0}{\partial\rho} = c_s^2$), this gives a simplified local equilibrium distribution function for density:

$$\begin{aligned} f_i^{eq} = w_i^{00}\rho + w_i\rho\left\{\left(\hat{e}_i\cdot\vec{u}\right) + \frac{3}{2}\left(\hat{e}_i\cdot\vec{u}\right)^2 - \frac{1}{2}u^2\right\} + w_i^p P_0 - w_i^t\kappa\phi\nabla^2\phi \\ + w_i^{xx}\kappa\left(\partial_x\phi\right)^2 + w_i^{yy}\kappa\left(\partial_y\phi\right)^2 + w_i^{zz}\kappa\left(\partial_z\phi\right)^2 + w_i^{xy}\kappa\partial_x\phi\partial_y\phi + w_i^{xz}\kappa\partial_x\phi\partial_z\phi + w_i^{yz}\kappa\partial_y\phi\partial_z\phi. \end{aligned} \tag{5.84}$$

As for single-fluid, two-phase systems, collisions can be applied using any scheme (with corrections to equilibrium moments for MRT schemes), using the concentration to find the relaxation time from values for the two fluids.

The second moment of the concentration distribution function gives

$$\sum_i g_i e_{i\alpha} e_{i\beta} = \Gamma\mu\delta_{ij} + \phi u_\alpha u_\beta \tag{5.85}$$

where Γ is a mobility parameter between the two fluid species. Along with the concentration relaxation time τ_ϕ, the mobility can be expressed as

$$M = \Gamma\Delta t\left(\tau_\phi - \frac{1}{2}\right). \tag{5.86}$$

As before, the concentration tensor can be implemented by selecting an equilibrium distribution function, which for square lattices can be chosen as

$$g_i^{eq} = w_i^{00}\phi + w_i\phi\left\{(\hat{e}_i\cdot\vec{u}) + \frac{3}{2}(\hat{e}_i\cdot\vec{u})^2 - \frac{1}{2}u^2\right\} + w_i^p\Gamma\mu \tag{5.87}$$

using the same weighting parameters as before. Collisions for the concentration distribution function can be applied using any scheme, although the BGK single-relaxation-time scheme is normally sufficient as the magnitude of mobility can be easily controlled using Γ[95].

DL_MESO_LBE currently only has the quartic form for the concentration part of the bulk free energy density and the equation of state, i.e.:

$$\psi = f(\rho) + a\left(-\frac{1}{2}\phi^2 + \frac{1}{4}\phi^4\right)$$

$$P_0 = f(\rho) + a\left(-\frac{1}{2}\phi^2 + \frac{3}{4}\phi^4\right)$$

which provide a double well potential:

$$\mu = a\left(-\phi + \phi^3\right) - \kappa\nabla^2\phi.$$

For this form of potential, the surface tension between the two fluids is given as $\gamma = \sqrt{\frac{8}{9}\kappa a}$ and the interfacial width is $\xi = \sqrt{\frac{\kappa}{a}}$.

The first and second order gradients of fluid density and concentration are calculated for all grid points, while the bulk pressure P_0, Galilean invariance corrector λ and chemical potential μ can also be calculated at each grid point for the given equation of state and form of chemical potential. As for one-fluid systems, these values can be used to calculate local equilibrium distribution functions (BGK and TRT collisions) or moment-transposed versions for density (MRT collisions): the concentration distribution function is always collided using BGK collision schemes and the local equilibrium function for concentration can also be calculated.

Surface wetting

Surface wetting can be accommodated into the free-energy model by an additional term to the Landau free-energy functional[10, 95], i.e.

$$\Psi_s = \int \Phi(\rho_s, \phi_s) dS, \tag{5.88}$$

where $\Phi(\rho_s, \phi_s)$ is a surface free energy density, which can be expressed as power series in surface density ρ_s and surface concentration ϕ_s. Expansions up to quadratic terms, i.e.

$$\Phi(\rho_s, \phi_s) = a_0\rho_s + \frac{1}{2}a_1\rho_s^2 + b_0\phi_s + \frac{1}{2}b_1\phi_s^2$$

are sufficient for surface wetting, with a value of zero for Φ giving a contact angle of 90°. Minimising the free energy gives an equilibrium boundary condition for the surface, e.g. for density (with $\hat{s}$ as the unit normal to the surface pointing into the fluid):

$$\kappa\hat{s}\cdot\nabla\rho_s = \frac{\partial\Phi}{\partial\rho_s} = a_0 + a_1\rho_s, \tag{5.89}$$

which can be applied at the collision step by using the resulting density (or concentration) gradient in the local equilibrium distribution function[95], with one-sided approximations for gradients used to calculate second-order gradients[10]. If a bounce-back boundary condition is used, the velocity in the local equilibrium distribution function is set to zero for the collision step.

5.4 Diffusion and heat transfer

In a similar fashion to multiple fluid systems, the Lattice Boltzmann Equation method can be applied to problems involving diffusion and/or heat transfers by using additional distribution functions for each solute and/or temperature[58, 144].

For a system consisting of a number of dilute solutes along with a bulk fluid, the governing equation for each solute is given as

$$g_i\left(\vec{x}+\hat{e}_i\Delta t, t+\Delta t\right) - g_i\left(\vec{x},t\right) = -\frac{\Delta t}{\tau_s}\left[g_i\left(\vec{x},t\right) - g_i^{eq}\right] \tag{5.90}$$

where g_i is the distribution function for the solute and τ_s the solute relaxation time, which is related to its diffusivity

$$D = \frac{1}{3}\left(\tau_s - \frac{1}{2}\right)\frac{\left(\Delta x\right)^2}{\Delta t}$$

and the Schmidt number can be determined by

$$\mathrm{Sc} = \frac{\nu}{D} = \frac{2\tau_f - 1}{2\tau_s - 1}.$$

Taking the concentration of the solute as $C_s = \sum_i g_i$, the equilibrium distribution function for the solute is given by a simpler form of Equation 4.11:

$$g_i^{eq} = w_i C_s\left[1 + 3\frac{\left(\hat{e}_i\cdot\vec{u}\right)}{c^2}\right] \tag{5.91}$$

where the velocity used is that of the bulk fluid.

Heat transfers can be coupled to the system in a similar manner, using a thermal distribution function h_i and a thermal relaxation time τ_t, which gives the governing equation

$$h_i\left(\vec{x}+\hat{e}_i\Delta t, t+\Delta t\right) - h_i\left(\vec{x},t\right) = -\frac{\Delta t}{\tau_t}\left[h_i\left(\vec{x},t\right) - h_i^{eq}\right] \tag{5.92}$$

The temperature at each lattice point (relative to a mean value) can be determined as the sum of the distribution functions, $T = \sum_i h_i$, which can be used to determine the equilibrium distribution function

$$h_i^{eq} = w_i T\left[1 + 3\frac{\left(\hat{e}_i\cdot\vec{u}\right)}{c^2}\right], \tag{5.93}$$

again using the bulk fluid velocity. The thermal relaxation time is related to the thermal diffusivity

$$\alpha = \frac{1}{3}\left(\tau_t - \frac{1}{2}\right)\frac{\left(\Delta x\right)^2}{\Delta t}$$

with the Prandtl number for the system determined by a ratio of relaxation times, i.e.

$$\mathrm{Pr} = \frac{\nu}{\alpha} = \frac{2\tau_f - 1}{2\tau_t - 1}.$$

It should be noted that if two-relaxation-time (TRT), multiple-relaxation-time (MRT) or cascaded LBE schemes are to be used, these only apply to fluids: all diffusion and heat transfer processes are calculated using the BGK single relaxation time schemes described in this section.

5.4.1 Boussinesq approximation

The coupling of fluid flows to heat transfer described above only produces heat conduction effects. To model convective heat transfer processes, an additional force on the fluid is required to link flow to thermal transport. The most common form is the Boussinesq approximation[45], which applies a buoyancy force on fluid a proportional to the temperature difference:

$$\vec{F}^a = -\rho\vec{g}\beta^a\left(\frac{T-T_0}{T_h-T_l}\right) \tag{5.94}$$

where $\vec{g}$ is gravitational acceleration, β^a is the volumetric expansion coefficient for fluid a, T_h and T_l are respectively the maximum and minimum temperatures of the system and $T_0 = \frac{1}{2}(T_h + T_l)$ is a reference temperature.

DL_MESO_LBE can calculate this force for all lattice points. The product of gravitational acceleration and volumetric expansion ($\vec{g}\beta^a$) for each fluid, as well as the maximum and minimum temperatures T_h and T_l, can be included in the `lbin.sys` file.

5.5 Compressible and incompressible fluids

The standard Lattice Boltzmann Equation scheme is capable of modelling compressible fluids. Incompressible fluids can be modelled by making a simple modification to the local equilibrium distribution function[49]:

$$f_i^{eq} = w_i\left[\rho + \rho_0\left(\frac{3\left(\hat{e}_i\cdot\vec{u}\right)}{c^2} + \frac{9\left(\hat{e}_i\cdot\vec{u}\right)^2}{2c^4} - \frac{3u^2}{2c^2}\right)\right] \tag{5.95}$$

where ρ_0 is the fixed density of the incompressible fluid and the density ρ becomes an analogue to pressure ($\frac{P}{\rho_0} = c_s^2\rho$). While Equation 4.1 is still applicable to calculate ρ, the fluid velocity is now calculated by

$$\rho_0 u_\alpha = \sum_{i=0}^{q} f_i e_{i\alpha} \tag{5.96}$$

DL_MESO_LBE allows users to select incompressible fluids using the keyword `incompressible_fluids` in the `lbin.sys` file. Additional collision and equilibrium distribution function routines are included to allow the user to model incompressible fluids. All of these routines use a specified constant density for each fluid in the system as the value of ρ_0, which can also be specified in the `lbin.sys` file. (Note that incompressible fluids are not available when using Swift free-energy-based interactions or cascaded LBE collisions.)

Chapter 6

DL_MESO_LBE Input and Output Files

6.1 Input files

All user-specified input files for DL_MESO_LBE must be in ANSI text format, with keywords (where necessary) and numerical values separated from each other with spaces or tabs.

Define system: `lbin.sys`

The use of the DL_MESO GUI is recommended for producing `lbin.sys`, although existing files of that name can also be edited. Its format consists of a keyword and an associated numerical parameter on each line separated by spaces or tabs. No allowances are made for typographical errors or abbreviations in keywords, which must be included in full and in the form described below.

Ten keywords are compulsory for all LBE simulations, as these determine the lattice scheme to be used, the number of lattices to use, and the sizes of the system and boundary regions.

keyword:	**meaning:**
space_dimension	sets the number of dimensions in the system (2 or 3)
discrete_speed	sets the number of lattice links per grid point (9, 15, 19 or 27)
number_of_fluid	sets the number of fluid lattices (N_f) for the system (if modelling solutes, this must be set to 1)
number_of_solute	sets the number of solutes (N_c) to be modelled
temperature_scalar	determines whether or not a lattice is needed to model heat transfers (set to 1 if needed, 0 if not)
phase_field	determines whether or not a lattice is needed to represent phase fields (set to 1 if needed, 0 if not)[1]
grid_number_x	sets the number of grid points in the x-dimension
grid_number_y	sets the number of grid points in the y-dimension
grid_number_z	sets the number of grid points in the z-dimension (if a two-dimensional system is modelled, this will be reset to 1)
domain_boundary_width	sets the size of the boundary region (if running DL_MESO in serial, this is usually reset to 0)

Additional keywords can be used to specify the algorithms for collisions, forcing and mesophase interactions, the format and data type for output files, whether fluids are compressible or incompressible, and whether or not restart files should be used. If these are omitted, DL_MESO_LBE will assume that a new simulation is to take place with compressible fluids subjected to BGK (single-relaxation-time) collisions using standard forcing and no mesophase interactions, producing VTK formatted files in big endian binary. (If using the customizable

[1] No multiple fluid phase scheme included in DL_MESO currently requires this lattice.

versions of DL_MESO_LBE, all of these keywords may be omitted except for `incompressible_fluids`, which is required to correctly calculate fluid velocities in initialization and output files and apply boundary conditions.) If all three flags for combining data from processor cores are switched on for three-dimensional calculations (or the x- and y-components are both switched on for two-dimensional systems), MPI-IO will be used to put together data slices in single output files.

keyword:	**meaning:**
collision_type	sets the type of collisions and forcing (`BGK` (0), `BGKEDM` (1), `BGKGuo` (2), `TRT` (3), `TRTEDM` (4), `TRTGuo` (5), `MRT` (6), `MRTEDM` (7), `MRTGuo` (8), `CLBE` (9), `CLBEEDM` (10) or `CLBEGuo` (11) [2])
interaction_type	sets the type of mesophase interactions[3] (`ShanChen` (1), `ShanChenQuadratic` (2), `Lishchuk` (10), `LishchukSpencer` (11), `LishchukSpencerTensor` (12), `LishchukLocal` (13) or `Swift` (20))
output_format	sets the format for output files (`VTK` (0), `LegacyVTK` (1), `Plot3D` (2))
output_type	sets the data type for output files (`Binary` (0), `Text` or `ANSI` (1))
output_combine_x	combines data from processor cores along the x-axis into single output files (0 = off, 1 = on)
output_combine_y	combines data from processor cores along the y-axis into single output files (0 = off, 1 = on)
output_combine_z	combines data from processor cores along the z-axis into single output files (0 = off, 1 = on)
incompressible_fluids	determines whether or not the fluids should be incompressible (set to 0 for compressible fluids, 1 for incompressible fluids)
restart_simulation	determines whether or not the simulation should be restarted (set to 0 for a new simulation, 1 to restart the simulation from the state given in the `lbout.dump` restart file)

The following keywords can be used to specify other information, such as fluid densities, velocities, relaxation times or frequencies etc. Superfluous parameters can be omitted, while new ones would require additions to the parameter recognition loop in the `fInputParameters` subroutine in `lbpIO`. Note that if there are duplicate entries for any keyword, the value associated with the last one in the `lbin.sys` file will be used.

keyword:	**meaning:**
total_step	sets total number of timesteps for the simulation
equilibration_step	sets number of timesteps for equilibration of the simulation (without solid boundary conditions or external forcing)
save_span	sets interval for writing output files
dump_span	sets interval for writing restart files (default: 10000 - if set to zero, restart file only written at end of simulation)
calculation_time	sets available calculation time (in seconds) before closing down the simulation (default: 0 - simulation will not stop until last timestep is reached[4])
boundary_type	sets type of boundary conditions for fluid flows (`ZouHe` (0), `Inamuro` (1), `Regularized`/`Regularised` (2), `SimpleZouHe` (10) or `Kinetic` (11))
solute_boundary_type	sets type of boundary conditions for solutes (`ZouHe` (0) or `Inamuro` (1))
thermal_boundary_type	sets type of boundary conditions for thermal flows (`ZouHe` (0) or `Inamuro` (1))
noise_intensity	gives maximum variation in initial fluid densities for multiple fluid systems

[2] Either the keyword or the number can be used to specify the types.
[3] If set to an unrecognised word or to 0, interactions will be switched off.
[4] Not recommended on computing platforms with queue managers as restart files might not be written in time before the calculation is halted.

evaporation_limit	gives minimum fluid density for non-continuous fluids when dealing with edge or corner boundaries (default value 10^{-8})
trt_magic_number	sets the TRT 'magic number' Λ_{eo}
gas_constant	sets the universal gas constant R for equations of state (Shan-Chen pseudopotential and Swift free-energy interactions: default value 1)
gradient_order	sets the order of gradient approximations (i.e. number of neighbouring grid points used) at boundary/near-boundary points (1 or 2: default value 1)
sound_speed	sets speed of sound for fluid 0 in real-life (i.e. non-lattice-based) units
kinetic_viscosity	sets kinematic viscosity for fluid 0 in real-life units
total_step	sets total number of timesteps for the simulation
oscillating_freq	sets frequency for sinusoidal oscillating forces across system
oscillating_period	sets period for sinusoidal oscillating forces across system (reciprocal of frequency)
oscillating_freq_top	sets frequency for sinusoidal oscillating velocity at top boundary
oscillating_period_top	sets period for sinusoidal oscillating velocity at top boundary (reciprocal of frequency)
oscillating_freq_bot	sets frequency for sinusoidal oscillating velocity at bottom boundary
oscillating_period_bot	sets period for sinusoidal oscillating velocity at bottom boundary (reciprocal of frequency)
oscillating_freq_lef	sets frequency for sinusoidal oscillating velocity at left boundary
oscillating_period_lef	sets period for sinusoidal oscillating velocity at left boundary (reciprocal of frequency)
oscillating_freq_rig	sets frequency for sinusoidal oscillating velocity at right boundary
oscillating_period_rig	sets period for sinusoidal oscillating velocity at right boundary (reciprocal of frequency)
oscillating_freq_fro	sets frequency for sinusoidal oscillating velocity at front boundary
oscillating_period_fro	sets period for sinusoidal oscillating velocity at front boundary (reciprocal of frequency)
oscillating_freq_bac	sets frequency for sinusoidal oscillating velocity at back boundary
oscillating_period_bac	sets period for sinusoidal oscillating velocity at back boundary (reciprocal of frequency)
speed_ini_n	sets initial velocity for all fluids ($n = 0$ for x-component, $n = 1$ for y-component, $n = 2$ for z-component)
speed_top_n	sets constant velocity at top boundary for all fluids
speed_oscil_top_n	sets oscillating velocity amplitude at top boundary for all fluids
speed_bot_n	sets constant velocity at bottom boundary for all fluids
speed_oscil_bot_n	sets oscillating velocity amplitude at bottom boundary for all fluids
speed_lef_n	sets constant velocity at left boundary for all fluids
speed_oscil_lef_n	sets oscillating velocity amplitude at left boundary for all fluids
speed_rig_n	sets constant velocity at right boundary for all fluids
speed_oscil_rig_n	sets oscillating velocity amplitude at right boundary for all fluids
speed_fro_n	sets constant velocity at front boundary for all fluids
speed_oscil_fro_n	sets oscillating velocity amplitude at front boundary for all fluids
speed_bac_n	sets constant velocity at back boundary for all fluids
speed_oscil_bac_n	sets oscillating velocity amplitude at back boundary for all fluids
density_ini_f	sets initial density for fluid f throughout system (f between 0 and $N_f - 1$)
density_inc_f	sets constant density for incompressible fluid f
density_top_f	sets density for fluid f at top boundary
density_bot_f	sets density for fluid f at bottom boundary
density_lef_f	sets density for fluid f at left boundary
density_rig_f	sets density for fluid f at right boundary

density_fro_f	sets density for fluid f at front boundary
density_bac_f	sets density for fluid f at back boundary
rheology_fluid_f	sets rheology model for fluid f (see below)
rheology_parameter_a_f	sets rheological model parameter a for fluid f
rheology_parameter_b_f	sets rheological model parameter b for fluid f
rheology_parameter_c_f	sets rheological model parameter c for fluid f
rheology_parameter_d_f	sets rheological model parameter d for fluid f
rheology_power_f	sets rheological model power index n for fluid f
relaxation_fluid_f	sets (initial) relaxation time (τ_f) for fluid f (symmetric relaxation time for TRT)
relax_freq_fluid_f	sets (initial) relaxation frequency (τ_f^{-1}) for fluid f (symmetric relaxation frequency for TRT)
bulk_relaxation_fluid_f	sets bulk relaxation time ($\tau_{f,bulk}$) for fluid f
bulk_relax_freq_fluid_f	sets bulk relaxation frequncy ($\tau_{f,bulk}^{-1}$) for fluid f
mrt_relax_i	sets ith relaxation time for MRT scheme, applicable for all fluids (see below for details: $i = 0, 1$ for D2Q9 lattice, $i = 0, 1, 2$ for D3Q15 and D3Q19 lattices, $i = 0, 1, 2, 3, 4, 5, 6, 7$ for D3Q27 lattice)
mrt_relax_freq_i	sets ith relaxation frequency for MRT scheme, applicable for all fluids (see below for details: $i = 0, 1$ for D2Q9 lattice, $i = 0, 1, 2$ for D3Q15 and D3Q19 lattices, $i = 0, 1, 2, 3, 4, 5, 6, 7$ for D3Q27 lattice)
clbe3_relaxation_fluid_f	sets CLBE third-order relaxation time (τ_3) for fluid f
clbe3_relax_freq_fluid_f	sets CLBE third-order relaxation frequency ($\omega_3 = \tau_3^{-1}$) for fluid f
clbe4_relaxation_fluid_f	sets CLBE fourth-order relaxation time (τ_4) for fluid f
clbe4_relax_freq_fluid_f	sets CLBE fourth-order relaxation frequency ($\omega_4 = \tau_4^{-1}$) for fluid f
relax_mobility	sets mobility relaxation time (τ_ϕ) for two-fluid Swift free-energy interactions
relax_freq_mobility	sets mobility relaxation frequency (τ_ϕ^{-1}) for two-fluid Swift free-energy interactions
mobility_parameter	sets mobility parameter Γ for two-fluid Swift free-energy interactions
surface_tension_parameter	sets the surface tension parameter κ for Swift free-energy interactions (both one and two fluid systems)
solute_ini_s	sets initial concentration for solute s throughout system (s between 0 and `lbsy.nc` $- 1$)
solute_top_s	sets concentration for solute s at top boundary
solute_bot_s	sets concentration for solute s at bottom boundary
solute_lef_s	sets concentration for solute s at left boundary
solute_rig_s	sets concentration for solute s at right boundary
solute_fro_s	sets concentration for solute s at front boundary
solute_bac_s	sets concentration for solute s at back boundary
relax_solute_s	sets relaxation time (τ_s) for solute s
relax_freq_solute_s	sets relaxation frequency (τ_s^{-1}) for solute s
temperature_ini	sets initial temperature throughout system
temperature_top	sets temperature at top boundary
temperature_bottom	sets temperature at bottom boundary
temperature_left	sets temperature at left boundary
temperature_right	sets temperature at right boundary
temperature_front	sets temperature at front boundary
temperature_back	sets temperature at back boundary
temperature_system	sets temperature of entire system if using equations of state and no temperature scalar
heating_rate_sys	sets rate of change in temperature (with time based on real-life units) throughout system

heating_rate_top	sets rate of change in temperature at top boundary
heating_rate_bottom	sets rate of change in temperature at bottom boundary
heating_rate_left	sets rate of change in temperature at left boundary
heating_rate_right	sets rate of change in temperature at right boundary
heating_rate_front	sets rate of change in temperature at front boundary
heating_rate_back	sets rate of change in temperature at back boundary
relax_thermal	sets thermal relaxation time (τ_t)
relax_freq_thermal	sets thermal relaxation frequency (τ_t^{-1})
body_force_n	sets constant external body force on fluid f: $n = 3f$ for x-component, $n = 3f+1$ for y-component, $n = 3f + 2$ for z-component
body_force_x_f	sets x-component of constant external body force on fluid f
body_force_y_f	sets y-component of constant external body force on fluid f
body_force_z_f	sets z-component of constant external body force on fluid f
oscillating_force_n	sets amplitude of sinusoidal oscillating body force on fluid f: $n = 3f$ for x-component, $n = 3f + 1$ for y-component, $n = 3f + 2$ for z-component
oscillating_force_x_f	sets x-component of amplitude of sinusoidal oscillating body force on fluid f
oscillating_force_y_f	sets y-component of amplitude of sinusoidal oscillating body force on fluid f
oscillating_force_z_f	sets z-component of amplitude of sinusoidal oscillating body force on fluid f
boussinesq_force_n	sets Boussinesq force constant ($\vec{g}\beta$) for fluid f: $n = 3f$ for x-component, $n = 3f + 1$ for y-component, $n = 3f + 2$ for z-component
boussinesq_force_x_f	sets x-component of Boussinesq force constant ($\vec{g}\beta$) for fluid f
boussinesq_force_y_f	sets y-component of Boussinesq force constant ($\vec{g}\beta$) for fluid f
boussinesq_force_z_f	sets z-component of Boussinesq force constant ($\vec{g}\beta$) for fluid f
boussinesq_boussinesq_high	sets high reference temperature for Boussinesq convection (T_h)
boussinesq_boussinesq_low	sets low reference temperature for Boussinesq convection (T_l)
interaction_n	sets interaction parameter between fluids f_1 and f_2: $n = N_f \times f_1 + f_2$
interaction_f_1_f_2	sets interaction parameter between fluids f_1 and f_2
quadratic_weight	sets Shan-Chen quadratic term weighting parameter β between all pairs of fluid species
quadratic_weight_n	sets Shan-Chen quadratic term weighting parameter β between fluids f_1 and f_2: $n = N_f \times f_1 + f_2$
quadratic_weight_f_1_f_2	sets Shan-Chen quadratic term weighting parameter β between fluids f_1 and f_2
potential_type	sets the pseudopotential type for Shan-Chen interactions (see below) or chemical potential type for Swift free-energy interactions (`None` (0), `Quartic` (1))
potential_type_f	sets the pseudopotential type for Shan-Chen interactions (see below) for fluid f
equation_of_state	sets equation of state for all fluids with Swift free-energy interactions (see below for options)
eos_parameter_a	sets equation-of-state parameter a for all fluid species
eos_parameter_a_f	sets equation of state parameter a for fluid f
eos_parameter_b	sets equation of state parameter b for all fluid species
eos_parameter_b_f	sets equation of state parameter b for fluid f
potential_parameter_a	sets chemical potential parameter a for all fluid species (Swift free-energy interactions)
potential_parameter_b	sets chemical potential parameter b for all fluid species (Swift free-energy interactions)
shanchen_psi0_f	sets Shan-Chen pseudopotential parameter ψ_0 for fluid f
critical_temperature_f	sets critical temperature T_c for fluid f
critical_pressure_f	sets critical pressure P_c for fluid f
acentric_factor_f	sets acentric factor ω for fluid f

segregation	sets fluid segregation parameter between all fluids species
segregation_n	sets fluid segregation parameter between fluids f_1 and f_2: $n = N_f \times f_1 + f_2$
segregation_f_1_f_2	sets fluid segregation parameter between fluids f_1 and f_2
wetting_type	sets the basis for wetting interactions between solid points and all fluid species for Shan-Chen interactions (`Density` (0), `Potential` (1), `ScreenedPotential` (2)) or for Swift free-energy interactions (`None` (0), `Quadratic` (1))
wetting_type_f	sets the basis for Shan-Chen wetting interactions between solid points and fluid species f (`Density` (0), `Potential` (1), `ScreenedPotential` (2))
wall_interaction_f	sets Shan-Chen interaction parameter between fluid f and solid walls
wetting_parameter_rho_n	sets Swift wetting potential parameter for fluid density (a_n, $n = 0$ or 1)
wetting_parameter_phi_n	sets Swift wetting potential parameter for fluid concentration (b_n, $n = 0$ or 1)

When using MRT collisions, the following relaxation times or frequencies can be specified for all fluids in the simulation (overriding the default values given here):

MRT parameter number i	Relaxation frequency for lattice model			
	D2Q9[68]	D3Q15[23]	D3Q19[23]	D3Q27[124]
0	$s_2 = 1.14$	$s_2 = 1.2$	$s_2 = 1.4$	$s_{10} = 1.5$
1	$s_4 = 1.92$	$s_4 = 1.6$	$s_4 = 1.4$	$s_{13} = 1.83$
2	-	$s_{14} = 1.2$	$s_{16} = 1.98$	$s_{16} = 1.4$
3	-	-	-	$s_{17} = 1.61$
4	-	-	-	$s_{18} = 1.98$
5	-	-	-	$s_{20} = 1.98$
6	-	-	-	$s_{23} = 1.74$
7	-	-	-	$s_{26} = 1.74$

with the relaxation times corresponding to those given in the original references. (More details about these can be found in Appendix D.)

The following rheological models are available, with the respective key numbers and words in the `lbin.sys` file and correspondence of parameters to those required in the input file (noting that the power gives n for all relevant models):

Model number	Key word	Equation	Parameters			
			a	b	c	d
0	`Simple`[5]	$\mu = c_s^2 \rho \left(\tau_f - \frac{1}{2}\right) \Delta t$	τ_f	-	-	-
1	`Newtonian`	$\mu = \mu_0$	μ_0	-	-	-
2	`PowerLaw` (`Power`)	$\mu = K\dot{\gamma}^{n-1}$	K	-	-	-
3	`Bingham`	$\mu = \mu_0 + \frac{\sigma_y}{\dot{\gamma}}\left(1 - e^{-m\dot{\gamma}}\right)$	μ_0	σ_y	m	-
4	`HerschelBulkley` (`Herschel`)	$\mu = K\dot{\gamma}^{n-1} + \frac{\sigma_y}{\dot{\gamma}}\left(1 - e^{-m\dot{\gamma}}\right)$	K	σ_y	m	-
5	`Casson`	$\mu = \left(\sqrt{\mu_0} + \sqrt{\frac{\sigma_y}{\dot{\gamma}}}\left(1 - e^{-m\dot{\gamma}}\right)\right)^2$	μ_0	σ_y	m	-
6	`CarreauYasuda` (`Carreau`)	$\mu = \mu_\infty + (\mu_0 - \mu_\infty)\left(1 + (\lambda\dot{\gamma})^d\right)^{\frac{n-1}{d}}$	μ_∞	μ_0	λ	d

where μ_0 and μ_∞ are specified constant viscosities (at zero-shear and infinite-shear), K is a consistency viscosity for power-law fluids, σ_y is a yield stress, λ is a yield relaxation time for Carreau fluids, m is a stress growth exponent for dealing with discontinuities in plastic models with yield stresses, n and a are power indices.

The following equations of state are available for Shan-Chen pseudopotentials and Swift free-energy bulk pressures, with the respective key numbers and words in the `lbin.sys` file (note that some numbers are skipped to give alternatives for constant and variable temperature systems):

[5] The density ρ is either the local value for a given lattice point or the constant value for incompressible fluids ρ_0.

Table 6.4: Boundary condition category

value	meaning
0	liquid
10	domain boundary
11	inside solid
12	on-grid bounce-back boundary
13	mid-link bounce-back boundary
21–99	outflow boundary
100–199	constant speed, composition and temperature boundary
200–299	constant speed, Neumann composition and temperature boundary
300–399	constant speed and composition, Neumann temperature boundary
400–499	constant speed and temperature, Neumann composition boundary
500–599	constant pressure, composition and temperature boundary
600–699	constant pressure, Neumann composition and temperature boundary
700–799	constant pressure and composition, Neumann temperature boundary
800–899	constant pressure and temperature, Neumann composition boundary

0	`IdealLattice`	$p = \rho c_s^2$
1	`ShanChen1993`	$p = \rho c_s^2 + \frac{1}{2} c_s^2 g \rho_0^2 \left(1 - e^{-\frac{\rho}{\rho_0}}\right)^2$
2	`ShanChen1994`	$p = \rho c_s^2 + \frac{1}{2} c_s^2 g \psi_0^2 \left(e^{\frac{-2\rho_0}{\rho}}\right)$
3	`Qian1995`	$p = \rho c_s^2 + \frac{1}{2} c_s^2 g \left(\frac{\rho_0 \rho}{\rho_0 + \rho}\right)^2$
4	`Rho`	$p = \rho c_s^2 + \frac{1}{2} c_s^2 g \rho^2$
5	`Ideal`	$p = \rho RT$
6	`vanderWaals` (`vdW`)	$p = \frac{\rho RT}{1-b\rho} - a\rho^2$
7	`CarnahanStarlingvanderWaals` (`CSvdW`)	$p = \rho RT \left(\frac{1+\phi+\phi^2-\phi^3}{(1-\phi)^3}\right) - a\rho^2$
8	`RedlichKwong` (`RK`)	$p = \frac{\rho RT}{1-b\rho} - \frac{a\rho^2}{\sqrt{T}(1+b\rho)}$
10	`SoaveRedlichKwong` (`SRK`)	$p = \frac{\rho RT}{1-b\rho} - \frac{a\alpha(T_r,\omega)\rho^2}{1+b\rho}$
12	`PengRobinson` (`PR`)	$p = \frac{\rho RT}{1-b\rho} - \frac{a\alpha(T_r,\omega)\rho^2}{1+2b\rho-b^2\rho^2}$
14	`CarnahanStarlingRedlichKwong` (`CSRK`)	$p = \rho RT \left(\frac{1+\phi+\phi^2-\phi^3}{(1-\phi)^3}\right) - \frac{a\rho^2}{\sqrt{T}(1+b\rho)}$

where $\phi = \frac{1}{4} b\rho$ for the Carnahan-Starling equations of state. For Swift free-energy interactions with Shan-Chen equations of state (numbers 1 to 4 in the above table), values for g and ψ_0 can either be specified for fluid 0 or the global equation-of-state parameters a and b (respectively) can be used.

Define space: `lbin.spa`

The GUI is recommended for creating the `lbin.spa` file, which stores the data in the following format:

```
x,y,z,space property
```

An empty `lbin.spa` file represents all boundaries as periodic. The space property of a grid point is represented by an integer value in DL_MESO_LBE. For example, 0 represents a liquid site and 12 represents an on-grid bounce-back boundary. Table 6.4 lists the categories of space properties.

The orientation of a solid-liquid boundary is also represented by the value of an integer. For example, a planar surface with normal vector along the y-axis is denoted by 21, while a concave corner face at the top-right-front corner is denoted by 31. It must be pointed out that only those space positions located in the surface of a face-centered cube have been included and translated in DL_MESO_LBE. Points with random orientations, e.g. 47° plane, have not been included.

The boundary condition number can be rather confusing and difficult to understand. To assist in producing the space properties, the DL_MESO GUI includes a translator that interprets a *word* as its corresponding

integer number. The *word* is made up of defined letters as listed in Table 6.5. The boundary conditions with combinations of type and orientation are listed in Table 6.6. The *letters* are in the order of:

1. Fluid property: constant speed or constant pressure.
2. Solute property: constant composition or bounceback boundary.
3. Temperature property: isothermal (constant) or heat bath (bounceback boundary).
4. Geometric property: planar surface, concave corner or concave edge.
5. Boundary orientation: one letter for planar surface, two letters for concave corners or three letters for concave edges.

For example, a shearing planar surface facing downwards along the y-axis with constant composition and temperature (i.e. isothermal) is represented as VCBPSD and translated as 322.

Table 6.5: Boundary condition category

letter	**meaning**
O	Outflow
V	Constant Velocity
P	Constant Pressure (Density)
C	Constant Solute Composition
T	Constant Temperature
B	Bounceback Boundary Condition (Solute Composition or Temperature)
PS	Planar Surface
CC	Concave Corner
CE	Concave Edge
T	Normal Vector Pointing to Top
D	Normal Vector Pointing Downwards
L	Normal Vector Pointing to Left
R	Normal Vector Pointing to Right
F	Normal Vector Pointing to Front
B	Normal Vector Pointing to Back

Table 6.6: Notation of boundary condition

OPST	21	OPSD	22	OPSL	23
OPSR	24	OPSF	25	OPSB	26
OCETF	47	OCELF	48	OCEDF	49
OCERF	50				
VCTPST	121	VCTPSD	122	VCTPSL	123
VCTPSR	124	VCTPSF	125	VCTPSB	126
VCTCCTRB	127	VCTCCTLB	128	VCTCCDLB	129
VCTCCDRB	130	VCTCCTRF	131	VCTCCTLF	132
VCTCCDLF	133	VCTCCDRF	134	VCTCETR	143
VCTCETL	144	VCTCEDL	145	VCTCEDR	146
VCTCETF	147	VCTCELF	148	VCTCEDF	149
VCTCERF	150	VCTCETB	151	VCTCELB	152
VCTCEDB	153	VCTCERB	154	VCBPST	221
VCBPSD	222	VCBPSL	223	VCBPSR	224
VCBPSF	225	VCBPSB	226	VCBCCTRB	227
VCBCCTLB	228	VCBCCDLB	229	VCBCCDRB	230
VCBCCTRF	231	VCBCCTLF	232	VCBCCDLF	233
VCBCCDRF	234	VCBCETR	243	VCBCETL	244
VCBCEDL	245	VCBCEDR	246	VCBCETF	247
VCBCELF	248	VCBCEDF	249	VCBCERF	250
VCBCETB	251	VCBCELB	252	VCBCEDB	253
VCBCERB	254	VBTPST	321	VBTPSD	322
VBTPSL	323	VBTPSR	324	VBTPSF	325
VBTPSB	326	VBTCCTRB	327	VBTCCTLB	328
VBTCCDLB	329	VBTCCDRB	330	VBTCCTRF	331
VBTCCTLF	332	VBTCCDLF	333	VBTCCDRF	334
VBTCETR	343	VBTCETL	344	VBTCEDL	345
VBTCEDR	346	VBTCETF	347	VBTCELF	348
VBTCEDF	349	VBTCERF	350	VBTCETB	351
VBTCELB	352	VBTCEDB	353	VBTCERB	354
VBBPST	421	VBBPSD	422	VBBPSL	423
VBBPSR	424	VBBPSF	425	VBBPSB	426
VBBCCTRB	427	VBBCCTLB	428	VBBCCDLB	429
VBBCCDRB	430	VBBCCTRF	431	VBBCCTLF	432
VBBCCDLF	433	VBBCCDRF	434	VBBCETR	443
VBBCETL	444	VBBCEDL	445	VBBCEDR	446
VBBCETF	447	VBBCELF	448	VBBCEDF	449
VBBCERF	450	VBBCETB	451	VBBCELB	452
VBBCEDB	453	VBBCERB	454	PCTPST	521
PCTPSD	522	PCTPSL	523	PCTPSR	524
PCTPSF	525	PCTPSB	526	PCTCCTRB	527
PCTCCTLB	528	PCTCCDLB	529	PCTCCDRB	530
PCTCCTRF	531	PCTCCTLF	532	PCTCCDLF	533
PCTCCDRF	534	PCTCETR	543	PCTCETL	544
PCTCEDL	545	PCTCEDR	546	PCTCETF	547
PCTCELF	548	PCTCEDF	549	PCTCERF	550
PCTCETB	551	PCTCELB	552	PCTCEDB	553
PCTCERB	554	PCBPST	621	PCBPSD	622
PCBPSL	623	PCBPSR	624	PCBPSF	625

Table 6.6: Notation of boundary condition (continued)

PCBPSB	626	PCBCCTRB	627	PCBCCTLB	628
PCBCCDLB	629	PCBCCDRB	630	PCBCCTRF	631
PCBCCTLF	632	PCBCCDLF	633	PCBCCDRF	634
PCBCETR	643	PCBCETL	644	PCBCEDL	645
PCBCEDR	646	PCBCETF	647	PCBCELF	648
PCBCEDF	649	PCBCERF	650	PCBCETB	651
PCBCELB	652	PCBCEDB	653	PCBCERB	654
PBTPST	721	PBTPSD	722	PBTPSL	723
PBTPSR	724	PBTPSF	725	PBTPSB	726
PBTCCTRB	727	PBTCCTLB	728	PBTCCDLB	729
PBTCCDRB	730	PBTCCTRF	731	PBTCCTLF	732
PBTCCDLF	733	PBTCCDRF	734	PBTCETR	743
PBTCETL	744	PBTCEDL	745	PBTCEDR	746
PBTCETF	747	PBTCELF	748	PBTCEDF	749
PBTCERF	750	PBTCETB	751	PBTCELB	752
PBTCEDB	753	PBTCERB	754	PBBPST	821
PBBPSD	822	PBBPSL	823	PBBPSR	824
PBBPSF	825	PBBPSB	826	PBBCCTRB	827
PBBCCTLB	828	PBBCCDLB	829	PBBCCDRB	830
PBBCCTRF	831	PBBCCTLF	832	PBBCCDLF	833
PBBCCDRF	834	PBBCETR	843	PBBCETL	844
PBBCEDL	845	PBBCEDR	846	PBBCETF	847
PBBCELF	848	PBBCEDF	849	PBBCERF	850
PBBCETB	851	PBBCELB	852	PBBCEDB	853
PPBCERB	854				

Define initial condition: `lbin.init`

This optional file cannot currently be created by the GUI: the user must create this file or use the utility `lbeinitcreate` if it is required. The following format is required for each lattice point whose default velocity, fluid densities, solute concentrations or temperature needs replacing:

$$\mathtt{x,y,z,u}_x\mathtt{,u}_y\mathtt{,u}_z\mathtt{,}\rho^0 \ldots \rho^{N_f-1}\mathtt{,}c^0 \ldots c^{N_c-1}\mathtt{,}T$$

Note that three values for each grid position and velocity must be included (the values for z-components in two-dimensional simulations must be set to zero). At each grid point specified, density/concentration/temperature values must be included for *all* lattices used in calculations: the total number of values in each line must be equal to $6 + N_f + N_c + N_t$ (where N_f and N_c are the numbers of fluids and solutes respectively, and N_t equals 1 if temperatures are in use or 0 if they are not).

6.2 Output files

DL_MESO_LBE prints information about the simulation to the screen or standard output:

- welcome messages
- a brief description of the simulation to be carried out, including system-wide parameters (e.g. relaxation times)
- details of domain decomposition if running in parallel

- reports on the masses and momentum of fluids in the system at user-specified intervals (with timings in seconds)
- a final summary including a calculation efficiency measure and a reminder to cite DL_MESO for any published results

This information can be directed to a file specified at the command line, e.g. by launching DL_MESO_LBE using the command

- `./lbe.exe > OUTPUT`

Restart capability: `lbout.dump`

A binary restart file (`lbout.dump`) containing data to allow an interrupted simulation to be restarted is generated at regular intervals (every `lbdump` time steps) and at the end of the allotted number of time steps. This can be read into DL_MESO_LBE if the value for **restart_simulation** in `lbin.sys` is set equal to 1. (The number of processor cores used to restart the simulation does not have to equal the number used to create the `lbout.dump` file.)

The file format consists of a header made up of 12 integers with the following properties:

- The number of dimensions
- The number of lattice speeds
- The number of grid points in the x-dimension
- The number of grid points in the y-dimension
- The number of grid points in the z-dimension
- The number of fluids (N_f)
- The number of solutes (N_c)
- The switch for temperature field (N_t)
- The (currently unused) switch for phase field (N_{pf})
- The timestep at the point when the file was created
- The number of snapshot output files previously written
- The switch for incompressible fluids

followed by the constant densities (in double precision format) for incompressible fluids (if these were in use). The positions for each grid point then follow (in the Cartesian coordinates x, y, z) as a single block of integers, before the distribution functions and local relaxation frequencies (used for non-Newtonian rheological models) follow in a block of double precision numbers. The grid points do not have to be ordered in any way but the order corresponds to the values of distribution function and relaxation frequencies found in the block of double precision numbers. (Each grid point's data start with distribution functions ordered by fluid/solute/temperature and then lattice link, followed by the relaxation frequencies for the fluids.)

The utility `lbedumpinit` in the `LBE/utility` directory can read `lbout.dump` files and produce a `lbin.init` initialization file for a new simulation. The utility `lbedumpvtk` can produce a Structured Grid XML VTK formatted file of the system at the timestep when the restart file was generated. Further details on both utilities can be found in Appendix C or the `README` file in the `LBE/utility` directory.

Simulation snapshots: `lbout*.vts`, `lbout*.vtk`, `lbout*.q`

Snapshots of the simulation can be written in Structured Grid XML VTK, Structured Grid Legacy VTK and standard Plot3D data format, either in big endian binary format (by default) or ANSI (text). These may be modified by the user as required.

By default, DL_MESO_LBE will produce an output file per frame per processor core: to reduce the number of files produced per frame, the **output_combine** options can be switched on in the `lbin.sys` file. Processor cores along a particular line or plane can gather together their data onto a single core in a group, which writes a file with their combined data instead of getting each core to write its own file. Each dimension (x, y and z) can be switched on independently of each other for data combination: switching on all three dimensions will gather all data together onto a single core for writing and thus produce one output file per frame. The option for one output file per frame is implemented by gathering data in one or two dimensions to give groups orthogonal to the other dimension that can be put together contiguously using MPI-IO (e.g. in three-dimensions, the x- and y- dimensions are combined into two-dimensional groups that are placed side-by-side in the z-direction).

The utility `lbeplot3dgather` in the `LBE/utility` directory can directly combine binary Plot3D files generated in parallel, while Parallel Structured Grid XML VTK files (`lbtout*.pvts`) that refer to the files from each processor can be created using the utility `lbevtkgather`: further details can be found in Appendix C or the `README` file in the same directory. By default *all* properties for a simulation – fluid densities, mass fractions, solute concentrations and temperatures – are written to each output file (or to individual output files for Plot3D for each property). The customizable version of DL_MESO_LBE allows users to select which properties should be written to output files: further details about this can be found in the Developer Manual.

Structured Grid XML VTK format: `lbout*.vts`

Structured Grid VTK files written by DL_MESO_LBE include the lattice dimensions (numbers of grid points in each direction), the Cartesian coordinates of the grid points in real-life units, and the same data as in Legacy VTK files (see below). Output files produced in ANSI text format include the data between XML tags, e.g. `<DataArray>`, while those produced in big endian binary use the `<DataArray>` tags to refer to the starting point for the data in a stream of binary numbers inside an `<AppendedData>` tag.

If each core or I/O group produces its own file per frame, these files represent the data in each subdomain (or collected subdomain) and should be retained when plotting the entire system, as the parallel VTK format links to these rather than creating combined files for the entire system. The utility `lbevtkgather` can be used to do this, which also requires the `lbout.info` and `lbout.ext` files generated when running DL_MESO_LBE in parallel to obtain information about the data and how the (collections of) subdomains fit together.

Legacy VTK format: `lbout*.vtk`

Legacy VTK files written by DL_MESO_LBE include the lattice dimensions (numbers of grid points in each direction), the Cartesian coordinates of the grid points in real-life units, and the following data:

- A scalar property (fluid density, mass fraction, solute concentration or scalar temperature), e.g.

$$\rho_{0,0,0}$$
$$\dots$$
$$\rho_{nx-1,ny-1,nz-1}$$

- Fluid velocity

$$U_{0,0,0}\ V_{0,0,0}\ W_{0,0,0}$$
$$\dots$$
$$U_{nx-1,ny-1,nz-1}\ V_{nx-1,ny-1,nz-1}\ W_{nx-1,ny-1,nz-1}$$

- The space (boundary condition) property

$$\phi_{0,0,0}$$
$$\dots$$
$$\phi_{nx-1,ny-1,nz-1}$$

If all properties are to be output, they are all included in the same file for each time step under unique names.

Plot3D format

This format consists of two file types: one indicating the grid positions, and the other providing solution data for macroscopic properties.

Output grid position: `lbout*.xyz`

$$nx, ny, nz$$
$$x_{0,0,0}, \dots x_{nx-1,ny-1,nz-1}$$
$$y_{0,0,0}, \dots y_{nx-1,ny-1,nz-1}$$
$$z_{0,0,0}, \dots z_{nx-1,ny-1,nz-1}$$

where nx is the total number of grid points in x-direction, ny is the total number of grid points in y-direction, nz is the total number of grid points in z-direction and $(x_{i,j,k}, y_{i,j,k}, z_{i,j,k})$ is the Cartesian coordinate of grid point (i, j, k).

Output macroscopic quantities: `lbout.q`

$$nx, ny, nz$$
$$c, 1.0, Re, t$$
$$\rho_{0,0,0}, \dots \rho_{nx-1,ny-1,nz-1}, U_{0,0,0}, \dots U_{nx-1,ny-1,nz-1}$$
$$V_{0,0,0}, \dots V_{nx-1,ny-1,nz-1}, W_{0,0,0}, \dots W_{nx-1,ny-1,nz-1}$$
$$\phi_{0,0,0}, \dots \phi_{nx-1,ny-1,nz-1}$$

where c is the speed of sound for the lattice, Re the Reynolds number for the flow, t the time step, ρ the density, U the x-component of velocity, V the y-component of velocity, W the z-component of velocity and ϕ the space (boundary condition) property. (The 1.0 between the lattice speed of sound and flow Reynolds number represents the freestream angle of attack.) The density of the fluid may be replaced with its concentration or scalar temperature.

If all properties are output, each property is given a uniquely named file based on the number of fluid or solute (e.g. `lbout00dens*.q` for the density of fluid 0) and its property (`lbout*dens*.q`, `lbout*frac*.q`, `lbout*conc*.q` and `lbouttemp*.q`).

Chapter 7

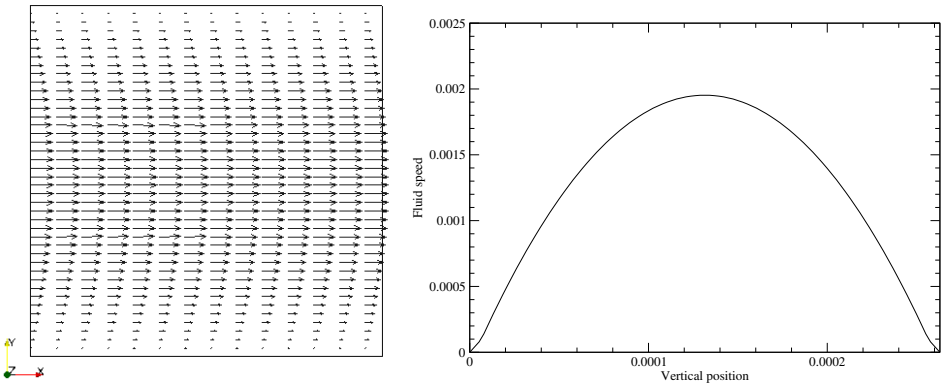
DL_MESO LBE Examples

Test cases for Lattice Boltzmann Equation simulations using DL_MESO – including the required input and sample output files – can be found in the `DEMO/LBE` subdirectory. They can be run using either the serial or parallel versions of DL_MESO_LBE.

Images of all test cases and videos for some can be found in the Example Simulations page of the DL_MESO website: a link to it can be found at `www.ccp5.ac.uk/DL_MESO`

7.1 2D_Pressure

This is a 2D simulation of a single fluid on a 42×42 grid with fixed pressure (density) boundary conditions on the left and right boundaries and solid walls (represented by bounce back) at the top and bottom. A visualization with vector glyphs and a plot of fluid speed against vertical position can be seen in Figure 7.1, which show the boundary conditions result in a laminar flow with a parabolic velocity profile.

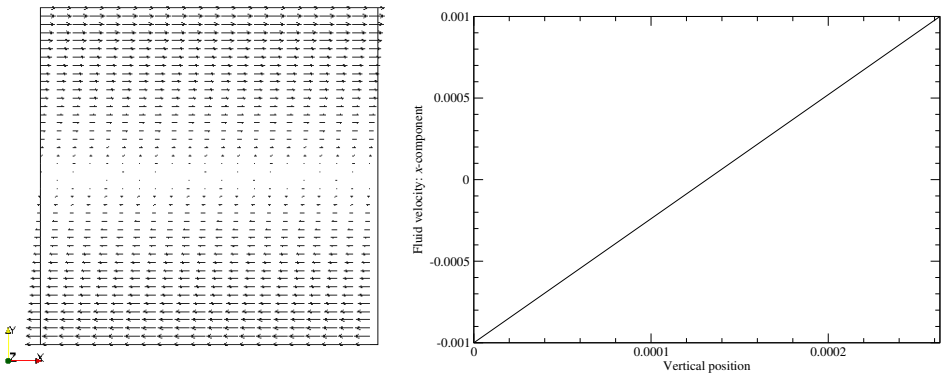


(a) Vector plot of system (b) Variation of fluid speed with vertical position

Figure 7.1: Results from LBE `2D_Pressure` test case

7.2 2D_Shear

This is a 2D simulation of a single fluid on a 42×42 grid with a shear boundary condition. The vector plot in Figure 7.2 demonstrates the ability of the applied boundary conditions to generate a linear shearing Couette flow throughout the grid, which is confirmed by the plot of horizontal velocity component as a function of vertical position at the last time step for the simulation.



(a) Vector plot of system

(b) Variation of horizontal velocity with vertical position

Figure 7.2: Results from LBE `2D_Shear` test case

7.3 2D_CylinderFlow

This is a 2D simulation of a single fluid on a 125×50 grid with a constant horizontal body force and a circular obstacle of radius 12, representing channel flow past an infinitely-long cylinder. Figure 7.3 shows this flow as streamlines coloured by density (from blue to red for low to high), with solid black lines representing the solid boundaries (both walls and the cylinder).

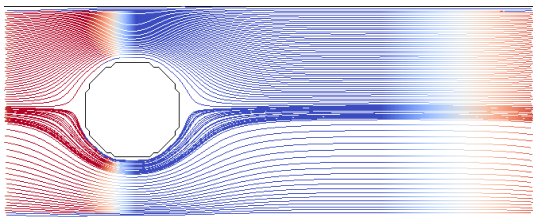


Figure 7.3: Density (scale: blue to red) and velocity streamline plot from LBE `2D_CylinderFlow` test case

7.4 2D_KarmanVortex

This is a 2D simulation of a single fluid on a 250×50 grid with a constant horizontal body force and a circular obstacle of radius 8, representing channel flow past an infinitely-long cylinder that eventually produces a von Kármán vortex street between two solid walls. Figure 7.4 shows the flow field at the final time step: an `.AVI` video file has been rendered from the calculation and can be found in the Example Simulations page of the DL_MESO website.

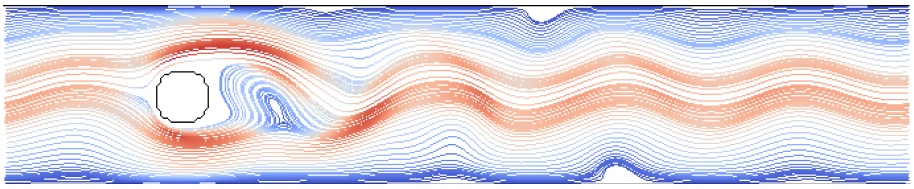


Figure 7.4: Velocity streamline plot from LBE `2D_KarmanVortex` test case (colour scale for velocity magnitude: blue to red)

7.5 2D_KarmanVortexOutflow

This is a 2D simulation of a single fluid using cascaded LBE collisions on a 520×180 grid with a constant velocity of 0.04 applied to the left boundary, an outflow boundary condition to the right, periodic boundaries at the top and bottom, and a circular obstacle of radius 20. This represents unbounded flow past an infinitely-long

cylinder that eventually produces a von Kármán vortex street. Figure 7.5 shows the flow field at the final time step.

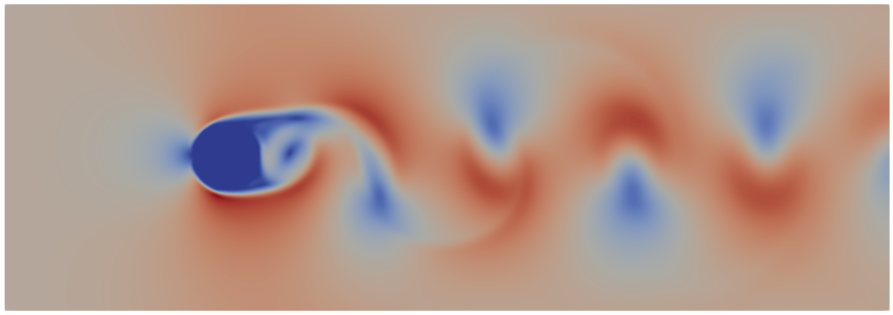


Figure 7.5: Velocity colour map plot from LBE `2D_KarmanVortexOutflow` test case (colour scale for velocity magnitude: blue to red)

7.6 2D_LidCavity

This is a 2D simulation of a single incompressible fluid on a 128×128 grid with a shear boundary condition at the top and solid walls surrounding the other edges of the system, resulting in lid-driven cavity flow. Figure 7.6 shows the fully-developed velocity field for a Reynolds number of 100 at the final time step.

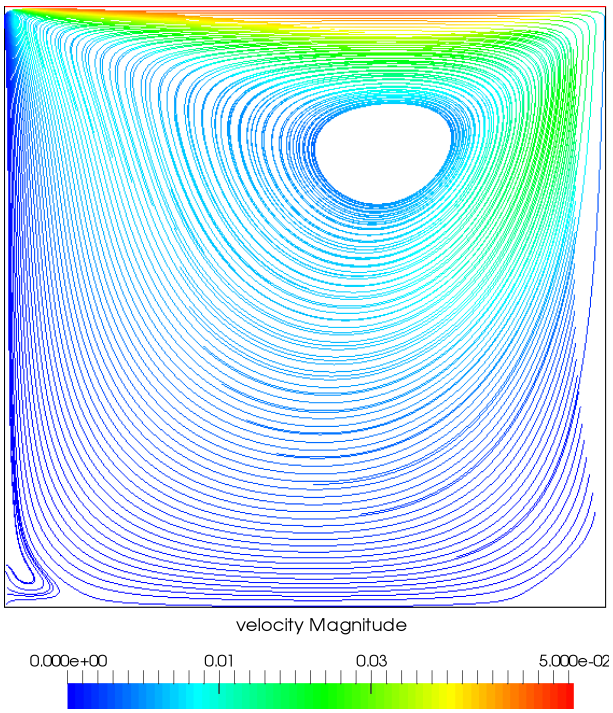


Figure 7.6: Velocity streamline and magnitude plot from LBE `2D_Lidcavity` test case

7.7 2D_RayleighBenard

This is a 2D simulation of a single fluid undergoing natural (Rayleigh-Bénard) convection on a 102×51 grid. The fluid is contained between two solid walls: the wall at the bottom of the system is maintained at a higher temperature than that at the top. Figure 7.7 shows the fully-developed temperature field with flow streamlines at the final time step for a Prandtl number of 1 and a Rayleigh number of ~ 21250.

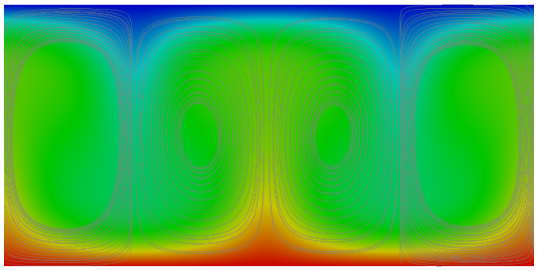
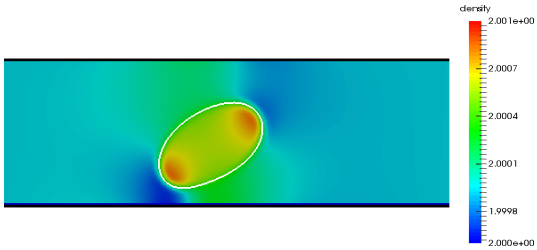


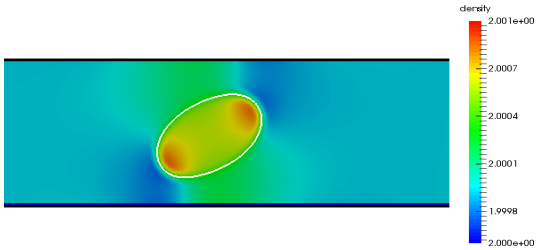
Figure 7.7: Plot of fluid temperature and streamlines for LBE `2D_RayleighBenard` test case (temperature scale: blue to red)

7.8 2D_DropShear

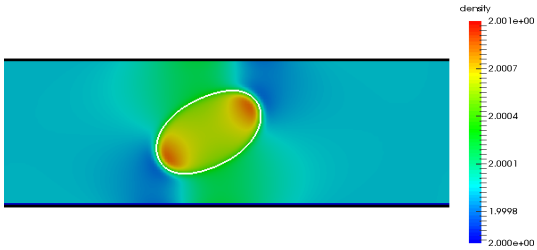
This is a 2D simulation of an initially static drop on a 150×50 grid undergoing linear shear[48] using Lishchuk continuum-based mesophase interactions with Guo forcing. The drop and continuous fluid are contained between two solid walls: after an equilibration period to allow the drop shape to settle, the wall at the top of the system is set to move horizontally while the wall at the bottom is kept stationary. Figure 7.8 shows the fluid density (pressure) field and drop positions at time steps throughout the simulation, demonstrating traverse migration (lift) due to linear shear, for a system with droplet Reynolds number of 0.135 and capillary number (ratio of inertial to interfacial stresses) of 0.147. An `.AVI` video file has been rendered from an example calculation, which can be found in the Example Simulations page of the DL_MESO website.



(a) $t = 17\ 000$



(b) $t = 86\ 500$



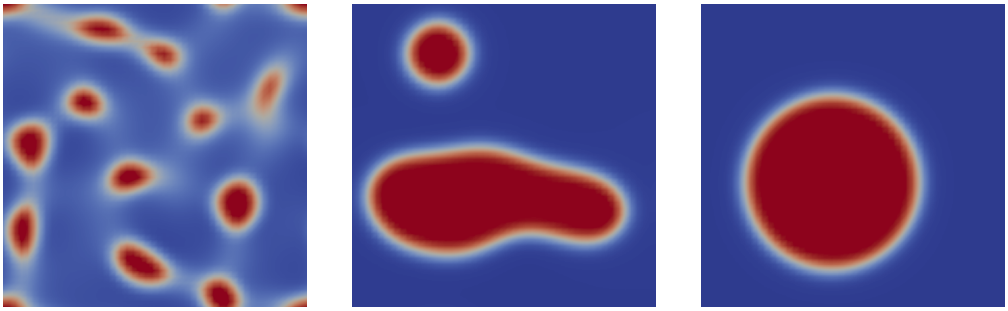
(c) $t = 390\ 000$

Figure 7.8: Plots of fluid density (pressure) and drop positions for LBE `2D_DropShear` test case

7.9 2D_PhaseSeparation

This is a 2D simulation of a single fluid on a 50×50 grid, which interacts using a Shan/Chen pseudopotential chosen to apply the cubic Peng-Robinson equation of state[145], using parameters of $a = \frac{2}{49}$, $b = \frac{2}{21}$, an acentric

factor $\omega = 0.344$ and gas constant $R = 1$. The temperature of the fluid is set to a value below the critical temperature, which causes the fluid to separate into liquid and vapour phases. Figure 7.9 shows the phase separation process in a number of snapshots, with the liquid phase forming into a circular drop. (This test case is courtesy of Christopher Stiles at the College of Nanoscale Science and Engineering, State University of New York.)



(a) $t = 100$ (b) $t = 500$ (c) $t = 5000$

Figure 7.9: Progressive density plots from LBE `2D_PhaseSeparation` test case (red for liquid phase $\rho_L = 7.633$, blue for vapour phase $\rho_V = 0.0462$)

7.10 2D_PowerLaw

This is a 2D simulation of a single fluid on a 64×128 grid with constant density boundaries at the left and right, mid-link bounce back conditions at the top and bottom, MRT collisions using local calculations of shear rate to modify relaxation times at grid points to obtain power law rheological behaviour. A pressure drop is applied using the constant density boundaries and the flow profile can be measured. Figure 7.10 shows how the velocity profile changes with different powers n.

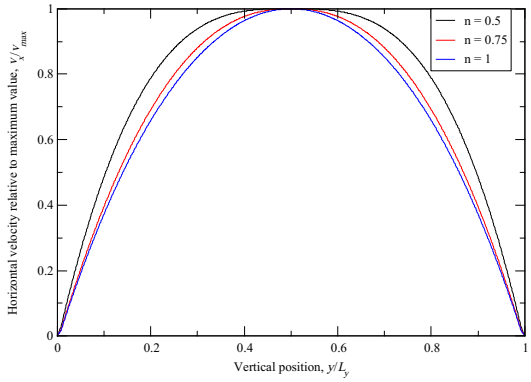
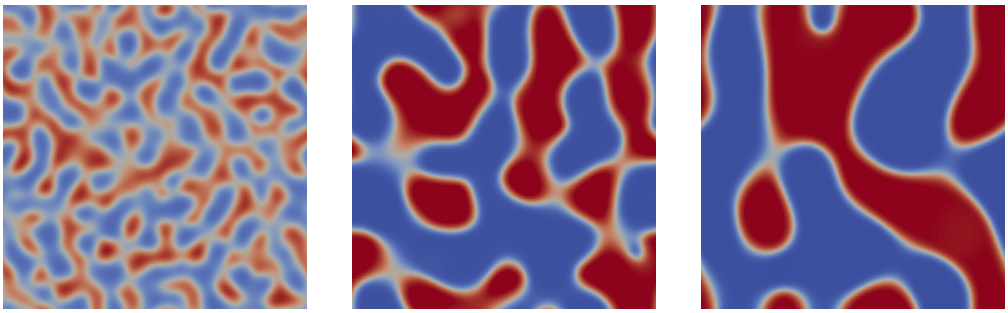


Figure 7.10: Plot of horizontal velocity as a function of vertical position for LBE `2D_PowerLaw` test case using different powers n

7.11 3D_PhaseSeparation

This is a 3D simulation of two fluids on a $100 \times 100 \times 100$ grid with periodic boundary conditions and Shan/Chen pseudopotential mesoscopic interactions (using the original pseudopotential[108]) that cause the fluids to separate. Figure 7.11 shows the phase separation process in a number of snapshots: two `.AVI` video files have been rendered from an example calculation (one giving a 3D view of the system, the other showing a plane normal to the x-axis) which can be found in the Example Simulations page of the DL_MESO website.

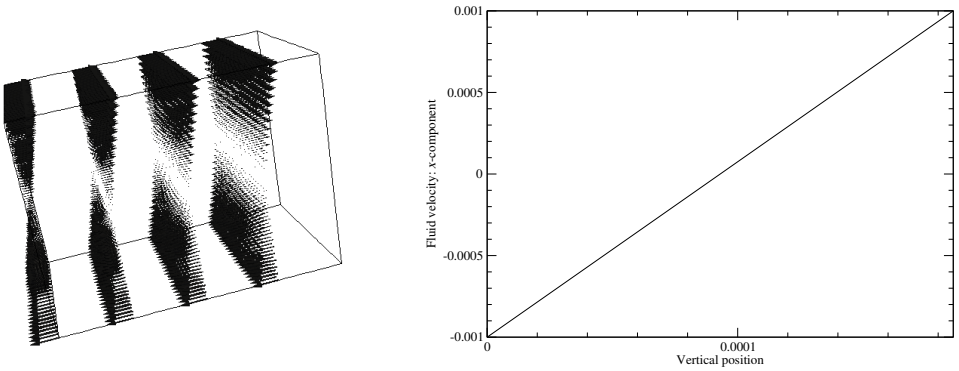


(a) $t = 400$ (b) $t = 1200$ (c) $t = 2000$

Figure 7.11: Progressive density plots in plane normal to y-axis from LBE `3D_PhaseSeparation` test case (red for fluid 0, blue for fluid 1)

7.12 3D_Shear

This is a 3D simulation of a single fluid on a $40 \times 30 \times 25$ grid with a shear boundary condition. Figure 7.12 shows a vector plot for this system, demonstrating that linear shear is generated and maintained by the moving boundaries in the planes normal to the y-axis, and a plot of the horizontal component of fluid velocity against vertical position at the last time step.



(a) Vector plot of system (b) Plot of horizontal velocity with vertical position

Figure 7.12: Results from LBE `3D_Shear` test case

7.13 3D_RayleighBenard

This is a 3D simulation of a single fluid undergoing natural (Rayleigh-Bénard) convection on a $80 \times 40 \times 80$ grid. The fluid is contained between two solid walls: the wall at the bottom of the system is maintained at a higher temperature than that at the top. Figure 7.13 shows the fully-developed temperature and convective flow fields at the final time step for a Prandtl number of 1 and a Rayleigh number of ~ 10000.

7.14 3D_DropShear

This is a 3D simulation of an initially static drop on a $150 \times 50 \times 50$ grid undergoing linear shear[48] using Lishchuk continuum-based mesophase interactions with Guo forcing. It operates in an identical manner to the `2D_DropShear` test case, with the initially stationary drop being subjected to linear shear. Figure 7.14 shows the fluid velocity streamlines (coloured using velocity magnitude) and the density (pressure) field applied to the surface of the deformed drop at the final time step.

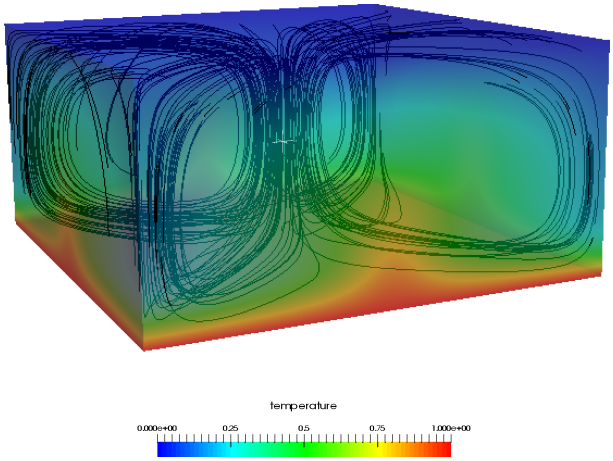


Figure 7.13: Plot of fluid temperature for LBE `3D_RayleighBenard` test case (scale: blue to red) with streamlines depicting convective flow

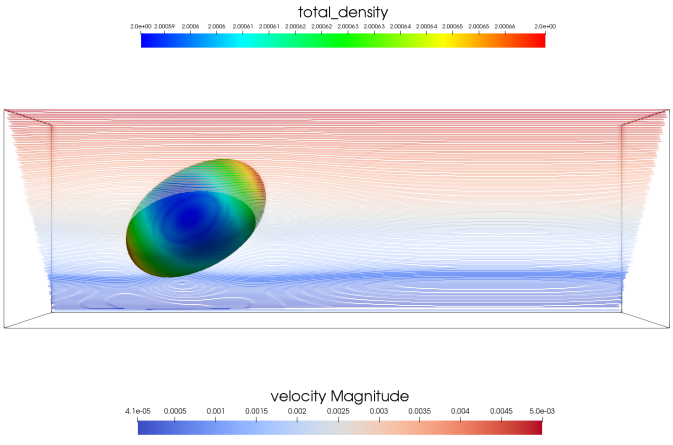


Figure 7.14: Plot of fluid streamlines and density on drop surface for LBE `3D_DropShear` test case (velocity and density scales: blue to red)

Part II

Dissipative Particle Dynamics (DPD)

Chapter 8

Dissipative Particle Dynamics: Basic Theory

8.1 Introduction

Dissipative Particle Dynamics (DPD) is an off-lattice, discrete particle method for modelling mesoscopic systems. It has little in common with Lattice Boltzmann methods, except in its application to systems of similar length and time scales.

The DPD method inherits its methodology from classical Molecular Dynamics (MD), particularly from Brownian Dynamics (BD). It differs from BD, however, in an important way: it is *Galilean invariant* and for this reason conserves hydrodynamic behaviour, while the BD method does not. Many systems are crucially dependent on hydrodynamic interactions and it is essential to retain this feature in the model. DPD is particularly useful for simulating 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 MD. DPD may also be used when such systems experience shear and flow gradients.

The DPD algorithm can be summarized by the following:

- A condensed phase system may be modelled as a system of free particles interacting directly through 'soft' forces.
- 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 Molecular Dynamics.

8.2 Outline of Method

In DPD[1] the system is modelled as a system of free particles, which are spherical and interact over a range that is of the same order as their diameters. The particles can be thought of as assemblies or aggregates of molecules, such as solvent molecules or polymers, or more simply as carriers of momentum.

The equations governing the time evolution in a DPD simulation resemble those of ordinary MD:

$$\frac{d\vec{v}_i}{dt} = \frac{\vec{F}_i}{m_i} \tag{8.1}$$

$$\frac{d\vec{r}_i}{dt} = \vec{v}_i \tag{8.2}$$

[1] The outline of the DPD method supplied here is based on [43].

in which $\vec{r}_i$, $\vec{v}_i$ and $\vec{F}_i$ are the position, velocity and force of the ith particle, which has mass m_i. The force on the particle is a sum of pair forces:

$$\vec{F}_i = \sum_{j \neq i}^{N} \left(\vec{F}_{ij}^C + \vec{F}_{ij}^D + \vec{F}_{ij}^R \right) \tag{8.3}$$

in which $\vec{F}_{ij}^C$, $\vec{F}_{ij}^D$ and $\vec{F}_{ij}^R$ 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. Additional pairwise forces may be included for more complicated systems, such as those involving chains of particles bonded together[103].

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:

$$V(r_{ij}) = \frac{1}{2} A_{ij} r_c \left(1 - \frac{r_{ij}}{r_c} \right)^2 \tag{8.4}$$

where $r_{ij} = |\vec{r}_j - \vec{r}_i|$, r_c is a cutoff radius and A_{ij} is the interaction strength. A_{ij} may be the same for all particle pairs or may be different for different particle types. The cutoff radius is related to the length scale for the particles: this can also vary for different particles and pairs.

Equation (8.4) gives rise to a repulsive force of the form

$$\vec{F}_{ij}^C = A_{ij} w^C(r_{ij}) \frac{\vec{r}_{ij}}{r_{ij}} = A_{ij} \left(1 - \frac{r_{ij}}{r_c} \right) \frac{\vec{r}_{ij}}{r_{ij}} \tag{8.5}$$

This is the deterministic or *conservative* force $\vec{F}_{ij}^C$ exerted on particle i by particle j. Note the switching function $w^C(r_{ij})$ and the force are zero when $r_{ij} > r_c$ and thus the particles have an effective radius of 1 in units of the cutoff radius r_c (often used as the length scale for the system).

The stochastic forces experienced by the particles is again pairwise in nature and takes the form

$$\vec{F}_{ij}^R = \sigma_{ij} w^R(r_{ij}) \zeta_{ij} \Delta t^{-\frac{1}{2}} \frac{\vec{r}_{ij}}{r_{ij}} \tag{8.6}$$

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:

$$\vec{F}_{ij}^D = -\gamma_{ij} w^D(r_{ij}) (\vec{r}_{ij} \cdot \vec{v}_{ij}) \frac{\vec{r}_{ij}}{r_{ij}^2} \tag{8.7}$$

where $w^D(r_{ij})$ is once again a switching function and $\vec{v_{ij}} = \vec{v_j} - \vec{v_i}$. 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.

$$\sigma_{ij}^2 = 2\gamma_{ij} k_B T \tag{8.8}$$

$$w^D(r_{ij}) = \left[w^R(r_{ij}) \right]^2 \tag{8.9}$$

In practice the switching functions are defined through

$$w^D(r_{ij}) = \left(1 - \frac{r_{ij}}{r_c} \right)^2 \qquad (r_{ij} < r_c) \tag{8.10}$$

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.

8.3 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:

$$p = \rho k_B T + \frac{1}{3V}\left\langle \sum_{j>i} (\vec{r}_i - \vec{r}_j)\cdot \vec{F}^C_{ij} \right\rangle \tag{8.11}$$

$$= \rho k_B T + \frac{2\pi}{3}\rho^2 \int_0^{r_c} rA\left(1-\frac{r}{r_c}\right) g(r) r^2 \; dr \tag{8.12}$$

where $g(r)$ is a radial distribution function for the soft sphere model[43] 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:

$$p = \rho k_B T + \alpha A \rho^2 \tag{8.13}$$

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 $A \approx \frac{75 k_B T}{\rho}$. Alternative equations of state may be obtained by modifying the functional form of conservative interactions to include localized densities (i.e. many-body DPD)[89, 130].

Transport coefficients for a DPD fluid can be derived using the expressions for the drag and stochastic forces[43, 64, 83]. For the given switching functions, the kinematic viscosity can be found to be

$$\nu \approx \frac{45 k_B T}{4\pi\gamma\rho r_c^3} + \frac{2\pi\gamma\rho r_c^5}{1575} \tag{8.14}$$

while the self-diffusion coefficient is given as

$$D \approx \frac{45 k_B T}{2\pi\gamma\rho r_c^3}. \tag{8.15}$$

The ratio of these two properties, the Schmidt number ($\mathrm{Sc} = \frac{\nu}{D}$), is therefore:

$$\mathrm{Sc} \approx \frac{1}{2} + \frac{(2\pi\gamma\rho r_c^4)^2}{70875 k_B T} \tag{8.16}$$

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[92]. Alternative thermostats are available which can model systems with higher Schmidt numbers[79, 121].

8.4 Derivation of Equilibrium

The derivation of the DPD algorithm is based on the Fokker-Planck equation

$$\frac{\partial \rho}{\partial t} = \mathcal{L}\rho \tag{8.17}$$

where ρ is the equilibrium distribution function and $\mathcal{L}$ is the evolution operator, which may be split into *conservative* and *dissipative* parts:

$$\mathcal{L} = \mathcal{L}^{\mathcal{C}} + \mathcal{L}^{\mathcal{D}} \tag{8.18}$$

with

$$\mathcal{L}^{\mathcal{C}} = -\sum_{i=1}^{N} \frac{\vec{p}_i}{m_i}\frac{\partial}{\partial \vec{r}_i} - \sum_{i\neq j}^{N} \vec{F}^C_{ij}\frac{\partial}{\partial \vec{p}_i} \tag{8.19}$$

$$\mathcal{L}^{\mathcal{D}} = \sum_{i=1}^{N} \hat{e}_{ij}\cdot\frac{\partial}{\partial \vec{p}_i}\left[\gamma w^D(\hat{e}_{ij}\cdot\vec{v}_{ij}) + \frac{\sigma^2}{2}\left\{w^R(r_{ij})\right\}^2 \hat{e}_{ij}\cdot\left\{\frac{\partial}{\partial \vec{p}_i} - \frac{\partial}{\partial \vec{p}_j}\right\}\right] \tag{8.20}$$

where $\hat{e}_{ij} = \frac{\vec{r_{ij}}}{r_{ij}}$.

When $\alpha = \gamma = 0$ then Equation (8.17) becomes

$$\frac{\partial \rho}{\partial t} = \mathcal{L}^{\mathcal{C}} \rho \tag{8.21}$$

for which the equilibrium solution is evidently

$$\rho^{eq} = \frac{1}{Z} \exp\left(\frac{1}{k_B T} \left[\sum_{i=1}^{N} \frac{p_i^2}{2m_i} + \frac{1}{2} \sum_{j \neq i}^{N} \phi\left(r_{ij}\right) \right] \right) \tag{8.22}$$

which is, of course, the Boltzmann distribution function for an equilibrium system. Thus it is apparent that for the simulation based on Equation (8.17) to maintain the same distribution function, the terms in the operator $\mathcal{L}^{\mathcal{D}}$ of Equation (8.20) must sum to zero. It follows that the conditions given in Equations (8.8) and (8.9) must apply.

8.5 Summary of Dissipative Particle Dynamics

DPD is a simple method. 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. In implementation it differs very little from Molecular Dynamics. 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.

Chapter 9

DL_MESO_DPD Features

9.1 Domain decomposition and linked-list cell calculations

The Domain Decomposition (DD) strategy is one of several ways to parallelize particle-based simulations[114]. Its basis is the division of the simulated system into equal-sized spatial blocks or domains, each of which is allocated to a specific processing unit of a parallel computer. The arrays defining the coordinates, velocities and forces for all N particles in the system are divided into sub-arrays of size $\approx \frac{N}{P}$ on each of the P processing units, with the particles allocated geometrically among them. In order for the strategy to work efficiently, the simulated system should possess a reasonably uniform density so that each processing unit is allocated as equal a portion of particle data as possible. The computation of forces and integration of the equations of motion are shared (more or less) equally between the processing units and to a large extent can be computed independently on each unit. While tricky to program, this method is conceptually simple and particularly suited to large-scale simulations.

The DD strategy which underpins DL_MESO_DPD is based on the link cell algorithm[53], which requires a relatively short-ranged cutoff for interparticle potentials and forces. There is a need for processing units to exchange 'halo data', i.e. sending the contents of link cells at the boundaries of each domain to neighbouring units so each may have all the necessary information to compute pairwise forces acting on the particles in its allotted domain. Similarly the force and virial contributions from particles in boundary halos need to be returned to their original processing units for summation. The link cell algorithm is also applied in serial by duplicating system data to create the boundary halo across periodic boundaries. Further parallelization of the link cell algorithm is possible by dividing up the link cells among the available threads when locating particle pairs and calculating forces between them.

The size of the boundary halo – which can be specified by the user in the `CONTROL` file – should not be greater than the minimum system dimension per domain; for good parallel performance, it is recommended that the halo size should be no larger than one-third of the smallest subdomain dimension. The maximum number of particles per domain is calculated after reading the input files: this gives the sizes of force, velocity and position arrays. This value should be large enough to hold all particles in each domain *plus* any particles in boundary halos, including duplicates when running in serial or using smaller numbers of processing units. If the density of the system is likely to be uneven, the user can increase the maximum number of particles by specifying an additional density variation in the `CONTROL` file.

For parallel running, the size of the transfer buffer needs to be sufficiently large to hold data for particles being transferred into boundary halos. This value is based upon the maximum number of particles likely to be transferred, which is initially estimated by assuming a constant particle density. While this initial estimate is usually sufficient for homogeneously distributed systems, it may need adjusting if the system becomes less evenly distributed: this value is checked either during equilibration or if many-body DPD interactions are in use and increased if required. (If a system is restarted with an `export` files or started using a `CONFIG` file, the likely number of buffer particles is checked and the initial size of the transfer buffer can be increased to better

suit the configuration at the start of the calculation.)

9.1.1 Intramolecular interactions

Intramolecular interactions may be handled in two different ways: either (1) locally with each processing unit being allocated a subset of bonds to deal with (including bonds across neighbouring units), or (2) globally with all units holding all bond data and sharing bonded particle positions, each carrying out all bond calculations and appropriately allocating forces to local particles. The former method may require larger boundary halo sizes for the bond lengths being simulated but is more efficient for larger numbers of molecules and processing units, while the latter method requires the sharing of information between all units but does not require halo information and is guaranteed to find all bonds.

Bookkeeping arrays list all particles involved in bonded interactions according to global index numbers and point to appropriate arrays of parameters to define the potential. If the 'key' bonded particle for a bond[1] moves from one processing unit to another, the entry in the bookkeeping array is also moved. At each time step a list of bonded particles in each domain is created to relate global index numbers to the local index numbers used by the processing unit in force, velocity and coordinate arrays. This global/local index list is sorted by global index number to allow cross-referencing to local index numbers by means of a binary search.

9.1.2 Electrostatic interactions

For systems with periodic boundary conditions DL_MESO_DPD uses Ewald sums to calculate Coulombic interactions (see Section 9.5). Calculation of real space components uses a similar but separate link cell algorithm as for other pairwise interactions. A separate cutoff for the real space part (r_e) should be specified, which can be larger than the maximum cutoff radius for non-electrostatic interactions ($r_{c,max}$). These interactions require a larger boundary halo than typically used for standard pairwise interactions.

9.2 Thermostats and integration algorithms

The integration algorithms in DL_MESO_DPD are based on the second-order Velocity Verlet (VV) scheme[131], which yields the positions, velocities and forces of particles at the same time and is generally used in molecular dynamics simulations. This algorithm has two stages. The first stage advances the particle velocities to time $t + \frac{1}{2}\Delta t$ by integrating the forces and uses the new half-step velocities to advance the position to time $t + \Delta t$:

$$\vec{v}_i\left(t + \tfrac{1}{2}\Delta t\right) = \vec{v}_i\left(t\right) + \frac{\Delta t}{2}\frac{\vec{F}_i\left(t\right)}{m_i} \tag{9.1}$$

$$\vec{r}_i\left(t + \Delta t\right) = \vec{r}_i\left(t\right) + \Delta t\vec{v}_i\left(t + \tfrac{1}{2}\Delta t\right) \tag{9.2}$$

The positions at the end of the time step allow the forces to be recalculated, before the second stage of the algorithm is applied to advance the half-step velocities to the end of the time step by integrating with the new force:

$$\vec{v}_i\left(t + \Delta t\right) = \vec{v}_i\left(t + \tfrac{1}{2}\Delta t\right) + \frac{\Delta t}{2}\frac{\vec{F}_i\left(t + \Delta t\right)}{m_i} \tag{9.3}$$

Five thermostatting algorithms are currently available in DL_MESO_DPD: two variants of the standard DPD thermostat and three alternative schemes which apply velocity corrections to the particles after force integration. The algorithm can be selected in the `CONTROL` file using the directive **ensemble** with the keyword **nvt** for constant volume simulations, **npt** for constant pressure simulations or **nst** for constant normal pressure and surface area or tension simulations. Dissipative force parameters and collision frequencies can be specified for each interacting species pair in the `FIELD` file. Frozen particles are involved in thermostatting algorithms due to the contributions they make to system virials and pressure; however they are excluded from the force integration algorithm and their velocities are reset to their previous values (usually zero).

[1] This is the first referenced particle in stretching bonds and the second for bond angles and dihedrals.

9.2.1 DPD thermostat with standard Velocity Verlet integration (MD-VV) (`mdvv`)

This algorithm uses the drag (dissipative) and random forces, $\vec{F}_{ij}^{D}$ and $\vec{F}_{ij}^{R}$ respectively as described in Chapter 8, as the system thermostat, i.e. the thermostatting force $\vec{F}_{ij}^{T} = \vec{F}_{ij}^{D} + \vec{F}_{ij}^{R}$. This thermostatting force is combined with all other forces between particles – pairwise conservative (standard and/or density-dependent), bonding, electrostatic, planar surface, external (body) forces – and integrated using the standard Velocity Verlet integrator.

The combination of the DPD thermostat with the standard MD-type VV algorithm is the simplest and least time-consuming thermostatting algorithm available in DL_MESO_DPD. (If no ensemble type is selected in the `CONTROL` file, DL_MESO_DPD will use this algorithm by default.) The drag force does, however, depend upon particle velocities and is therefore only approximated using the mid-step values: this frequently produces a system temperature higher than that specified by the user and requires a small time step Δt to reduce the offset to tolerable levels.

9.2.2 DPD thermostat with DPD Velocity Verlet integration (DPD-VV) (`dpdvv`)

As with the MD-VV scheme, this algorithm uses the drag and random forces as the system thermostat, which are combined with all other forces before being integrated using the Velocity Verlet scheme. The drag force is subsequently recalculated after the second stage using the velocities at the end of the time step[37].

The recalculation of drag forces after force integration helps to alleviate the temperature offset produced by the MD-VV, and hence larger time steps may be used for reasonable temperature control. It does require the re-use of the linked-list cells and inter-processor communications to recalculate the drag forces, which can increase the time required per time step compared to the MD-VV scheme.

9.2.3 DPD thermostat using Shardlow splitting (`dpds1` and `dpds2`)

Shardlow splitting[110] exploits an operator splitting approach to integrate the drag and random forces in the DPD thermostat. The changes in particle velocities due to the thermostat during a timestep can be determined separately from changes due to other forces (conservative, bonding interactions etc.) by rigorously expanding velocity Verlet integration of the thermostatting forces. The separate integration of drag and random forces can be described as a Trotter (first-order) or Strang (second-order) operator.

The first-order Shardlow operator can be achieved in two stages, equivalent to the first and second stages of velocity Verlet integration of drag and random forces ($\vec{F}_{ij}^{D}$ and $\vec{F}_{ij}^{R}$). The first stage is simply integration of the forces over half of the timestep Δt for all particle pairs (i and j) within the thermostat cutoff:

$$\vec{v}_i(t+\tfrac{1}{2}\Delta t) = \vec{v}_i(t) - \frac{\Delta t}{2m_i}\left[-\gamma_{ij} w^D(r_{ij})\left(\frac{\vec{r}_{ij}}{r_{ij}}\cdot\vec{v}_{ij}(t)\right) + \sigma_{ij} w^R(r_{ij})\,\zeta_{ij}\Delta t^{-\frac{1}{2}}\right]\frac{\vec{r}_{ij}}{r_{ij}} \tag{9.4}$$

$$\vec{v}_j(t+\tfrac{1}{2}\Delta t) = \vec{v}_j(t) + \frac{\Delta t}{2m_j}\left[-\gamma_{ij} w^D(r_{ij})\left(\frac{\vec{r}_{ij}}{r_{ij}}\cdot\vec{v}_{ij}(t)\right) + \sigma_{ij} w^R(r_{ij})\,\zeta_{ij}\Delta t^{-\frac{1}{2}}\right]\frac{\vec{r}_{ij}}{r_{ij}}. \tag{9.5}$$

The equations for the second velocity Verlet stage would ordinarily be implicit due to the inclusion of the relative velocity between particle pairs at the end of the timestep in the dissipative force. However, these equations can be rewritten in an explicit form that gives the velocity changes based on the mid-step relative velocities:

$$\begin{aligned}\vec{v}_i(t+\Delta t) &= \vec{v}_i(t+\tfrac{1}{2}\Delta t) - \frac{\Delta t}{2m_i\left(1+\frac{1}{2}\mu_{ij}\gamma_{ij}w^D(r_{ij})\,\Delta t\right)}\times \\ &\quad\left[-\gamma_{ij} w^D(r_{ij})\left(\frac{\vec{r}_{ij}}{r_{ij}}\cdot\vec{v}_{ij}(t+\tfrac{1}{2}\Delta t)\right) + \sigma_{ij} w^R(r_{ij})\,\zeta_{ij}\Delta t^{-\frac{1}{2}}\right]\frac{\vec{r}_{ij}}{r_{ij}}\end{aligned} \tag{9.6}$$

$$\begin{aligned}\vec{v}_j(t+\Delta t) &= \vec{v}_j(t+\tfrac{1}{2}\Delta t) + \frac{\Delta t}{2m_j\left(1+\frac{1}{2}\mu_{ij}\gamma_{ij}w^D(r_{ij})\,\Delta t\right)}\times \\ &\quad\left[-\gamma_{ij} w^D(r_{ij})\left(\frac{\vec{r}_{ij}}{r_{ij}}\cdot\vec{v}_{ij}(t+\tfrac{1}{2}\Delta t)\right) + \sigma_{ij} w^R(r_{ij})\,\zeta_{ij}\Delta t^{-\frac{1}{2}}\right]\frac{\vec{r}_{ij}}{r_{ij}},\end{aligned} \tag{9.7}$$

where $\mu_{ij} = \frac{m_i m_j}{m_i+m_j}$ is the reduced mass between the two particles. Other forces can subsequently be integrated using the standard velocity Verlet algorithm, which do not depend upon the particle velocities.

The second-order Shardlow operator is virtually identical to the first-order operator, except for replacing all instances of Δt with $\frac{\Delta t}{2}$ in the above velocity corrections and applying them twice, both before (starting at t) and after (starting at $t + \frac{1}{2}\Delta t$) integration of other forces. In both cases, correct integration of drag and random forces from known velocity data allows larger timesteps to be used than for standard Velocity Verlet integration. Both first-order and second-order Shardlow operators can be coupled with a barostat to allow for constant pressure, surface area or surface tension simulations[76].

9.2.4 Lowe-Andersen thermostat (`lowe`)

The Lowe-Andersen thermostat[79] is an alternative to the use of drag and random forces in the DPD thermostat, which uses a variant of the Andersen thermostat[1]. After all other forces (conservative, bonding interactions etc.) are integrated using the Velocity Verlet scheme, a random sample of particle pairs have their relative velocity replaced by a value from a Maxwellian distribution, i.e.

$$v_{ij}^{\circ} = \zeta_{ij}\sqrt{\frac{k_B T}{\mu_{ij}}} \tag{9.8}$$

where $\mu_{ij} = \frac{m_i m_j}{m_i + m_j}$ is the reduced mass between the two particles. The velocities of particles i and j thus become:

$$\vec{v}_i = \vec{v}_i - \frac{\mu_{ij}}{m_i}\left(-\left(\hat{e}_{ij}\cdot\vec{v}_{ij}\right) + v_{ij}^{\circ}\right)\hat{e}_{ij} \tag{9.9}$$

$$\vec{v}_j = \vec{v}_j + \frac{\mu_{ij}}{m_j}\left(-\left(\hat{e}_{ij}\cdot\vec{v}_{ij}\right) + v_{ij}^{\circ}\right)\hat{e}_{ij} \tag{9.10}$$

The probability of a particle pair being thermostatted is equal to $\Gamma\Delta t$, where Γ is defined as the collision frequency (with a maximum effective value of $\frac{1}{\Delta t}$), and the velocity corrections to particle pairs are applied in a random order to prevent biasing.

The above pairwise correction of velocities is equivalent to applying a thermostatting force equal to

$$\vec{F}_{ij}^{T} = \frac{\mu_{ij}}{\Delta t}\left(-\left(\hat{e}_{ij}\cdot\vec{v}_{ij}\right) + v_{ij}^{\circ}\right)\hat{e}_{ij} \tag{9.11}$$

and thus a virial correction of $-\vec{F}_{ij}^{T}\cdot\vec{r}_{ij}$ is applied for each particle pair being thermostatted.

The viscosity and self-diffusivity generated by this thermostat for a single species are

$$\nu = \frac{\pi\rho\Gamma r_c^5}{75m} \tag{9.12}$$

$$D = \frac{k_B T \tau_D}{m} \tag{9.13}$$

where τ_D is the decay time for velocity correlations and inversely proportional to the collision frequency. The Schmidt number is therefore proportional to $\frac{\Gamma^2}{k_B T}$ and can thus reach values up to $\mathcal{O}(10^7)$.

This thermostat is suited to systems with higher viscosities and low diffusivities while giving the correct system temperature for a wide range of time step sizes (within numerical errors due to Velocity Verlet force integration). Its implementation requires a randomized list of particle pairs due for thermostatting, which must be identical on all processing units and has to be worked through sequentially, using the latest available values for the velocities. Additional memory is thus required on each processing unit to store this list, which includes the particle numbers, processor numbers and $\vec{r}_{ij}$, and whose size will depend on the number of particles in the entire system and the value of Γ. Communication of particle velocities, the random relative velocity, v_{ij}°, and masses between processing units is necessary if a pair of particles crosses domain boundaries.

9.2.5 Peters thermostat (`peters`)

The Peters thermostat[92] is a modification of the Lowe-Andersen thermostat that reduces to standard DPD as the time step tends to zero. After integrating all forces using the Velocity Verlet scheme, all particle pairs have

their velocities modified (in a random order) using:

$$\vec{v}_i = \vec{v}_i - \frac{1}{m_i}\left(-a_{ij}\left(\hat{e}_{ij}\cdot\vec{v}_{ij}\right)\Delta t + b_{ij}\zeta_{ij}\sqrt{\Delta t}\right)\hat{e}_{ij} \tag{9.14}$$

$$\vec{v}_j = \vec{v}_j + \frac{1}{m_j}\left(-a_{ij}\left(\hat{e}_{ij}\cdot\vec{v}_{ij}\right)\Delta t + b_{ij}\zeta_{ij}\sqrt{\Delta t}\right)\hat{e}_{ij} \tag{9.15}$$

where the coefficients a_{ij} and b_{ij} are chosen so that

$$b_{ij} = \sqrt{2k_BTa_{ij}\left(1-\frac{a_{ij}\Delta t}{2\mu_{ij}}\right)}.$$

To ensure that the thermostat both reduces to the DPD thermostat as the time step reduces to zero and is not restricted by the choice of time step, the coefficients are chosen as follows:

$$a_{ij} = \frac{\mu_{ij}}{\Delta t}\left(1-\exp\left[-\frac{\gamma_{ij}\omega(r_{ij})\Delta t}{\mu_{ij}}\right]\right) \tag{9.16}$$

$$b_{ij} = \sqrt{\frac{k_BT\mu_{ij}}{\Delta t}\left(1-\exp\left[-\frac{2\gamma_{ij}\omega(r_{ij})\Delta t}{\mu_{ij}}\right]\right)} \tag{9.17}$$

The above velocity corrections give an equivalent thermostatting force of

$$\vec{F}_{ij}^{T} = \left(-a_{ij}\left(\hat{e}_{ij}\cdot\vec{v}_{ij}\right) + \frac{b_{ij}\zeta_{ij}}{\sqrt{\Delta t}}\right)\hat{e}_{ij} \tag{9.18}$$

and a correction to the virial of $-\vec{F}_{ij}^{T}\cdot\vec{r}_{ij}$ is also applied for each particle pair.

This thermostat can be used with larger time steps than the standard DPD thermostat but with similarly low system viscosities. As for the Lowe-Andersen thermostat, its implementation in parallel running uses a replicated data strategy for the list of particle pairs to be thermostatted, which requires additional memory on each processing unit for storing this list. The efficiency of the Peters thermostat therefore depends upon the total number of particles in the system: since all particle pairs are modified, calculation times for this thermostat will be comparable to those for the Lowe-Andersen thermostat when $\Gamma\Delta t \approx 1$.

9.2.6 Stoyanov-Groot thermostat (`stoyanov`)

The Stoyanov-Groot thermostat[121] is a combination of the Lowe-Andersen thermostat and a Galilean-invariant Nosé-Hoover thermostat which acts locally and on pairs of particles. During force calculations after the first Velocity Verlet stage, the choice to use either the Lowe-Andersen or Nosé-Hoover thermostats for each particle pair is made at random; the Lowe-Andersen thermostat is selected with a probability of $\Gamma\Delta t$. The system temperature is also determined in terms of relative velocities for all particle pairs, i.e.

$$k_BT^* = \frac{\sum_{i>j}\psi^T(r_{ij})\mu_{ij}\vec{v}_ij^{\,2}}{3\sum_{i>j}\psi^T(r_{ij})} \tag{9.19}$$

where $\psi^T(r_{ij})$ is a smearing function for the temperature, chosen to reduce to zero when $r_{ij} \geq r_c$: by default this is set as $\psi^T(r_{ij}) = 1$ for $r < r_c$. For all particle pairs that are to be subjected to the Nosé-Hoover thermostat (i.e. those not subjected to the Lowe-Andersen thermostat), an additional thermostatting force is included:

$$\vec{F}_{ij}^{T} = -\alpha w^T(r_{ij})\left(1-\frac{k_BT^*}{k_BT}\right)\left[\vec{v}_{ij}\cdot\hat{e}_{ij}\right]\hat{e}_{ij} \tag{9.20}$$

with α as a system-wide thermostat coupling parameter and $w^T(r_{ij})$ as a switching function, which by default is equivalent to $w^R(r_{ij}) = 1 - \frac{r_{ij}}{r_c}$ for standard DPD. A virial correction of $-\vec{F}_{ij}^{T}\cdot\vec{r}_{ij}$ is also made for each particle pair.

This thermostat can produce a wide range of system viscosities and diffusivities with good temperature control and hydrodynamics, using the collision frequency Γ to obtain the required Schmidt number. The replicated data strategy is again used for the Lowe-Andersen part, which requires memory in every processing unit to store the data for particle pair modification using the Lowe-Andersen scheme: the Nosé-Hoover scheme calculates the thermostatting forces locally.

9.3 Barostats

In addition to a thermostat, a barostat may be included in simulations to obtain a desired average pressure (P_0) by adjusting the size (and shape) of the simulation cell. DL_MESO_DPD includes two such algorithms: a Langevin-type barostat[60] and the Berendsen barostat[6], both of which have been coupled to all six available thermostats.

The isotropic pressure in a system is calculated using the virial theorem:

$$P(t) = \frac{1}{3V(t)}\left[\sum_i m_i v_i^2(t) + \sum_i \vec{F}_i(t)\cdot\vec{r}_i(t)\right] \tag{9.21}$$

while for anisotropic orthorhombic systems the pressure in dimension α, related to the instantaneous stress tensor component $\sigma_{\alpha\alpha}(t)$, is defined as

$$P_\alpha(t) = \frac{1}{V(t)}\left[\sum_i m_i v_{i,\alpha}^2(t) + \sum_i F_{i,\alpha}(t) r_{i,\alpha}(t)\right] \tag{9.22}$$

$$= \frac{\sigma_{\alpha\alpha}}{V(t)}. \tag{9.23}$$

In both equations, the instantaneous values required for barostats include only the interaction forces (e.g. soft pairwise interactions, bonds, electrostatics): they do not include virial contributions from thermostatting, which are included in reported values of system pressure.

All barostat definitions are expressed for the general anisotropic case with surface tensions (γ_0) acting in the z-dimension (i.e. NσT ensembles) and the simulation box remaining orthogonal (monoclinic): unlike solids, fluid systems typically modelled using DPD cannot resist shear. The anisotropic definitions can be applied to isotropic systems (NPT ensembles) by setting $P_x(t)$, $P_y(t)$ and $P_z(t)$ all equal to $P(t)$ and $\gamma_0 = 0$. Semi-isotropic systems with even variations in the x- and y-dimensions can be obtained by replacing $P_x(t)$ and $P_y(t)$ with the arithmetic mean value of these two pressure components. Constant normal pressure and surface area systems (NP$_n$AT ensembles) can be modelled by only applying the barostat in the z-dimension, keeping the box sizes in the x- and y-dimensions constant.

The barostat can be selected in the `CONTROL` file using either **ensemble npt** or **ensemble nst** as directives: the barostat type should be specified after the coupled thermostat. If using a constant normal pressure and surface tension ensemble, the surface tension should be specified in the same line. The word **semi** can be used to apply semi-isotropic variations in simulation box size in the x- and y-dimensions for constant surface tension simulations. The target system pressure must also be specified in the same file using the directive **pres**sure: DL_MESO_DPD will close with an error message if no target pressure is specified when using a barostat. Frozen particles are moved when a barostat is applied but their positions relative to the dimensions of the system remain constant during calculations.

9.3.1 Langevin barostat (`langevin`)

The governing equation for the Langevin barostat on an orthorhombic simulation cell[60] is the force exerted by the piston (expressed as the time-derivative of its momentum $p_{g,\alpha} = W_g u_{g,\alpha}$):

$$\dot{p}_{g,\alpha} = V\left(P_\alpha - P_0 + \frac{\gamma_0}{L_z}\right) + \frac{1}{N_f}\sum_i m_i v_i^2 - \gamma_p p_{g,\alpha} + \sigma_p \zeta_{p,\alpha}\Delta t^{-\frac{1}{2}} \quad (\alpha = x, y)$$

$$\dot{p}_{g,z} = V\left(P_z - P_0\right) + \frac{1}{N_f}\sum_i m_i v_i^2 - \gamma_p p_{g,z} + \sigma_p \zeta_{p,g}\Delta t^{-\frac{1}{2}}$$

where N_f is the number of degrees of freedom: for a three-dimensional box containing N moving (i.e. non-frozen) particles, $N_f = 3(N-1)$. γ_p and $\sigma_p \equiv \sqrt{\frac{2}{3}\gamma_p W_g k_B T}$ are respectively the drag and random coefficients for the piston and $\zeta_{p,\alpha}$ is a Gaussian random number for dimension α. The Gaussian random number is set to the same value for all three dimensions if operating isotropically or the same value for the x- and y-dimensions

if operating semi-isotropically. When both γ_p and σ_p are set to zero, the Langevin barostat reduces to the **extended system** method.

The subsequent simulation cell size L_α can be determined by

$$\dot{L}_\alpha = \frac{p_{g,\alpha} L_\alpha}{W_g} = u_{g,\alpha} L_\alpha \tag{9.24}$$

with the barostat mass W_g chosen to be equal to $N k_B T \tau_p^2$, where τ_p is the characteristic barostat time and should be set equal to between $\frac{2}{\gamma_p}$ and $\frac{10}{\gamma_p}$.

The velocities and positions of the particles are calculated by integration of slightly modified differential equations:

$$\frac{dv_{i,\alpha}}{dt} = \frac{F_{i,\alpha}}{m_i} - u_{g,\alpha} v_{i,\alpha} - \frac{1}{N_f} v_{i,\alpha} \sum_\alpha u_{g,\alpha} \tag{9.25}$$

$$\frac{dr_{i,\alpha}}{dt} = v_{i,\alpha} + u_{g,\alpha} r_{i,\alpha} \tag{9.26}$$

where the force on particle i, $\vec{F}_i$, includes any thermostatting forces: the time integral of these forces can be determined for all thermostat types.

The implementation of this barostat is carried out using the Velocity Verlet scheme to integrate the equations of motion for both the particles and the barostat. The first Velocity Verlet stage integrates the forces on the particles

$$v_{i,\alpha}\left(t + \tfrac{1}{2}\Delta t\right) = v_{i,\alpha}(t) + \frac{\Delta t}{2} \frac{F_{i,\alpha}(t)}{m_i} - \Delta t u_{g,\alpha} v_{i,\alpha} \tag{9.27}$$

which is followed by a similar integration for the barostat velocity:

$$u_{g,\alpha}\left(t + \tfrac{1}{2}\Delta t\right) = u_{g,\alpha}(t) + \frac{\Delta t}{2} \frac{F_{g,\alpha}(t)}{W_g} \tag{9.28}$$

before the positions and simulation box dimensions are updated:

$$r_{i,\alpha}(t + \Delta t) = \exp\left(u_{g,\alpha}\left(t + \tfrac{1}{2}\Delta t\right)\Delta t\right)\left\{r_{i,\alpha}(t) + \Delta t v_{i,\alpha}\left(t + \tfrac{1}{2}\Delta t\right)\right\} \tag{9.29}$$

$$L_\alpha(t + \Delta t) = \exp\left(u_{g,\alpha}\left(t + \tfrac{1}{2}\Delta t\right)\Delta t\right) L_\alpha(t) \tag{9.30}$$

At this point the forces at the end of the time step are calculated (including thermostatting forces), along with the system pressure. Since the the barostat force requires correct velocities for both the particles and barostat, an iterative procedure is required which begins by calculating an initial guess for the barostat velocity at the end of the time step:

$$u_{g,\alpha}^{(0)}(t + \Delta t) = u_{g,\alpha}(t - \Delta t) + \frac{2F_{g,\alpha}(t)\Delta t}{W_g}$$

Each iteration starts by calculating the particle velocities in a slightly modified second Velocity Verlet step:

$$v_{i,\alpha}^{(n+1)}(t + \Delta t) = \frac{\exp\left(u_{g,\alpha}\left(t + \frac{1}{2}\Delta t\right)\Delta t\right) v_{i,\alpha}\left(t + \frac{1}{2}\Delta t\right) + \frac{\Delta t}{2}\frac{F_{i,\alpha}(t+\Delta t)}{m_i}}{1 + u_{g,\alpha}^{(n)}\Delta t} \tag{9.31}$$

The barostat force is then calculated using the same Gaussian random numbers for each iteration:

$$F_{g,\alpha}^{(n+1)}(t + \Delta t) = V(P_\alpha - P_0) + \frac{1}{N_f}\sum_i m_i \left(v_i^{(n+1)}\right)^2 - \gamma_p u_{g,\alpha}^{(n)} W_g + \sigma_p \zeta_{p,\alpha} \tag{9.32}$$

and the barostat velocity is recalculated:

$$u_{g,\alpha}^{(n+1)}(t + \Delta t) = u_{g,\alpha}\left(t + \tfrac{1}{2}\Delta t\right) + \frac{F_{g,\alpha}^{(n+1)}(t + \Delta t)\Delta t}{2W_g} \tag{9.33}$$

Equations 9.31 to 9.33 are repeated until convergence in particle velocities is achieved, i.e. when

$$\frac{\sum_i \left(\vec{v}_i^{(n+1)}(t + \Delta t) - \vec{v}_i^{(n)}(t + \Delta t)\right)^2}{3N_f} \leq \epsilon$$

with ϵ as a numerical tolerance (set to 10^{-6} by default). This normally takes a few iterations per time step without requiring recalculation of particle forces or rescaling of particle coordinates.

9.3.2 Berendsen barostat (`berendsen`)

The governing equation for the Berendsen barostat[6] is a simple differential equation for the pressure:

$$\frac{dP_\alpha}{dt} = \frac{P_0 - P_\alpha}{\tau_p} \tag{9.34}$$

which can be solved to give a scaling factor for the simulation volume, $\vec{\eta}(t)$:

$$\begin{aligned} \eta_\alpha(t) &= 1 - \frac{\beta\Delta t}{3\tau_p}\left(P_0 - P_\alpha(t) - \frac{\gamma_0}{L_z}\right) \quad (\alpha = x, y) \\ \eta_z(t) &= 1 - \frac{\beta\Delta t}{3\tau_p}\left(P_0 - P_z(t)\right) \end{aligned} \tag{9.35}$$

where β is the isothermal compressibility of the system. The exact value of this property is not critical to the algorithm, since it relies on the ratio $\frac{\beta}{\tau_p}$.

The barostat is implemented using a variant of the Velocity Verlet algorithm; after the midstep velocities are determined, the scaling factor for time t is used to modify the particle positions and resize the simulation volume

$$r_{i,\alpha}\left(t + \Delta t\right) = \eta_\alpha(t) r_{i,\alpha}\left(t\right) + \Delta t v_{i,\alpha}\left(t + \tfrac{1}{2}\Delta t\right) \tag{9.36}$$

$$L_\alpha\left(t + \Delta t\right) = \eta_\alpha(t) L_\alpha\left(t\right) \tag{9.37}$$

The remainder of the Velocity Verlet algorithm is unchanged, although the scaling factor for the beginning of the next time step can be calculated at this point using Equation 9.35. No iteration is required for this barostat.

9.4 Particle-particle interactions

Pairwise particle interaction parameters in DL_MESO_DPD are specified in the `FIELD` file for each species pair using the directive **interact**ions. Interaction parameter values and lengthscales are stored in preparation for DPD calculations, with the maximum number of parameters per interaction dependent on the unbonded potential models in use. Energy parameters (ϵ_{ij} or A_{ij}) can either use the absolute values given in the `FIELD` files or use these values scaled by the specified system temperature ($k_B T$).

If interaction parameters between different particle species are not specified in the `FIELD` file, these can be determined by mixing rules. Energy and dissipative parameters (e.g. $A_{\alpha\beta}$ and $\gamma_{\alpha\beta}$ for DPD) can be determined for unlike particle pairs as geometric means of these parameters for same-species interactions, e.g.

$$A_{\alpha\beta} = \sqrt{A_{\alpha\alpha} A_{\beta\beta}}$$

while interaction lengths are set to the arithmetic mean, e.g.

$$r_{c,\alpha\beta} = \frac{r_{c,\alpha\alpha} + r_{c,\beta\beta}}{2}$$

It should be noted that interaction lengths should be less than or equal to the maximum interaction cut-off radius $r_{c,max}$: if any interaction lengths are larger, the potentials and forces will be truncated at $r_{c,max}$. If the maximum interaction cut-off radius is not specified in the `CONTROL` file, the largest interaction length specified in the `FIELD` file ($r_{c,\alpha\beta}$) will be used. Frozen particles are included in all interactions but the resultant forces on these particles are not subsequently integrated.

Four types of pairwise interactions between particles are available in DL_MESO_DPD: Lennard-Jones, Weeks-Chandler-Andersen, Groot-Warren (standard) DPD and many-body (density-dependent) DPD. In the case that many-body DPD interactions are used for any particle pair, mixing rules cannot be used and thus interaction parameters for *all* particle pairs must be specified by the user.

9.4.1 Lennard-Jones (`lj`)

The Lennard-Jones potential[62] is a mathematically simple model that approximates interactions (both attractive and repulsive) between pairs of neutral atoms or molecules:

$$U(r_{ij}) = 4\epsilon_{ij}\left[\left(\frac{\sigma_{ij}}{r_{ij}}\right)^{12} - \left(\frac{\sigma_{ij}}{r_{ij}}\right)^{6}\right] \tag{9.38}$$

where ϵ_{ij} is the depth of the potential well and σ_{ij} is the finite distance at which the potential is zero between particles i and j. This potential and its related force are calculated for all interparticle distances (r_{ij}) less than the maximum interaction cutoff radius $r_{c,max}$. Long-range system-wide corrections to the potential and virial are required:

$$U^{lr} = \frac{8\pi}{V}\sum_{\alpha}\sum_{\beta\geq\alpha}(2-\delta_{\alpha\beta})N_\alpha N_\beta \epsilon_{\alpha\beta}\left(\frac{\sigma_{\alpha\beta}^{12}}{9r_{c,max}^{9}} - \frac{\sigma_{\alpha\beta}^{6}}{3r_{c,max}^{3}}\right) \tag{9.39}$$

$$\mathcal{W}^{lr} = -\frac{16\pi}{V}\sum_{\alpha}\sum_{\beta\geq\alpha}(2-\delta_{\alpha\beta})N_\alpha N_\beta \epsilon_{\alpha\beta}\left(\frac{2\sigma_{\alpha\beta}^{12}}{3r_{c,max}^{9}} - \frac{\sigma_{\alpha\beta}^{6}}{r_{c,max}^{3}}\right) \tag{9.40}$$

where $\delta_{\alpha\beta}$ is the Kronecker delta (1 when $\alpha = \beta$, 0 when $\alpha \neq \beta$) and N_α the total number of particles of species α. These corrections multiplied by the volume are stored in the array `clr` to eliminate the need to adjust them if the system volume is changed by a barostat.

9.4.2 Weeks-Chandler-Andersen (`wca`)

The Weeks-Chandler-Andersen potential[137] is a modification of the Lennard-Jones potential to produce purely repulsive, short-range interactions:

$$U(r_{ij}) = 4\epsilon_{ij}\left[\left(\frac{\sigma_{ij}}{r_{ij}}\right)^{12} - \left(\frac{\sigma_{ij}}{r_{ij}}\right)^{6}\right] + \epsilon_{ij}. \tag{9.41}$$

This interaction is applied for interparticle distances up to $2^{\frac{1}{6}}\sigma_{ij}$, which should be less than or equal to the maximum cutoff radius $r_{c,max}$. No long-range corrections to potential energy or virials are required.

9.4.3 Standard DPD (`dpd`)

The Groot-Warren (standard) form of DPD[43] uses the following purely repulsive, soft potential:

$$U\left(r_{ij}\right) = \frac{1}{2}A_{ij}r_{c,ij}\left(1 - \frac{r_{ij}}{r_{c,ij}}\right)^{2}. \tag{9.42}$$

This conservative interaction is applied for interparticle distances up to $r_{c,ij}$, which should be less than or equal to the maximum value $r_{c,max}$.

9.4.4 Many-body DPD (`mdpd`)

The conservative force in standard DPD depends only upon the species interacting and the interparticle separation, which yields a quadratic equation of state. Many-body DPD[89, 130] is a method of providing alternative thermodynamic behaviours to DPD particles by making conservative forces additionally dependent on local densities.

The free energy of an inhomogeneous system with density $\rho(r)$ can be defined as the following in both continuous and ensemble-averaged discrete forms:

$$\mathcal{F} = \int d\vec{r}\rho(\vec{r})\psi(\overline{\rho}(\vec{r})) \tag{9.43}$$

$$= \left\langle \sum_i \psi(\overline{\rho}(\vec{r_i})) \right\rangle \tag{9.44}$$

where $\psi(\rho)$ is the free energy per particle in a homogeneous system and $\overline{\rho}(\vec{r_i})$ is a function related to the density at (and near) the position of particle i ($\vec{r_i}$). The latter can be approximated by a function dependent on the positions of particles close to particle i ($\tilde{\rho}_i$) to allow the calculation of an instantaneous free-energy:

$$\tilde{\mathcal{F}} = \sum_i \psi(\tilde{\rho}_i).$$

The effective (conservative) force on particle i can be obtained from the spatial derivative of the free energy, although only the excess part (equivalent to the potential energy U) is required since the kinetic motion of particles automatically accounts for the ideal contribution:

$$\vec{F}_i^C = -\frac{\partial \tilde{\mathcal{F}}^{ex}(\{\vec{r}_k\})}{\partial \vec{r}_i} = -\sum_j \frac{\partial \psi^{ex}(\tilde{\rho}_j)}{\partial \vec{r}_i} \tag{9.45}$$

The force can also be expressed in terms of pairwise interactions, taking a form similar to the standard DPD conservative force (Equation 8.5):

$$\vec{F}_{ij}^C = \left(\frac{\partial \psi^{ex}(\tilde{\rho}_i)}{\partial \tilde{\rho}_i} + \frac{\partial \psi^{ex}(\tilde{\rho}_j)}{\partial \tilde{\rho}_j} \right) w^C(r_{ij}) \frac{\vec{r}_{ij}}{r_{ij}} \tag{9.46}$$

The local-density approximation can be defined as a weighted average of instantaneous densities:

$$\begin{aligned} \tilde{\rho}_i &= \int d\vec{r}\, w^\rho(\mid \vec{r} - \vec{r}_i \mid) \rho(\vec{r}, \{\vec{r}_k\}) \\ &= \sum_{j \neq i} \int d\vec{r}\, w^\rho(\mid \vec{r} - \vec{r}_i \mid) \delta(\vec{r} - \vec{r}_j) \\ \tilde{\rho}_i &= \sum_{j \neq i} w^\rho(r_{ij}) \end{aligned} \tag{9.47}$$

with $w^\rho(r)$ as the weight function vanishing beyond a cutoff r_d (which can be equal to r_c or smaller) and normalized so that $\int_0^\infty 4\pi r^2 w^\rho(r) dr = 1$. The most frequently used form for the weight function is

$$w^\rho(r_{ij}) = \frac{15}{2\pi r_d^3} \left(1 - \frac{r_{ij}}{r_d} \right)^2 \qquad (r_{ij} < r_d) \tag{9.48}$$

which reduces to standard DPD when the excess free energy per particle is set to $\psi^{ex}(\tilde{\rho}) = \frac{\pi}{30} A \tilde{\rho}$.

Multiple-component many-body DPD is also possible by defining partial local densities, e.g. for component α

$$\tilde{\rho}_i^\alpha = \sum_{j \in \alpha, j \neq i} w^\rho(r_{ij}) \tag{9.49}$$

and generalizing Equation 9.46:

$$\vec{F}_{ij}^C = \left(\frac{\partial \psi_{c(i)}^{ex}(\{\tilde{\rho}^\alpha\}_i)}{\partial \tilde{\rho}_{c(j)}} + \frac{\partial \psi_{c(j)}^{ex}(\{\tilde{\rho}^\alpha\}_j)}{\partial \tilde{\rho}_{c(i)}} \right) w^C(r_{ij}) \frac{\vec{r}_{ij}}{r_{ij}} \tag{9.50}$$

with $c(i)$ as the component to which particle i belongs (e.g. if $i \in \alpha$, $c(i) = \alpha$) and $\{\tilde{\rho}^\alpha\}_i$ as the set of local densities of different components at the position of particle i.

The `manybody_module` includes a routine (`local_density`) to calculate local densities for each species using Equation 9.49: the overall density for each particle can be obtained by a simple sum over all species. The user can modify the routine `manybody_potential` in this module and `conservativeforce` in `field_module` to apply their own choices for many-body potentials and forces respectively. Up to five many-body interaction parameters per species pair can be specified in the `FIELD` file.

The many-body DPD example provided with DL_MESO_DPD produces a van der Waals-like equation of state and can be used to model vapour/liquid interfaces[132]. The potential (excess free energy) per particle is given by:

$$\psi^{ex}(\tilde{\rho}) = \frac{\pi}{30} A_{ij} \overline{\tilde{\rho}} + \frac{\pi r_d^4}{30} B_{ij} \tilde{\rho}^2 \tag{9.51}$$

where $\overline{\tilde{\rho}}$ is equivalent to $\tilde{\rho}$ but with the cutoff set to r_c instead of r_d. The associated pairwise force is equal to

$$\vec{F}_{ij}^C = \left[A_{ij} \left(1 - \frac{r_{ij}}{r_{c,ij}} \right) + B_{ij} \left(\rho_i + \rho_j \right) \left(1 - \frac{r_{ij}}{r_d} \right) \right] \frac{\vec{r_{ij}}}{r_{ij}} \tag{9.52}$$

where ρ_i and ρ_j are the local densities considering *all* particle species. In the routine provided, the terms with A_{ij} for both force and potential are calculated as though they are standard DPD. By setting $A_{ij} < 0$ and $B_{ij} > 0$, a vapour/liquid mixture can be modelled and its equation of state for a single component is given as

$$p = \rho k_B T + \alpha A \rho^2 + 2\alpha B r_d^4 \left(\rho^3 - c\rho^2 + d\right)$$

where $\alpha \approx 0.101$, c and d are numerical offsets. Parameters A_{ij} and B_{ij} for each interacting pair of species can be specified in the `FIELD` file, although it is recommended that the values of B_{ij} are kept the same for all species pairs to ensure the many-body DPD forces are conservative (i.e. $\nabla_j \vec{F}_i = \nabla_i \vec{F}_j$)[134].

9.5 Long-ranged Electrostatic (Coulombic) Potentials

Compared to other interactions in DPD, electrostatic interactions act over considerably longer ranges, which can also include periodic images of the system. The governing equation for finding the electric potential is the Poisson equation, shown here in dimensionless form[42]:

$$\nabla \cdot (p(\vec{r}) \nabla \varphi) = -\Gamma \rho \tag{9.53}$$

where φ is the electric potential, ρ the charge density (concentration of cations minus concentration of anions per unit volume), $p(\vec{r})$ the local polarizability relative to a reference medium (e.g. water) and Γ the coupling constant for the reference medium[2]. The latter is given by

$$\Gamma = \frac{e^2}{k_B T \epsilon_0 \epsilon_r r_c}$$

with e as the electron charge, ϵ_0 the dielectric constant of a vacuum and ϵ_r the relative permittivity of the reference medium. For water at room temperature (298K) with N_m molecules per DPD particle, $\Gamma \approx 20.00 N_m^{-\frac{1}{3}}$. (The value of Γ can be scaled with the specified temperature if the relevant option is selected in the `FIELD` file.)

The total electrostatic potential energy can be expressed as a sum of Coulombic energies (which also include periodic images), which can be derived from the charge distributions for each ion by integrating the Poisson equation. The most frequently used solution in molecular dynamics is for point charges, i.e. defining $\delta = (\vec{r} - \vec{r}_i)$ as the Dirac delta function, when $\rho_i = q_i \delta(\vec{r} - \vec{r}_i)$ the total electrostatic potential energy of the system is given by

$$U_{tot}^E = \frac{\Gamma}{4\pi} \sum_i \sum_{j>i} \frac{q_i q_j}{r_{ij}}. \tag{9.54}$$

This gives the standard Coulombic potential between two point charges:

$$U_{ij}^E = \frac{\Gamma}{4\pi} \frac{q_i q_j}{r_{ij}}. \tag{9.55}$$

The Coulombic potential between point charges often cannot be used unmodified for DPD simulations: soft beads used in combination with unlike point charges may collapse on top of each other, forming infinitely strong ion pairs[42]. To prevent this from happening, the charges can be spread out over a finite volume using a smearing charge distribution.

9.5.1 Standard Ewald sum

The electrostatic potential can often be determined using an Ewald summation[29], which can be applied to neutral periodic (or pseudo-periodic) systems with charged particles. The basic model is a system of point charges mutually interacting via the Coulomb potential. The Ewald sum method makes two changes to this system:

1. Each point charge is effectively neutralised at long range by superimposing a spherical Gaussian cloud of opposite charge centred on the particle.

[2] This property is related to the Bjerrum length: $\lambda_B = \frac{\Gamma r_c}{4\pi}$.

2. A set of further Gaussian charges is superimposed on the point charges with the same sign to nullify the first set of Gaussians.

The assembly of point ions and Gaussian charges produced by the first modification becomes the *real space* part of the Ewald sum: this is now short-ranged and can be calculated in a pairwise fashion within a cutoff radius. The potential from the second set of Gaussian charges is obtained by solving the Poisson equation using a Fourier series in *reciprocal space*: this can be calculated directly by analytical means as the Fourier transform of a Gaussian is an exponential. To complete the Ewald sum, an additional self-energy correction is needed to remove the potential of a Gaussian acting on its own site.

The electrostatic potential for the system is therefore split into a number of terms:

$$\begin{aligned} U^E_{tot} &\equiv q\varphi \\ &= U^{sr} + U^{lr} + U^{sc} + U^{cc} + U^{lr,corr} + U^{dip} \end{aligned} \tag{9.56}$$

where U^{sr}, U^{lr} and U^{sc} are the short-range (real space), long range (reciprocal space) and self-energy correction terms respectively. An additional term for systems with a net charge, U^{cc}, can also be included, while a dipole-based term U^{dip} can be included for systems with slab-like geometries (e.g. with non-periodic boundary conditions in one dimension). The electrostatic interactions between pairs of frozen particles should be excluded and while these can be simply omitted from the real space term, additional corrections need to be made to the reciprocal space term ($U^{lr,corr}$): these cannot be calculated easily in reciprocal space so they are calculated in real space.

For systems with point charges, the real space (short-range) potential energy between particles i and j is given as

$$U^{sr}_{ij} = \frac{\Gamma q_i q_j}{4\pi r_{ij}} \mathrm{erfc}(\alpha r_{ij}) \tag{9.57}$$

where α is the Ewald convergence parameter[3] and erfc is the complementary error function (note that $\mathrm{erf}(x) + \mathrm{erfc}(x) = 1$). This term is evaluated for all pairs within an electrostatic short-range cutoff, r_e. The real space force is given by

$$\vec{F}^{E,sr}_{ij} = \frac{\Gamma q_i q_j}{4\pi r^2_{ij}} \left(\frac{2\alpha r_{ij}}{\sqrt{\pi}} \exp(-\alpha^2 r^2_{ij}) + \mathrm{erfc}(\alpha r_{ij}) \right) \frac{\vec{r}_{ij}}{r_{ij}} \tag{9.58}$$

and the associated virial term by

$$\begin{aligned} \mathcal{W}^{E,sr}_{ij} &= -\vec{F}^{E,sr}_{ij} \cdot \vec{r}_{ij} \\ &= -\frac{\Gamma q_i q_j}{4\pi r_{ij}} \left(\frac{2\alpha r_{ij}}{\sqrt{\pi}} \exp(-\alpha^2 r^2_{ij}) + \mathrm{erfc}(\alpha r_{ij}) \right) \end{aligned} \tag{9.59}$$

The total long range contribution to electrostatic potential for the point charge system is given as follows:

$$U^{lr}_{tot} = \frac{\Gamma}{2V} \sum_{\vec{k} \neq 0}^{\vec{k}_{max}} \frac{\exp\left(-\frac{k^2}{4\alpha^2}\right)}{k^2} \left| \sum_j q_j \exp\left(-i\vec{k} \cdot \vec{r}_j\right) \right|^2 \tag{9.60}$$

where i is the imaginary constant ($\sqrt{-1}$) and $\vec{k}$ is a reciprocal vector, which can be defined for an orthogonal periodic simulation box of dimensions $L_x \times L_y \times L_z$ as

$$\vec{k} = \begin{bmatrix} \frac{2\pi k_1}{L_x} \\ \frac{2\pi k_2}{L_y} \\ \frac{2\pi k_3}{L_z} \end{bmatrix}$$

where k_1, k_2 and k_3 are integers (positive and negative) from zero to maximum values specified by the user as k_1^{max}, k_2^{max} and k_3^{max} respectively for x-, y- and z-dimensions. (Adjustments can be made to the reciprocal

[3] This parameter is related to the standard deviation of the Gaussian function, σ, by $\alpha \equiv \frac{1}{\sqrt{2}\sigma}$.

vector to account for shearing boundaries[138].) A maximum reciprocal vector, $\vec{k}_{max}$, can be defined using the maximum k values: while an infinite number of images should theoretically be used, in practice smaller finite numbers can be used as the reciprocal space term converges rapidly. The reciprocal space force acting on each charged particle can be determined by differentiation, i.e.

$$\vec{F}_j^{E,lr} = -\frac{\Gamma q_j}{V} \sum_{\vec{k}\neq 0}^{\vec{k}_{max}} i\vec{k} \exp\left(i\vec{k}\cdot\vec{r}_j\right) \frac{\exp\left(-\frac{k^2}{4\alpha^2}\right)}{k^2} \sum_n q_n \exp\left(-i\vec{k}\cdot\vec{r}_n\right) \tag{9.61}$$

and the virial can be obtained by differentiation of the potential with volume[113]:

$$\mathcal{W}_{tot}^{E,lr} = -\frac{\Gamma}{2V} \sum_{\vec{k}\neq 0}^{\vec{k}_{max}} \frac{\exp\left(-\frac{k^2}{4\alpha^2}\right)}{k^2} \left|\sum_j q_j \exp\left(-i\vec{k}\cdot\vec{r}_j\right)\right|^2 \left(1 - \frac{k^2}{2\alpha^2}\right). \tag{9.62}$$

The self-correction term required to remove all Gaussian distributions in real space is given by the limit of the pairwise reciprocal space term (i.e. expressed similarly to the real space term) for zero separation, i.e.

$$U_{tot}^{sc} = -\frac{\Gamma\alpha}{4\pi^{\frac{3}{2}}} \sum_i q_i^2 \tag{9.63}$$

and is constant for a given system over all time steps. No additional forces or virial terms are required for self-correction since this potential term is not dependent on the positions of charge particles.

If the system is not charge neutral, a correction for systems with net charge[73] can also be made:

$$U_{tot}^{cc} = -\frac{\Gamma}{8\alpha^2 V} \left(\sum_i q_i\right)^2 \tag{9.64}$$

and again no additional force or virial terms are required.

If a non-periodic boundary condition is included in dimension α, a potential correction based on the dipole moment in that dimension[143] can be made:

$$U_{tot,\alpha}^{dip} = \frac{\Gamma}{2V} \left(\sum_i q_i r_{i,\alpha}\right)^2 \tag{9.65}$$

which also involves an additional force on each particle in that dimension, i.e.

$$F_{j,\alpha}^{dip} = \frac{\Gamma q_j}{V} \sum_i q_i r_{i,\alpha}. \tag{9.66}$$

To minimise periodic effects due to the multiple images of the reciprocal space potential, an additional vacuum gap can be included to the volume used for reciprocal space calculations to put some empty space between images of the slab in dimension α: this gap can be specified for DL_MESO_DPD without increasing the system volume.

To remove the reciprocal space terms between pairs of frozen particles, the reciprocal space potential can be expressed in a similar fashion to the real space term and then subtracted, i.e.

$$U_{ij}^{lr,corr} = -\frac{\Gamma q_i q_j}{4\pi r_{ij}} \mathrm{erf}(\alpha r_{ij}). \tag{9.67}$$

Similarly the forces and virial contributions for frozen particle pairs need to be removed:

$$\vec{F}_{ij}^{lr,corr} = \frac{\Gamma q_i q_j}{4\pi r_{ij}^2} \left(\frac{2\alpha r_{ij}}{\sqrt{\pi}} \exp(-\alpha^2 r_{ij}^2) - \mathrm{erf}(\alpha r_{ij})\right) \frac{\vec{r}_{ij}}{r_{ij}} \tag{9.68}$$

$$\mathcal{W}_{ij}^{lr,corr} = \frac{\Gamma q_i q_j}{4\pi r_{ij}} \left(\mathrm{erf}(\alpha r_{ij}) - \frac{2\alpha r_{ij}}{\sqrt{\pi}} \exp(-\alpha^2 r_{ij}^2)\right). \tag{9.69}$$

Even though the forces on frozen particles are ignored during integration, they do contribute to the system potential energy, virial and stress tensor terms and should therefore be evaluated. These corrections only need

to be calculated once for constant volume systems (as the frozen particles do not move during the simulation) but would need to be recalculated at every time step if a barostat is used.

Since smeared charge distributions are required for DPD simulations, the above expressions for the various terms of the Ewald sum would need modification. However, it is possible to select appropriate smearing functions that converge towards the standard Coulombic potential for larger particle separations; by doing this, the point charge forms for the reciprocal space, self-energy and net system charge terms in the Ewald sum can be used unmodified and only the real space and reciprocal space corrections for frozen particles would need to be changed. Two examples of distributions that can be used for DPD simulations in DL_MESO_DPD are given below.

The real space (short-range) component of electrostatic interactions is calculated using a link cell algorithm with a cutoff radius of r_e, which can be larger than that used for standard pairwise interactions ($r_{c,max}$). The reciprocal space (long-range) component is calculated using the scheme described by [115] and parallelized by distribution over particle sites, which requires global summations but is more efficient in terms of memory usage than distribution of $\vec{k}$ vectors. The same routine also adds the self-interaction and charged system corrections, which are calculated before the simulation starts. Corrections to forces, potential energy, virial and stress tensors to exclude interactions between charged frozen particles can be calculated: these only need to be calculated once for constant volume systems (since the frozen particles do not move) but have to be recalculated at each time step if a barostat is applied.

This method can be invoked by using the directives **ewald** in the `CONTROL` file: this can be used either to directly set the real-space convergence parameter (α) and k-space vector range ($\vec{k}_{max}$), or specify a required precision (i.e. a maximum relative error, ϵ) to calculate the convergence parameter and/or k-space vector.

$$\epsilon_{real} \approx \frac{\mathrm{erfc}(\alpha r_e)}{r_e} \tag{9.70}$$

$$\epsilon_{recip} \approx \frac{\exp\left(-\frac{k^2}{4\alpha^2}\right)}{k^2} \tag{9.71}$$

where $k = \frac{2\pi}{L} k_{max}$. If Gaussian charge smearing is selected with the equal length scale option for charge smearing and Ewald summation, the value of α specified by the **ewald** directive (either directly specified or using the precision option) will be ignored if the smearing length σ_G is specified.

The permittivity coupling constant Γ can be set using the **perm**ittivity directive in the `CONTROL` file, or alternatively the Bjerrum length λ_B can be specified with the **bjer**rum (length) directive. Vacuum gaps for systems with non-periodic boundary conditions (i.e. hard surfaces or frozen bead walls) can be specified using the directive **vacu**um (gap) in the `CONTROL` file, although these will be ignored for systems with shearing boundaries and have no effect on the physical system volumes being modelled: non-zero values can be used to invoke dipole moment corrections to potentials and forces.

9.5.2 Smooth Particle Mesh Ewald (SPME)

Smooth Particle Mesh Ewald is a modification of the standard Ewald method[28], which changes the way in which the reciprocal space terms are calculated. Rather than directly calculating the Fourier transforms of the Gaussian charges, the charges are added to a three-dimensional rectangular grid using an interpolation procedure involving (complex) B-splines and a Fast Fourier Transform (FFT) is applied to the charge grid. For larger k-space vectors, this can greatly reduce the computational cost compared to the direct Ewald sum.

The complex exponential terms for positions in the direct sum – $\exp\left(-i\vec{k}\cdot\vec{r}_j\right)$ – are interpolated using the approximation (shown here for one dimension):

$$\exp\left(2\pi i u_j k/L\right) \approx b(k) \sum_{\ell=-\infty}^{\infty} M_n\left(u_j - \ell\right) \exp\left(2\pi i k \ell / K\right) \tag{9.72}$$

where k is the integer index of the reciprocal space vector in a principal direction, K and L the total number of grid points and the box length respectively in the same direction, and u_j the fractional coordinate of ion j

scaled by the factor K (i.e. $u_j = Kr_j$). The coefficients $M_n(u)$ are B-splines of order n – noting that these are dependent on K and thus limit the infinite sum over ℓ – and the factor $b(k)$ is a constant that can be computed as follows:

$$b(k) = \exp\left(2\pi i(n-1)k/K\right)\left[\sum_{\ell=0}^{n-2} M_n(\ell+1)\exp\left(2\pi ik\ell/K\right)\right]^{-1}. \tag{9.73}$$

The structure factor, ordinarily defined as the following for the standard Ewald sum:

$$S\left(\vec{k}\right) = \sum_n q_n \exp\left(-i\vec{k}\cdot\vec{r}_n\right), \tag{9.74}$$

can be approximated by:

$$S\left(\vec{k}\right) \approx b_1(k_1)b_2(k_2)b_3(k_3)Q^\dagger(k_1,k_2,k_3) \tag{9.75}$$

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

$$Q(\ell_1,\ell_2,\ell_3) = \sum_{j=1}^{N} q_j \sum_{n_1,n_2,n_3} M_n(u_{j,x}-\ell_1-n_1L_x)M_n(u_{j,y}-\ell_2-n_2L_y)M_n(u_{j,z}-\ell_3-n_3L_z) \tag{9.76}$$

where the sums over n_1, n_2 and n_3 are required to capture contributions from all relevant periodic cell images (i.e. nearest images).

The total long range (reciprocal space) contribution to electrostatic potential for the point charge system can be approximated as:

$$U_{tot}^{lr} = \frac{\Gamma}{2V}\sum_{k_1,k_2,k_3} G^\dagger(k_1,k_2,k_3)Q(k_1,k_2,k_3) \tag{9.77}$$

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

$$G(k_1,k_2,k_3) = \frac{\exp\left(-\frac{k^2}{4\alpha^2}\right)}{k^2}B(k_1,k_2,k_3)\left(Q^\dagger(k_1,k_2,k_3)\right)^* \tag{9.78}$$

with $\left(Q^\dagger(k_1,k_2,k_3)\right)^*$ as the complex conjugate of $Q^\dagger(k_1,k_2,k_3)$ and

$$B(k_1,k_2,k_3) = \left|b_1(k_1)\right|^2\left|b_2(k_2)\right|^2\left|b_3(k_3)\right|^2. \tag{9.79}$$

The function $G(k_1,k_2,k_3)$ is 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 $Q(k_1,k_2,k_3)$.

The forces on each charged particle can be calculated from

$$\vec{F}_j^{E,lr} = -\nabla_j U^{lr} = -\frac{\Gamma}{V}\sum_{k_1,k_2,k_3} G^\dagger(k_1,k_2,k_3)\nabla_j Q(k_1,k_2,k_3), \tag{9.80}$$

which is straightforward due to the recursive properties of B-splines. The approximations made for SPME mean that the total forces on all charged particles may not be zero: this can be overcome by subtracting the net force from each particle. The virials and stress tensor are calculated in a similar way to the standard Ewald sum but using the charge grid to determine the structure factor as shown above, e.g.

$$\mathcal{W}_{tot}^{E,lr} = -\frac{\Gamma}{2V}\sum_{k_1,k_2,k_3}\frac{\exp\left(-\frac{k^2}{4\alpha^2}\right)}{k^2}B(k_1,k_2,k_3)\left|Q^\dagger(k_1,k_2,k_3)\right|^2\left(1-\frac{k^2}{2\alpha^2}\right). \tag{9.81}$$

Electrostatic interactions using the Smooth Particle Mesh Ewald technique are currently calculated in DL_MESO_DPD by putting together a charge grid replicated over all processor cores, which is also solved by every core with a three-dimensional FFT solver[4]. Different three-dimensional Fast Fourier Transform (FFT) solvers are available at compile time with the necessary parts of the code activated using compiler flags:

[4]Work is ongoing to allow for distributed charge arrays with parallel FFT solvers, e,g, [11], that can determine Fourier transforms for locally held grid points.

- An internal (serial) 3D FFT solver (no compiler flag required)[112];
- IBM Engineering and Scientific Subroutine Library (ESSL);
- The Fastest Fourier Transform in the West (FFTW)[33].

There are restrictions on the size of the maximum k-space vector: certain FFT solvers may require the values in this vector to be products of powers of 2, 3 and/or 5, and DL_MESO will adjust any non-qualifying values upwards. The real space part is calculated in the same manner as the standard Ewald sum, as are corrections to forces, potentials, virials and stress tensors for pairs of frozen particles.

SPME can be invoked by using the directive **spme** in the `CONTROL` file. One option is to specify the real-space convergence parameter (α and k-space vector range ($\vec{k}_{max}$), while the other is to specify the required precision (maximum relative error) and estimate values for α and $\vec{k}_{max}$. Note that because the maximum k-vector in SPME represents the sides of a cube rather than a radius of convergence, the maximum reciprocal vector should be double the size of that used for the standard Ewald sum. Both options for the **spme** directive in the `CONTROL` file can also be used to specify the B-spline interpolation order n: if no value is specified, a default value of 8 will be assumed, while a minimum value of 4 will be accepted and any odd number will be increased by 1 to ensure an even order is used.

9.5.3 Charge smearing

Charge smearing for DPD can be selected to reduce the possibility of opposite-charge collapse at shorter separations between ion pairs while still ensuring the potential is Coulombic (proportional to the reciprocal of separation) at larger distances and allowing for unmodified calculations of reciprocal space terms in Ewald sums. Five types of charge smearing are currently available in DL_MESO_DPD: no charge smearing (point charges), linear[42], Slater-type[41] (in both truncated and exact forms), Gaussian[21, 136] and sinusoidal dampling[34].

All forms of charge smearing can be expressed in terms of corrections to the standard (point-charge) Coulombic potential, i.e.

$$U(r_{ij}) = \frac{\Gamma q_i q_j}{4\pi r_{ij}} \left[1 - f(r_{ij})\right] \tag{9.82}$$

where $f(r_{ij})$ is a distance-dependent correction. The short-range (real space) potential between particles i and j for Ewald summation can be expressed as:

$$U_{ij}^{sr} = \frac{\Gamma q_i q_j}{4\pi r_{ij}} \left[\mathrm{erfc}(\alpha r_{ij}) - f(r_{ij})\right] \tag{9.83}$$

and the equivalent force and virial contribution between the particles is given by:

$$\vec{F}_{ij}^{E,sr} = \frac{\Gamma q_i q_j}{4\pi r_{ij}^2} \left(\frac{2\alpha r_{ij}}{\sqrt{\pi}} \exp(-\alpha^2 r_{ij}^2) + \mathrm{erfc}(\alpha r_{ij}) + r_{ij} \frac{df}{dr_{ij}} - f(r_{ij}) \right) \frac{\vec{r}_{ij}}{r_{ij}} \tag{9.84}$$

$$\mathcal{W}_{ij}^{E,sr} = -\frac{\Gamma q_i q_j}{4\pi r_{ij}} \left(\frac{2\alpha r_{ij}}{\sqrt{\pi}} \exp(-\alpha^2 r_{ij}^2) + \mathrm{erfc}(\alpha r_{ij}) + r_{ij} \frac{df}{dr_{ij}} - f(r_{ij}) \right). \tag{9.85}$$

Corrections to reciprocal-space terms for pairs of frozen beads can be made using equations (9.67), (9.68) and (9.69) for all smearing types.

9.5.3.1 Point charges: no charge smearing

If repulsions between particles are generally high enough to ensure sufficiently large excluded volumes (e.g. higher values of A_{ij} or using hard-core potentials), point charges can be safely used as ion collapse will not occur[127]. In this case, no charge smearing scheme will be used, i.e. $f(r_{ij}) = 0$, and thus the standard equations for the real space contributions for potential, force and virial – equations (9.57), (9.58) and (9.59) respectively – can be used unmodified.

If no **smear** directive is included in the `CONTROL` file, point charges will be assumed, but the user can still optionally include **smear none** in this file. No smearing length needs to be included: any specified value will be ignored for this charge smearing scheme.

9.5.3.2 Linear charge smearing

Linear charge smearing is based on using the following charge distribution:

$$\rho(r) = \begin{cases} \frac{3q}{\pi R^3}\left(1 - \frac{r}{R}\right) & (r < R) \\ 0 & (r \geq R) \end{cases} \tag{9.86}$$

where R is a smearing distance. While this charge distribution was devised for solution with a Particle-Particle Particle-Mesh method[42], a pairwise potential can be devised for solution with methods based on Ewald summation. A good mathematical description of the pairwise potential is as follows:

$$U(r_{ij}) = \begin{cases} \frac{\Gamma q_i q_j}{4\pi r_{ij}}\left[\frac{52}{35}\frac{r_{ij}}{R} - \frac{4}{5}\left(\frac{r_{ij}}{R}\right)^3 + \frac{2}{5}\left(\frac{r_{ij}}{R}\right)^5 - \frac{2120}{15603}\left(\frac{r_{ij}}{R}\right)^{6.145}\right] & (r_{ij} < R) \\ \frac{\Gamma q_i q_j}{4\pi r_{ij}}\left[1 - \frac{36813504}{11468205}\frac{r_{ij}}{R}\left(1 - \frac{r_{ij}}{2R}\right)^6\right] & (R \leq r_{ij} < 2R) \\ \frac{\Gamma q_i q_j}{4\pi r_{ij}} & (r_{ij} \geq 2R). \end{cases} \tag{9.87}$$

and the resulting pairwise force between particles i and j is

$$\vec{F}_{ij}^{E}(r_{ij}) = \begin{cases} \frac{\Gamma q_i q_j}{4\pi r_{ij}^2}\left[\frac{8}{5}\left(\frac{r_{ij}}{R}\right)^3 - \frac{8}{5}\left(\frac{r_{ij}}{R}\right)^5 + \frac{2597}{3715}\left(\frac{r_{ij}}{R}\right)^{6.145}\right]\frac{\vec{r}_{ij}}{r_{ij}} & (r_{ij} < R) \\ \frac{\Gamma q_i q_j}{4\pi r_{ij}^2}\left[1 - \frac{35776}{3715}\left(\frac{r_{ij}}{R}\right)^2\left(1 - \frac{r_{ij}}{2R}\right)^5\right]\frac{\vec{r}_{ij}}{r_{ij}} & (R \leq r_{ij} < 2R) \\ \frac{\Gamma q_i q_j}{4\pi r_{ij}^2}\frac{\vec{r}_{ij}}{r_{ij}} & (r_{ij} \geq 2R) \end{cases} \tag{9.88}$$

The associated correction function for comparison with the Coulombic potential is given as:

$$f(r_{ij}) = \begin{cases} 1 - \frac{52}{35}\frac{r_{ij}}{R} + \frac{4}{5}\left(\frac{r_{ij}}{R}\right)^3 - \frac{2}{5}\left(\frac{r_{ij}}{R}\right)^5 + \frac{2120}{15603}\left(\frac{r_{ij}}{R}\right)^{6.145} & (r_{ij} < R) \\ \frac{36813504}{11468205}\frac{r_{ij}}{R}\left(1 - \frac{r_{ij}}{2R}\right)^6 & (R \leq r_{ij} < 2R) \\ 0 & (R \geq 2R). \end{cases}$$

and leads to the following equations for the real space potential for Ewald sums between pairs of charged particles:

$$U_{ij}^{sr} = \begin{cases} \frac{\Gamma q_i q_j}{4\pi r_{ij}}\left[\frac{52}{35}\frac{r_{ij}}{R} - \frac{4}{5}\left(\frac{r_{ij}}{R}\right)^3 + \frac{2}{5}\left(\frac{r_{ij}}{R}\right)^5 - \frac{2120}{15603}\left(\frac{r_{ij}}{R}\right)^{6.145} - \mathrm{erf}(\alpha r_{ij})\right] & (r_{ij} < R) \\ \frac{\Gamma q_i q_j}{4\pi r_{ij}}\left[\mathrm{erfc}(\alpha r_{ij}) - \frac{36813504}{11468205}\frac{r_{ij}}{R}\left(1 - \frac{r_{ij}}{2R}\right)^6\right] & (R \leq r_{ij} < 2R) \\ \frac{\Gamma q_i q_j}{4\pi r_{ij}}\mathrm{erfc}(\alpha r_{ij}) & (R \geq 2R) \end{cases} \tag{9.89}$$

the pairwise force:

$$\vec{F}_{ij}^{E,sr} = \begin{cases} \frac{\Gamma q_i q_j}{4\pi r_{ij}^2}\left[\frac{2\alpha r_{ij}}{\sqrt{\pi}}\exp(-\alpha^2 r_{ij}^2) - \mathrm{erf}(\alpha r_{ij}) + \frac{8}{5}\left(\frac{r_{ij}}{R}\right)^3 - \frac{8}{5}\left(\frac{r_{ij}}{R}\right)^5 + \frac{2597}{3715}\left(\frac{r_{ij}}{R}\right)^{6.145}\right]\frac{\vec{r}_{ij}}{r_{ij}} & (r_{ij} < R) \\ \frac{\Gamma q_i q_j}{4\pi r_{ij}^2}\left[\frac{2\alpha r_{ij}}{\sqrt{\pi}}\exp(-\alpha^2 r_{ij}^2) + \mathrm{erfc}(\alpha r_{ij}) - \frac{35776}{3715}\left(\frac{r_{ij}}{R}\right)^2\left(1 - \frac{r_{ij}}{2R}\right)^5\right]\frac{\vec{r}_{ij}}{r_{ij}} & (R \leq r_{ij} < 2R) \\ \frac{\Gamma q_i q_j}{4\pi r_{ij}^2}\left[\frac{2\alpha r_{ij}}{\sqrt{\pi}}\exp(-\alpha^2 r_{ij}^2) + \mathrm{erfc}(\alpha r_{ij})\right]\frac{\vec{r}_{ij}}{r_{ij}} & (r_{ij} \geq 2R) \end{cases} \tag{9.90}$$

and contributions to the virial:

$$\mathcal{W}_{ij}^{E,sr} = \begin{cases} -\frac{\Gamma q_i q_j}{4\pi r_{ij}}\left[\frac{2\alpha r_{ij}}{\sqrt{\pi}}\exp(-\alpha^2 r_{ij}^2) - \mathrm{erf}(\alpha r_{ij}) + \frac{8}{5}\left(\frac{r_{ij}}{R}\right)^3 - \frac{8}{5}\left(\frac{r_{ij}}{R}\right)^5 + \frac{2597}{3715}\left(\frac{r_{ij}}{R}\right)^{6.145}\right] & (r_{ij} < R) \\ -\frac{\Gamma q_i q_j}{4\pi r_{ij}}\left[\frac{2\alpha r_{ij}}{\sqrt{\pi}}\exp(-\alpha^2 r_{ij}^2) + \mathrm{erfc}(\alpha r_{ij}) - \frac{35776}{3715}\left(\frac{r_{ij}}{R}\right)^2\left(1 - \frac{r_{ij}}{2R}\right)^5\right] & (R \leq r_{ij} < 2R) \\ -\frac{\Gamma q_i q_j}{4\pi r_{ij}}\left[\frac{2\alpha r_{ij}}{\sqrt{\pi}}\exp(-\alpha^2 r_{ij}^2) + \mathrm{erfc}(\alpha r_{ij})\right] & (r_{ij} \geq 2R) \end{cases} \tag{9.91}$$

These forms of charge smearing can be invoked using the directive **smear** in the `CONTROL` file with the keyword **lin**ear. The smearing length R can be given using the directive **smear len**gth: the electrostatic real space cutoff distance for the Ewald sum r_e should have a value of at least $2D$ to ensure the modified pairwise potential at shorter distances is applied correctly.

9.5.3.3 Slater-type (exponential) charge smearing

Slater-type charge smearing is based on using a decaying exponential function for the correction function $f(r_{ij})$ in the electrostatic potential. The Slater charge distribution is given as:

$$\rho(r) = \frac{q}{\pi\lambda^3}\exp\left(-\frac{2r}{\lambda}\right) \tag{9.92}$$

where λ is the decay length of the charge. This gives a potential energy between charged particles i and j[135] of

$$U(r_{ij}) = \frac{\Gamma q_i q_j}{4\pi r_{ij}}\left[1-\exp\left(-2\beta r_{ij}\right)\left(1+\tfrac{11}{8}\beta r_{ij}+\tfrac{3}{4}\beta^2 r_{ij}^2+\tfrac{1}{6}\beta^3 r_{ij}^3\right)\right] \tag{9.93}$$

and the corresponding electrostatic force is

$$\vec{F}_{ij}^{E}(r_{ij}) = \frac{\Gamma q_i q_j}{4\pi r_{ij}^2}\left[1-\exp\left(-2\beta r_{ij}\right)\left(1+2\beta r_{ij}+2\beta^2 r_{ij}^2+\tfrac{7}{6}\beta^3 r_{ij}^3+\tfrac{1}{3}\beta^4 r_{ij}^4\right)\right]\frac{\vec{r}_{ij}}{r_{ij}}, \tag{9.94}$$

where $\beta = \frac{1}{\lambda}$ in both equations.

The related correction function to the standard Coulombic potential in this case is

$$f(r_{ij}) = \exp\left(-2\beta r_{ij}\right)\left(1+\tfrac{11}{8}\beta r_{ij}+\tfrac{3}{4}\beta^2 r_{ij}^2+\tfrac{1}{6}\beta^3 r_{ij}^3\right)$$

and leads to the following equations for the real space potential for Ewald sums between pairs of charged particles:

$$U_{ij}^{sr} = \frac{\Gamma q_i q_j}{4\pi r_{ij}}\left[\mathrm{erfc}(\alpha r_{ij})-\exp\left(-2\beta r_{ij}\right)\left(1+\tfrac{11}{8}\beta r_{ij}+\tfrac{3}{4}\beta^2 r_{ij}^2+\tfrac{1}{6}\beta^3 r_{ij}^3\right)\right] \tag{9.95}$$

the corresponding pairwise force:

$$\vec{F}_{ij}^{E,sr} = \frac{\Gamma q_i q_j}{4\pi r_{ij}^2}\left[\frac{2\alpha r_{ij}}{\sqrt{\pi}}\exp(-\alpha^2 r_{ij}^2)+\mathrm{erfc}(\alpha r_{ij})-\exp\left(-2\beta r_{ij}\right)\left(1+2\beta r_{ij}+2\beta^2 r_{ij}^2+\tfrac{7}{6}\beta^3 r_{ij}^3+\tfrac{1}{3}\beta^4 r_{ij}^4\right)\right]\frac{\vec{r}_{ij}}{r_{ij}} \tag{9.96}$$

and the real space contribution to the virial:

$$\mathcal{W}_{ij}^{E,sr} = -\frac{\Gamma q_i q_j}{4\pi r_{ij}}\left[\frac{2\alpha r_{ij}}{\sqrt{\pi}}\exp(-\alpha^2 r_{ij}^2)+\mathrm{erfc}(\alpha r_{ij})-\exp\left(-2\beta r_{ij}\right)\left(1+2\beta r_{ij}+2\beta^2 r_{ij}^2+\tfrac{7}{6}\beta^3 r_{ij}^3+\tfrac{1}{3}\beta^4 r_{ij}^4\right)\right]. \tag{9.97}$$

An approximation to the above Slater potential can be obtained by truncating the polynomial multiplied to the exponential. The following frequently-used form has been proposed[41]:

$$U(r_{ij}) = \frac{\Gamma q_i q_j}{4\pi r_{ij}}\left[1-(1+\beta r_{ij})\exp(-2\beta r_{ij})\right] \tag{9.98}$$

where β is related to the charge decay length λ in one of three possible ways:

1. $\beta = \frac{1}{\lambda}$: direct correspondence to the exact Slater potential.
2. $\beta = \frac{5}{8\lambda}$: correspondence to the overlap potential (when $r_{ij}=0$)[5].
3. $\beta = \frac{1}{\sqrt{2}\lambda}$: correspondence to the second moment of charge distribution ($\sigma = \frac{1}{3}\int_0^\infty 4\pi r^4\rho(r)dr$), which allows comparisons with other charge smearing models.

The resulting electrostatic force is given as

$$\vec{F}_{ij}^{E}(r_{ij}) = \frac{\Gamma q_i q_j}{4\pi r_{ij}^2}\left[1-\exp(-2\beta r_{ij})\left(1+2\beta r_{ij}(1+\beta r_{ij})\right)\right]\frac{\vec{r}_{ij}}{r_{ij}}, \tag{9.99}$$

which is also a truncated form of the force for the exact Slater potential. The charge density in this case is slightly different to the exact Slater model:

$$\rho(r) = \frac{q\beta^2}{\pi r}\exp\left(-2\beta r\right). \tag{9.100}$$

[5]Mixing of charge decay lengths also corresponds with this model, e.g. [15].

The related correction function to the standard Coulombic potential is

$$f(r_{ij}) = \exp\left(-2\beta r_{ij}\right)\left(1+\beta r_{ij}\right)$$

with the short-range potential energy between particles i and j given as

$$U_{ij}^{sr} = \frac{\Gamma q_i q_j}{4\pi r_{ij}}\left[\mathrm{erfc}(\alpha r_{ij}) - \exp(-2\beta r_{ij})(1+\beta r_{ij})\right] \tag{9.101}$$

and the pairwise force is

$$\vec{F}_{ij}^{E,sr} = \frac{\Gamma q_i q_j}{4\pi r_{ij}^2}\left[\frac{2\alpha r_{ij}}{\sqrt{\pi}}\exp(-\alpha^2 r_{ij}^2) + \mathrm{erfc}(\alpha r_{ij}) - \exp(-2\beta r_{ij})\left(1+2\beta r_{ij}(1+\beta r_{ij})\right)\right]\frac{\vec{r}_{ij}}{r_{ij}}. \tag{9.102}$$

The associated virial contribution is given as

$$\mathcal{W}_{ij}^{E,sr} = -\frac{\Gamma q_i q_j}{4\pi r_{ij}}\left[\frac{2\alpha r_{ij}}{\sqrt{\pi}}\exp(-\alpha^2 r_{ij}^2) + \mathrm{erfc}(\alpha r_{ij}) - \exp(-2\beta r_{ij})\left(1+2\beta r_{ij}(1+\beta r_{ij})\right)\right]. \tag{9.103}$$

These forms of charge smearing can be invoked using the directive **smear** in the `CONTROL` file with the keyword **slater** and followed by either **exact** or **approx** for the exact and approximate (truncated) forms of the Slater-type potential: if this additional word is omitted, the exact form is assumed. The smearing length λ for both the exact and approximate models can be given using the directive **smear len**gth, with an additional keyword that can optionally be placed after the value of λ:

- **orig**inal (or no word): use original relationship between λ and β, i.e. $\beta = \frac{1}{\lambda}$.
- **over**lap: use $\beta = \frac{5}{8\lambda}$ to match overlap potentials.
- **dist**ribution: use $\beta = \frac{1}{\sqrt{2}\lambda}$ to match charge distributions.

Alternatively, the value of β can be specified directly using the directive **smear beta**: the same words after the value can be used to identify the relationship between β and λ.

9.5.3.4 Gaussian charge smearing

A Gaussian charge distribution is described as the following:

$$\rho(r) = q\left(\frac{1}{2\pi\sigma_G}\right)^{\frac{3}{2}}\exp\left(-\frac{r^2}{2\sigma_G^2}\right) \tag{9.104}$$

where σ_G is the length scale of the charge. This gives a potential energy between charged particles i and j of

$$U(r_{ij}) = \frac{\Gamma q_i q_j}{4\pi r_{ij}}\mathrm{erf}\left(\frac{r_{ij}}{2\sigma_G}\right) \tag{9.105}$$

and the corresponding electrostatic force is

$$\vec{F}_{ij}^{E}(r_{ij}) = \frac{\Gamma q_i q_j}{4\pi r_{ij}^2}\left[\mathrm{erf}\left(\frac{r_{ij}}{2\sigma_G}\right) - \frac{r_{ij}}{\sigma_G\sqrt{\pi}}\exp\left(-\frac{r_{ij}^2}{4\sigma_G^2}\right)\right]\frac{\vec{r}_{ij}}{r_{ij}}. \tag{9.106}$$

The correction function for the Coulombic potential is given as

$$f(r_{ij}) = 1 - \mathrm{erf}\left(\frac{r_{ij}}{2\sigma_G}\right) = \mathrm{erfc}\left(\frac{r_{ij}}{2\sigma_G}\right)$$

resulting in the following real space potential energy between particles i and j

$$U_{ij}^{sr} = \frac{\Gamma q_i q_j}{4\pi r_{ij}}\left[\mathrm{erfc}(\alpha r_{ij}) - \mathrm{erfc}\left(\frac{r_{ij}}{2\sigma_G}\right)\right], \tag{9.107}$$

the corresponding pairwise force

$$\vec{F}_{ij}^{E,sr} = \frac{\Gamma q_i q_j}{4\pi r_{ij}^2}\left[\mathrm{erfc}(\alpha r_{ij}) - \mathrm{erfc}\left(\frac{r_{ij}}{2\sigma_G}\right) + \frac{2\alpha r_{ij}}{\sqrt{\pi}}\exp(-\alpha^2 r_{ij}^2) - \frac{r_{ij}}{\sigma_G\sqrt{\pi}}\exp\left(-\frac{r_{ij}^2}{4\sigma_G^2}\right)\right]\frac{\vec{r}_{ij}}{r_{ij}}, \tag{9.108}$$

and an associated virial contribution as

$$\mathcal{W}_{ij}^{E,sr} = -\frac{\Gamma q_i q_j}{4\pi r_{ij}}\left[\mathrm{erfc}(\alpha r_{ij}) - \mathrm{erfc}\left(\frac{r_{ij}}{2\sigma_G}\right) + \frac{2\alpha r_{ij}}{\sqrt{\pi}}\exp(-\alpha^2 r_{ij}^2) - \frac{r_{ij}}{\sigma_G\sqrt{\pi}}\exp\left(-\frac{r_{ij}^2}{4\sigma_G^2}\right)\right]. \tag{9.109}$$

It should be noted that when $\sigma_G = \frac{1}{2\alpha}$, all real space terms (potential, force, virial) reduce to zero and therefore do not need to be evaluated: in this situation, all of the electrostatic interactions can be dealt with solely in reciprocal space, which can reduce the required computation time.

This form of charge smearing can be invoked using the directive **smear** in the `CONTROL` file with the keyword **gauss**. The smearing length scale σ_G can be specified using the directive **smear length**: if the word **equal** follows the value of σ_G, the Ewald sum real-space convergence factor α will be set equal to $\frac{1}{2\sigma_G}$ to eliminate real space contributions and the value given in the **ewald** or **spme** directive will be ignored.

9.5.3.5 Sinusoidal charge smearing

For high charge density systems with small Debye screening lengths, the following pairwise force scheme is proposed[34]:

$$\vec{F}_{ij}^{E}(r_{ij}) = \begin{cases} \frac{\Gamma q_i q_j}{4\pi r_{ij}^2}\sin^6\left(\frac{2\pi r_{ij}}{4D}\right)\frac{\vec{r}_{ij}}{r_{ij}} & (r_{ij} < D) \\ \frac{\Gamma q_i q_j}{4\pi r_{ij}^2}\frac{\vec{r}_{ij}}{r_{ij}} & (r_{ij} \geq D) \end{cases} \tag{9.110}$$

where the Coulombic force is recovered exactly beyond the charge smearing length D. Assuming the potential for $r_{ij} \geq D$ corresponds to the standard Coulombic potential, this implies the following potential for the case when $r_{ij} < D$:

$$U(r_{ij}) = -\frac{\Gamma q_i q_j}{4\pi}\int_D^{r_{ij}} \frac{\sin^6\left(\frac{2\pi r}{4D}\right)}{r^2}dr \tag{9.111}$$

$$= \frac{\Gamma q_i q_j}{4\pi r_{ij}}\sin^6\left(\frac{2\pi r_{ij}}{4D}\right) + \frac{\Gamma q_i q_j}{4\pi D}\sum_{k=2}^{\infty} C_k\left(1 - \left(\frac{r_{ij}}{D}\right)^{2k+1}\right) \tag{9.112}$$

where the coefficients for the infinite series required to solve the integral are given by

$$C_k = \frac{(-1)^k \pi^{2(k+1)}}{32(2k+1)!(2k+1)}\left(15 - 6(2)^{2(k+1)} + 3^{2(k+1)}\right).$$

The correction term compared to the Coulombic potential is given as

$$f(r_{ij}) = \begin{cases} 1 - \sin^6\left(\frac{2\pi r_{ij}}{4D}\right) - \frac{r_{ij}}{D}\sum_{k=2}^{\infty} C_k\left(1 - \left(\frac{r_{ij}}{D}\right)^{2k+1}\right) & (r_{ij} < D) \\ 0 & (r_{ij} \geq D) \end{cases}$$

which leads to the following expression for the Ewald real space potential contribution between particles i and j:

$$U_{ij}^{sr} = \begin{cases} \frac{\Gamma q_i q_j}{4\pi r_{ij}}\left[\sin^6\left(\frac{2\pi r_{ij}}{4D}\right) + \frac{r_{ij}}{D}\sum_{k=2}^{\infty} C_k\left(1 - \left(\frac{r_{ij}}{D}\right)^{2k+1}\right) - \mathrm{erf}(\alpha r_{ij})\right] & (r_{ij} < D) \\ \frac{\Gamma q_i q_j}{4\pi r_{ij}}\mathrm{erfc}(\alpha r_{ij}) & (r_{ij} \geq D) \end{cases} \tag{9.113}$$

the related force between the particles:

$$\vec{F}_{ij}^{E,sr} = \begin{cases} \frac{\Gamma q_i q_j}{4\pi r_{ij}^2}\left[\frac{2\alpha r_{ij}}{\sqrt{\pi}}\exp(-\alpha^2 r_{ij}^2) - \mathrm{erf}(\alpha r_{ij}) + \sin^6\left(\frac{2\pi r_{ij}}{4D}\right)\right]\frac{\vec{r}_{ij}}{r_{ij}} & (r_{ij} < D) \\ \frac{\Gamma q_i q_j}{4\pi r_{ij}^2}\left[\frac{2\alpha r_{ij}}{\sqrt{\pi}}\exp(-\alpha^2 r_{ij}^2) + \mathrm{erfc}(\alpha r_{ij})\right] & (r_{ij} \geq D) \end{cases} \tag{9.114}$$

and the associated virial contribution:

$$\mathcal{W}_{ij}^{E,sr} = \begin{cases} -\frac{\Gamma q_i q_j}{4\pi r_{ij}}\left[\frac{2\alpha r_{ij}}{\sqrt{\pi}}\exp(-\alpha^2 r_{ij}^2) + \sin^6\left(\frac{2\pi r_{ij}}{4D}\right) - \mathrm{erf}(\alpha r_{ij})\right] & (r_{ij} < D) \\ -\frac{\Gamma q_i q_j}{4\pi r_{ij}}\left[\frac{2\alpha r_{ij}}{\sqrt{\pi}}\exp(-\alpha^2 r_{ij}^2) + \mathrm{erfc}(\alpha r_{ij})\right] & (r_{ij} \geq D) \end{cases} \tag{9.115}$$

This form of charge smearing can be invoked using the directive **smear** in the `CONTROL` file with the keyword **sin**usoidal. The smearing length scale D can be specified using the directive **smear length**.

9.6 Bond interactions between particles

Molecules of particles bonded together can be included in calculations using a `FIELD` file to define the properties and topologies of the bonds, angles and dihedrals between them. These data are used to add the specified numbers of molecules into the system before DPD calculations commence and to create tables listing the particles that are included in bonds, angles and dihedrals.

9.6.1 Stretching bonds

DL_MESO_DPD can model four forms of bond potential (and corresponding force) between specified particles, all of which are functions of the distance between them. The available bond forms between particles i and j are as follows:

1. Harmonic (Hookean/Fraenkel) bond:

$$U\left(r_{ij}\right) = \frac{\kappa}{2}\left(r_{ij} - r_0\right)^2 \tag{9.116}$$

2. (Shifted) Finitely Extendible Non-linear Elastic (FENE) bond:

$$U\left(r_{ij}\right) = \begin{cases} -\frac{1}{2}\kappa r_{max}^2 \ln\left[1 - \frac{\left(r_{ij}-r_0\right)^2}{r_{max}^2}\right] & r_{ij} < r_0 + r_{max} \\ \infty & r_{ij} \geq r_0 + r_{max} \end{cases} \tag{9.117}$$

3. Marko-Siggia Worm-Like Chain (WLC)[82]:

$$U\left(r_{ij}\right) = \begin{cases} \frac{k_B T}{2A_p}\left[\frac{1}{2\left(1-\frac{r_{ij}}{r_{max}}\right)} - \frac{1}{2}\left(1 + \frac{r_{ij}}{r_{max}}\right) + \frac{r_{ij}^2}{r_{max}^2}\right] & r_{ij} < r_{max} \\ \infty & r_{ij} \geq r_{max} \end{cases} \tag{9.118}$$

4. Morse potential bond[86]:

$$U\left(r_{ij}\right) = D_e\left[1 - \exp\left(-\beta\left(r_{ij} - r_0\right)\right)\right]^2 \tag{9.119}$$

where $\vec{r}_{ij} = \vec{r}_i - \vec{r}_j$, r_0 is an equilibrium bond length, r_{max} the maximum specified bond length or extension, κ is a spring force constant, A_p the persistence length of a wormlike chain, D_e the potential well depth and β the potential 'width'. The spring force constants or potential well depths can be scaled with temperature if this option is selected in the `FIELD` file. (The effective spring force constant for worm-like chains already includes a temperature dependence, so persistence lengths do not need to be scaled with temperature.)

The force on particle i due to a bond potential is obtained from the general formula:

$$\vec{F}_i = -\frac{1}{r_{ij}}\left[\frac{\partial}{\partial r_{ij}} U\left(r_{ij}\right)\right]\vec{r}_{ij} \tag{9.120}$$

with the force acting on particle j equal to the negative of this, and the virial contribution from the stretching bond given by

$$\mathcal{W} = -\vec{r}_{ij} \cdot \vec{F}_i, \tag{9.121}$$

with only *one* contribution per bond[6].

[6]This expression is also used for the virial contribution from the standard DPD pairwise forces, i.e. Equation (8.3), again only applying a single contribution per particle pair.

9.6.2 Bond angles

DL_MESO_DPD includes three methods for modelling potentials and forces between three bonded particles due to the angle formed between them, θ_{ijk}. The potentials are given as follows:

1. Harmonic:
$$U\left(\theta_{ijk}\right) = \frac{\kappa}{2}\left(\theta_{ijk} - \theta_0\right)^2 \tag{9.122}$$

2. Harmonic cosine:
$$U\left(\theta_{ijk}\right) = \frac{\kappa}{2}\left(\cos\theta_{ijk} - \cos\theta_0\right)^2 \tag{9.123}$$

3. Cosine:
$$U\left(\theta_{ijk}\right) = A\left[1 + \cos\left(m\theta_{ijk} - \delta\right)\right] \tag{9.124}$$

where A and κ are angle force constants, m is the multiplicity, δ the angle at minimum potential and θ_0 an equilibrium bond angle. The angle force constants can be scaled with temperature if this option is selected in the `FIELD` file.

The angle across particles i, j and k can be determined from the bond vectors $\vec{r}_{ij} = \vec{r}_i - \vec{r}_j$ and $\vec{r}_{kj} = \vec{r}_k - \vec{r}_j$:

$$\theta_{ijk} = \cos^{-1}\left\{\frac{\vec{r}_{ij}\cdot\vec{r}_{kj}}{r_{ij}r_{kj}}\right\} \tag{9.125}$$

The most general form for the bond angle potential is given thus:

$$U\left(\theta_{ijk}, r_{ij}, r_{kj}\right) = A\left(\theta_{ijk}\right) S\left(r_{ij}\right) S\left(r_{kj}\right) S\left(r_{ik}\right) \tag{9.126}$$

with $A\left(\theta\right)$ as a purely angular function and $S\left(r\right)$ a screening or truncation function. The force on particle ℓ in dimension α is thus given by:

$$\begin{aligned} F_\ell^\alpha &= -\frac{\partial}{\partial x_\alpha} U\left(\theta_{ijk}, r_{ij}, r_{kj}\right) && (9.127)\\ &= -S\left(r_{ij}\right) S\left(r_{kj}\right) S\left(r_{ik}\right) \frac{\partial}{\partial r_\ell^\alpha} A\left(\theta_{ijk}\right) \\ &\quad -A\left(\theta_{ijk}\right) S\left(r_{kj}\right) S\left(r_{ik}\right) \left(\delta_{\ell i} - \delta_{\ell j}\right) \frac{r_{ij}^\alpha}{r_{ij}} \frac{\partial}{\partial r_{ij}} S\left(r_{ij}\right) \\ &\quad -A\left(\theta_{ijk}\right) S\left(r_{ij}\right) S\left(r_{ik}\right) \left(\delta_{\ell k} - \delta_{\ell j}\right) \frac{r_{kj}^\alpha}{r_{kj}} \frac{\partial}{\partial r_{kj}} S\left(r_{kj}\right) \\ &\quad -A\left(\theta_{ijk}\right) S\left(r_{ij}\right) S\left(r_{kj}\right) \left(\delta_{\ell k} - \delta_{\ell i}\right) \frac{r_{ik}^\alpha}{r_{ik}} \frac{\partial}{\partial r_{ik}} S\left(r_{ik}\right) && (9.128) \end{aligned}$$

with $\delta_{ab} = 1$ if $a = b$ and $\delta_{ab} = 0$ if $a \neq b$. In the absence of screening terms, the above formula reduces to

$$\begin{aligned} F_\ell^\alpha &= -\frac{\partial}{\partial r_\ell^\alpha} A\left(\theta_{ijk}\right) && (9.129)\\ &= \frac{1}{\sin\theta_{ijk}} \frac{\partial}{\partial\theta_{ijk}} A\left(\theta_{ijk}\right) \times \\ &\quad \left\{\left(\delta_{\ell j} - \delta_{\ell i}\right)\frac{r_{kj}^\alpha}{r_{ij}r_{kj}} + \left(\delta_{\ell j} - \delta_{\ell k}\right)\frac{r_{ij}^\alpha}{r_{ij}r_{kj}} - \cos\left(\theta_{ijk}\right)\left[\left(\delta_{\ell j} - \delta_{\ell i}\right)\frac{r_{ij}^\alpha}{r_{ij}^2} + \left(\delta_{\ell j} - \delta_{\ell k}\right)\frac{r_{kj}^\alpha}{r_{kj}^2}\right]\right\} && (9.130) \end{aligned}$$

The contribution to the virial from the angle is given by

$$\mathcal{W} = -\left(\vec{r}_{ij}\cdot\vec{F}_i + \vec{r}_{kj}\cdot\vec{F}_k\right) \tag{9.131}$$

which is equal to zero for bond angle potentials without screening terms[116].

9.6.3 Bond dihedrals

Three potential models for bond dihedrals along particles i, j, k and l are provided in DL_MESO_DPD as follows:

1. Cosine torsion:
$$U\left(\phi_{ijkl}\right) = A\left[1 + \cos\left(m\phi_{ijkl} - \delta\right)\right] \tag{9.132}$$

2. Harmonic:
$$U\left(\phi_{ijkl}\right) = \frac{\kappa}{2}\left(\phi_{ijkl} - \phi_0\right)^2 \tag{9.133}$$

3. Harmonic cosine:
$$U\left(\phi_{ijkl}\right) = \frac{\kappa}{2}\left(\cos\phi_{ijkl} - \cos\phi_0\right)^2 \tag{9.134}$$

where A and κ are dihedral force constants, m is the multiplicity, δ the dihedral at minimum potential and ϕ_0 an equilibrium bond dihedral. The dihedral force constants can be scaled with temperature if this option is selected in the `FIELD` file.

The dihedral angle across all four particles (or between planes ij and kl) is given by

$$\phi_{ijkl} = \cos^{-1} B\left(\vec{r}_{ij}, \vec{r}_{jk}, \vec{r}_{kl}\right) \tag{9.135}$$

where

$$B\left(\vec{r}_{ij}, \vec{r}_{jk}, \vec{r}_{kl}\right) = \frac{\left(\vec{r}_{ij} \times \vec{r}_{jk}\right) \cdot \left(\vec{r}_{jk} \times \vec{r}_{kl}\right)}{|\vec{r}_{ij} \times \vec{r}_{jk}||\vec{r}_{jk} \times \vec{r}_{kl}|} \tag{9.136}$$

which gives a negative value for ϕ_{ijkl} if the vector $\left(\vec{r}_{ij} \times \vec{r}_{jk}\right) \cdot \left(\vec{r}_{jk} \times \vec{r}_{kl}\right)$ is in the same direction as the bond vector $\vec{r}_{jk}$ and positive if in the opposite direction.

The force on particle ℓ acting in the α direction is given by

$$\begin{aligned} F_\ell^\alpha &= -\frac{\partial}{\partial x_\alpha} U\left(\phi_{ijkl}\right) && (9.137)\\ &= \frac{1}{\sin\phi_{ijkl}} \frac{\partial}{\partial \phi_{ijkl}} U\left(\phi_{ijkl}\right) \frac{\partial}{\partial r_\ell^\alpha} B\left(\vec{r}_{ij}, \vec{r}_{jk}, \vec{r}_{kl}\right). && (9.138) \end{aligned}$$

Using the following definition

$$\left[\vec{a}\ \vec{b}\right]_\alpha \equiv \sum_\beta \left(1 - \delta_{\alpha\beta}\right) a^\beta b^\beta$$

the derivative of $B\left(\vec{r}_{ij}, \vec{r}_{jk}, \vec{r}_{kl}\right)$ is given by

$$\begin{aligned} \frac{\partial}{\partial r_\ell^\alpha} B\left(\vec{r}_{ij}, \vec{r}_{jk}, \vec{r}_{kl}\right) &= \frac{1}{|\vec{r}_{ij} \times \vec{r}_{jk}||\vec{r}_{jk} \times \vec{r}_{kl}|} \frac{\partial}{\partial r_\ell^\alpha} \left\{\left(\vec{r}_{ij} \times \vec{r}_{jk}\right) \cdot \left(\vec{r}_{jk} \times \vec{r}_{kl}\right)\right\} \\ -\frac{\cos\phi_{ijkl}}{2} &\left\{ \frac{1}{|\vec{r}_{ij} \times \vec{r}_{jk}|^2} \frac{\partial}{\partial r_\ell^\alpha} |\vec{r}_{ij} \times \vec{r}_{jk}|^2 + \frac{1}{|\vec{r}_{jk} \times \vec{r}_{kl}|^2} \frac{\partial}{\partial r_\ell^\alpha} |\vec{r}_{jk} \times \vec{r}_{kl}|^2 \right\} \end{aligned} \tag{9.139}$$

with

$$\begin{aligned} \frac{\partial}{\partial r_\ell^\alpha} \left\{\left(\vec{r}_{ij} \times \vec{r}_{jk}\right) \cdot \left(\vec{r}_{jk} \times \vec{r}_{kl}\right)\right\} &= r_{ij}^\alpha \left(\left[\vec{r}_{jk}\vec{r}_{jk}\right]_\alpha \left(\delta_{\ell k} - \delta_{\ell l}\right) + \left[\vec{r}_{jk}\vec{r}_{kl}\right]_\alpha \left(\delta_{\ell k} - \delta_{\ell j}\right)\right) + \\ & r_{jk}^\alpha \left(\left[\vec{r}_{ij}\vec{r}_{jk}\right]_\alpha \left(\delta_{\ell l} - \delta_{\ell k}\right) + \left[\vec{r}_{jk}\vec{r}_{kl}\right]_\alpha \left(\delta_{\ell j} - \delta_{\ell i}\right)\right) + \\ & r_{kl}^\alpha \left(\left[\vec{r}_{ij}\vec{r}_{jk}\right]_\alpha \left(\delta_{\ell k} - \delta_{\ell j}\right) + \left[\vec{r}_{jk}\vec{r}_{jk}\right]_\alpha \left(\delta_{\ell i} - \delta_{\ell j}\right)\right) + \\ & 2r_{jk}^\alpha \left[\vec{r}_{ij}\vec{r}_{kl}\right]_\alpha \left(\delta_{\ell l} - \delta_{\ell k}\right) \end{aligned} \tag{9.140}$$

$$\begin{aligned} \frac{\partial}{\partial r_\ell^\alpha} |\vec{r}_{ij} \times \vec{r}_{jk}|^2 &= 2r_{ij}^\alpha \left(\left[\vec{r}_{jk}\vec{r}_{jk}\right]_\alpha \left(\delta_{\ell j} - \delta_{\ell i}\right) + \left[\vec{r}_{ij}\vec{r}_{jk}\right]_\alpha \left(\delta_{\ell j} - \delta_{\ell k}\right)\right) + \\ & 2r_{jk}^\alpha \left(\left[\vec{r}_{ij}\vec{r}_{ij}\right]_\alpha \left(\delta_{\ell k} - \delta_{\ell j}\right) + \left[\vec{r}_{ij}\vec{r}_{jk}\right]_\alpha \left(\delta_{\ell i} - \delta_{\ell j}\right)\right) \end{aligned} \tag{9.141}$$

$$\frac{\partial}{\partial r_\ell^\alpha}|\vec{r}_{jk}\times\vec{r}_{kl}|^2 = 2r_{kl}^\alpha\left([\vec{r}_{jk}\vec{r}_{jk}]_\alpha(\delta_{\ell l}-\delta_{\ell k})+[\vec{r}_{jk}\vec{r}_{kl}]_\alpha(\delta_{\ell j}-\delta_{\ell k})\right)+ 2r_{jk}^\alpha\left([\vec{r}_{kl}\vec{r}_{kl}]_\alpha(\delta_{\ell k}-\delta_{\ell j})+[\vec{r}_{jk}\vec{r}_{kl}]_\alpha(\delta_{\ell k}-\delta_{\ell l})\right) \quad (9.142)$$

It can be shown both algebraically and thermodynamically that the dihedral makes no contribution to the virial[116].

Improper dihedrals — which limit the geometry of molecules — can be applied using the same procedure as standard dihedrals and no distinction is made between them in DL_MESO_DPD.

9.7 Frozen bead walls

One method of representing solid boundaries is to include layers of frozen beads that do not move during the simulation but still interact with all other particles. On their own, frozen bead walls can achieve no-slip boundaries if appropriate choices for the density of frozen beads in the walls and interactions between frozen and non-frozen particles are made. Non-porous boundaries with no slip conditions can be obtained, albeit at the cost of density fluctuations near the walls[102]. Frozen bead walls can also be supplemented by hard reflecting boundaries (see below) shifted by the wall thickness or a larger distance to ensure they are non-porous and reduce density fluctuations.

To use this boundary condition, a species of frozen beads needs to be specified in the `FIELD` file, along with the interaction types and parameters between this species and all others. The directive **froz**en (walls) is used in the `CONTROL` file to specify which boundary surfaces should include frozen beads, while the same directive in the `FIELD` file identifies the frozen bead species, the bead density and thickness of the wall regions: the number of beads required is automatically determined and the size of the system is adjusted to include the additional wall regions, i.e. the user does not have to include the walls in the system dimensions given in the `CONTROL` file.

This boundary condition can be set up in this manner for new simulations, either starting from scratch or using a `CONFIG` file. If creating a `CONFIG` file from a previous simulation, users are advised to use the hard reflecting boundary condition described above to ensure molecules remain within the required non-periodic boundaries. Simulations with frozen bead walls can be restarted but these walls must already be included in restart (`export`) files and cannot subsequently be added.

9.8 Surface interactions

The default boundaries for a simulation box are periodic, i.e. particles leaving the system are replaced at the opposite face with the same velocity. Certain systems may require alternative boundary conditions and DL_MESO_DPD can include these at the system boundaries. The directive **surf**ace in the `CONTROL` file can be used to specify the type of surface interaction (`shear` or `hard`) and which surface(s) are to be included: for hard surfaces, the type of particle reflection (specular or bounce-back) and displacement of the boundary from the edges of the simulation volume can also be specified.

Care should be taken to ensure that the initial configuration does not include bonds crossing any non-periodic solid boundary. DL_MESO_DPD will ensure this is the case for simulations starting from scratch but cannot check for bonds crossing solid boundaries in `CONFIG` files.

9.8.1 Shearing periodic walls (`shear`)

Shearing walls moving at a specified velocity are applied using the Lees-Edwards boundary condition[71]: each particle that moves through the otherwise periodic boundary has its velocity modified and is shifted by a distance related to the wall velocity, i.e.

$$\Delta\vec{x}_w = \vec{V}_w t \quad (9.143)$$

where $\vec{V}_w$ is the velocity of the moving boundary. All interactions between pairs of particles across the periodic boundary are calculated. To correctly apply pairwise thermostats and avoid excessive damping in regions near

the boundary, the tangential components for relative velocities between pairs of particles crossing the boundary are adjusted to remove the change due to the boundary scheme's velocity modification[72], i.e. $\vec{v}_{ij}^{adj} = \vec{v}_{ij} \pm 2\vec{V}_w$[7].

The directive **surf**ace is used in the `CONTROL` file to specify which boundaries should move and at which time step shearing should start, while the **extern**al directive in the `FIELD` file is used to specify the velocity of the moving wall at $x_i = L_i$ for dimension i (the wall at $x_i = 0$ is given the negative value).

9.8.2 Hard reflecting boundaries (`hard`)

This boundary condition is applied by using a combination of reflection (either specular or bounce-back, giving free-slip or no-slip boundaries respectively) at the system boundaries or at a user-specified displacement from those boundaries with a short-range wall potential. Specular reflections are achieved by moving any particle leaving the system at a particular boundary back into it and inverting the velocity component normal to the wall (but maintaining the tangential momentum). Bounce-back reflections reverse particle momentum from the point at which the particle would move through the boundary, completely reversing the particle velocity. In both cases, since the boundaries are assumed to be non-porous, no interactions across them are included and thus adjacent boundary halos are not used. Both types of reflection can be used in conjunction with frozen bead walls, which is generally applied at or near the wall surface.

A short-range wall potential can be included to reduce density oscillations[97]. This can either be a DPD-like soft short-range wall repulsion

$$U_{wall}(z) = \tfrac{1}{2} A_{wall,\alpha} z_{c,\alpha} \left(1 - \frac{z}{z_{c,\alpha}}\right)^2, \qquad (z < z_{c,\alpha}) \tag{9.144}$$

or a Weeks-Chandler-Andersen (WCA) repulsion

$$U_{wall}(z) = 4\epsilon_{wall,\alpha} \left[\left(\frac{\sigma_\alpha}{z}\right)^{12} - \left(\frac{\sigma_\alpha}{z}\right)^{6}\right] + \epsilon_{wall,\alpha}, \qquad \left(z < 2^{\frac{1}{6}}\sigma_\alpha\right). \tag{9.145}$$

In the above, z is the distance between the particle and the boundary, $A_{wall,\alpha}$ is the DPD repulsive force magnitude with species α, $z_{c,\alpha}$ the surface repulsion range for the given species, $\epsilon_{wall,\alpha}$ the potential well depth for species α with the boundary and σ_α the distance at which the potential is equal to $\epsilon_{wall,\alpha}$. The type of wall repulsion and its parameters can be specified in the `FIELD` file for each species using the directive **surf**ace: the same directive in the `CONTROL` file is used to determine the maximum surface repulsion cutoff, which boundary surfaces should be hard and reflecting, the type of reflection (specular or bounceback) to be used and the displacement from the system boundary.

[7] The adjustment in relative velocities supersedes a previously implemented scheme of switching off pairwise thermostats across shearing boundaries[19].

Chapter 10

DL_MESO_DPD Input and Output Files

10.1 Input files

All user-specified input files for DL_MESO_DPD must be in ANSI text format, with keywords (where necessary) and numerical values separated from each other with spaces, tabs or commas.

Define system: `CONTROL`

The `CONTROL` file contains the control variables for running a DPD simulation and is read by the subroutine `read_control` in `config_module`. Such files can can be obtained either via use of the DL_MESO GUI or by editing existing files of that name, such as those in the `DEMO/DPD` directory. These consist primarily of directives: character strings that appear as the first entry of a data record and invoke a particular operation or provide numerical parameters. Extra options may be added by the inclusion of keywords to qualify a particular directive. Directives can be included in any order except for the simulation name (up to 80 characters long) on the first line of the file and the **finish** directive which marks the end of the file.

A list of the directives available follows, with bold type specifying the minimum number of letters required by DL_MESO_DPD. Some directives may include optional words or parameters as indicated by brackets.

directive:	**meaning:**
bjerrum (length) f	sets the Bjerrum length $\lambda_B = \frac{\Gamma r_c}{4\pi}$ for the system to f (if specified in `FIELD` file, this property can be scaled with temperature)
boundary **halo** f	sets size of boundary halo (overriding default values determined from interaction cutoff, maximum bond lengths and short-range electrostatics) as f length units
close time f	sets job closure time to f seconds
config (**orig**in) **zero**	sets origin of `CONFIG` file as bottom left back corner instead of default of box centre: this option can be used for backwards compatibility with older versions of DL_MESO
cutoff f	sets maximum interaction cutoff radius, $r_{c,max}$, to f length units
densvar f	allows for local variation of $\approx f$ % in the system density of particles (useful for non-homogeneous or non-equilibrium simulations, default $f = 0$
electrostatic **cut**off f	sets required short-range electrostatic cutoff radius, r_e, to f length units (default $f = r_c$)
ensemble nvt mdvv	selects NVT (constant volume/temperature) ensemble, DPD thermostat with standard MD-like Velocity Verlet integration (default ensemble if otherwise not specified)
ensemble nvt dpdvv	selects NVT ensemble, DPD thermostat with DPD Velocity Verlet integration

ensemble nvt dpds1	selects NVT ensemble, DPD thermostat with first-order Shardlow splitting
ensemble nvt dpds2	selects NVT ensemble, DPD thermostat with second-order Shardlow splitting
ensemble nvt lowe	selects NVT ensemble, Lowe-Andersen thermostat
ensemble nvt peters	selects NVT ensemble, Peters thermostat
ensemble nvt stoyanov α	selects NVT ensemble, Stoyanov-Groot thermostat with coupling parameter α
ensemble npt Q **lang**evin f_1 f_2	selects NPT (constant pressure/temperature) ensemble, thermostat type Q (i.e. *mdvv*, *dpdvv*, *dpds1*, *dpds2*, *lowe*, *peters* or *stoyanov* α) with Langevin barostat, setting relaxation time (τ_p) and viscosity parameter (γ_p) as f_1 and f_2 respectively
ensemble npt Q **bere**ndsen f	selects NPT ensemble, thermostat type Q with Berendsen barostat, setting ratio of compressibility to relaxation time, $\frac{\beta}{\tau_p}$, to f
ensemble nst Q **lang**evin f_1 f_2	selects (orthogonally constrained) NσT (constant stress/temperature) ensemble, thermostat type Q with Langevin barostat, setting relaxation time (τ_p) and viscosity parameter (γ_p) as f_1 and f_2 respectively
ensemble nst Q **bere**ndsen f	selects (orthogonally constrained) NσT ensemble, thermostat type Q with Berendsen barostat, setting ratio of compressibility to relaxation time, $\frac{\beta}{\tau_p}$, to f
ensemble nst Q_1 Q_2 **area**	selects NP$_n$AT (constant normal pressure/surface area/temperature) ensemble, thermostat type Q_1 and barostat type Q_2 (i.e. *langevin* f_1 f_2 or *berendsen* f)
ensemble nst Q_1 Q_2 **tens**ion γ_0	selects NP$_n\gamma$T (constant normal pressure/surface tension/temperature) anisotropic ensemble, thermostat type Q_1 and barostat type Q_2 with target surface tension γ_0
ensemble nst Q_1 Q_2 **tens**ion γ_0 **semi**	selects NP$_n\gamma$T (constant normal pressure/surface tension/temperature) semi-isotropic ensemble, thermostat type Q_1 and barostat type Q_2 with target surface tension γ_0, varying the (x, y) plane isotropically
ensemble nst Q_1 Q_2 **orth**ogonal	selects NP$_n\gamma$T (constant normal pressure/surface tension/temperature) anisotropic ensemble, thermostat type Q_1 and barostat type Q_2 with target surface tension $\gamma_0 = 0$[1]
ensemble nst Q_1 Q_2 **orth semi**	selects NP$_n\gamma$T (constant normal pressure/surface tension/temperature) semi-isotropic ensemble, thermostat type Q_1 and barostat type Q_2 with target surface tension $\gamma_0 = 0$
equilibration (steps) n	equilibrates system for the first n timesteps (default $n = 0$)
ewald precision f	calculates electrostatic forces using Ewald sum, setting the real-space convergence parameter α and reciprocal space (k-vector) range (k_1, k_2, k_3) based on achieving the relative error f
ewald (sum) α k_1 k_2 k_3	calculates electrostatic forces using Ewald sum with real-space convergence parameter α and reciprocal space (k-vector) range (k_1, k_2, k_3)
finish	closes the `CONTROL` file (last data record)
frozen (wall) i	sets frozen bead walls orthogonal to i-axis (`x`, `y`, `z`) if specified (multiple walls can be specified if separated with spaces or commas). Note that this directive can only be used for new simulations: this directive is ignored if a simulation is restarted.
global bonds	calculates bonded interactions globally by storing bond data on all processors and sharing bonded particle positions (default: calculate bonded interactions locally)
job time f	sets maximum job time to f seconds

[1] This option is equivalent to the orthogonal NσT ensemble and is included for equivalence to DL_POLY.

l_init	creates `CONFIG` file (called `CFGINI`) of initial system configuration (with velocities and forces) for new simulations[2]
l_scr	redirects simulation output to the standard (default) output of the machine and operating system, e.g. the screen
manybody **cut**off f	sets required many-body DPD interaction radius, r_d, to f length units (default $f = r_c$)
ndump n	writes restart data to `export` files every n timesteps (default $n = 1000$)
nfold i j k	option to create volumetrically expanded version of current system (described by `CONFIG` and `FIELD` files) by replicating `CONFIG` file's contents (i, j, k) times while preserving topology of `FIELD` file[3]
no config	ignores contents of `CONFIG` file and create initial configuration based purely on `FIELD` file
no electrostatics	ignores electrostatics in simulation
no index	ignores particles' indices as read from `CONFIG` file and set their indexing according to order of reading
openmp (critical)	uses OpenMP critical regions to assign forces in multithreaded calculations (instead of using additional memory per thread)
permittivity (constant) f	sets permittivity constant for system, $\Gamma = \frac{4\pi\lambda_B}{r_c}$, to f (if specified in `FIELD` file, this property can be scaled with temperature)
pressure f	sets required system pressure to f (target pressure for constant pressure ensembles)
print (every) n	prints system data every n timesteps
print partial (temperatures)	prints partial temperatures (for each dimension) in both `OUTPUT` and `CORREL`. (This option is automatically switched on if a constant force or shear is applied to the system.)
rcut f	see **cut**off
restart	restarts job from end point of previous run (i.e. continue current simulation using `export` files)
restart noscale	restarts job from previous run without rescaling to system temperature (i.e. begin a new simulation from older run without temperature reset)
restart scale	restarts job from previous run after rescaling to system temperature (i.e. begin a new simulation from older run with temperature reset)
scale (temperature) (every) n	rescales system temperature every n steps during equilibration
seed n	modifies random number generator seeds used in DPD calculations
smear Q	applies charge smearing type to Q (**none**, **lin**ear, **slater** (**exact**), **slater approx**, **gauss** or **sin**usoidal)
smear beta f (Q)	sets Slater-type smearing parameter β: can be followed by Q (**orig**inal, **over**lap or **dist**ribution) to specify how β is related to the smearing length λ ($\lambda = \frac{1}{\beta}$, $\lambda = \frac{5}{8\beta}$ or $\lambda = \frac{1}{\sqrt{2}\beta}$ respectively)
smear length f	sets smearing length (R, λ, σ_G or D) to f
smear length f **equ**al	sets smearing length for Gaussian smearing (σ_G) to f and set Ewald real-space convergence parameter $\alpha = \frac{1}{2\sigma_G}$ to override any other specified value[4]
smear length f Q	sets smearing length for Slater-type smearing (λ) to f: can be followed by Q (**orig**inal, **over**lap or **dist**ribution) to specify how β is related to the smearing length λ ($\beta = \frac{1}{\lambda}$, $\beta = \frac{5}{8\lambda}$ or $\beta = \frac{1}{\sqrt{2}\lambda}$ respectively)

[2] If simulation is started using a `CONFIG` file with velocities and forces without volumetric expansion, this option is ignored.

[3] If restarting a system with an `export` file that was originally set up this way, this option can be used to avoid modifying the `FIELD` file.

[4] The real space terms for the Ewald sum are omitted when this option is selected.

spme precision f n	calculates electrostatic forces using Smooth Particle Mesh Ewald, setting the real-space convergence parameter α and reciprocal space (k-vector) range (k_1, k_2, k_3) based on achieving the relative error f, and maximum B-spline order n (default value of 8 if this value is omitted or set to less than 4)
spme (sum) α k_1 k_2 k_3 n	calculates electrostatic forces using Smooth Particle Mesh Ewald with real-space convergence parameter α, reciprocal space (k-vector) range (k_1, k_2, k_3)[5] and maximum B-spline order n (default value of 8 if this value is omitted or set to less than 4)
stack (size) n	sets rolling average stack to n timesteps
stats (every) n	accumulates statistics data and write to `CORREL` file every n timesteps
steps n	runs simulation for n timesteps
stress i j Q	accumulates separated stress tensors and writes to `Stress_*.d` file(s) with controls: i = start timestep for writing stress tensors, j = timestep interval between stress tensor data, Q = (up to four) separated stress tensor sets to write to files (**pot**ential, **dis**sipative, **ran**dom, **kin**etic, **all** four)
surface **cut**off f	sets required maximum surface repulsive range, z_c, to f length units (default $f = r_c$)
surface **hard** i **boun**ceback (f)	sets hard adsorbing walls orthogonal to i-axis (`x`, `y`, `z`) if specified (multiple walls can be specified if separated with spaces or commas) with bounce-back reflections at a distance f from simulation boundaries (optional: default $f = 0$)
surface **hard** i (**spec**ular) (f)	sets hard adsorbing walls orthogonal to i-axis (`x`, `y`, `z`) if specified (multiple walls can be specified if separated with spaces or commas) with specular reflections at a distance f from simulation boundaries (optional: default $f = 0$)
surface **shea**r i j	sets moving Lees-Edwards periodic walls orthogonal to i-axis (`x`, `y`, `z`) if specified (a single wall can be specified and only the first specified dimension will be used), starting shearing from time step j
temperature f	sets required simulation temperature (k_BT) to f (target temperature for constant temperature ensembles)
thermostat **cut**off f	sets thermostat cutoff radius, r_c, to f length units
trajectory i j k	writes trajectory data to `HISTORY` file with controls: i = start timestep for dumping configurations (default: equilibration time), j = timestep interval between configurations, k = data level to be included (default: 0; 0 = positions, 1 = positions and velocities, 2 = positions, velocities and forces)
timestep f	set timestep to f time units
vacuum (gap) f_1 f_2 f_3	sets size of vacuum gap (additional volume) for reciprocal space Ewald/SPME calculations of slab-like systems as orthorhombic dimensions (f_1, f_2, f_3)
volume f_1 (f_2 f_3)	sets system size to either cubic volume f_1 or orthorhombic dimensions (f_1, f_2, f_3)

While not every directive has to be included in the `CONTROL` file for a valid simulation and many hold default values if unspecified, the following are mandatory and must be set to values greater than zero:

- **temp**erature
- **timestep**
- **vol**ume (if no `CONFIG` file is available or the supplied `CONFIG` omits volume data)

[5] This vector should be double the size of that used for the standard Ewald sum, as the k-vector is applied cubically rather than spherically.

The maximum interaction **cut**off should also be set greater than zero, but if this value is omitted in the `CONTROL` file, this value will be set as the maximum interaction length specified in the `FIELD` file (σ_{ij} or $r_{c,ij}$). Superfluous parameters and switches for particular systems (e.g. specified pressure for constant volume simulations) can be safely omitted from the `CONTROL` file without causing runtime problems. If the user wishes to include new directives in the `CONTROL` file, modifications to the parameter recognition loop in the `read_config` subroutine (`config_module`) will be required.

Define interactions: `FIELD`

The `FIELD` file contains the species and force field information required for both bonded and unbonded interactions, and is read by the `scan_field` and `read_field` subroutines in `config_module`. Apart from the name of the simulation (up to 80 characters) in the first line, this file contains a number of directives, each indicating the type and number of interactions to follow.

If the values of all energy-based parameters for interactions (e.g. A_{ij}, κ) specified in the `FIELD` file need to be scaled, an optional directive **units** can be used to select the scaling. No scaling is ordinarily applied – which can be reinforced using the directive **units const**ant – while the directive **units kT** will scale the parameters with the required system temperature (k_BT).

The species information *must* be provided before any interaction data (which can be included in any order), using the directive **species** n. This indicates that data for n species are to follow, each species given in a single line using the following format:

name	a8	name of species
mass	real	particle mass for species
charge	real	particle charge for species
populations	integer	unbonded population of species
frozen	integer	determines whether particles of this species are frozen (1) or not (0)

The unbonded population can be omitted for species which are wholly contained in molecules, as can the frozen particle parameter for unfrozen species.

Non-bonded interactions are specified with the directive **interact**ions n, with n pairwise interactions to follow; each is given in a single line using the format:

species 1	a8	name of species 1
species 2	a8	name of species 2
key	a4	interaction key, see Table 10.2
variable 1	real	interaction parameter, see Table 10.2
variable 2	real	interaction parameter, see Table 10.2
variable 3	real	interaction parameter, see Table 10.2
variable 4	real	interaction parameter, see Table 10.2
variable 5	real	interaction parameter, see Table 10.2
variable 6	real	interaction parameter, see Table 10.2
variable 7	real	interaction parameter, see Table 10.2

If any interactions are many-body DPD, the interactions for *all* possible species pairs must be specified in the `FIELD` file and values for all parameters must be given, even if not all of them are required for the many-body DPD model in use. Otherwise only like-like (same species, i.e. $i = j$) interactions are required, as any missing interaction data can be derived using mixing rules: the mixing rule for interaction lengths can be invoked

by setting its value (σ_{ij} or $r_{c,ij}$) equal to zero for a particular interaction. If using the Lowe-Andersen or Stoyanov-Groot thermostats, the dissipative factor γ_{ij} should be replaced with the bath collision frequency Γ_{ij}.

Table 10.2: Non-bonded interactions

key	interaction type	Parameters (1-7)						
lj	Lennard-Jones	ϵ_{ij}	σ_{ij}	γ_{ij}	-	-	-	-
wca	Weeks-Chandler-Andersen	ϵ_{ij}	σ_{ij}	γ_{ij}	-	-	-	-
dpd	Groot-Warren DPD	A_{ij}	$r_{c,ij}$	γ_{ij}	-	-	-	-
mdpd	Many-body DPD	A_{ij}	B_{ij}	C_{ij}	D_{ij}	E_{ij}	$r_{c,ij}$	γ_{ij}

Molecules are specified using the directive **molecul**es n, with data for n molecule types to follow. Immediately after this directive, the following records are included to define each molecule type:

1. Molecule name
   which can be a character string of up to 8 characters in length

2. **nummols** n
   where n is the number of times a molecule of this type appears in the system. This is followed by the data for the molecule type:

3. **bead**s n
   where n gives the number of beads (particles) in this molecule type. A number of records follow for each bead:

name	a8	name of species
x	real	relative x-coordinate for bead
y	real	relative y-coordinate for bead
z	real	relative z-coordinate for bead

   The relative coordinates are used to define the initial shape of the molecule when it is inserted into the system: these are not used if an initial configuration is already available.

4. **no iso**mer
   indicates that the molecule shape should not be reflected or otherwise modified when added to the system. This directive is optional and should be left out if no restrictions on molecule insertion are to apply.

5. **bond**s n
   where n gives the number of flexible bonds in the molecule. Each of the subsequent n records contains:

bond key	a4	potential key, see Table 10.3
index 1 (i)	integer	first bead index in bond
index 2 (j)	integer	second bead index in bond
variable 1	real	potential parameter, see Table 10.3
variable 2	real	potential parameter, see Table 10.3
variable 3	real	potential parameter, see Table 10.3
variable 4	real	potential parameter, see Table 10.3

Note that the bead indices are those arising from numbering each bead in the molecule from 1 to the number specified in the **bead**s directive for this molecule. The same numbering scheme applies for all descriptions of the molecule: DL_MESO_DPD will itself construct the global numbers for all particles in the system.

6. **angle**s n
   where n gives the number of angle bonds in the molecule. Each of the n records following contains:

angle key	a4	potential key, see Table 10.4
index 1 (i)	integer	first bead index in bond angle
index 2 (j)	integer	second bead index in bond angle (central site)
index 3 (k)	integer	third bead index in bond angle
variable 1	real	potential parameter, see Table 10.4
variable 2	real	potential parameter, see Table 10.4
variable 3	real	potential parameter, see Table 10.4
variable 4	real	potential parameter, see Table 10.4

   Angle-based parameters, e.g. θ_0, should be given in degrees. This directive and associated data records need not be specified if the molecule contains no bond angles.

7. **dihed**rals n
   where n gives the number of dihedral interactions in the molecule. Each of the following n records contains:

dihedral key	a4	potential key, see Table 10.5
index 1 (i)	integer	first bead index in bond dihedral
index 2 (j)	integer	second bead index in bond dihedral (central site)
index 3 (k)	integer	third bead index in bond dihedral
index 4 (l)	integer	fourth bead index in bond dihedral
variable 1	real	potential parameter, see Table 10.5
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

   Angle-based parameters, e.g. ϕ_0, should be given in degrees. This directive and associated data records need not be specified if the molecule contains no bond dihedrals.

8. **finish**
   indicates the end of details for a molecule type. Each subsequent molecule type can be entered after this directive, beginning with its name and ending with the **finish** directive.

Surface interactions can be specified using the directive **surf**ace n. If hard adsorbing surfaces are to be used, this directive should be followed by entries specifying the soft short-range repulsions for n species:

name	a8	name of species
key	a4	interaction key
variable 1	real	interaction parameter, see Table 10.6
variable 2	real	interaction parameter, see Table 10.6

where the interaction keys currently available are `dpd` and `wca`.

If frozen bead surfaces are in use, the **froz**en directive should be followed by a single line specifying the properties for the walls to be constructed:

Table 10.3: Bond potentials

key	potential type	Variables (1-4)				functional form
harm	Harmonic	κ	r_0	-	-	$U(r) = \frac{1}{2}\kappa(r - r_0)^2$
fene	(Shifted) FENE	κ	r_{max}	r_0	-	$U(r) = -0.5\kappa r_{max} \ln\left[1 - \left(\frac{r-r_0}{r_{max}}\right)^2\right] : r < r_{max} + r_0$ $U(r) = \infty : r \geq r_{max} + r_0$
wlc	Worm-like chain	A_p	r_{max}	-	-	$U(r) = \frac{k_B T}{2A_p}\left[\frac{1}{2\left(1-\frac{r}{r_{max}}\right)} - \frac{1}{2}\left(1 + \frac{r}{r_{max}}\right) + \frac{r^2}{r_{max}^2}\right] : r < r_{max}$ $U(r) = \infty : r \geq r_{max}$
mors	Morse	D_e	r_0	β	-	$U(r) = D_e[1 - \exp(-\beta(r - r_0))]^2$

Table 10.4: Bond angle potentials

key	potential type	Variables (1-4)				functional form
harm	Harmonic	κ	θ_0	-	-	$U(\theta) = \frac{1}{2}\kappa(\theta - \theta_0)^2$
hcos	Harmonic cosine	κ	θ_0	-	-	$U(\theta) = \frac{1}{2}\kappa(\cos\theta - \cos\theta_0)^2$
cos	Cosine	A	δ	m	-	$U(\theta) = A\left[1 + \cos\left(m\theta - \delta\right)\right]$

Table 10.5: Bond dihedral potentials

key	potential type	Variables (1-4)				functional form
cos	Cosine torsion	A	δ	m	-	$U(\phi) = A[1 + \cos(m\phi - \delta)]$
harm	Harmonic	κ	ϕ_0	-	-	$U(\phi) = \frac{1}{2}\kappa(\phi - \phi_0)^2$
hcos	Harmonic cosine	κ	ϕ_0	-	-	$U(\phi) = \frac{1}{2}\kappa(\cos\phi - \cos\phi_0)^2$

Table 10.6: Surface interactions

key	interaction type	Parameters (1-2)	
dpd	Groot-Warren DPD	$A_{wall,i}$	$z_{c,i}$
wca	Weeks-Chandler-Andersen	$\epsilon_{wall,i}$	σ_i

name	a8	name of frozen bead species
ρ_{wall}	real	density of frozen beads in wall regions
x_{wall}	real	thickness of wall region

External fields are flagged by the directive **extern**al, followed by a line with a keyword indicating the type of field to be applied and the field parameters. A gravitational field can be specified using the keyword **grav** and three real values representing the x-, y- and z-components of gravitational acceleration (adding an additional force of $m\vec{G}$ to each particle), i.e.

$$\texttt{grav}\ \ G_x\ \ G_y\ \ G_z$$

while a constant electric field can be specified using the keyword **elec** and three real values representing the x-, y- and z-components of the electric field (adding an additional force of $q\vec{E}$ to each particle with non-zero charge q), i.e.

$$\texttt{elec}\ \ E_x\ \ E_y\ \ E_z$$

If using the Lees-Edwards shearing boundary condition, the velocity of the walls in dimension α can be specified using the keyword **shear** and three real values representing the x-, y- and z-components for the velocity at $x_\alpha = L_\alpha$, i.e.

$$\texttt{shear}\ \ V_{w,x}\ \ V_{w,y}\ \ V_{w,z}$$

Note that the velocity at $x_\alpha = 0$ will be equal in magnitude but opposite in direction, and that the velocity component for dimension α will be ignored.

The `FIELD` file must be closed with the directive **close**.

If molecules are to be included in the system, the supplied C++ program `molecule-generate` in the directory `DPD/utility` can be used to either create a new `FIELD` file with the required data or append it to a pre-existing file: see Appendix C for more details. Example files in the `DEMO/DPD` directory can be examined for this purpose.

Define initial state: `CONFIG`

An optional `CONFIG` file can be included to define the initial state of the system, which can include the positions, velocities and forces for each particle[6]. This file is read by the subroutine `read_config` in `start_module` and scanned by the subroutine `scan_config` in `config_module`.

At the beginning of the file, five lines of information (of which the first two are mandatory) have to be included:

- The simulation name (80 characters)
- The `CONFIG` file key `levcfg` (integer), the periodic boundary key `imcon` (integer), the number of particles in the file (integer, optional) and the configuration energy (real, optional)
- The x-, y- and z-components for the x-axis vector (real, optional)
- The x-, y- and z-components for the y-axis vector (real, optional)
- The x-, y- and z-components for the z-axis vector (real, optional)

The file key `levcfg` is set depending on the information available for each particle: 0 for positions only, 1 for positions and velocities or 2 for positions, velocities and forces. If particle velocities are not specified, these

[6]This file is formatted identically to `CONFIG` files used in DL_POLY[117, 129], with the origin is set as the centre of the simulation volume.

are generated at random to produce a distribution corresponding to the required system temperature, while unknown forces are set to zero. The simulation name, number of particles in the file and configuration energy are not read by DL_MESO_DPD and can thus be ignored (although the line for the simulation name must remain). If axes vectors are included in the `CONFIG` file and the value of `imcon` is greater than zero, these will be read on the assumption that the simulation volume is orthorhombic (the only possible shape available in DL_MESO_DPD).

Each particle is represented by a block record, with at least two lines of information:

- The species name (8 characters) or number (integer) and the global particle number (integer, optional)
- The x-, y- and z-coordinates for the particle (real)
- The x-, y- and z-components of particle velocity (real, if `levcfg`> 0)
- The x-, y- and z-components of force on the particle (real, if `levcfg`> 1)

If global particle numbers are not included or the **no ind**ex option is invoked in the `CONTROL` file, these are generated automatically for the particles in the order specified by the `CONFIG` file. Care should be taken that any particles belonging to molecules are numbered correctly, since the bond information is assigned in an identical fashion to unspecified systems, i.e. numbering after all loose particles in the relative order specified by the `FIELD` file. If the **nfold** option is invoked in the `CONTROL` file, DL_MESO_DPD will duplicate the given configuration in each Cartesian direction and assign global particle numbers to the enlarged system in a similar fashion, i.e. unbonded particles precede bonded ones and molecules are ordered according to the `FIELD` file.

`CONFIG` files can be created from restart files or trajectory files from previous simulations using the supplied Fortran90 programs `export_config` or `history_config` respectively in the directory `DPD/utility`; see Appendix C for more details. Frozen particle walls, if specified in the `CONTROL` file, can be added to systems with `CONFIG` files (with or without duplication) but users have to ensure that any molecules do not cross boundaries where frozen particle walls will be placed: no checks are available to prevent this from happening but `CONFIG` files could be created from previous simulations involving hard adsorbing surfaces.

If the **l_init** option is selected in the `CONTROL` file, a file `CFGINI` will be created in the same `CONFIG` file format to provide the initial configuration of the simulation with particle velocities and forces. This option is automatically switched off if a `CONFIG` file with forces acting on particles is supplied and no volume expansion (with the **nfold** option) is selected, since the `CFGINI` file would be identical to the `CONFIG` file.

10.2 Output files

General output file: `OUTPUT`

This ANSI text file is generated by all DPD calculations and contains:

- The number of processors available and their endianness.
- The system and bond/angle/dihedral properties used for calculations.
- Domain decomposition details (Parallel version only).
- The starting positions and velocities of a particle sample.
- The calculation time, current values and rolling averages for the total energy, potential energies (total, electrostatic, and from bond stretching, angles and dihedrals), virial, kinetic energy, pressure and temperature (overall and partial for each dimension) every `nsbpo` time steps.
- Final averages and fluctuations (standard deviations) for all reported properties plus stress tensors over all time steps after equilibration.

- The final positions and velocities of a particle sample.
- Elapsed and average times for the calculation.

Only the properties relevant to the simulation will be printed: electrostatic and bond energies will be output if the simulation includes these kinds of interactions, while the partial temperatures will be output if the system is likely to include dynamics (e.g. if a constant force or shear is applied) or if this option is switched on using the **print part**ial directive in the `CONTROL` file.

If the **l_scr** directive is included in the `CONTROL` file, the above simulation information will be redirected to the standard output for the machine and its operating system (e.g. to the screen) and no `OUTPUT` file will be generated. This directive may be useful when a simulation crashes but no error messages or other information are printed to the `OUTPUT` file[7].

Restart files: `export` and `REVIVE`

A restart file with the name `export` is produced every `ndump` time steps. This binary file contains the following information for the time step:

- Name of DPD calculation.
- Total number of particles, specified temperature, size of timestep, system volume and boundary displacement due to shear.
- Particle global identify numbers, species and molecule numbers, Cartesian coordinates, velocities and forces.

An additional streamed binary file called `REVIVE` is also produced at the same time and contains the following information:

- Name of DPD calculation.
- Current values of barostat parameters.
- Current statistical properties (current values, cumulative sums and fluctuations, rolling averages and stacks).
- Numbers of processor cores and threads in use.
- Current state of random number generators on all cores and threads.

In combination, the `export` and `REVIVE` files can restore a stopped DPD calculation, while the `export` file on its own can be used as an initial configuration for a new simulation, with or without velocity scaling to account for changes in system temperature. The `export` file can also be used to generate a plot at the given time step or a `CONFIG` file for subsequent simulations using the utilities `export_image` and `export_config` respectively in the `DPD/utility` directory; Appendix C gives instructions for their use.

The `export` file is written in parallel by assigning processor cores to groups, which each gather their particle data onto a root processor: the root processors subsequently write their data into the `export` file collaboratively using MPI-IO. The majority of the `REVIVE` file is written by a single processor core, but all cores collaboratively write their random number generator states using MPI-IO.

[7] This directive can also be used to direct this output to a user-named file as an alternative to `OUTPUT`: see Appendix B for more details.

Trajectory file: HISTORY

If the **traj**ectory option is specified in the CONTROL file, a trajectory file with the name HISTORY will be generated and data added to it periodically: the timestep intervals and the initial timestep for writing to this file are specified by the user using this directive. This streamed binary file contains some of the system properties — including unique indices for particles and molecules — followed by data for every particle in the domain at each timestep: the level of data (positions, velocities, forces) is chosen by the user. If multiple processor cores are used for the simulation, processor cores are assigned into groups and each group gathers its data onto a single root processor, which subsequently writes the data in collaboration with other root processors to the HISTORY file using MPI-IO.

The HISTORY file can be used with eight utilities provided in the DPD/utility directory. The utilities traject_vtf and traject_selected_vtf produce plottable VMD files [56], with sets of bonded particles represented as residues. The utility traject_xml produces plottable XML files formatted for the coarse-grained molecular dynamics code GALAMOST [147], which can be read into the visualisation software package OVITO [122]. The utility history_config can produce a CONFIG file from a selected time step in the HISTORY file. The utility local can calculate localized properties (e.g. temperature, composition) and produces VTK files with these properties as cell data. The utility isosurfaces can produce isovolumetric plots based on a given species in VTK files and calculate order parameters to determine the structure of a mesophase. The utility radius calculates the end-to-end distances and radii of gyration for all molecules, as well as give normalised distributions of end-to-end distances. The utility dipole calculates the dipole moments for all molecules in the system, autocorrelation functions of dipole moments and their Fourier transforms. The utilities rdf and rdfmol calculate radial distribution functions (RDFs) for pairs of particle species and molecule types respectively, with the option of calculating Fourier transforms of the RDFs to give structure factors. The utility widom_insertion calculates excess chemical potentials of individual particles or molecules by random insertions into particle configurations supplied from trajectory frames. Appendix C gives instructions for their use.

Statistical data file: CORREL

If specified by the **stats** directive in the CONTROL file, DL_MESO_DPD will generate an ANSI text file containing statistical data at a timestep interval specified by the user, which can later be imported into a spreadsheet or used by graph-plotting software. The formatting of the data varies depending on which kinds of interactions (bonds, angles, dihedrals, electrostatics) were used and whether a barostat was applied, based on the overall format (in a single line):

$$\begin{array}{rrrrrrrrrrrrrrrrrrr} & & & & & & & & & t & E_{tot} & E_{pot,tot} & E_{pot,elec} & E_{pot,bond} & E_{pot,angle} & E_{pot,dihed} & \\ P & P_{xx} & P_{xy} & P_{xz} & P_{yx} & P_{yy} & P_{yz} & P_{zx} & P_{zy} & P_{zz} & V & L_x & L_y & L_z & \gamma_z & T & T_x & T_y & T_z \\ & & & & & & & & & \langle r_{bond}\rangle & r_{bond,max} & r_{bond,min} & \langle\theta_{angle}\rangle & \langle\phi_{dihed}\rangle & & & & \end{array}$$

where t is the time, E energy (*tot* denoting total, *pot* potential), P pressure, P_{ij} pressure tensor (ratio of stress tensor to volume), V volume, L_α box length in dimension α ($\alpha = x, y, z$), γ_z interfacial tension along z-axis, T temperature, $T_\alpha = \langle \sum_i m_i v_{i,\alpha}^2 \rangle$ partial temperature for dimension α, $\langle r_{bond}\rangle$ the mean bond length, $r_{bond,max}$ and $r_{bond,min}$ the maximum and minimum bond lengths, $\langle\theta_{angle}\rangle$ the mean bond angle (in degrees) and $\langle\phi_{dihed}\rangle$ the mean bond dihedral (in degrees).

Any property which does not vary or is not measured during the simulation, e.g. volume for NVT ensembles, is omitted from each line of data. Headers are placed at the top of the file to indicate which properties are included, starting with a hash character to allow easier reading by graphing software.

Separated stress tensor files: Stress_*.d

If specified by the **stress** directive in the CONTROL, DL_MESO_DPD will generate at least one ANSI text file containing stress tensors at a timestep interval from an initial timestep, both of which are specified by the user. These stress tensors are separated out into up to four components with associated filenames:

- Interaction potential (conservative, many-body, bonded and electrostatic) components: `Stress_pot.d`
- Dissipative components: `Stress_diss.d`
- Random components: `Stress_rn.d`
- Kinetic components: `Stress_kin.d`

and the user can select which components are written to these files in the `CONTROL` file. The data is formatted in a similar form to the `CORREL` file:

$$t \;\; P \;\; P^Q_{xx} \;\; P^Q_{xy} \;\; P^Q_{xz} \;\; P^Q_{yx} \;\; P^Q_{yy} \;\; P^Q_{yz} \;\; P^Q_{zx} \;\; P^Q_{zy} \;\; P^Q_{zz} \;\; V$$

where t is the time, P the total pressure, $P^Q_{ij} = \sigma^Q_{ij}/V$ the separated pressure tensor (where i and j can each be x, y or z, Q representing potential, dissipative, random or kinetic types) and V the system volume. Headers are placed at the top of each file to indicate the properties in each column, starting with a hash character to allow easier reading by graphing software. The data in these files can be used to analyse the rheological behaviour of the system, e.g. integrating autocorrelation functions of the potential and dissipative components of stress tensors to find the zero-shear viscosity[27].

Chapter 11

DL_MESO DPD Examples

Test cases for Dissipative Particle Dynamics simulations using DL_MESO – including the required input and sample output files – can be found in the `DEMO/DPD` subdirectory. All of the following examples can be run using either the serial or parallel versions of DL_MESO_DPD, with or without OpenMP multithreading. The smaller problems (i.e. with up to 20 000 particles) are best suited to running in serial or on a small number of processor cores (e.g. 16 or less) to limit the times required for interprocess communication, while larger problems are better suited to running in parallel on larger numbers of processor cores to reduce the memory requirements per processor core.

Images of all test cases and videos for some can be found in the Example Simulations page of the DL_MESO website: a link to it can be found at `www.ccp5.ac.uk/DL_MESO`

11.1 Mixture_Small

This simulation consists of 1000 particles with 3 species with populations of 333, 333 and 334 respectively. All particle types have identical sizes and masses but different energy parameters, using the default mixing rules for unlike particle parameters. Figure 11.1 gives a snapshot of the system at the end of the simulation, demonstrating mixing between the three particle types (represented by different colours).

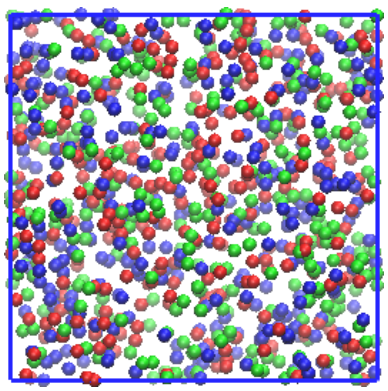


Figure 11.1: Visualisation of system at final time step from DPD `Mixture_Ser` test case

11.2 Mixture_Large

This simulation example is similar to `Mixture_Small` but larger: it consists of 512 000 particles with 2 species, each with a population of 256 000 particles. The particle types have identical sizes and masses but different energy parameters, using the default mixing rules for unlike particle parameters. Figure 11.2 gives a snapshot of the system at the end of the simulation, demonstrating good mixing between the two particle types.

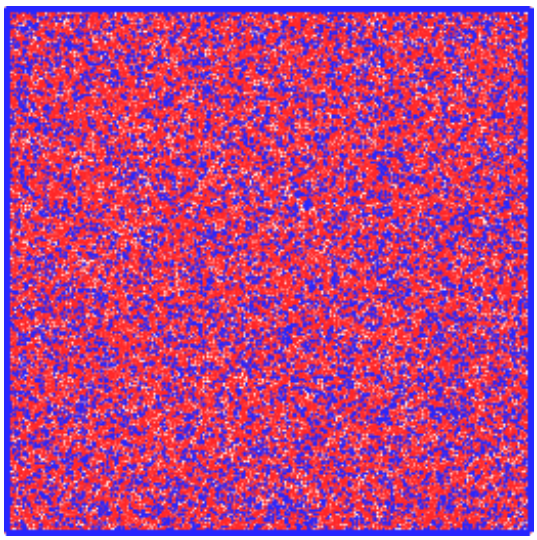
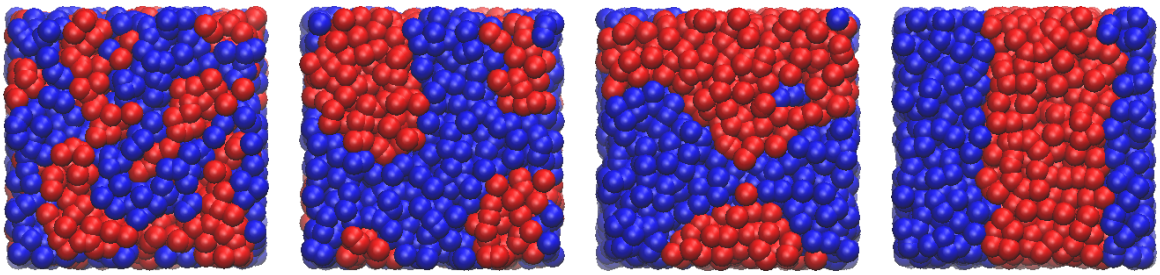


Figure 11.2: Visualisation of system at final time step from DPD `Mixture_Par` test case

11.3 PhaseSeparation

This simulation example consists of 3000 particles with 2 species, each with a population of 1500. Both particle types have identical sizes, masses and like-like energy parameters, but the unlike energy parameter has been set to a larger value to produce phase separation, which can clearly be seen in Figure 11.3. The initial state of this simulation has been provided in a `CONFIG` file. An `.AVI` video file of the simulation can be found in the Example Simulations page of the DL_MESO website.



(a) $t = 2$ (b) $t = 10$ (c) $t = 30$ (d) $t = 60$

Figure 11.3: Visualisations of DPD `PhaseSeparation` test case (red for particle type 1, blue for type 2)

11.4 Aggregate

This simulation consists of 3000 unbonded particles and 30 molecules of 10 particles each with harmonic bonds between them. The unbonded particles and molecules are made up of different species with a higher energy parameter for unlike particle interactions. This causes the molecules to aggregate, which can be seen in Figure 11.4, a snapshot of the simulation.

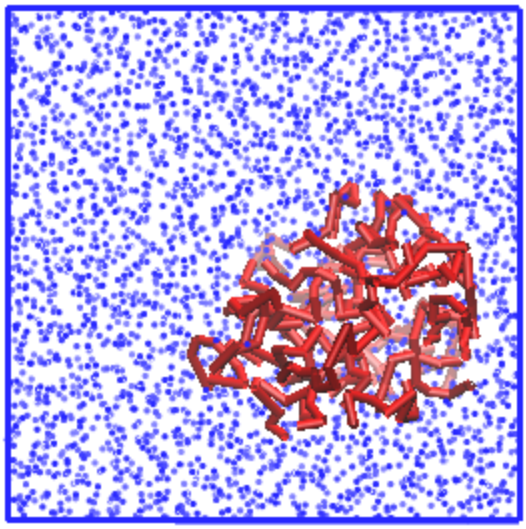
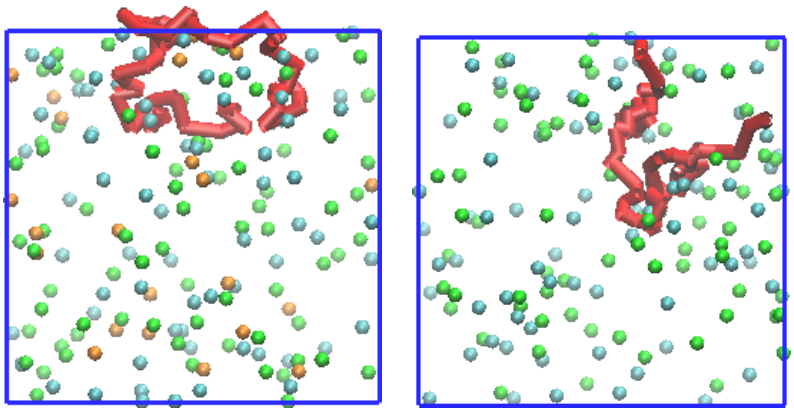


Figure 11.4: Visualisation of system during the DPD `Aggregate` test case

11.5 Polyelectrolyte

This simulation example consists of a slightly hydrophobic polyelectrolyte molecule of 50 particles, each with a relative charge of +0.5, immersed in a salt solution of concentration $0.14M$[41]. The salt solution consists of 9900 neutral water particles, 75 cationic salt particles (net charge +1), 75 anions of charge −1 and 25 counterions of charge −1 to keep the system neutral. A similar simulation is included with the polyelectrolyte replaced with a neutral polymer of the same number of particles and the counterions replaced with water (`FIELD-neutral`: this should be renamed to `FIELD` to run the simulation and used with the same `CONTROL` file). Figure 11.5 gives a comparison between the polyelectrolyte and neutral polymer at the final time step, which have measured radii of gyration of 3.21 and 2.58 respectively. The evolution of radii of gyration can be seen in the files `radius` and `radius-neutral` for the polyelectrolyte and neutral polymer respectively, given in the fourth column of each file.



(a) Polyelectrolyte (b) Neutral polymer

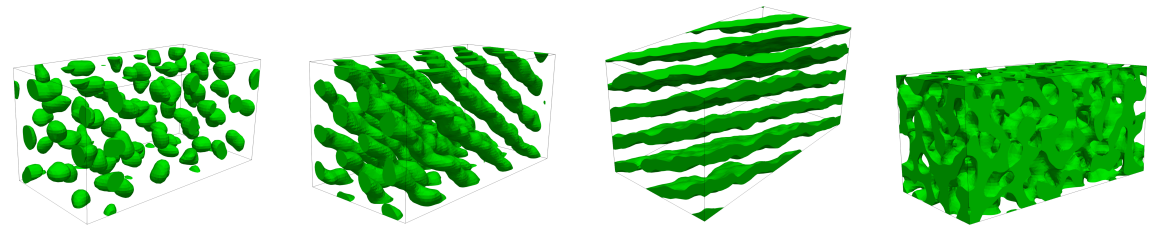
Figure 11.5: Visualisations of DPD `Polyelectrolyte` test case: red for polyelectrolyte/polymer, green for salt cations, cyan for anions, orange for counterions (water omitted for clarity)

11.6 AmphiphileMesophases

This example consists of four separate simulations, each with 12 000 particles consisting of dimers (molecules consisting of two particles, one hydrophilic and the other hydrophobic, with harmonic bonds of equilibrium length 0.5 between them) and unbonded monomers[63]. Defining the composition ϕ as the ratio of DPD particles within dimers to the total number of particles in the system, the interaction data for simulations with

dimer compositions of 30%, 55%, 75% and 90% are provided with filenames `FIELD-30`, `FIELD-55`, `FIELD-75` and `FIELD-90` respectively. (Each of these files should be renamed to `FIELD` to run the simulation, while the `CONTROL` defining the simulation properties can be used for all four simulations.)

These systems provide four points on a phase diagram corresponding to isotropic dimer, hexagonal, lamellar and isotropic monomer phases respectively. The final configurations obtained for each phase can be seen in Figure 11.6, shown as isosurfaces of the hydrophobic particles to highlight the distinctions between the phases. The three eigenvalues of the second moment of isosurface normal distributions can be used as order parameters for these mesophases[97]: the evolution of these values over time are given by the files `moment-30`, `moment-55`, `moment-75` and `moment-90`.



(a) $\phi = 0.30$, $\mu_1 \approx 0.323$, $\mu_2 \approx 0.335$, $\mu_3 \approx 0.341$

(b) $\phi = 0.50$, $\mu_1 \approx 0.053$, $\mu_2 \approx 0.445$, $\mu_3 \approx 0.502$

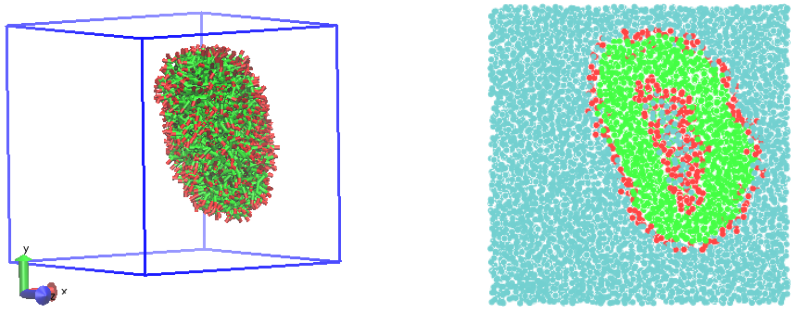
(c) $\phi = 0.75$, $\mu_1 \approx 0.032$, $\mu_2 \approx 0.037$, $\mu_3 \approx 0.931$

(d) $\phi = 0.90$, $\mu_1 \approx 0.316$, $\mu_2 \approx 0.338$, $\mu_3 \approx 0.346$

Figure 11.6: Visualisations of DPD `AmphiphileMesophases` test case at final time step (isosurfaces of hydrophobic particles)

11.7 VesicleFormation

This simulation example consists of 37 440 unbonded water particles and 1008 molecules, each consisting of one hydrophilic head particle and three hydrophobic tail particles bonded together with stiff harmonic bonds of equilibrium length 1.0 between them[141]. The molecules represent amphiphiles and during the course of the simulation self-assemble into a vesicle and encapsulate a number of water particles. Figure 11.7 shows the self-assembled vesicle, both in three dimensions and in a cross-section to show the encapsulated water. An `.AVI` video file of the simulation can be found in the Example Simulations page of the DL_MESO website.



(a) Vesicle (water omitted for clarity) (b) Cross-section orthogonal to z-axis

Figure 11.7: Visualisations of DPD `VesicleFormation` test case at $t = 50\ 000$ (red for hydrophile, green for hydrophobe, cyan for water)

11.8 LipidBilayer

This simulation example consists of 35 152 unbonded water particles and 2000 molecules, each consisting of one hydrophilic head particle and six hydrophobic tail particles bonded together with stiff harmonic bonds of equilibrium length 0.5 between them and cosine angle potentials between each pair of bonds[111]. The molecules represent amphiphilic lipids and self-assemble into a bilayer. Figure 11.8 shows the self-assembled lipid bilayer.

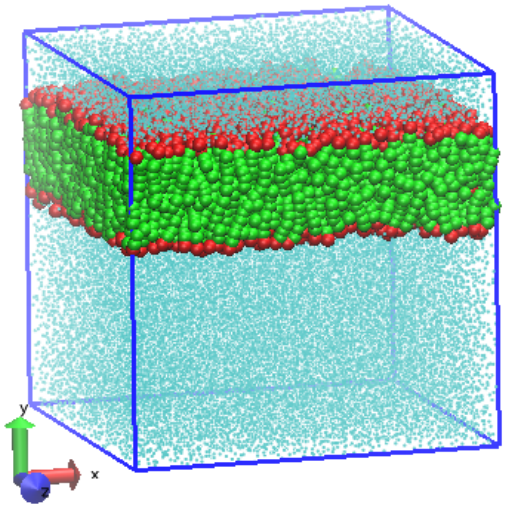


Figure 11.8: Visualisation of system at the end of the DPD `LipidBilayer` test case

11.9 AlkylSulphate

This simulation example consists of 22,200 unbonded water particles and 360 molecules, each consisting of a terminating alkyl bead (CH_3), two two-carbon alkyl beads (CH_2CH_2) and a combined alkyl/sulphate bead with a negative charge ($CH_2OSO_3^-$) to represent sodium hexyl sulphate (S6S)[2]: each molecule is also accompanied

by a free-moving positively-charged sodium ion (Na^+). The simulation is carried out in a periodic box of $20 \times 20 \times 20$ DPD length units using Smooth Particle Mesh Ewald calculations with approximate Slater-type (exponential) charge smearing to determine electrostatic interactions. The molecular concentration is higher than the critical micelle concentration, so several micelles are formed as shown in Figure 11.9.

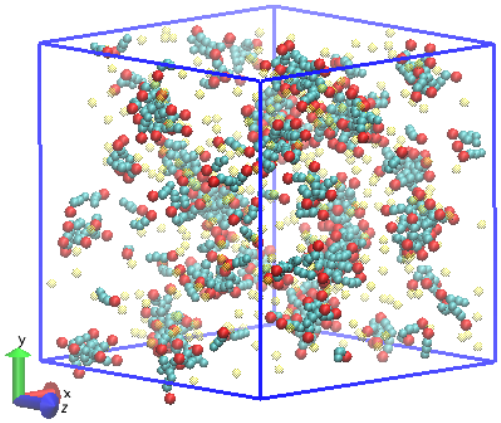


Figure 11.9: Visualisation of system at the end of the DPD `AlkylSulphate` test case (red for alkyl/sulphate beads, cyan for alkyl beads, translucent yellow for sodium ions)

11.10 FloryHuggins

This simulation example consists of 3840 unbonded particles in a box of 20×8 DPD length units: each half of the box consists of one particle type. The conservative force parameter between the two species is set to a higher value than that between pairs of same-species particles. The volume fraction of either component away from the interface between them – as shown in Figure 11.10 – allows for calculation of the Flory-Huggins χ parameter: varying the conservative force parameter between the two components allows the relationship between A_{ij} and χ to be determined[43].

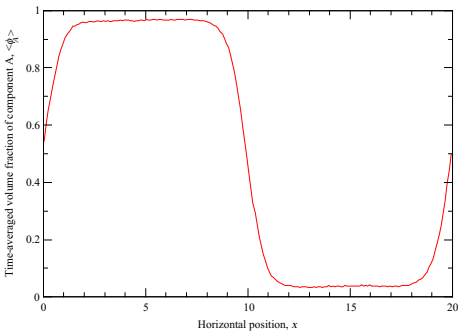
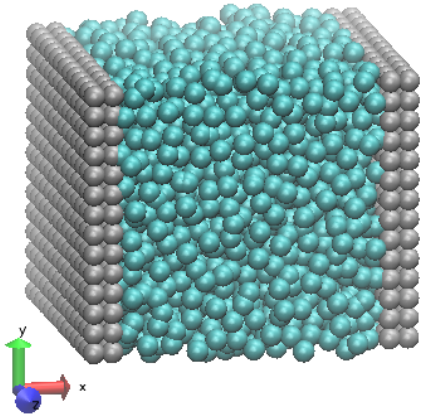


Figure 11.10: Time-averaged plot of species concentration along x-axis of box in `FloryHuggins` test case: $A_{ij} = 37$, $A_{ii} = A_{jj} = 25$, $\phi_A \approx 0.96578$, $\chi \approx 3.585$

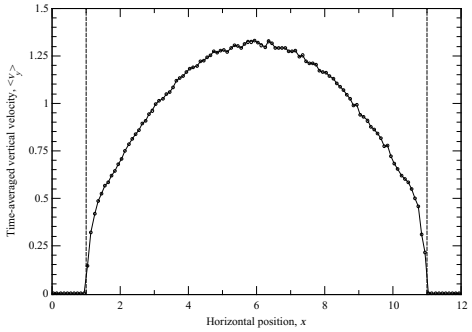
11.11 PoiseuilleFlow

This simulation example consists of 3000 unbonded particles in a box of $10 \times 10 \times 10$ DPD length units with walls of frozen particles added to the surfaces orthogonal to the x-axis of thickness 1 DPD length unit and particle density of 3, as well as bounce back reflections at those surfaces to prevent particles penetrating the walls. A constant body force in the direction of the y-axis is added to each non-frozen particle which, in combination with the frozen particle walls approximating no-slip boundaries, gives Poiseuille flow of the DPD fluid. Figure 11.11 gives a snapshot of the system at the final time step, as well as plots of y-component velocity, density of

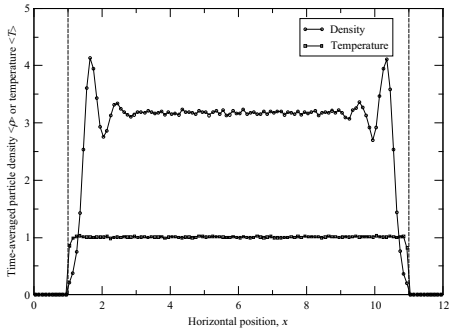
the fluid particles and temperature (defined only by x- and z-components of velocity in this case). The emergent velocity profile is similar to that expected for Poiseuille flow, while the temperature and density profiles are mainly flat across the entire spacing between the walls, apart from significant density fluctuations close to the walls.



(a) Visualisation of system at $t = 2\ 000$



(b) Time-averaged y-component of velocity



(c) Time-averaged fluid density and temperature

Figure 11.11: Visualisation and plots from DPD `PoiseuilleFlow` test case: broken lines denote positions of no-slip boundaries due to frozen particle walls

11.12 ShearFlow

This simulation example consists of 3000 unbonded particles in a box of $10 \times 10 \times 10$ DPD length units with Lees-Edwards shearing boundaries orthogonal to the y-axis. The Stoyanov-Groot thermostat is used for this system to control both the fluid viscosity and system temperature. Figure 11.12 gives the emergent time-averaged velocity profile, yielding a shear rate of 0.2013 (in DPD units, close to the applied shear rate of 0.2) against a measured stress component $\langle \sigma_{yx} \rangle = -0.1985$.

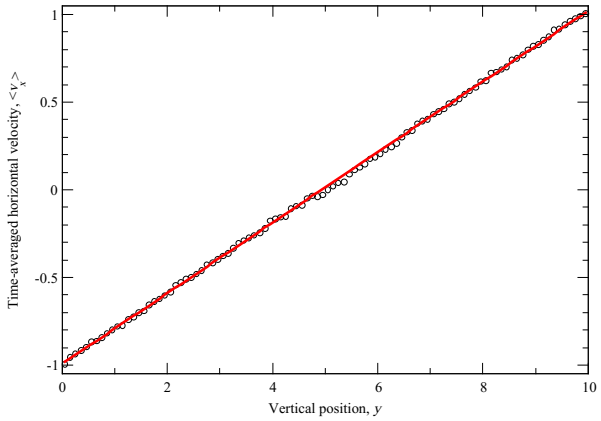


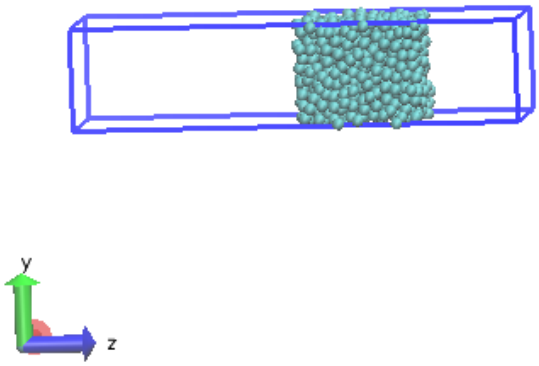
Figure 11.12: Plot of x-component velocity from DPD `ShearFlow` test case: red line denotes best-fit for determining shear rate

11.13 VapourLiquid

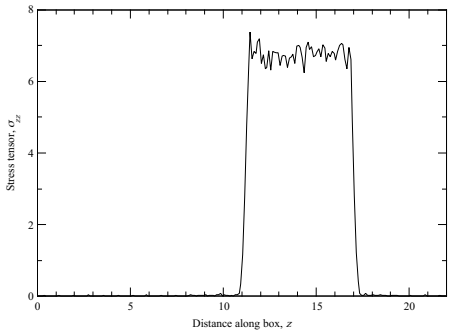
This simulation example consists of 1000 unbonded water particles initially distributed uniformly in a box of $5 \times 5 \times 22$ DPD length units, using the default many-body DPD interactions with $A_{ij} = -50$ and $B_{ij} = 25$ to apply vapour-liquid interactions and surface tension[132, 36]. Figure 11.13 shows the system at the final timestep after the water particles have coalesced into a single body surrounded by empty space, along with the time-averaged zz stress component (σ_{zz}) along the z-dimension of the box.

11.14 SurfaceDrop

This simulation example consists of 4000 unbonded water particles initially distributed uniformly in a box of $27 \times 13 \times 27$ DPD length units, with walls of frozen particles added to the surfaces orthogonal to the y-axis of thickness 1 DPD length unit and particle density of 3. A constant body (gravitational) force is applied downwards to each non-frozen particle and many-body DPD interactions similar to those in the `VapourLiquid` example are used, with a higher value of A_{ij} between the water and wall particles to give a hydrophobic surface. Figure 11.14 shows the system at the final timestep after the water particles have coalesced into drops sitting on the bottom wall. An MPEG-4 video file of the simulation can be found in the Example Simulations page of the DL_MESO website. (This test case is courtesy of Erik Johansson at the Department of Energy Sciences, Faculty of Engineering, Lund University, Sweden[61].)



(a) Visualisation at final time step



(b) Time-averaged stress component σ_{zz}

Figure 11.13: Visualisation of system at final time step from DPD `VapourLiquid` test case

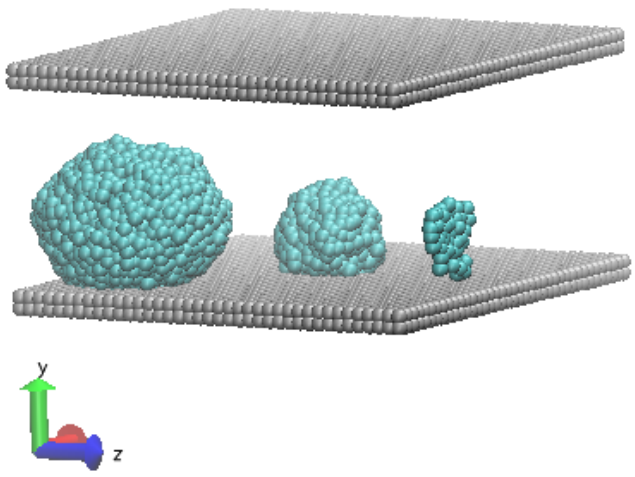


Figure 11.14: Visualisation of DPD `SurfaceDrop` test case at final time step

Appendix A

Changes to input files from previous versions of DL_MESO

This appendix the major differences in input file formats between previous versions of DL_MESO and the current one, particularly those that may affect users' abilities to repeat previous calculations.

A.1 DL_MESO_LBE

All `lbin.sys` and `lbin.spa` files from at least version 2.2 of DL_MESO onwards are backwards compatible, i.e. these can still be read and used by the current version of DL_MESO_LBE (with some caveats: see below), as are `lbin.init` files from version 2.5 onwards. No fundamental changes have ever been made to `lbin.spa` and `lbin.init` file formats: additional boundary types have been added to `lbin.spa` (e.g. mid-grid bounce back since version 2.4, outflow conditions in version 2.7), but these have not required modifications to previously defined boundary condition codes.

As it was not possible to specify collision algorithm, mesophase interaction type, output file format etc. in `lbin.sys` files prior to version 2.5, older versions of the LBE code had to have these variations hard-coded by the user in the main program source file. The input file format for `lbin.sys` up until version 2.5 could only read numbers (double precision or integer) for each keyword. DL_MESO will apply particular defaults for information that is not included in a `lbin.sys` file, but it is the user's responsibility to ensure these defaults are suitable for the given simulation. Salient information about the simulation is written to the screen or standard output prior to calculation, which gives the user the opportunity to see what DL_MESO has selected based on the input files. The user may also wish to use the Java GUI to read the `lbin.sys` file and see how it might be interpreted by `DL_MESO_LBE`, as well as save an updated version of the same file.

A number of significant changes have occurred between versions 2.6 and 2.7 of DL_MESO_LBE that are reflected in changes to `lbin.sys` files:

- Multiple types of constant density/velocity boundary condition for fluids (Zou/He, Inamuro, regularised, kinetic) can now be selected (compared to previously fixed Zou/He conditions, which are still the default) using the keyword **boundary_type** in `lbin.sys`.
- Multiple types of constant solute concentration and temperature conditions (Zou/He, Inamuro) can now be selected (compared to previously fixed Inamuro conditions from version 2.4): the default is now Zou/He instead of Inamuro. These can be specified using the keywords **solute_boundary_type** and **thermal_boundary_type** in `lbin.sys`.
- The number of output files written per frame can be reduced by collecting together data among processors in one, two or three directions using the keywords **output_combine_x**, **output_combine_y** and **output_combine_z**. If all directions for a simulation are selected for collecting data, MPI-IO is used to put the data together into a single file.

- Simulations can be restarted from a new `lbout.dump` file containing the distribution functions at all grid points: the frequency of creating this file is specified with the keyword **dump_span**, while the use of the file for a simulation restart is specified with **restart_simulation**.
- Maximum calculation runtimes can be specified with **calculation_time** in `lbin.sys` to ensure simulations come to a stop, e.g. if running on a computing platform with a job submission system.

Chapter 6 gives more details about the keywords and values that can now be specified in the `lbin.sys` file.

A.2 DL_MESO_DPD

Since version 2.5 of DL_MESO, the `CONTROL` and `FIELD` input files for the DPD code have been formatted in a similar manner to DL_POLY: the utility `convert-input.cpp` is available to convert the input files for earlier versions of DL_MESO_DPD (up to and including version 2.4) to the DL_POLY-based form used by version 2.7.

A few minor changes to `CONTROL` and `FIELD` files have been made between versions 2.6 and 2.7:

- The specification of charge smearing type and parameter has changed from one line in the `CONTROL` file to two, one specifying the smearing type and the other specifying the parameter (which can either be the smearing length or its reciprocal). For instance, if approximate Slater smearing with the parameter $\beta = 0.929$ is required, the directive

  ```
  smear slater 0.929
  ```

  in version 2.6 should now be changed to

  ```
  smear slater approx
  smear beta 0.929
  ```

  for version 2.7. If this change is not made, DL_MESO_DPD will close with an error message stating that the charge smearing length has not been specified.

- Frozen bead walls can now be specified separately from other boundary conditions, allowing reflective boundaries (either specular or bounceback) to be applied at the edge of a frozen bead wall to prevent penetration by fast-moving particles. This has necessitated the introduction of new keywords in the `CONTROL` and FIELD files. For instance, frozen bead walls in the y-dimension that were specified in the `CONTROL` file in version 2.6 by

  ```
  surface frozen y
  ```

  should now be changed to

  ```
  frozen walls y
  ```

  for version 2.7. Similarly, the following two lines in the `FIELD` file in version 2.6 specifying the frozen wall species, density and thickness:

  ```
  SURFACES
  WALL     3.0 1.0
  ```

  should be changed to the following for version 2.7:

  ```
  FROZEN
  WALL     3.0 1.0
  ```

- Surface interactions have been expanded for version 2.7 to allow different types (DPD and Weeks-Chandler-Andersen) and length scales (e.g. cutoff distances) to be specified. The **surf**ace directive in the `FIELD` file has been altered for version 2.7: instead of specifying the interactions strengths for *all* particle species in versions 2.5 and 2.6, e.g.

  ```
  surface
  A 25.0
  B 30.0
  ```

  this directive is now similar to the **interact**ions directive with a number in the same line indicating the number of parameter sets to follow, each of which consists of the species, the type, the interaction strength and length scale, e.g.

  ```
  surface 2
  A dpd 25.0 1.0
  B dpd 30.0 1.0
  ```

Chapter 10 gives more details about the keywords and values that can now be specified in the `CONTROL` and `FIELD` files.

Appendix B

Manual compiliation and running of DL_MESO

B.1 DL_MESO_LBE

DL_MESO_LBE has been written in C++ in a modular form and the main program codes – `slbe.cpp` for serial running, `plbe.cpp` for parallel running – are designed to allow the user to change algorithms for collision, mesophases etc. by specifying them in input files. Customised codes (`slbecustom.cpp`[1] and `plbecustom.cpp`) are also available to allow users to 'hardwire' the algorithms for collisions, mesophases etc. into the code, which might improve computational efficiency.

To compile the code and produce an executable in the working directory, at a command line type:

- `c++ ../LBE/slbe.cpp -o lbe.exe`[2]

assuming that `c++` is the name of the available C++ compiler, `slbe.cpp` is the version of the code being compiled and `lbe.exe` is the name of the executable required. If compiling the parallel version of the code, the command for the C++ compiler 'wrapped' with MPI is required, which is commonly `mpiCC` or `mpicxx`. Additional compiler flags may be used between the compiler name and the reference to the code to improve computation speed or assist in debugging. To compile the OpenMP multithreaded versions of the codes, an additional compiler flag is needed to invoke the OpenMP libraries (e.g. `-fopenmp`).

Two compile-time options are available for the parallel version of the code:

- `MPIold`: this is a backwards compatibility option to use MPI-1.x subroutines for creating MPI derived data types (used in communications) instead of the default subroutines for MPI-2.x and later.
- `Packbuf`: this replaces communications using MPI derived data types with packed and unpacked buffers. If OpenMP multithreading is used for the code, the packing and unpacking of buffers will be split between threads.

and these can be invoked using compiler flags, e.g. `-DPackbuf`.

Before running the executable, the required input files (`lbin.sys`, `lbin.spa` and optionally `lbin.init`) need to be copied or moved into the working directory. If running in serial, the executable can just be run using the command:

- `lbe.exe` if running in Windows, or
- `./lbe.exe` if running in Unix-like operating systems,

while the parallel version requires a command to run N identical copies of the program on N processors, e.g.

[1] If preferred, an alternative version that uses a boundary layer, `slbecombine.cpp`, is available.
[2] `/` may need to be replaced by \ on computers running Windows.

- `mpirun -np` N `./lbe.exe`

and may need to be launched via a batch job script: please consult your machine administrator or documentation for further details.

The diagnostic output from DL_MESO_LBE (indicating the masses and momentum of the fluids in the system) is ordinarily sent to the machine and operating system's standard output, which can either be the screen or (if running using a batch job script) an automatically generated text file. To redirect this output to a specific file, the command used to run the simulation (either in the terminal or in the batch job script file) can be modified to 'pipe' its output, e.g.

- `./lbe.exe > output`

Modifications may be made to the customised versions of the code to select or leave out routines for e.g. specific collision, propagation and mesophase interaction algorithms instead of relying on keywords in the `lbin.sys`. The user can also create new subroutines and functions in the `lbpUSER.cpp` and `lbpUSER.hpp` files. (Further details about these can be found in the Developer Manual.)

B.2 DL_MESO_DPD

The Fortran modules for DL_MESO_DPD (ending in `*.F90` to automatically invoke preprocessing) must be compiled in a particular order to satisfy dependencies of shared variables and arrays:

- `constants`
- `variables`
- `numeric_container`
- `comms_module`
- `error_module`
- `parse_utils`
- `bond_module`
- `surface_module`
- `domain_module`
- `manybody_module`
- `ewald_module`
- `spme_module`
- `start_module`
- `config_module`
- `field_module`
- `integrate_dpd_mdvv`
- `integrate_dpd_dpdvv`
- `integrate_dpd_shardlow`
- `integrate_lowe`

- integrate_peters
- integrate_stoyanov
- statistics_module
- run_module
- dlmesodpd

If running DL_MESO_DPD in serial, the modules comms_module and domain_module should be replaced by comms_module_ser and domain_module_ser respectively. If using the OpenMP multithreaded version of DL_MESO_DPD, the modules numeric_container, bond_module, manybody_module, ewald_module, spme_module and field_module should be replaced by similarly named modules ending with _omp, and dlmesodpd should be replaced with dlmesodpd_omp.

To simplify the process, a makefile may be created either in the DPD directory or in the working directory to automatically compile the modules and build the executable. Examples of these for running in the DPD directory may be found in the DPD/makefiles directory and modified by the user. The compiler (after CFC=) and flags (after FFLAGS=) may need changing depending on the Fortran compiler available: if MPI is available, the Fortran compiler 'wrapped' with MPI (most commonly mpifort or mpif90) is required, while if OpenMP is available, the correct compiler flag to include its libraries is needed. If invoking from the working directory, the modules for DL_MESO_DPD should either be preceded by the path, i.e. ../DPD/, in the list of compile sources or the directive VPATH=../DPD/ can be used before the source list; the latter strategy is used by the DL_MESO GUI when creating makefiles.

If an alternative FFT solver to that supplied with DL_MESO_DPD is to be used, compile time options can be used to invoke the correct lines when compiling the code:

- FFTW: this uses the Fastest Fourier Transform in the West (FFTW)[33], linking the C libraries rather than legacy Fortran versions;
- ESSL: this uses the FFT solver supplied with the IBM Engineering and Scientific Subroutine Library (ESSL).

which can be invoked by adding a compiler flag to the list (after FFLAGS=), e.g. -DFFTW: the same line may also require the location of any include files for the FFT solver, e.g. -I/usr/local/fftw/3.3.8/include/. The relevant library also has to be linked in when the executable is built: both the location of the library and the compiler flag to link the library can be added to the LFLAGS= line, e.g. LFLAGS= $(FFLAGS) -L/usr/local/fftw/3.3.8/lib/ -lfftw3.

DL_MESO_DPD can be compiled using the command make if the makefile is called Makefile, or if it has a custom name (e.g. Makefile-custom) by the command

- make -f Makefile-custom

The example makefiles will produce an executable with the name dpd.exe, which can be copied to the working directory (if necessary). The required input files (CONTROL and FIELD) will also need to be created in or copied into the same directory, as well as an optional CONFIG file to specify an initial configuration for a new simulation. export and REVIVE files from a previous run can be used for restarting a previous simulation: the number of processing units does not have to remain the same. The utility export_config can be used to convert the restart files into a CONFIG file for a new simulation.

If running in serial, the executable can just be run using the command:

- dpd.exe if running in Windows, or

- `./dpd.exe` if running in Unix-like operating systems,

while the parallel version requires a command to run N identical copies of the program on N processors, e.g.

- `mpirun -np` N `./dpd.exe`

and on distributed computers may need to be launched via a batch job script: please consult your machine administrator or documentation for further details.

If the **l_scr** directive has been included in the `CONTROL` file, the diagnostic output that is ordinarily written to the `OUTPUT` file will be sent to the machine and operating system's standard output, i.e. the screen or an automatically generated text file for batch jobs. This output can be redirected to a user-specified output file by modifying the command used to launch DL_MESO_DPD, e.g.

- `mpirun -np` N `./dpd.exe > output`

The maximum numbers of particles per process (`maxdim`), pairs of unbonded interactions (`maxpair`) and the initial values for the maximum number of particles in buffers (`maxbfbd`) and the maximum sizes of transfer buffers (`maxbuf`) are automatically set according to the total number of particles in the system and the number of processing units to be used for simulations. The value of `maxdim` can be increased by the user to allow for non-evenly distributed systems by setting a value for **densvar** in the `CONTROL` file. The values of `maxbfbd` and `maxbuf` are automatically adjusted during equilibration and for systems with many-body DPD interactions.

If using alternative many-body DPD interactions to the vapour-liquid example provided, the subroutines `manybody_potential` in `manybody_module.f90` and `conservative force` in `field_module.f90` should be modified by the user as necessary; the routine `local_density` should not be altered by the user but the function `weight_rho` can be changed if an alternative weighting function for calculating local densities is required. Additional bond, angle and dihedral types can also be added to the subroutines `bond_force`, `angle_force` and `dihedral_force` respectively in `bond_module.f90`, but this will also require changes to `scan_field` and `read_field` in `config_module.f90` to include a four-letter code for the bond/angle/dihedral type that can be read from the `FIELD` file. (Further details about these can be found in the Developer Manual.)

Appendix C

DL_MESO Utilities

DL_MESO includes a number of utility programs which are not directly needed for Lattice Boltzmann or DPD simulations but are useful both for producing files required as inputs for those calculations and to process output files for visualization and analysis. These may be found in the `/LBE/utility` and `/DPD/utility` directories.

Compilation can either be carried out individually or collectively using makefiles: each utility directory includes a makefile to compile all the utilities therein and the working directory `/WORK` includes one to compile both sets for use with the GUI[1]. The latter can be invoked using the command

- `make -f Makefile-utils`

Some further details on these utilities can be found in the `README` files in the source directories.

C.1 DL_MESO_LBE

All utilities for the LBE code can be run at the command line with optional arguments, e.g.

- `utility.exe [arguments]`

for Windows machines, or

- `./utility.exe [arguments]`

for machines running Unix, Linux or Mac (Mac OS X or macOS) operating systems. One command line argument all LBE utilities have in common is `-h` (for help), which will give a brief description of the utility and its available command line arguments before quitting. All utilities that work on input or output files should be run in the same directory as those files, which should have the default names for those types (`lbin.sys`, `lbout*.vts`, `lbout*.q` etc.).

lbeinitcreate

`lbeinitcreate` is a utility written in C++ to create initialisation files (`lbin.init`) to override the default initial conditions. This utility can add fluid drops to the system (either circular in 2D or spherical in 3D) and rectangular 'sources' of specified solute concentrations or temperature to a system.

If `c++` is the command for the available C++ compiler, the executable `init.exe` can be produced by typing

- `c++ -o init.exe lbeinitcreate.cpp`

[1] If using the GUI and the utilties are to be compiled manually or in their source directories, copies of the executables are required in the directory from which the GUI is to be launched, e.g. `/WORK`.

and run at the command line (`init.exe` or `./init.exe`).

A pre-existing `lbin.sys` file needs to exist in the directory where the utility is run, as this provides information on the dimensions and size of the simulation system, the numbers of fluids and initial and constant densities for each fluid, the number of solutes, whether or not a thermal lattice is included and the default initial velocity. This information is displayed on the screen when the utility is run: if no `lbin.sys` file can be found, an error message will be displayed and the utility will terminate.

If a single fluid is specified in the `lbin.sys` file with different initial and constant densities, the utility will ask for the number of drops to be added to the system: for each drop, the user will need to specify its radius, where its centre is located on the lattice grid and its density. If more than one fluid is specified in the `lbin.sys` file, the utility will then attempt to determine the continuous fluid for the system from the initial densities and, if necessary, ask the user to identify it. The utility will then ask for the number of drops to be added to the system: for each drop, the user will need to specify the fluid, its radius, where its centre is located on the lattice grid and its density. (Note that it is possible for a drop to extend beyond the grid boundaries if periodic boundaries are in use, but the drop centre must be within those boundaries.)

If any solutes are to be included, the utility will ask for the number of solute 'sources' (i.e. regions of constant solute concentration): for each source, it will then ask for the solute number, the required concentration, the location of one corner of the rectangular source and its extent in each dimension (which can extend beyond periodic boundaries). Similarly, if a temperature grid is included in the system, the utility will ask for the number of temperature 'sources', followed by the required temperature and the location of the corner and the extent of the source.

Once all of the above information is obtained, the utility will then create the `lbin.init` file, which specifies the grid points, velocities, fluid densities, solute concentrations and temperatures for any locations in the system that require non-default initial conditions.

lbeplot3dgather

`lbeplot3dgather` is a utility written in C++ to gather Plot3D output files produced by the parallel version of DL_MESO_LBE and produce a single structure file (`lbtout.xyz` or `lbtout.xy`) and a single set of solution files (`lbtout*.q`) for visualisation of the entire system.

If `c++` is the command for the available C++ compiler, the executable `plot3d.exe` can be produced by typing

- `c++ -o plot3d.exe lbeplot3dgather.cpp`

and either run at the command line or via the GUI under **Gather LBE Data**.

All `lbout*.xyz` and `lbout*.q` files should be copied to the directory including the executable (if necessary) before running, as well as the `lbout.info` file to give information on the sizes of integers and floating point numbers. No user input is required, although the utility will stop with an error message if no `lbout.info` file is available. No other error messages are produced, so care should be taken to ensure no solution files are missing.

lbevtkgather

`lbevtkgather` is a utility written in C++ to gather Structured Grid XML-formatted VTK output files produced by the parallel version of DL_MESO_LBE (`lbout*.vts`) and produce a set of linking files (`lbtout*.pvts`) for visualisation of the entire system.

If `c++` is the command for the available C++ compiler, the executable `vtk.exe` can be produced by typing

- `c++ -o vtk.exe lbevtkgather.cpp`

and either run at the command line or via the GUI under **Gather LBE Data**.

All `lbout*.vts` files should be copied to the directory including the executable (if necessary) before running, as well as the `lbout.info` and `lbout.ext` files to give information about the number of writing processors used for the simulation (i.e. the number of files per frame) and the extents of each piece.

The executable for this utility can be run with any of the following command line arguments:

- `-a`
  Create linking files for output files that contain all properties from the LBE simulation (fluid densities, fluid mass fractions, solute concentrations and temperatures)
- `-d`
  Create linking files for output files that contain a single fluid density
- `-f`
  Create linking files for output files that contain a single fluid mass fraction
- `-c`
  Create linking files for output files that contain a single solute concentration
- `-t`
  Create linking files for output files that just contain temperatures

If no command-line argument is specified, the utility will assume all properties are contained in the output files. (If the GUI is used, the level of data to link together can be selected using the pulldown list in the **Gather LBE Data** panel.) No other user input is required, but error messages will be produced if either of the files `lbout.info` and `lbout.ext` are missing. No other error messages are produced, so care should be taken to ensure no VTK files for the pieces are missing, particularly since these files are required for plotting as the linking files do not include the data.

lbedumpvtk

`lbedumpvtk` is a utility written in C++ to create an XML-formatted structured VTK file `lbdump.vts` from the `lbout.dump` restart file generated by DL_MESO_LBE. The resulting file can be used to visualise the system at the point when the restart data was written (e.g. when the simulation was terminated) and verify the simulation is proceeding as expected, even when no other output files are generated.

If `c++` is the command for the available C++ compiler, the executable `dump_to_vtk.exe` can be produced by typing

- `c++ -o dump_to_vtk.exe lbedumpvtk.cpp`

and either run at the command line or via the GUI under **Gather LBE Data**. No other files are needed to create the VTK file, as the `lbout.dump` file includes all required data for visualisation.

lbedumpinit

`lbedumpvtk` is a utility written in C++ to create an initialisation file `lbin.init` from the `lbout.dump` restart file generated by DL_MESO_LBE. The resulting file can be used to start a new simulation from the state of a previous one.

If `c++` is the command for the available C++ compiler, the executable `dump_to_init.exe` can be produced by typing

- `c++ -o dump_to_init.exe lbedumpinit.cpp`

and either run at the command line or via the GUI under **Gather LBE Data**. No other files are needed to create the `lbin.init` file, as the `lbout.dump` file includes all required data for determining the state of the simulation (i.e. macroscopic properties at each grid point).

C.2 DL_MESO_DPD

All utilities for the DPD code can be run at the command line with optional arguments, e.g.

- `utility.exe [arguments]`

for Windows machines, or

- `./utility.exe [arguments]`

for machines running Unix, Linux or Mac (Mac OS X or macOS) operating systems. One command line argument all DPD utilities have in common is `-h` (for help), which will give a brief description of the utility and its available command line arguments before quitting. All utilities that work on input or output files should be run in the same directory as those files, which should have the default names for those types (`CONTROL`, `FIELD`, `export`, `HISTORY` etc.).

convert-input

`convert-input` is a utility written in C++ to read DPD input files created for earlier versions of DL_MESO (up to version 2.4) and create `CONTROL` and `FIELD` files formatted in the style for versions 2.7 and later.

This utility can be compiled to produce the executable `convert.exe` with the command

- `c++ -o convert.exe convert-input.cpp`

if `c++` is the command for the available C++ compiler.

This utility can be run with any (or all) of these optional command line arguments:

- `-c [CONTROL]`
  Uses a `CONTROL` file as input called `[CONTROL]`
- `-f [FIELD]`
  Uses a `FIELD` file as input called `[FIELD]`
- `-m [MOLECULE]`
  Uses a `MOLECULE` file as input called `[MOLECULE]`
- `-v`
  Verbose option: writes out data read from input files onto screen or standard output

Note that if the default names are used for input, the old `CONTROL` and `FIELD` files are renamed after being read to prevent them being overwritten with the new versions of those files. (The `MOLECULE` file is no longer required and therefore does not need to be renamed.)

molecule-generate

`molecule-generate` is a utility written in C++ to generate the input files required for modelling particles in DPD simulations that are bonded together, i.e. molecules. A random flight generation system is used to generate the coordinates of bonded beads – which can form branched molecule chains – a constant distance

apart within a cube of a size specified by the user, which will be used by DL_MESO_DPD to insert the molecule into the system.

This utility can be compiled to produce the executable `molecule.exe` with the command

- `c++ -o molecule.exe molecule-generate.cpp`

if `c++` is the command for the available C++ compiler. This utility can be run from the command line or via the GUI in **Set DPD Molecules** (which runs the utility in a new command line/shell window).

This utility can be run with any (or all) of these optional command line arguments:

- `-p`
  Write data to a separate file (`molecule`) instead of a `FIELD` file: this is used by the GUI to insert molecular data into the `FIELD` file it generates
- `-s` n `[SPEC1] [SPEC2] ... [SPEC`n`]`
  Define n particle species and provide their names (`[SPEC1]`, `[SPEC2]` to `[SPEC`n`]`)

If a `FIELD` file exists in the same directory as the executable with species data and the `-s` command-line option is not used, the number of species and their names will be read from it; otherwise the user will be asked to enter this information and this will be written to a new `FIELD` file (if the separate file is not specified). The user will then be asked for the number of molecules required, the numbers of bond, angles and dihedrals and their types and parameters.

For each molecule, the user is asked for its name, the number to be included in the system and whether or not isomers of the molecule can be included. The side length for the cube inside which the molecule will fit is then required, followed by the bond length, the number of molecule chains and the number of particles for each chain. If the chain in question is not the first (primary) chain, the user will also be asked for a pre-existing bead number as the starting point for the chain.

After this, the default species for the beads in the molecule will be requested: the user will then be asked enter the bead numbers for each of the other species (0 can be entered to finish specifying bead numbers). If more than one bond type is to be included, the user will be asked to select the default bond type and then select the bonds that are of different types by typing the index bead number (and optionally the destination bead if more than one is available). Bond angles and/or dihedrals can also be selected by typing in the index bead number and then selecting the required bead triple or quadruple if more than one is available.

The molecules will either be appended to the `FIELD` file in the correct format (see Section 10.1 for more details) with positions for the beads relative to each molecule's centre of mass, or the data will be written to the separate `molecule` file. Note that the `FIELD` file will not be quite complete after running this utility: data for unbonded interactions and external force fields may be required (if it is created from scratch using the utility) and a **close** directive will be required at the end.

export_config

`export_config` is a utility written in Fortran to produce a configuration file in DL_POLY format (`CONFIG`) from DL_MESO_DPD restart files (`export`), which can be used as a starting point for new simulations. Since a limited amount of data is included in restart files, the `FIELD` file for the simulation is also needed to provide some additional information.

The source code for this utility, `export_config.F90`, can be used for `export` files created by both the serial and parallel versions of DL_MESO_DPD. If the utility is to be run on a different machine to the one used for DPD calculations, care should be taken to ensure the utility can read files with the same endianness by using a compiler flag to force big or little endianness.

If the available Fortran compiler is invoked by the command `f90`, the executable `export_config.exe` can be produced by typing

- `f90 -o export_config.exe export_config.F90`

and either run at the command line or by using the GUI in **Process DPD Data** after entering the number of processes used in the required field and selecting the required `CONFIG` file key in the pulldown list.

The executable for this utility can be run with any of the following command line arguments:

- `-k` i
  Set `CONFIG` file key, `levcfg`, to i (0 = positions only, 1 = positions and velocities, 2 = positions, velocities and forces)
- `-s`
  Write particles to `CONFIG` file after sorting them in order by particle number
- `-u`
  Write particles to `CONFIG` file without sorting them

If no command-line argument is given, the utility will ask the user to type in the `CONFIG` file key and assume the particles are not sorted before writing to the file. (Note that particles in `export` files produced by the serial version of DL_MESO will be automatically in numerical order.)

export_image_vtf

`export_image_vtf` is a utility written in Fortran to produce a VTF format trajectory file (`export.vtf`) from DL_MESO_DPD restart files (`export`) that can be visualized in VMD [56] to give a snapshot of the last simulation timestep. Since a limited amount of data is included in restart files, the `FIELD` file for the simulation is needed to provide some additional information.

The source code for this utility, `export_image_vtf.F90`, can be used for `export` created by both the serial and parallel versions of DL_MESO_DPD: if the utility is to be run on a different machine to the one used for DPD calculations, care should be taken to ensure the utility can read files with the same endianness by using a compiler flag to force big or little endianness.

If the available Fortran compiler is invoked by the command `f90`, the executable `export_image_vtf.exe` can be produced by typing

- `f90 -o export_image_vtf.exe export_image_vtf.F90`

and either run at the command line (`export_image_vtf.exe` or `./export_image_vtf.exe`) or by using the GUI in **Process DPD Data**. The optional commend-line argument `-s` can be used to sort the particles into order of global particle index. So long as both the `export` and `FIELD` files are available in the same directory, the utility will produce the VTF trajectory file without any prompting.

export_image_xml

`export_image_xml` is a utility written in Fortran to produce a trajectory file in GALAMOST [147] XML format (`export.xml`) from DL_MESO_DPD restart files (`export`) that can be visualized in OVITO [122] to give a snapshot of the last simulation timestep. Since a limited amount of data is included in restart files, the `FIELD` file for the simulation is needed to provide some additional information.

The source code for this utility, `export_image_xml.F90`, can be used for `export` created by both the serial and parallel versions of DL_MESO_DPD: if the utility is to be run on a different machine to the one used for

DPD calculations, care should be taken to ensure the utility can read files with the same endianness by using a compiler flag to force big or little endianness.

If the available Fortran compiler is invoked by the command `f90`, the executable `export_image_xml.exe` can be produced by typing

- `f90 -o export_image_xml.exe export_image_xml.F90`

and either run at the command line (`export_image_xml.exe` or `./export_image_xml.exe`) or by using the GUI in **Process DPD Data**. So long as both the `export` and `FIELD` files are available in the same directory, the utility will produce the XML trajectory file without any prompting.

history_config

`history_config` is a utility written in Fortran to take DL_MESO_DPD trajectory output data files (`HISTORY`) and produce a configuration file in DL_POLY format (`CONFIG`) from them, which can be used as a starting point for new simulations (including restarting simulations on different numbers of processes).

The source code for this utility, `history_config.F90`, reads `HISTORY` files generated by the serial and parallel versions of DL_MESO_DPD respectively: because `HISTORY` files include a number in their headers as an endianness check, the utility should be able to read these files created on different machines. The utility outputs data for every particle for the selected timestep: the maximum level of data (particle positions, velocities and forces) is fixed by the `HISTORY` file but the user can choose up to that particular level to write to the `CONFIG` file.

If `f90` is the command for the available Fortran compiler, the executable `history_config.exe` can be produced by typing

- `f90 -o history_config.exe history_config.F90`

and either run at the command line or via the GUI in **Process DPD Data**.

The executable for this utility can be run with any of the following command line arguments:

- `-k` i
  Set `CONFIG` file key, `levcfg`, to i (0 = positions only, 1 = positions and velocities, 2 = positions, velocities and forces)
- `-f` i
  Use trajectory data in frame i of `HISTORY` file for configuration in `CONFIG` file
- `-l` i
  Use last frame of trajectory data in `HISTORY` file for configuration in `CONFIG` file
- `-s`
  Write particles to `CONFIG` file after sorting them in order by particle number
- `-u`
  Write particles to `CONFIG` file without sorting them

If no command-line argument is given or the `CONFIG` file key or trajectory frame number are out of range for the given `HISTORY` file, the utility will ask the user to type in valid numbers for the `CONFIG` file key (if more than positions are available) and trajectory frame number (unless the last one is selected at the command line), as well as assume that the particles are not sorted when writing to the `CONFIG` file.

traject_vtf

The Fortran utility `traject_vtf` reads in `HISTORY` output data files generated by DL_MESO_DPD and produces a VTF format trajectory file (`traject.vtf`) that can visualize the simulation using VMD [56], such that snapshots at the recorded timesteps and animations can be produced. An option exists to produce separate VTF format trajectory files for unbonded particles (`traject_bead.vtf`) and bonded particles in molecules (`traject_mole.vtf`), and another option exists to write the structure and coordinates into separate files (`traject.vsf` and `traject.vcf` respectively).

The source code for this utility, `traject_vtf.F90`, reads `HISTORY` files generated by the serial and parallel versions of DL_MESO_DPD respectively: because `HISTORY` files include a number in their headers as an endianness check, the utility should be able to read these files created on different machines. The utility outputs every particle for all recorded timesteps, including bond data for particles in molecules represented as residues.

If `f90` is the command for the available Fortran compiler, the executable `traject_vtf.exe` can be produced by typing

- `f90 -o traject_vtf.exe traject_vtf.F90`

and either run at the command line or via the GUI in **Process DPD Data**.

The executable for this utility can be run with the following optional command line arguments:

- `-b`
  Write trajectory data to separate files for bonded and unbonded particles
- `-s`
  Write particles to `traject.vtf` file after sorting them in order by particle number
- `-u`
  Write particles to `traject.vtf` file without sorting them
- `-sc`
  Separate structure and coordinates into `traject.vsf` and `traject.vcf` files respectively (option can be used with `-b`)

By default, a single trajectory file will be written and the particles will not be sorted into order of global index number.

traject_selected_vtf

The Fortran utility `traject_selected_vtf` works in a similar fashion to `traject_vtf` but allows the user to select which particles and timesteps to output to the trajectory file. If `f90` is the command for the available Fortran compiler, the executable `trajects_vtf.exe` can be produced by typing

- `f90 -o trajects_vtf.exe traject_selected_vtf.F90`

and run at the command line (note that it cannot be invoked using the GUI). The executable for this utility can be run with the following optional command line arguments:

- `-s`
  Write particles to `traject.vtf` file after sorting them in order by particle number
- `-u`
  Write particles to `traject.vtf` file without sorting them

- `-sc`
  Separate structure and coordinates into `traject.vsf` and `traject.vcf` files respectively

By default the particles will not be sorted into order of global index number. Beyond these, the user will be shown information about the contents of the `HISTORY` file and asked to type in what is required in the `traject.vtf` file.

The user will then need to choose which particles are to be written to the VTF file (or VSF and VCF files): the utility will first show the user how many particles (total and unbonded) are in each frame of the `HISTORY` file, as well as the names and numbers of particle species and molecule types. The user will then be asked to select one of the following options to include in the file(s):

1. All particles
2. A contiguous range of particles based on global particle indices
3. All particles belonging to particular species and molecule types
4. Individual particle numbers
5. Individual molecule numbers

and based on this choice, further questions will be asked to select the range of particle indices, species or molecule types, particle numbers or molecule numbers respectively. The user will then be shown the number of available trajectory frames in the `HISTORY` file and asked to select which one to start with, which one to end with and the frequency to write to the output file(s) (i.e. whether or not to skip over frames).

traject_xml

The Fortran utility `traject_xml` reads in `HISTORY` output data files generated by DL_MESO_DPD and produces a series of numbered trajectory files in GALAMOST [147] XML format (`traject_*.xml`) that can be used to visualize the simulation using OVITO [122], such that snapshots at the recorded timesteps and animations can be produced with the correct bond connectivity if molecules are present.

The source code for this utility, `traject_xml.F90`, reads `HISTORY` files generated by the serial and parallel versions of DL_MESO_DPD respectively: because `HISTORY` files include a number in their headers as an endianness check, the utility should be able to read these files created on different machines. The utility outputs every particle for all recorded timesteps, including bond data for particles in molecules. (Each molecule type is used to identify the types of bonds between particles.)

If `f90` is the command for the available Fortran compiler, the executable `traject_xml.exe` can be produced by typing

- `f90 -o traject_xml.exe traject_xml.F90`

and either run at the command line or via the GUI in **Process DPD Data**.

No command line arguments are required to run this utility. A series of trajectory files will be written: because global index numbers cannot be explicitly assigned in this format, the particles' data are sorted by global index before being written to the files .If velocities are included in the `HISTORY` file, these will also be written to the trajectory files.

traject_selected_xml

The Fortran utility `traject_selected_xml` works in a similar fashion to `traject_xml` but allows the user to select which particles and timesteps to output to the trajectory files. If `f90` is the command for the available Fortran compiler, the executable `trajects_xml.exe` can be produced by typing

- `f90 -o trajects_xml.exe traject_selected_xml.F90`

and run at the command line (note that it cannot be invoked using the GUI). The user will be shown information about the contents of the `HISTORY` file and asked to type in what is required in the `traject.vtf` file.

The user will then need to choose which particles are to be written to the XML files: the utility will first show the user how many particles (total and unbonded) are in each frame of the `HISTORY` file, as well as the names and numbers of particle species and molecule types. The user will then be asked to select one of the following options to include in the file(s):

1. All particles
2. A contiguous range of particles based on global particle indices
3. All particles belonging to particular species and molecule types
4. Individual particle numbers
5. Individual molecule numbers

and based on this choice, further questions will be asked to select the range of particle indices, species or molecule types, particle numbers or molecule numbers respectively. The user will then be shown the number of available trajectory frames in the `HISTORY` file and asked to select which one to start with, which one to end with and the frequency to write to the output file(s) (i.e. whether or not to skip over frames).

isosurfaces

The Fortran utility `isosurfaces` reads in `HISTORY` output data files generated by DL_MESO_DPD and produces grid-based density maps at each recorded timestep for a specified species in Legacy VTK format for visualization of isosurfaces. It also calculates the second moment of the isosurface normal distribution, whose eigenvalues can be used to determine the mesophase for the system.

The source code for this utility, `isosurfaces.F90`, can read `HISTORY` files generated by the serial and parallel versions of DL_MESO_DPD respectively: an endianness check to their headers will allow the utility to work out which endianness to use for reading them. An executable `isosurfaces.exe` can be created by typing

- `f90 -o isosurfaces.exe isosurfaces.F90`

if `f90` is the command for the available Fortran compiler. This can be run either at the command line or via the GUI in **Process DPD Data**.

The executable for this utility can be run with the following command line arguments:

- `-b [SPECIES]`
  Use particle species `[SPECIES]` to create density maps (either name or number based on its order in the `FIELD` file)
- `-p` f
  Set spacing between grid points to f (overriding default value of 0.25)
- `-s` f
  Set Gaussian standard deviation σ to f (overriding default value of 0.4)
- `-sf` i
  Start collecting and writing data from frame i of `HISTORY` file (overriding default value of 1)

- `-sl` i
  Finish collecting and writing data at frame i of `HISTORY` file (overriding default value of last available frame)

- `-tf` i
  Set frequency of frames used from `HISTORY` file to i (overriding default value of 1)

For instance, if isosurfaces of species `TAIL` are required with a target grid spacing of 0.2 and a Gaussian standard deviation of 0.5 between frames 4 and 10 every 2 frames, the utility can be launched using one of the following commands:

- `isosurfaces.exe -b TAIL -p 0.2 -s 0.5 -sf 4 -sl 10 -tf 2`

- `./isosurfaces.exe -b TAIL -p 0.2 -s 0.5 -sf 4 -sl 10 -tf 2`

Note that if no species is specified at the command line, the user will be asked to choose one based on the contents of the `HISTORY` file.

For each recorded timestep, the volume of every particle is smeared using a Gaussian function of standard deviation σ:

$$f(\vec{r}) = \frac{1}{(2\pi\sigma^2)^{\frac{3}{2}}} \exp\left(-\frac{|\vec{r}-\vec{r}_i|^2}{2\sigma^2}\right)$$

where $\vec{r}_i$ is the position of particle i. All points on a regular orthogonal grid within a distance of 3σ from the particle position are assigned contributions from this smearing function. The resulting totals for all particles of the specified species are subsequently written to Legacy VTK files named `density_*.vtk`. The densities can then be used to construct the isosurface normal distribution $p(\vec{n})$, using the mean value of density over the system as the threshold for isosurfaces. Its second moment

$$\mathbf{M} = \int \vec{n}\vec{n}p(\vec{n})d\vec{n}$$

gives an indication of how the particles in the species are arranged in the system. The three eigenvalues of the second moment (μ_1, μ_2, μ_3) can be used as mesophase order parameters[97, 133]: $\mu_1 \approx \mu_2 \approx \mu_3$ indicates an isotropic mesophase, $\mu_1 \ll \mu_2, \mu_3$ indicates a hexagonal mesophase and $\mu_1, \mu_2 \ll \mu_3$ indicates a lamellar mesophase. The eigenvalues are written to a file named `moment`, which contains columns for the time and the three eigenvalues in numerical order: this file can be imported into graph plotting software to display how the system mesophase changes over time.

radius

The Fortran utility `radius` reads in `HISTORY` output data files generated by DL_MESO_DPD and calculates the end-to-end distances and radii of gyration at each recorded timestep for all molecules in the system, as well as finding the time-averaged distributions of end-to-end distances for each molecule type.

The source code for this utility, `radius.F90`, can read both `HISTORY` and `HISTORY*` files generated by the serial and parallel versions of DL_MESO_DPD respectively. If `f90` is the command for the available Fortran compiler, the executable `radius.exe` can be produced by typing

- `f90 -o radius.exe radius.F90`

and run either at the command line or via the GUI in **Process DPD Data**.

The executable for this utility can be run with the following command line arguments:

- `-c` f
  Set maximum end-to-end distance for distribution to f (overriding default value of 2.0)

- `-d` f
  Set histogram spacing for distribution to f (overriding default value of 0.05)

- `-sf` i
  Start collecting and writing data from frame i of `HISTORY` file (overriding default value of 1)

- `-sl` i
  Finish collecting and writing data at frame i of `HISTORY` file (overriding default value of last available frame)

- `-tf` i
  Set frequency of frames used from `HISTORY` file to i (overriding default value of 1)

Note that if the `HISTORY` file includes no molecules, the utility will stop with an error message.

For each recorded timestep and all molecules, the end-to-end distance along the main molecular branch and the radius of gyration

$$R_g^2 = \frac{1}{N}\sum_i^N (\vec{r}_i - \vec{r}_{mean})^2$$

are calculated, where N is the number of particles in the molecule and $\vec{r}_{mean} = \frac{\sum_i^N m_i \vec{r}_i}{\sum_i^N m_i}$ is the centre-of-mass for the molecule. Files named `radius_*` are produced for each molecule type: these text files contain columns for the time, root mean squared end-to-end distance, the mean squared end-to-end distance and root mean squared radius of gyration over all molecules of the specified type, which can be plotted using graph plotting software.

A single file named `MOLDIST` is also produced to give the time-averaged distributions of end-to-end distance for every molecule type, normalised to give $\int_0^\infty 4\pi r^2 dr\, g(r) = 1$. The file starts with two lines, the first giving the name of the simulation and the second with the number of timesteps used and the number of distance divisions used (the number of shells). The distributions for each molecule type are then given, starting with a line giving the molecule name and followed by columns with the radius r (the mid-point for each shell) and the distribution $g(r)$: each type is separated by two blank lines.

dipole

The Fortran utility `dipole` reads in `HISTORY` output data files generated by DL_MESO_DPD and calculates the total dipole moments for each molecule type, autocorrelation functions of dipole moments and (optionally) their Fourier transforms.

The source code for this utility, `dipole.F90`, can read `HISTORY` files generated by the serial and parallel versions of DL_MESO_DPD respectively, regardless of which endianness is used. If `f90` is the command for the available Fortran compiler, the executable `dipole.exe` can be produced by typing

- `f90 -o dipole.exe dipole.F90`

and run either at the command line or via the GUI in **Process DPD Data**.

The executable for this utility can be run with the following command line arguments:

- `-n` i
  Set number of bins to calculate dipole autocorrelation functions (DAFs) to i

- `-fft`
  Calculate Fourier Transforms of dipole autocorrelation functions

- `-fc` i
  Set number of bins for DAFs to i (overriding default value of maximum between 500 and double the number of bins for DAF calculations)

- `-sf` i
  Start collecting and writing data from frame i of `HISTORY` file (overriding default value of 1)

- `-sl` i
  Finish collecting and writing data at frame i of `HISTORY` file (overriding default value of last available frame)

- `-tf` i
  Set frequency of frames used from `HISTORY` file to i (overriding default value of 1)

If no value for the number of bins for dipole autocorrelation functions is given in the command line or it exceeds the number of frames in the `HISTORY` file (or the number used based on starting/finishing frames and frequency), the user will be asked to type one in. Note that no Fourier transforms for the dipole autocorrelation functions will be calculated unless the `-fft` command line argument is used.

For each recorded timestep, the total dipole moment for each molecule is calculated using the formula $\vec{p} = \sum_i q_i \vec{r}_i$, where q_i is the charge on particle i (obtained from the `HISTORY` file) and $\vec{r}_i$ is the position of that particle with adjustments made for periodic boundary conditions where necessary. These dipole moments are summed up for each molecule type to give the total dipole moment $\vec{P}$. Files named `dipole_*` are produced for each molecule type: these text files contain columns for the time, x-, y- and z-components of the dipole moment, the squared dipole moment P^2 and the ratio of the squared dipole moment P^2 to volume ($\frac{P^2}{V}$).

The dipole autocorrelation function for each molecule type is calculated as $C(t) = \left\langle \vec{P}(0) \cdot \vec{P}(t) \right\rangle$ over the number of time steps given by the user, ensemble averaging across all possible samples from the given trajectory data. A single file called `DIPOLEDAT` is produced, containing the DAFs for all available molecule types. Starting with two lines – the first with the simulation name, the second with the total number of timesteps used and the number of time steps used for DAFs – the names and data for each molecule type is given, the latter in columns for time t, the absolute value of the ensemble averaged DAF $C(t)$ and the value scaled with the value at $t = 0$ (i.e. $Z(t) = \frac{C(t)}{C(0)}$). The data for each molecule type are separated by two blank lines.

If the Fourier transforms are requested, this appears in an additional file called `DIPOLEFFT`, which is similarly formatted to `DIPOLEDAT` apart from the data for each molecule type, which are the frequency k and the real and imaginary terms of the Fourier transform $S(k)$ (i.e. $\Re(S(k))$ and $\Im(S(k))$).

rdf

`rdf` is a utility written in Fortran that can read in `HISTORY` output data files generated by DL_MESO_DPD and determine radial distribution functions (RDFs) between all pairs of particle species (including self-interactions).

The source code for this utility `rdf.F90` can read `HISTORY` files generated by the serial and parallel versions of DL_MESO_DPD respectively. It can exploit OpenMP multithreading to speed up RDF calculations: the source code includes directives to use different blocks of code depending on whether or not it is compiled with OpenMP, which require preprocessing. (The source code filename extension `.F90` should automatically invoke the compiler's C preprocessor.)

If `f90` is the command for the available Fortran compiler, the executable `rdf.exe` can be produced by typing either

- `f90 -o rdf.exe rdf.F90`

for the serial (single thread) version, or by typing

- `f90 -o rdf.exe -openmp rdf.F90`

for the OpenMP multithreaded version, substituting `-openmp` with the required compiler flag for invoking OpenMP (e.g. `-fopenmp` for `gfortran`).

The utility can either be run at the command line or via the GUI in **Process DPD data**. A number of command line arguments can be used after the command:

- `-c` f
  Set maximum distance between pairs of particles to f (overriding default value of 2.0)
- `-d` f
  Set histogram spacing for radial distribution function calculations to f (overriding default value of 0.05)
- `-fft`
  Calculate Fourier Transforms of radial distribution functions (structure factors)
- `-fc` i
  Set number of bins for structure factors to i (overriding default value of maximum between 500 and double the number of bins for RDF calculations)
- `-sf` i
  Start collecting and writing data from frame i of `HISTORY` file (overriding default value of 1)
- `-sl` i
  Finish collecting and writing data at frame i of `HISTORY` file (overriding default value of last available frame)
- `-tf` i
  Set frequency of frames used from `HISTORY` file to i (overriding default value of 1)

Note that no Fourier transforms for the radial distribution functions will be calculated unless the `-fft` command line argument is used.

At each recorded timestep, linked-cell lists for the particles are created and used to determine the particle-particle distances between pairs within the maximum distance r_c. These distances are used to increase counters for the appropriate histogram bins (of width Δr) for each species pair. The sums of these histograms are averaged over time to give $n(r)$ (the mean number of particles in the bin at distance r) and the radial distribution function is given as

$$g(r) = \frac{n(r)}{4\pi r^2 \Delta r \rho}$$

where ρ is the mean particle density. The Fourier transform of $g(r)$ gives the structure factor:

$$S(k) = 1 + \frac{4\pi\rho}{k} \int_0^\infty (g(r) - 1) r \sin(kr) dr$$

which is a property that can be measured experimentally using e.g. X-ray diffraction.

A single file `RDFDAT` containing all radial distribution function data is produced. Starting with two lines – the first giving the simulation name, the second giving the number of timesteps and the number of histogram bins used – the names and the data for each species pair are given, the latter in columns for the radius r (the mid-point for each shell), the RDF $g(r)$ and the sum of $g(r)$ (the average number of particles of one type within r around a particle of the other). The data for each species pair are separated by two blank lines and an additional data set of the same form is given for all particles regardless of species type at the end of the file.

If the Fourier transforms are requested, an additional `RDFFFT` file is created. This file has a similar format to `RDFDAT`, except that the data is given in two columns: one for the frequency k and the other for $S(k)$ (the Fourier transform of $g(r) - 1$).

rdfmol

`rdfmol` is a utility written in Fortran that can read in `HISTORY` output data files generated by DL_MESO_DPD and determine radial distribution functions (RDFs) between all pairs of molecules by type (including self-interactions).

The source code for this utility `rdfmol.F90` can read `HISTORY` files generated by the serial and parallel versions of DL_MESO_DPD. It can exploit OpenMP multithreading to speed up calculations: the source code includes directives to use different blocks of code depending on whether or not it is compiled with OpenMP.

If `f90` is the command for the available Fortran compiler, the executable `rdfmol.exe` can be produced by typing either

- `f90 -o rdfmol.exe rdfmol.F90`

for the serial (single thread) version, or by typing

- `f90 -o rdfmol.exe -openmp rdfmol.F90`

for the OpenMP multithreaded version, substituting `-openmp` with the required compiler flag for invoking OpenMP.

The utility can either be run at the command line or via the GUI in **Process DPD data**. A number of command line arguments can be used after the command:

- `-c` f
  Set maximum distance between pairs of particles to f (overriding default value of 2.0)
- `-d` f
  Set histogram spacing for radial distribution function calculations to f (overriding default value of 0.05)
- `-fft`
  Calculate Fourier Transforms of radial distribution functions (structure factors)
- `-fc` i
  Set number of bins for DAFs to i (overriding default value of maximum between 500 and double the number of bins for RDF calculations)
- `-sf` i
  Start collecting and writing data from frame i of `HISTORY` file (overriding default value of 1)
- `-sl` i
  Finish collecting and writing data at frame i of `HISTORY` file (overriding default value of last available frame)
- `-tf` i
  Set frequency of frames used from `HISTORY` file to i (overriding default value of 1)

Note that no Fourier transforms for the radial distribution functions will be calculated unless the `-fft` command line argument is used.

At each recorded timestep, the centres of mass for each molecule are determined (taking periodic or shearing boundaries into account) and these coordinates are assigned to linked-cell lists, which are used to determine the distances between pairs of molecules within the maximum distance r_c. These distances are used to fill histograms with bin sizes of Δr and are later time-averaged and divided by the shell volumes (as for standard RDFs) to give the radial distribution functions between molecule types. Fourier transforms of molecular RDFs give structure factors for the molecules.

The standard output from this utility is the file `RDFMOLDAT`: this is formatted in a similar manner to `RDFDAT` except the RDF data is given by molecule type pairs. If Fourier transforms are requested, these are given in the file `RDFMOLFFT` and formatted similarly to `RDFFFT`.

local

`local` is a utility written in Fortran that can read in `HISTORY` output data files generated by DL_MESO_DPD and produce series of Legacy VTK format files containing statistical properties – number of beads, density, compositions per particle and molecule types, temperature and mean velocity, local stress tensors – in cuboidal subdivisions of the simulation volume for plotting and/or visualization.

The source code for this utility, `local.F90`, can read both `HISTORY` and `HISTORY*` files generated by the serial and parallel versions of DL_MESO_DPD respectively: if the utility is to be run on a different machine to the one used for DPD calculations, care should be taken to ensure the utility can read files with the same endianness by using a compiler flag to force big or little endianness. If `f90` is the command for the available Fortran compiler, the executable `local.exe` can be produced by typing

- `f90 -o local.exe local.F90`

and either run at the command line or via the GUI in **Process DPD Data** after entering the number of divisions required in each dimension.

This utility can be run with the following command line arguments specifying the number of divisions in each dimension:

- `-nx` i
  Set number of system divisions in x-dimension to i
- `-ny` i
  Set number of system divisions in y-dimension to i
- `-nz` i
  Set number of system divisions in z-dimension to i
- `-av` i
  Only write time-averaged data (suppress files for each frame)
- `-sf` i
  Start collecting and writing data from frame i of `HISTORY` file (overriding default value of 1)
- `-sl` i
  Finish collecting and writing data at frame i of `HISTORY` file (overriding default value of last available frame)
- `-tf` i
  Set frequency of frames used from `HISTORY` file to i (overriding default value of 1)

If these values are not set at the command line, the user will be asked to enter these values.

Unless the `-av` option is selected, files named `local_*.vtk` are produced for all the specified time steps after equilibration containing the data for each cuboidal cell. The level of data available in the `HISTORY` files (i.e. `keytrj`: 0 = particle positions, 1 = particle positions and velocities, 2 = particle positions, velocities and forces) dictates which of the following properties are output:

- the number of unfrozen beads
- densities for each bead species

- volume fractions for bead species
- volume fractions for molecule types (including a 'type' for all unbonded beads)
- the mean velocity for all unfrozen beads
- overall temperature
- partial temperatures for each dimension (i.e. for dimension α, $T_\alpha = \frac{\sum_i m_i v_{i,\alpha}^2}{N}$)
- local pressure tensors (based on Method of Planes[128])

An additional file, `averages.vtk`, is also produced with time-averaged values for the species densities, velocities, overall and partial temperatures, and local pressure tensor in each cuboidal cell based on the starting/finishing frames and frequency selected.

The scalar properties (including compositions) may be considered to act across the entire volumes of the cells, while the velocities and pressure tensors are representative of the cell centres.

widom_insertion

`widom_insertion` is a utility written in Fortran that can carry out Widom insertions to determine excess chemical potentials[139, 140]. This utility reads in trajectories from `HISTORY` output data files generated by DL_MESO_DPD, along with relevant simulation and interaction information from `CONTROL` and `FIELD` files, and randomly inserts a user-selected particle or molecule at different positions (and orientations) in the simulation box. The change in potential energy due to each insertion of particle or molecule i (ΔU_i) is measured and an ensemble average can be used to calculate the excess chemical potential, i.e.

$$\mu_i^{ex} = -k_B T \ln \left\langle \exp\left(-\frac{\Delta U_i}{k_B T}\right)\right\rangle$$

for a constant volume ensemble, or

$$\mu_i^{ex} = -k_B T \ln \left(\frac{\left\langle V \exp\left(-\frac{\Delta U_i}{k_B T}\right)\right\rangle}{\langle V \rangle} \right)$$

for simulations with varying volume (constant pressure, surface area or surface tension).

The source code for this utility, `widom_insertion.f90`, can read `HISTORY` files generated by the serial and parallel versions of DL_MESO_DPD respectively: endianness checks in the utility mean it should be possible to run the utility on a different machine to that used to generate the trajectory data. It can exploit OpenMP multithreading to speed up calculations: the source code includes directives to use different blocks of code depending on whether or not it is compiled with OpenMP.

If `f90` is the command for the available Fortran compiler, the executable `widom.exe` can be produced by typing

- `f90 -o widom.exe widom_insertion.F90`

for the serial (single thread) version, or by typing

- `f90 -o widom.exe -openmp widom_insertion.F90`

for the OpenMP multithreaded version, substituting `-openmp` with the required compiler flag for invoking OpenMP (e.g. `-fopenmp` for `gfortran`). The utility can be run either at the command line or via the GUI in **Process DPD Data**.

This utility can be run with the following command line arguments:

- `-p [SPECIES]`
  Insert a single particle of species `[SPECIES]` for each trial insertion
- `-m [MOLE]`
  Insert a molecule of type `[MOLE]` for each trial insertion
- `-rm`
  Use a randomly chosen molecule in each trajectory frame as a template for molecule trial insertion (overriding default of configuration given in `FIELD` file)
- `-n` i
  Set number of trial insertions per trajectory frame to i
- `-sf` i
  Start accumulating statistics on trial insertions from trajectory frame i (overriding default of 1)
- `-sl` i
  Stop accumulating statistics on trial insertions at trajectory frame i (overriding default of last available frame)
- `-r` i
  Set random number generator seed to i (only if `RNDSEED` file is unavailable)

The particle or molecule type and the number of trial insertions per frame are essential to carry out Widom insertions: if these are not included in the command line, the user will be asked to type in these data.

The utility will produce two files: a file called `RNDSEED` giving the final state of the random number generator (which can be used as the initial state for future calculations), and one called `CHEMPOT_*`, ending with the name of the bead species or molecule being inserted. The latter plottable file contains five columns with the time (in DPD units), the 'instantaneous' block-averaged excess chemical potential for the trajectory frame and its standard deviation, the time-averaged excess chemical potential and its standard deviation. The utility will also output the same information for each trajectory timestep to the screen or standard output.

Appendix D

Lattice schemes

This appendix gives details of the various lattice schemes implemented in the LBE code of DL_MESO: D2Q9, D3Q15, D3Q19 and D3Q27. These include details of the defined lattice vectors, weight factors used in local equilibrium distribution functions (including those for Swift free-energy interactions), the transformation matrices, moments, relaxation frequencies and forcing terms used for MRT and cascaded LBE collisions.

D.1 D2Q9

Speed vectors

i	$e_{i,x}$	$e_{i,y}$
0	0	0
1	-1	1
2	-1	0
3	-1	-1
4	0	-1
5	1	-1
6	1	0
7	1	1
8	0	1

Weight factors

i	w_i
0	$\frac{4}{9}$
2,4,6,8	$\frac{1}{9}$
1,3,5,7	$\frac{1}{36}$

Weight factors for Swift free-energy model

i	w_i	w_i^{00}	γ_i	δ_i	w_i^p	w_i^t	w_i^{xx}	w_i^{yy}	w_i^{zz}	w_i^{xy}	w_i^{xz}	w_i^{yz}
0	$\frac{4}{3}$	1	0	$-\frac{21}{8}$	$-\frac{5}{3}$	$-\frac{5}{3}$	$-\frac{1}{6}$	$-\frac{1}{6}$	0	0	0	0
1,5	$\frac{1}{12}$	0	$\frac{3}{2}$	$-\frac{3}{2}$	$\frac{1}{12}$	$\frac{1}{12}$	$-\frac{1}{24}$	$-\frac{1}{24}$	0	$-\frac{1}{4}$	0	0
2,6	$\frac{1}{3}$	0	$\frac{3}{2}$	$-\frac{3}{2}$	$\frac{1}{3}$	$\frac{1}{3}$	$\frac{1}{3}$	$-\frac{1}{6}$	0	0	0	0
3,7	$\frac{1}{12}$	0	$\frac{3}{2}$	$-\frac{3}{2}$	$\frac{1}{12}$	$\frac{1}{12}$	$-\frac{1}{24}$	$-\frac{1}{24}$	0	$\frac{1}{4}$	0	0
4,8	$\frac{1}{3}$	0	$\frac{3}{2}$	$-\frac{3}{2}$	$\frac{1}{3}$	$\frac{1}{3}$	$-\frac{1}{6}$	$\frac{1}{3}$	0	0	0	0

Multiple relaxation time scheme

Definition of transformation matrix based on [68]:

$$\mathbf{T} = \begin{bmatrix} 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 \\ -4 & 2 & -1 & 2 & -1 & 2 & -1 & 2 & -1 \\ 4 & 1 & -2 & 1 & -2 & 1 & -2 & 1 & -2 \\ 0 & -1 & -1 & -1 & 0 & 1 & 1 & 1 & 0 \\ 0 & -1 & 2 & -1 & 0 & 1 & -2 & 1 & 0 \\ 0 & 1 & 0 & -1 & -1 & -1 & 0 & 1 & 1 \\ 0 & 1 & 0 & -1 & 2 & -1 & 0 & 1 & -2 \\ 0 & 0 & 1 & 0 & -1 & 0 & 1 & 0 & -1 \\ 0 & -1 & 0 & 1 & 0 & -1 & 0 & 1 & 0 \end{bmatrix}$$

For the standard local equilibrium distribution functions, the equilibrium moments are expressed for incompressible fluids as:

$$\vec{M}^{eq} = \begin{pmatrix} \rho \\ e^{eq} \\ \epsilon^{eq} \\ j_x \\ q_x^{eq} \\ j_y \\ q_y^{eq} \\ p_{xx}^{eq} \\ p_{xy}^{eq} \end{pmatrix} = \begin{pmatrix} \rho \\ -2\rho + \frac{3}{\rho_0}\left(j_x^2 + j_y^2\right) \\ w_\epsilon \rho + \frac{w_{\epsilon j}}{\rho_0}(j_x^2 + j_y^2) \\ j_x \\ -j_x \\ j_y \\ -j_y \\ \frac{j_x^2 - j_y^2}{\rho_0} \\ \frac{j_x j_y}{\rho_0} \end{pmatrix}$$

where ρ is used in place of ρ_0 for mildly compressible fluids, $j_x = \rho_0 u_x$, $j_y = \rho_0 u_y$ and, by default, $w_\epsilon = 1$ and $w_{\epsilon j} = -3$. If using Swift free-energy interactions, the equilibrium moments are expressed as:

$$\vec{M}^{eq} = \begin{pmatrix} \rho \\ e^{eq} \\ \epsilon^{eq} \\ j_x \\ q_x^{eq} \\ j_y \\ q_y^{eq} \\ p_{xx}^{eq} \\ p_{xy}^{eq} \end{pmatrix} = \begin{pmatrix} \rho \\ -4\rho + \frac{3}{\rho}\left(j_x^2 + j_y^2\right) + 6(P_0 - \kappa(\rho\nabla^2\rho + \phi\nabla^2\phi)) + 15\frac{\lambda}{\rho}(\vec{p}\cdot\nabla\rho) \\ 4\rho - \frac{3}{\rho}(j_x^2 + j_y^2) - 9(P_0 - \kappa(\rho\nabla^2\rho + \phi\nabla^2\phi)) - \frac{3}{2}\kappa\left(|\nabla\rho|^2 + |\nabla\phi|^2\right) - \frac{33\lambda}{2\rho}(\vec{p}\cdot\nabla\rho) \\ j_x \\ -j_x \\ j_y \\ -j_y \\ \frac{j_x^2 - j_y^2}{\rho} + \kappa((\partial_x\rho)^2 - (\partial_y\rho)^2 + (\partial_x\phi)^2 - (\partial_y\phi)^2) + 2\frac{\lambda}{\rho}(p_x\partial_x\rho - p_y\partial_y\rho) \\ \frac{j_x j_y}{\rho} + \kappa(\partial_x\rho\partial_y\rho + \partial_x\phi\partial_y\phi) + \frac{\lambda}{\rho}(p_x\partial_y\rho + p_y\partial_x\rho) \end{pmatrix}$$

The relaxation frequencies for the above moments can be expressed by the following diagonal matrix:

$$\vec{s} = \mathrm{diag}\left(1, \tau_{f,bulk}^{-1}, s_2, 1, s_4, 1, s_4, \tau_f^{-1}, \tau_f^{-1}\right)$$

where the bulk viscosity can be related to the associated relaxation time by:

$$\nu' = \frac{1}{6}\left(\tau_{f,bulk} - \frac{1}{2}\right)\frac{(\Delta x)^2}{\Delta t}.$$

Recommended default values for the two variable relaxation frequencies s_2 and s_4 are 1.14 and 1.92 respectively for standard simulations, while both can be set to 1 for simulations with Swift free-energy interactions[95].

Guo forcing can be applied using the following moment transformations of the associated source terms:

$$\vec{S}^m = \begin{pmatrix} 0 \\ 6(v_x F_x + v_y F_y) \\ -6(v_x F_x + v_y F_y) \\ F_x \\ -F_x \\ F_y \\ -F_y \\ 2(v_x F_x - v_y F_y) \\ v_x F_y + v_y F_x \end{pmatrix},$$

and He forcing can be applied using these moment terms:

$$\vec{S}^m = \begin{pmatrix} 0 \\ 6(v_x F_x + v_y F_y) \\ -6(v_x F_x + v_y F_y) \\ F_x \\ -F_x(1 - 3v_y^2) + 6v_x v_y F_y \\ F_y \\ -F_y(1 - 3v_x^2) + 6v_x v_y F_x \\ 2(v_x F_x - v_y F_y) \\ v_x F_y + v_y F_x \end{pmatrix}.$$

Cascaded LBE scheme

Definitions of transformation and shift matrices based on [32]:

$$\mathbf{T} = \begin{bmatrix} 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 \\ 0 & -1 & -1 & -1 & 0 & 1 & 1 & 1 & 0 \\ 0 & 1 & 0 & -1 & -1 & -1 & 0 & 1 & 1 \\ 0 & 1 & 1 & 1 & 0 & 1 & 1 & 1 & 0 \\ 0 & 1 & 0 & 1 & 1 & 1 & 0 & 1 & 1 \\ 0 & -1 & 0 & 1 & 0 & -1 & 0 & 1 & 0 \\ 0 & 1 & 0 & -1 & 0 & -1 & 0 & 1 & 0 \\ 0 & 1 & 0 & -1 & 0 & 1 & 0 & 1 & 0 \\ 0 & -1 & 0 & 1 & 0 & 1 & 0 & 1 & 0 \end{bmatrix}$$

$$\mathbf{N} = \begin{bmatrix} 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ -u_x & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ -u_y & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ u_x^2 & -2u_x & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ u_y^2 & 0 & -2u_y & 0 & 1 & 0 & 0 & 0 & 0 \\ u_x u_y & -u_y & -u_x & 0 & 0 & 1 & 0 & 0 & 0 \\ -u_x^2 u_y & .2u_x u_y & u_x^2 & -u_y & 0 & -2u_x & 1 & 0 & 0 \\ -u_x u_y^2 & u_y^2 & 2u_x u_y & 0 & -u_x & -2u_y & 0 & 1 & 0 \\ u_x^2 u_y^2 & -2u_x u_y^2 & -2u_x^2 u_y & u_y^2 & u_x^2 & 4u_x u_y & -2u_y & -2u_x & 1 \end{bmatrix}.$$

The equilibrium central moments are expressed as follows:

$$\vec{\tilde{M}}^{eq} = \begin{pmatrix} \tilde{M}^{eq}_{00} \\ \tilde{M}^{eq}_{10} \\ \tilde{M}^{eq}_{01} \\ \tilde{M}^{eq}_{20} \\ \tilde{M}^{eq}_{02} \\ \tilde{M}^{eq}_{11} \\ \tilde{M}^{eq}_{21} \\ \tilde{M}^{eq}_{12} \\ \tilde{M}^{eq}_{22} \end{pmatrix} = \begin{pmatrix} \rho \\ 0 \\ 0 \\ \frac{1}{3}\rho \\ \frac{1}{3}\rho \\ 0 \\ 0 \\ 0 \\ \frac{1}{9}\rho \end{pmatrix}$$

and transformation of the above leads to the following expressions for the local equilibrium distribution functions:

$$\begin{aligned}
f_0^{eq} &= \tfrac{4}{9}\rho - \tfrac{2}{3}\rho u_x^2 - \tfrac{2}{3}\rho u_y^2 + \rho u_x^2 u_y^2 \\
f_1^{eq} &= \tfrac{1}{36}\rho - \tfrac{1}{12}\rho u_x + \tfrac{1}{12}\rho u_y + \tfrac{1}{12}\rho u_x^2 + \tfrac{1}{12}\rho u_y^2 - \tfrac{1}{4}\rho u_x u_y + \tfrac{1}{4}\rho u_x^2 u_y - \tfrac{1}{4}\rho u_x u_y^2 + \tfrac{1}{4}\rho u_x^2 u_y^2 \\
f_2^{eq} &= \tfrac{1}{9}\rho - \tfrac{1}{3}\rho u_x + \tfrac{1}{3}\rho u_x^2 - \tfrac{1}{6}\rho u_y^2 + \tfrac{1}{2}\rho u_x u_y^2 - \tfrac{1}{2}\rho u_x^2 u_y^2 \\
f_3^{eq} &= \tfrac{1}{36}\rho - \tfrac{1}{12}\rho u_x - \tfrac{1}{12}\rho u_y + \tfrac{1}{12}\rho u_x^2 + \tfrac{1}{12}\rho u_y^2 + \tfrac{1}{4}\rho u_x u_y - \tfrac{1}{4}\rho u_x^2 u_y - \tfrac{1}{4}\rho u_x u_y^2 + \tfrac{1}{4}\rho u_x^2 u_y^2 \\
f_4^{eq} &= \tfrac{1}{9}\rho - \tfrac{1}{3}\rho u_y - \tfrac{1}{6}\rho u_x^2 + \tfrac{1}{3}\rho u_y^2 + \tfrac{1}{2}\rho u_x^2 u_y - \tfrac{1}{2}\rho u_x^2 u_y^2 \\
f_5^{eq} &= \tfrac{1}{36}\rho + \tfrac{1}{12}\rho u_x - \tfrac{1}{12}\rho u_y + \tfrac{1}{12}\rho u_x^2 + \tfrac{1}{12}\rho u_y^2 - \tfrac{1}{4}\rho u_x u_y - \tfrac{1}{4}\rho u_x^2 u_y + \tfrac{1}{4}\rho u_x u_y^2 + \tfrac{1}{4}\rho u_x^2 u_y^2 \\
f_6^{eq} &= \tfrac{1}{9}\rho + \tfrac{1}{3}\rho u_x + \tfrac{1}{3}\rho u_x^2 - \tfrac{1}{6}\rho u_y^2 - \tfrac{1}{2}\rho u_x u_y^2 - \tfrac{1}{2}\rho u_x^2 u_y^2 \\
f_7^{eq} &= \tfrac{1}{36}\rho + \tfrac{1}{12}\rho u_x + \tfrac{1}{12}\rho u_y + \tfrac{1}{12}\rho u_x^2 + \tfrac{1}{12}\rho u_y^2 + \tfrac{1}{4}\rho u_x u_y + \tfrac{1}{4}\rho u_x^2 u_y + \tfrac{1}{4}\rho u_x u_y^2 + \tfrac{1}{4}\rho u_x^2 u_y^2 \\
f_8^{eq} &= \tfrac{1}{9}\rho + \tfrac{1}{3}\rho u_y - \tfrac{1}{6}\rho u_x^2 + \tfrac{1}{3}\rho u_y^2 - \tfrac{1}{2}\rho u_x^2 u_y - \tfrac{1}{2}\rho u_x^2 u_y^2
\end{aligned}$$

The relaxation frequencies can be expressed by the following block diagonal matrix:

$$\mathbf{\Lambda} = \text{diag}\left(1, 1, 1, \begin{bmatrix} s_+ & s_- \\ s_- & s_+ \end{bmatrix} \tau_f^{-1}, \omega_3, \omega_3, \omega_4\right)$$

where $s_+ = \frac{1}{2}\left(\tau_{f,bulk}^{-1} + \tau_f^{-1}\right)$ and $s_- = \frac{1}{2}\left(\tau_{f,bulk}^{-1} - \tau_f^{-1}\right)$. The bulk viscosity can be related to the associated relaxation time by:

$$\nu' = \frac{1}{3}\left(\tau_{f,bulk} - \frac{1}{2}\right)\frac{(\Delta x)^2}{\Delta t}.$$

Guo forcing can be applied using the following central moment transformations of the associated source terms:

$$\vec{S}^m = \begin{pmatrix} 0 \\ F_x \\ F_y \\ 0 \\ 0 \\ 0 \\ \left(\frac{1}{3} - v_x^2\right) F_y - 2 v_x v_y F_x \\ \left(\frac{1}{3} - v_y^2\right) F_x - 2 v_x v_y F_y \\ 4 v_x v_y \left(v_y F_x + v_x F_y\right) \end{pmatrix},$$

and He forcing can be applied using these central moment terms:

$$\vec{S}^m = \begin{pmatrix} 0 \\ F_x \\ F_y \\ 0 \\ 0 \\ 0 \\ \frac{1}{3}F_y \\ \frac{1}{3}F_x \\ 0 \end{pmatrix}.$$

D.2 D3Q15

Speed vectors

i	$e_{i,x}$	$e_{i,y}$	$e_{i,z}$
0	0	0	0
1	-1	0	0
2	0	-1	0
3	0	0	-1
4	-1	-1	-1
5	-1	-1	1
6	-1	1	-1
7	-1	1	1
8	1	0	0
9	0	1	0
10	0	0	1
11	1	1	1
12	1	1	-1
13	1	-1	1
14	1	-1	-1

Weight factors

i	w_i
0	$\frac{2}{9}$
1–3, 8–10	$\frac{1}{9}$
4–7, 11–14	$\frac{1}{72}$

Weight factors for Swift free-energy model

i	w_i	w_i^{00}	γ_i	δ_i	w_i^p	w_i^t	w_i^{xx}	w_i^{yy}	w_i^{zz}	w_i^{xy}	w_i^{xz}	w_i^{yz}
0	$\frac{2}{3}$	1	0	$-\frac{13}{2}$	$-\frac{7}{3}$	$-\frac{7}{3}$	$\frac{1}{6}$	$\frac{1}{6}$	$\frac{1}{6}$	0	0	0
1,8	$\frac{1}{3}$	0	0	1	$\frac{1}{3}$	$\frac{1}{3}$	$\frac{1}{3}$	$-\frac{1}{6}$	$-\frac{1}{6}$	0	0	0
2,9	$\frac{1}{3}$	0	0	1	$\frac{1}{3}$	$\frac{1}{3}$	$-\frac{1}{6}$	$\frac{1}{3}$	$-\frac{1}{6}$	0	0	0
3,10	$\frac{1}{3}$	0	0	1	$\frac{1}{3}$	$\frac{1}{3}$	$-\frac{1}{6}$	$-\frac{1}{6}$	$\frac{1}{3}$	0	0	0
4,11	$\frac{1}{24}$	0	0	-2	$\frac{1}{24}$	$\frac{1}{24}$	$-\frac{1}{48}$	$-\frac{1}{48}$	$-\frac{1}{48}$	$\frac{1}{8}$	$\frac{1}{8}$	$\frac{1}{8}$
5,12	$\frac{1}{24}$	0	0	-2	$\frac{1}{24}$	$\frac{1}{24}$	$-\frac{1}{48}$	$-\frac{1}{48}$	$-\frac{1}{48}$	$\frac{1}{8}$	$-\frac{1}{8}$	$-\frac{1}{8}$
6,13	$\frac{1}{24}$	0	0	-2	$\frac{1}{24}$	$\frac{1}{24}$	$-\frac{1}{48}$	$-\frac{1}{48}$	$-\frac{1}{48}$	$-\frac{1}{8}$	$\frac{1}{8}$	$-\frac{1}{8}$
7,14	$\frac{1}{24}$	0	0	-2	$\frac{1}{24}$	$\frac{1}{24}$	$-\frac{1}{48}$	$-\frac{1}{48}$	$-\frac{1}{48}$	$-\frac{1}{8}$	$-\frac{1}{8}$	$\frac{1}{8}$

Multiple relaxation time scheme

Definition of transformation matrix based on [23]:

$$\mathbf{T} = \begin{bmatrix}
1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 \\
-2 & -1 & -1 & -1 & 1 & 1 & 1 & 1 & -1 & -1 & -1 & 1 & 1 & 1 & 1 \\
16 & -4 & -4 & -4 & 1 & 1 & 1 & 1 & -4 & -4 & -4 & 1 & 1 & 1 & 1 \\
0 & -1 & 0 & 0 & -1 & -1 & -1 & -1 & 1 & 0 & 0 & 1 & 1 & 1 & 1 \\
0 & 4 & 0 & 0 & -1 & -1 & -1 & -1 & -4 & 0 & 0 & 1 & 1 & 1 & 1 \\
0 & 0 & -1 & 0 & -1 & -1 & 1 & 1 & 0 & 1 & 0 & 1 & 1 & -1 & -1 \\
0 & 0 & 4 & 0 & -1 & -1 & 1 & 1 & 0 & -4 & 0 & 1 & 1 & -1 & -1 \\
0 & 0 & 0 & -1 & -1 & 1 & -1 & 1 & 0 & 0 & 1 & 1 & -1 & 1 & -1 \\
0 & 0 & 0 & 4 & -1 & 1 & -1 & 1 & 0 & 0 & -4 & 1 & -1 & 1 & -1 \\
0 & 2 & -1 & -1 & 0 & 0 & 0 & 0 & 2 & -1 & -1 & 0 & 0 & 0 & 0 \\
0 & 0 & 1 & -1 & 0 & 0 & 0 & 0 & 0 & 1 & -1 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 1 & 1 & -1 & -1 & 0 & 0 & 0 & 1 & 1 & -1 & -1 \\
0 & 0 & 0 & 0 & 1 & -1 & -1 & 1 & 0 & 0 & 0 & 1 & -1 & -1 & 1 \\
0 & 0 & 0 & 0 & 1 & -1 & 1 & -1 & 0 & 0 & 0 & 1 & -1 & 1 & -1 \\
0 & 0 & 0 & 0 & -1 & 1 & 1 & -1 & 0 & 0 & 0 & 1 & -1 & -1 & 1
\end{bmatrix}$$

For the standard local equilibrium distribution functions, the equilibrium moments are expressed for incompressible fluids as:

$$\vec{M}^{eq} = \begin{pmatrix} \rho \\ e^{eq} \\ \epsilon^{eq} \\ j_x \\ q_x^{eq} \\ j_y \\ q_y^{eq} \\ j_z \\ q_z^{eq} \\ 3p_{xx}^{eq} \\ p_{ww}^{eq} \\ p_{xy}^{eq} \\ p_{yz}^{eq} \\ p_{zx}^{eq} \\ m_{xyz}^{eq} \end{pmatrix} = \begin{pmatrix} \rho \\ -\rho + \frac{\left(j_x^2+j_y^2+j_z^2\right)}{\rho_0} \\ w_\epsilon \rho + \frac{w_{\epsilon j}}{\rho_0}(j_x^2+j_y^2+j_z^2) \\ j_x \\ -\frac{7}{3}j_x \\ j_y \\ -\frac{7}{3}j_y \\ j_z \\ -\frac{7}{3}j_z \\ \frac{2j_x^2-j_y^2-j_z^2}{\rho_0} \\ \frac{j_y^2-j_z^2}{\rho_0} \\ \frac{j_x j_y}{\rho_0} \\ \frac{j_y j_z}{\rho_0} \\ \frac{j_z j_x}{\rho_0} \\ 0 \end{pmatrix}$$

where ρ is used in place of ρ_0 for mildly compressible fluids, $j_x = \rho_0 u_x$, $j_y = \rho_0 u_y$, $j_z = \rho_0 u_z$ and, by default, $w_\epsilon = 1$ and $w_{\epsilon j} = -5$. If using Swift free-energy interactions, the equilibrium moments are expressed as:

$$\vec{M}^{eq} = \begin{pmatrix} \rho \\ e^{eq} \\ \epsilon^{eq} \\ j_x \\ q_x^{eq} \\ j_y \\ q_y^{eq} \\ j_z \\ q_z^{eq} \\ 3p_{xx}^{eq} \\ p_{ww}^{eq} \\ p_{xy}^{eq} \\ p_{yz}^{eq} \\ p_{zx}^{eq} \\ m_{xyz}^{eq} \end{pmatrix} = \begin{pmatrix} \rho \\ -2\rho + \frac{3}{\rho}\left(j_x^2+j_y^2+j_z^2\right) + 3(P_0 - \kappa(\rho\nabla^2\rho + \phi\nabla^2\phi)) - \frac{1}{2}\kappa\left(|\nabla\rho|^2 + |\nabla\phi|^2\right) + 5\frac{\lambda}{\rho}(\vec{p}\cdot\nabla\rho) \\ 16\rho - \frac{5}{\rho}(j_x^2+j_y^2+j_z^2) - 45(P_0 - \kappa(\rho\nabla^2\rho + \phi\nabla^2\phi)) + \frac{5}{2}\kappa\left(|\nabla\rho|^2 + |\nabla\phi|^2\right) - 85\frac{\lambda}{\rho}(\vec{p}\cdot\nabla\rho) \\ j_x \\ -\frac{7}{3}j_x \\ j_y \\ -\frac{7}{3}j_y \\ j_z \\ -\frac{7}{3}j_z \\ \frac{2j_x^2-j_y^2-j_z^2}{\rho} + \kappa(2(\partial_x\rho)^2 - (\partial_y\rho)^2 - (\partial_z\rho)^2 + 2(\partial_x\phi)^2 - (\partial_y\phi)^2 - (\partial_z\phi)^2) + \frac{2\lambda}{\rho}(2p_x\partial_x\rho - p_y\partial_y\rho - p_z\partial_z\rho) \\ \frac{j_y^2-j_z^2}{\rho} + \kappa((\partial_y\rho)^2 - (\partial_z\rho)^2 + (\partial_y\phi)^2 - (\partial_z\phi)^2) + \frac{2\lambda}{\rho}(p_y\partial_y\rho - p_z\partial_z\rho) \\ \frac{j_x j_y}{\rho} + \kappa(\partial_x\rho\partial_y\rho + \partial_x\phi\partial_y\phi) + \frac{\lambda}{\rho}(p_x\partial_y\rho + p_y\partial_x\rho) \\ \frac{j_y j_z}{\rho} + \kappa(\partial_y\rho\partial_z\rho + \partial_y\phi\partial_z\phi) + \frac{\lambda}{\rho}(p_y\partial_z\rho + p_z\partial_y\rho) \\ \frac{j_x j_z}{\rho} + \kappa(\partial_x\rho\partial_z\rho + \partial_x\phi\partial_z\phi) + \frac{\lambda}{\rho}(p_x\partial_z\rho + p_z\partial_x\rho) \\ 0 \end{pmatrix}$$

The relaxation frequencies for the above moments can be expressed by the following diagonal matrix:

$$\vec{s} = \text{diag}\left(1, \tau_{f,bulk}^{-1}, s_2, 1, s_4, 1, s_4, 1, s_4, \tau_f^{-1}, \tau_f^{-1}, \tau_f^{-1}, \tau_f^{-1}, \tau_f^{-1}, s_{14}\right)$$

where the bulk viscosity can be related to the associated relaxation time by:

$$\nu' = \frac{2}{9}\left(\tau_{f,bulk} - \frac{1}{2}\right)\frac{(\Delta x)^2}{\Delta t}$$

Recommended default values for the three variable relaxation frequencies s_2, s_4 and s_{14} are 1.2, 1.6 and 1.2 respectively.

Guo forcing can be applied using the following moment transformations of the associated source terms:

$$\vec{S}^m = \begin{pmatrix} 0 \\ 2(v_xF_x + v_yF_y + v_zF_z) \\ -10(v_xF_x + v_yF_y + v_zF_z) \\ F_x \\ -\frac{7}{3}F_x \\ F_y \\ -\frac{7}{3}F_y \\ F_z \\ -\frac{7}{3}F_z \\ 2(2v_xF_x - v_yF_y - v_zF_z) \\ 2(v_yF_y - v_zF_z) \\ v_xF_y + v_yF_x \\ v_yF_z + v_zF_y \\ v_zF_x + v_xF_z \\ 0 \end{pmatrix},$$

and He forcing can be applied using these moment terms:

$$\vec{S}^m = \begin{pmatrix} 0 \\ 2(v_xF_x + v_yF_y + v_zF_z) \\ -10(v_xF_x + v_yF_y + v_zF_z) \\ F_x \\ \left(-\frac{7}{3} + 5v_y^2 + 5v_z^2\right)F_x + 10v_xv_yF_y + 10v_xv_zF_z \\ F_y \\ \left(-\frac{7}{3} + 5v_x^2 + 5v_z^2\right)F_y + 10v_xv_yF_x + 10v_yv_zF_z \\ F_z \\ \left(-\frac{7}{3} + 5v_x^2 + 5v_y^2\right)F_z + 10v_xv_zF_x + 10v_yv_zF_y \\ 2(2v_xF_x - v_yF_y - v_zF_z) \\ 2(v_yF_y - v_zF_z) \\ v_xF_y + v_yF_x \\ v_yF_z + v_zF_y \\ v_zF_x + v_xF_z \\ 3v_yv_zF_x + 3v_xv_zF_y + 3v_xv_yF_z \end{pmatrix}.$$

D.3 D3Q19

Speed vectors

i	$e_{i,x}$	$e_{i,y}$	$e_{i,z}$
0	0	0	0
1	-1	0	0
2	0	-1	0
3	0	0	-1
4	-1	-1	0
5	-1	1	0
6	-1	0	-1
7	-1	0	1
8	0	-1	-1
9	0	-1	1
10	1	0	0
11	0	1	0
12	0	0	1
13	1	1	0
14	1	-1	0
15	1	0	1
16	1	0	-1
17	0	1	1
18	0	1	-1

Weight factors

i	w_i
0	$\frac{1}{3}$
1–3, 10–12	$\frac{1}{18}$
4–9, 13–18	$\frac{1}{36}$

Weight factors for Swift free-energy model

i	w_i	w_i^{00}	γ_i	δ_i	w_i^p	w_i^t	w_i^{xx}	w_i^{yy}	w_i^{zz}	w_i^{xy}	w_i^{xz}	w_i^{yz}
0	1	1	0	$-\frac{9}{2}$	-2	-2	$\frac{1}{2}$	$\frac{1}{2}$	$\frac{1}{2}$	0	0	0
1,10	$\frac{1}{6}$	0	$\frac{3}{2}$	$-\frac{3}{2}$	$\frac{1}{6}$	$\frac{1}{6}$	$\frac{5}{12}$	$-\frac{1}{3}$	$-\frac{1}{3}$	0	0	0
2,11	$\frac{1}{6}$	0	$\frac{3}{2}$	$-\frac{3}{2}$	$\frac{1}{6}$	$\frac{1}{6}$	$-\frac{1}{3}$	$\frac{5}{12}$	$-\frac{1}{3}$	0	0	0
3,12	$\frac{1}{6}$	0	$\frac{3}{2}$	$-\frac{3}{2}$	$\frac{1}{6}$	$\frac{1}{6}$	$-\frac{1}{3}$	$-\frac{1}{3}$	$\frac{5}{12}$	0	0	0
4,13	$\frac{1}{12}$	0	$\frac{3}{2}$	$-\frac{3}{2}$	$\frac{1}{12}$	$\frac{1}{12}$	$-\frac{1}{24}$	$-\frac{1}{24}$	$\frac{1}{12}$	$\frac{1}{4}$	0	0
5,14	$\frac{1}{12}$	0	$\frac{3}{2}$	$-\frac{3}{2}$	$\frac{1}{12}$	$\frac{1}{12}$	$-\frac{1}{24}$	$-\frac{1}{24}$	$\frac{1}{12}$	$-\frac{1}{4}$	0	0
6,15	$\frac{1}{12}$	0	$\frac{3}{2}$	$-\frac{3}{2}$	$\frac{1}{12}$	$\frac{1}{12}$	$-\frac{1}{24}$	$\frac{1}{12}$	$-\frac{1}{24}$	0	$\frac{1}{4}$	0
7,16	$\frac{1}{12}$	0	$\frac{3}{2}$	$-\frac{3}{2}$	$\frac{1}{12}$	$\frac{1}{12}$	$-\frac{1}{24}$	$\frac{1}{12}$	$-\frac{1}{24}$	0	$-\frac{1}{4}$	0
8,17	$\frac{1}{12}$	0	$\frac{3}{2}$	$-\frac{3}{2}$	$\frac{1}{12}$	$\frac{1}{12}$	$\frac{1}{12}$	$-\frac{1}{24}$	$-\frac{1}{24}$	0	0	$\frac{1}{4}$
9,18	$\frac{1}{12}$	0	$\frac{3}{2}$	$-\frac{3}{2}$	$\frac{1}{12}$	$\frac{1}{12}$	$\frac{1}{12}$	$-\frac{1}{24}$	$-\frac{1}{24}$	0	0	$-\frac{1}{4}$

Multiple relaxation time scheme

Definition of transformation matrix based on [23]:

$$\mathbf{T} = \begin{bmatrix}
1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 \\
-30 & -11 & -11 & -11 & 8 & 8 & 8 & 8 & 8 & 8 & -11 & -11 & -11 & 8 & 8 & 8 & 8 & 8 & 8 \\
12 & -4 & -4 & -4 & 1 & 1 & 1 & 1 & 1 & 1 & -4 & -4 & -4 & 1 & 1 & 1 & 1 & 1 & 1 \\
0 & -1 & 0 & 0 & -1 & -1 & -1 & -1 & 0 & 0 & 1 & 0 & 0 & 1 & 1 & 1 & 1 & 0 & 0 \\
0 & 4 & 0 & 0 & -1 & -1 & -1 & -1 & 0 & 0 & -4 & 0 & 0 & 1 & 1 & 1 & 1 & 0 & 0 \\
0 & 0 & -1 & 0 & -1 & 1 & 0 & 0 & -1 & -1 & 0 & 1 & 0 & 1 & -1 & 0 & 0 & 1 & 1 \\
0 & 0 & 4 & 0 & -1 & 1 & 0 & 0 & -1 & -1 & 0 & -4 & 0 & 1 & -1 & 0 & 0 & 1 & 1 \\
0 & 0 & 0 & -1 & 0 & 0 & -1 & 1 & -1 & 1 & 0 & 0 & 1 & 0 & 0 & 1 & -1 & 1 & -1 \\
0 & 0 & 0 & 4 & 0 & 0 & -1 & 1 & -1 & 1 & 0 & 0 & -4 & 0 & 0 & 1 & -1 & 1 & -1 \\
0 & 2 & -1 & -1 & 1 & 1 & 1 & 1 & -2 & -2 & 2 & -1 & -1 & 1 & 1 & 1 & 1 & -2 & -2 \\
0 & -4 & 2 & 2 & 1 & 1 & 1 & 1 & -2 & -2 & -4 & 2 & 2 & 1 & 1 & 1 & 1 & -2 & -2 \\
0 & 0 & 1 & -1 & 1 & 1 & -1 & -1 & 0 & 0 & 0 & 1 & -1 & 1 & 1 & -1 & -1 & 0 & 0 \\
0 & 0 & -2 & 2 & 1 & 1 & -1 & -1 & 0 & 0 & 0 & -2 & 2 & 1 & 1 & -1 & -1 & 0 & 0 \\
0 & 0 & 0 & 0 & 1 & -1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & -1 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & -1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & -1 \\
0 & 0 & 0 & 0 & 0 & 0 & 1 & -1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & -1 & 0 & 0 \\
0 & 0 & 0 & 0 & -1 & -1 & 1 & 1 & 0 & 0 & 0 & 0 & 0 & 1 & 1 & -1 & -1 & 0 & 0 \\
0 & 0 & 0 & 0 & 1 & -1 & 0 & 0 & -1 & -1 & 0 & 0 & 0 & -1 & 1 & 0 & 0 & 1 & 1 \\
0 & 0 & 0 & 0 & 0 & 0 & -1 & 1 & 1 & -1 & 0 & 0 & 0 & 0 & 0 & 1 & -1 & -1 & 1
\end{bmatrix}$$

For the standard local equilibrium distribution functions, the equilibrium moments are expressed for incompressible fluids as:

$$vecM^{eq} = \begin{pmatrix} \rho \\ e^{eq} \\ \epsilon^{eq} \\ j_x \\ q_x^{eq} \\ j_y \\ q_y^{eq} \\ j_z \\ q_z^{eq} \\ 3p_{xx}^{eq} \\ 3\pi_{xx}^{eq} \\ p_{ww}^{eq} \\ \pi_{ww}^{eq} \\ p_{xy}^{eq} \\ p_{yz}^{eq} \\ p_{zx}^{eq} \\ m_x^{eq} \\ m_y^{eq} \\ m_z^{eq} \end{pmatrix} = \begin{pmatrix} \rho \\ -11\rho + \frac{19}{\rho_0}\left(j_x^2 + j_y^2 + j_z^2\right) \\ w_\epsilon \rho + \frac{w_{\epsilon j}}{\rho_0}(j_x^2 + j_y^2 + j_z^2) \\ j_x \\ -\frac{2}{3}j_x \\ j_y \\ -\frac{2}{3}j_y \\ j_z \\ -\frac{2}{3}j_z \\ \frac{2j_x^2 - j_y^2 - j_z^2}{\rho_0} \\ \frac{w_{xx}}{\rho_0}\left(2j_x^2 - j_y^2 - j_z^2\right) \\ \frac{j_y^2 - j_z^2}{\rho_0} \\ \frac{w_{xx}}{\rho_0}\left(j_y^2 - j_z^2\right) \\ \frac{j_x j_y}{\rho_0} \\ \frac{j_y j_z}{\rho_0} \\ \frac{j_z j_x}{\rho_0} \\ 0 \\ 0 \\ 0 \end{pmatrix}$$

where ρ is used in place of ρ_0 for mildly compressible fluids, $j_x = \rho_0 u_x$, $j_y = \rho_0 u_y$, $j_z = \rho_0 u_z$ and, by default, $w_\epsilon = 1$, $w_{\epsilon j} = -\frac{11}{2}$ and $w_{xx} = \frac{1}{2}$. If using Swift free-energy interactions, the equilibrium moments are expressed

as:

$$
vecM^{eq} = \begin{pmatrix} \rho \\ e^{eq} \\ \epsilon^{eq} \\ j_x \\ q_x^{eq} \\ j_y \\ q_y^{eq} \\ j_z \\ q_z^{eq} \\ 3p_{xx}^{eq} \\ 3\pi_{xx}^{eq} \\ p_{ww}^{eq} \\ \pi_{ww}^{eq} \\ p_{xy}^{eq} \\ p_{yz}^{eq} \\ p_{zx}^{eq} \\ m_x^{eq} \\ m_y^{eq} \\ m_z^{eq} \end{pmatrix} = \begin{pmatrix} \rho \\ -30\rho + \frac{19}{\rho}\left(j_x^2 + j_y^2 + j_z^2\right) + 57(P_0 - \kappa(\rho\nabla^2\rho + \phi\nabla^2\phi)) - \frac{19}{2}\kappa\left(|\nabla\rho|^2 + |\nabla\phi|^2\right) + 152\frac{\lambda}{\rho}(\vec{p}\cdot\nabla\rho) \\ 12\rho - \frac{11}{2\rho}(j_x^2 + j_y^2 + j_z^2) - 27(P_0 - \kappa(\rho\nabla^2\rho + \phi\nabla^2\phi)) + 8\kappa\left(|\nabla\rho|^2 + |\nabla\phi|^2\right) - \frac{109\lambda}{2\rho}(\vec{p}\cdot\nabla\rho) \\ j_x \\ -\frac{2}{3}j_x \\ j_y \\ -\frac{2}{3}j_y \\ j_z \\ -\frac{2}{3}j_z \\ \frac{2j_x^2 - j_y^2 - j_z^2}{\rho} + \kappa(2(\partial_x\rho)^2 - (\partial_y\rho)^2 - (\partial_z\rho)^2 + 2(\partial_x\phi)^2 - (\partial_y\phi)^2 - (\partial_z\phi)^2) + \frac{2\lambda}{\rho}(2p_x\partial_x\rho - p_y\partial_y\rho - p_z\partial_z\rho \\ -\frac{2j_x^2 - j_y^2 - j_z^2}{2\rho} - \frac{7}{2}\kappa(2(\partial_x\rho)^2 - (\partial_y\rho)^2 - (\partial_z\rho)^2 + 2(\partial_x\phi)^2 - (\partial_y\phi)^2 - (\partial_z\phi)^2) - \frac{\lambda}{\rho}(2p_x\partial_x\rho - p_y\partial_y\rho - p_z\partial_z\rho) \\ \frac{j_y^2 - j_z^2}{\rho} + \kappa((\partial_y\rho)^2 - (\partial_z\rho)^2 + (\partial_y\phi)^2 - (\partial_z\phi)^2) + \frac{2\lambda}{\rho}(p_y\partial_y\rho - p_z\partial_z\rho) \\ -\frac{j_y^2 - j_z^2}{2\rho} - \frac{7}{2}\kappa((\partial_y\rho)^2 - (\partial_z\rho)^2 + (\partial_y\phi)^2 - (\partial_z\phi)^2) - \frac{\lambda}{\rho}(p_y\partial_y\rho - p_z\partial_z\rho)\left(j_y^2 - j_z^2\right) \\ \frac{j_x j_y}{\rho} + \kappa(\partial_x\rho\partial_y\rho + \partial_x\phi\partial_y\phi) + \frac{\lambda}{\rho}(p_x\partial_y\rho + p_y\partial_x\rho) \\ \frac{j_y j_z}{\rho} + \kappa(\partial_y\rho\partial_z\rho + \partial_y\phi\partial_z\phi) + \frac{\lambda}{\rho}(p_y\partial_z\rho + p_z\partial_y\rho) \\ \frac{j_x j_z}{\rho} + \kappa(\partial_x\rho\partial_z\rho + \partial_x\phi\partial_z\phi) + \frac{\lambda}{\rho}(p_x\partial_z\rho + p_z\partial_x\rho) \\ 0 \\ 0 \\ 0 \end{pmatrix}
$$

The relaxation frequencies for the above moments can be expressed by the following diagonal matrix:

$$
\vec{s} = \text{diag}\left(1, \tau_{f,bulk}^{-1}, s_2, 1, s_4, 1, s_4, 1, s_4, \tau_f^{-1}, s_4, \tau_f^{-1}, s_4, \tau_f^{-1}, \tau_f^{-1}, \tau_f^{-1}, s_{16}, s_{16}, s_{16}\right)
$$

where the bulk viscosity can be related to the associated relaxation time by:

$$
\nu' = \frac{2}{9}\left(\tau_{f,bulk} - \frac{1}{2}\right)\frac{(\Delta x)^2}{\Delta t}
$$

Recommended default values for the three variable relaxation frequencies s_2, s_4 and s_{16} are 1.4, 1.4 and 1.98 respectively.

Guo forcing can be applied using the following moment transformations of the associated source terms:

$$
\vec{S}^m = \begin{pmatrix} 0 \\ 38(v_xF_x + v_yF_y + v_zF_z) \\ -11(v_xF_x + v_yF_y + v_zF_z) \\ F_x \\ -\frac{2}{3}F_x \\ F_y \\ -\frac{2}{3}F_y \\ F_z \\ -\frac{2}{3}F_z \\ 2(2v_xF_x - v_yF_y - v_zF_z) \\ -(2v_xF_x - v_yF_y - v_zF_z) \\ 2(v_yF_y - v_zF_z) \\ -(v_yF_y - v_zF_z) \\ v_xF_y + v_yF_x \\ v_yF_z + v_zF_y \\ v_zF_x + v_xF_z \\ 0 \\ 0 \\ 0 \end{pmatrix},
$$

and He forcing can be applied using these moment terms:

$$\vec{S}^m = \begin{pmatrix} 0 \\ 38(v_xF_x + v_yF_y + v_zF_z) \\ -11(v_xF_x + v_yF_y + v_zF_z) \\ F_x \\ \left(-\frac{2}{3} + \frac{5}{2}v_y^2 + \frac{5}{2}v_z^2\right)F_x + 5v_xv_yF_y + 5v_xv_zF_z \\ F_y \\ \left(-\frac{2}{3} + \frac{5}{2}v_x^2 + \frac{5}{2}v_z^2\right)F_y + 5v_xv_yF_x + 5v_yv_zF_z \\ F_z \\ \left(-\frac{2}{3} + \frac{5}{2}v_x^2 + \frac{5}{2}v_y^2\right)F_z + 5v_xv_zF_x + 5v_yv_zF_y \\ 2(2v_xF_x - v_yF_y - v_zF_z) \\ -(2v_xF_x - v_yF_y - v_zF_z) \\ 2(v_yF_y - v_zF_z) \\ -(v_yF_y - v_zF_z) \\ v_xF_y + v_yF_x \\ v_yF_z + v_zF_y \\ v_zF_x + v_xF_z \\ \frac{3}{2}\left(v_y^2 - v_z^2\right)F_x + 3v_xv_yF_y - 3v_xv_zF_z \\ \frac{3}{2}\left(v_z^2 - v_x^2\right)F_y + 3v_yv_zF_z - 3v_xv_yF_x \\ \frac{3}{2}\left(v_x^2 - v_y^2\right)F_z + 3v_xv_zF_x - 3v_yv_zF_y \end{pmatrix}.$$

Cascaded LBE scheme

Definitions of transformation and shift matrices based on [32]:

$$\mathbf{T} = \begin{bmatrix} 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 \\ 0 & -1 & 0 & 0 & -1 & -1 & -1 & -1 & 0 & 0 & 1 & 0 & 0 & 1 & 1 & 1 & 1 & 0 & 0 \\ 0 & 0 & -1 & 0 & -1 & 1 & 0 & 0 & -1 & -1 & 0 & 1 & 0 & 1 & -1 & 0 & 0 & 1 & 1 \\ 0 & 0 & 0 & -1 & 0 & 0 & -1 & 1 & -1 & 1 & 0 & 0 & 1 & 0 & 0 & 1 & -1 & 1 & -1 \\ 0 & 0 & 0 & 0 & 1 & -1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & -1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & -1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & -1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & -1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & -1 \\ 0 & 1 & 0 & 0 & 1 & 1 & 1 & 1 & 0 & 0 & 1 & 0 & 0 & 1 & 1 & 1 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 & 1 & 1 & 0 & 0 & 1 & 1 & 0 & 1 & 0 & 1 & 1 & 0 & 0 & 1 & 1 \\ 0 & 0 & 0 & 1 & 0 & 0 & 1 & 1 & 1 & 1 & 0 & 0 & 1 & 0 & 0 & 1 & 1 & 1 & 1 \\ 0 & 0 & 0 & 0 & -1 & -1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & -1 & -1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & -1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & -1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & -1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & -1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & -1 & -1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 1 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & -1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & -1 \\ 0 & 0 & 0 & 0 & 1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 1 \end{bmatrix}$$

$$
\mathbf{N} = \left[\begin{array}{cccccccccccccccccc}
1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
-u_x & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
-u_y & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
-u_z & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
u_x u_y & -u_y & -u_x & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
u_x u_z & -u_z & 0 & -u_x & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
u_y u_z & 0 & -u_z & -u_y & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
u_x^2 & -2u_x & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
u_y^2 & 0 & -2u_y & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
u_z^2 & 0 & 0 & -2u_z & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
-u_x u_y^2 & u_y^2 & 2u_x u_y & 0 & -2u_y & 0 & 0 & 0 & -u_x & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
-u_x u_z^2 & u_z^2 & 0 & 2u_x u_z & 0 & -2u_z & 0 & 0 & 0 & -u_x & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\
-u_x^2 u_y & 2u_x u_y & u_x^2 & 0 & -2u_x & 0 & 0 & 0 & -u_y & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\
-u_x^2 u_z & 2u_x u_z & 0 & u_x^2 & 0 & -2u_x & 0 & 0 & -u_z & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \\
-u_y u_z^2 & 0 & u_z^2 & 2u_y u_z & 0 & 0 & -2u_z & 0 & 0 & -u_y & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \\
-u_y^2 u_z & 0 & 2u_y u_z & u_y^2 & 0 & 0 & -2u_y & 0 & 0 & -u_z & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 \\
u_x^2 u_y^2 & -2u_x u_y^2 & -2u_x^2 u_y & 0 & 4u_x u_y & 0 & 0 & u_y^2 & u_x^2 & 0 & -2u_x & 0 & -2u_y & 0 & 0 & 0 & 1 & 0 \\
u_x^2 u_z^2 & -2u_x u_z^2 & 0 & -2u_x^2 u_z & 0 & 4u_x u_z & 0 & u_z^2 & 0 & u_x^2 & 0 & -2u_x & 0 & -2u_z & 0 & 0 & 0 & 1 \\
u_y^2 u_z^2 & 0 & -2u_y u_z^2 & -2u_y^2 u_z & 0 & 0 & 4u_y u_z & 0 & u_z^2 & 0 & u_y^2 & 0 & 0 & 0 & -2u_y & -2u_z & 0 & 0
\end{array}\right].
$$

The equilibrium central moments are expressed as follows:

$$\vec{\tilde{M}}^{eq} = \begin{pmatrix} \tilde{M}^{eq}_{000} \\ \tilde{M}^{eq}_{100} \\ \tilde{M}^{eq}_{010} \\ \tilde{M}^{eq}_{001} \\ \tilde{M}^{eq}_{110} \\ \tilde{M}^{eq}_{101} \\ \tilde{M}^{eq}_{011} \\ \tilde{M}^{eq}_{200} \\ \tilde{M}^{eq}_{020} \\ \tilde{M}^{eq}_{002} \\ \tilde{M}^{eq}_{120} \\ \tilde{M}^{eq}_{102} \\ \tilde{M}^{eq}_{210} \\ \tilde{M}^{eq}_{201} \\ \tilde{M}^{eq}_{012} \\ \tilde{M}^{eq}_{021} \\ \tilde{M}^{eq}_{220} \\ \tilde{M}^{eq}_{202} \\ \tilde{M}^{eq}_{022} \end{pmatrix} = \begin{pmatrix} \rho \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ \frac{1}{3}\rho \\ \frac{1}{3}\rho \\ \frac{1}{3}\rho \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ \frac{1}{9}\rho \\ \frac{1}{9}\rho \\ \frac{1}{9}\rho \end{pmatrix}$$

and transformation of the above leads to the following expressions for the local equilibrium distribution functions:

$$\begin{aligned}
f^{eq}_0 &= \tfrac{1}{3}\rho - \tfrac{1}{3}\rho u_x^2 - \tfrac{1}{3}\rho u_y^2 - \tfrac{1}{3}\rho u_z^2 + \rho u_x^2 u_y^2 + \rho u_x^2 u_z^2 + \rho u_y^2 u_z^2 \\
f^{eq}_1 &= \tfrac{1}{18}\rho - \tfrac{1}{6}\rho u_x + \tfrac{1}{6}\rho u_x^2 - \tfrac{1}{6}\rho u_y^2 - \tfrac{1}{6}\rho u_z^2 + \tfrac{1}{2}\rho u_x u_y^2 + \tfrac{1}{2}\rho u_x u_z^2 - \tfrac{1}{2}\rho u_x^2 u_y^2 - \tfrac{1}{2}\rho u_x^2 u_z^2 \\
f^{eq}_2 &= \tfrac{1}{18}\rho - \tfrac{1}{6}\rho u_y - \tfrac{1}{6}\rho u_x^2 + \tfrac{1}{6}\rho u_y^2 - \tfrac{1}{6}\rho u_z^2 + \tfrac{1}{2}\rho u_x^2 u_y + \tfrac{1}{2}\rho u_y u_z^2 - \tfrac{1}{2}\rho u_x^2 u_y^2 - \tfrac{1}{2}\rho u_y^2 u_z^2 \\
f^{eq}_3 &= \tfrac{1}{18}\rho - \tfrac{1}{6}\rho u_z - \tfrac{1}{6}\rho u_x^2 - \tfrac{1}{6}\rho u_y^2 + \tfrac{1}{6}\rho u_z^2 + \tfrac{1}{2}\rho u_x^2 u_z + \tfrac{1}{2}\rho u_y^2 u_z - \tfrac{1}{2}\rho u_x^2 u_z^2 - \tfrac{1}{2}\rho u_y^2 u_z^2 \\
f^{eq}_4 &= \tfrac{1}{36}\rho - \tfrac{1}{12}\rho u_x - \tfrac{1}{12}\rho u_y + \tfrac{1}{12}\rho u_x^2 + \tfrac{1}{12}\rho u_y^2 + \tfrac{1}{4}\rho u_x u_y - \tfrac{1}{4}\rho u_x^2 u_y - \tfrac{1}{4}\rho u_x u_y^2 + \tfrac{1}{4}\rho u_x^2 u_y^2 \\
f^{eq}_5 &= \tfrac{1}{36}\rho - \tfrac{1}{12}\rho u_x + \tfrac{1}{12}\rho u_y + \tfrac{1}{12}\rho u_x^2 + \tfrac{1}{12}\rho u_y^2 - \tfrac{1}{4}\rho u_x u_y + \tfrac{1}{4}\rho u_x^2 u_y - \tfrac{1}{4}\rho u_x u_y^2 + \tfrac{1}{4}\rho u_x^2 u_y^2 \\
f^{eq}_6 &= \tfrac{1}{36}\rho - \tfrac{1}{12}\rho u_x - \tfrac{1}{12}\rho u_z + \tfrac{1}{12}\rho u_x^2 + \tfrac{1}{12}\rho u_z^2 + \tfrac{1}{4}\rho u_x u_z - \tfrac{1}{4}\rho u_x^2 u_z - \tfrac{1}{4}\rho u_x u_z^2 + \tfrac{1}{4}\rho u_x^2 u_z^2 \\
f^{eq}_7 &= \tfrac{1}{36}\rho - \tfrac{1}{12}\rho u_x + \tfrac{1}{12}\rho u_z + \tfrac{1}{12}\rho u_x^2 + \tfrac{1}{12}\rho u_z^2 - \tfrac{1}{4}\rho u_x u_z + \tfrac{1}{4}\rho u_x^2 u_z - \tfrac{1}{4}\rho u_x u_z^2 + \tfrac{1}{4}\rho u_x^2 u_z^2 \\
f^{eq}_8 &= \tfrac{1}{36}\rho - \tfrac{1}{12}\rho u_y - \tfrac{1}{12}\rho u_z + \tfrac{1}{12}\rho u_y^2 + \tfrac{1}{12}\rho u_z^2 + \tfrac{1}{4}\rho u_y u_z - \tfrac{1}{4}\rho u_y^2 u_z - \tfrac{1}{4}\rho u_y u_z^2 + \tfrac{1}{4}\rho u_y^2 u_z^2 \\
f^{eq}_9 &= \tfrac{1}{36}\rho - \tfrac{1}{12}\rho u_y + \tfrac{1}{12}\rho u_z + \tfrac{1}{12}\rho u_y^2 + \tfrac{1}{12}\rho u_z^2 - \tfrac{1}{4}\rho u_y u_z + \tfrac{1}{4}\rho u_y^2 u_z - \tfrac{1}{4}\rho u_y u_z^2 + \tfrac{1}{4}\rho u_y^2 u_z^2 \\
f^{eq}_{10} &= \tfrac{1}{18}\rho + \tfrac{1}{6}\rho u_x + \tfrac{1}{6}\rho u_x^2 - \tfrac{1}{6}\rho u_y^2 - \tfrac{1}{6}\rho u_z^2 - \tfrac{1}{2}\rho u_x u_y^2 - \tfrac{1}{2}\rho u_x u_z^2 - \tfrac{1}{2}\rho u_x^2 u_y^2 - \tfrac{1}{2}\rho u_x^2 u_z^2 \\
f^{eq}_{11} &= \tfrac{1}{18}\rho + \tfrac{1}{6}\rho u_y - \tfrac{1}{6}\rho u_x^2 + \tfrac{1}{6}\rho u_y^2 - \tfrac{1}{6}\rho u_z^2 - \tfrac{1}{2}\rho u_x^2 u_y - \tfrac{1}{2}\rho u_y u_z^2 - \tfrac{1}{2}\rho u_x^2 u_y^2 - \tfrac{1}{2}\rho u_y^2 u_z^2 \\
f^{eq}_{12} &= \tfrac{1}{18}\rho + \tfrac{1}{6}\rho u_z - \tfrac{1}{6}\rho u_x^2 - \tfrac{1}{6}\rho u_y^2 + \tfrac{1}{6}\rho u_z^2 - \tfrac{1}{2}\rho u_x^2 u_z - \tfrac{1}{2}\rho u_y^2 u_z - \tfrac{1}{2}\rho u_x^2 u_z^2 - \tfrac{1}{2}\rho u_y^2 u_z^2 \\
f^{eq}_{13} &= \tfrac{1}{36}\rho + \tfrac{1}{12}\rho u_x + \tfrac{1}{12}\rho u_y + \tfrac{1}{12}\rho u_x^2 + \tfrac{1}{12}\rho u_y^2 + \tfrac{1}{4}\rho u_x u_y + \tfrac{1}{4}\rho u_x^2 u_y + \tfrac{1}{4}\rho u_x u_y^2 + \tfrac{1}{4}\rho u_x^2 u_y^2 \\
f^{eq}_{14} &= \tfrac{1}{36}\rho + \tfrac{1}{12}\rho u_x - \tfrac{1}{12}\rho u_y + \tfrac{1}{12}\rho u_x^2 + \tfrac{1}{12}\rho u_y^2 - \tfrac{1}{4}\rho u_x u_y - \tfrac{1}{4}\rho u_x^2 u_y + \tfrac{1}{4}\rho u_x u_y^2 + \tfrac{1}{4}\rho u_x^2 u_y^2 \\
f^{eq}_{15} &= \tfrac{1}{36}\rho + \tfrac{1}{12}\rho u_x + \tfrac{1}{12}\rho u_z + \tfrac{1}{12}\rho u_x^2 + \tfrac{1}{12}\rho u_z^2 + \tfrac{1}{4}\rho u_x u_z + \tfrac{1}{4}\rho u_x^2 u_z + \tfrac{1}{4}\rho u_x u_z^2 + \tfrac{1}{4}\rho u_x^2 u_z^2 \\
f^{eq}_{16} &= \tfrac{1}{36}\rho + \tfrac{1}{12}\rho u_x - \tfrac{1}{12}\rho u_z + \tfrac{1}{12}\rho u_x^2 + \tfrac{1}{12}\rho u_z^2 - \tfrac{1}{4}\rho u_x u_z - \tfrac{1}{4}\rho u_x^2 u_z + \tfrac{1}{4}\rho u_x u_z^2 + \tfrac{1}{4}\rho u_x^2 u_z^2 \\
f^{eq}_{17} &= \tfrac{1}{36}\rho + \tfrac{1}{12}\rho u_y + \tfrac{1}{12}\rho u_z + \tfrac{1}{12}\rho u_y^2 + \tfrac{1}{12}\rho u_z^2 + \tfrac{1}{4}\rho u_y u_z + \tfrac{1}{4}\rho u_y^2 u_z + \tfrac{1}{4}\rho u_y u_z^2 + \tfrac{1}{4}\rho u_y^2 u_z^2 \\
f^{eq}_{18} &= \tfrac{1}{36}\rho + \tfrac{1}{12}\rho u_y - \tfrac{1}{12}\rho u_z + \tfrac{1}{12}\rho u_y^2 + \tfrac{1}{12}\rho u_z^2 - \tfrac{1}{4}\rho u_y u_z - \tfrac{1}{4}\rho u_y^2 u_z + \tfrac{1}{4}\rho u_y u_z^2 + \tfrac{1}{4}\rho u_y^2 u_z^2
\end{aligned}$$

The relaxation frequencies can be expressed by the following block diagonal matrix:

$$\mathbf{\Lambda} = \mathrm{diag}\left(1, 1, 1, 1, \tau_f^{-1}, \tau_f^{-1}, \tau_f^{-1}, \begin{bmatrix} s_+ & s_- & s_- \\ s_- & s_+ & s_- \\ s_- & s_- & s_+ \end{bmatrix}, \omega_3, \omega_3, \omega_3, \omega_3, \omega_3, \omega_3, \omega_4, \omega_4, \omega_4\right)$$

where $s_+ = \frac{1}{3}\left(\tau_{f,bulk}^{-1} + 2\tau_f^{-1}\right)$ and $s_- = \frac{1}{3}\left(\tau_{f,bulk}^{-1} - \tau_f^{-1}\right)$. The bulk viscosity can be related to the associated relaxation time by:

$$\nu' = \frac{2}{9}\left(\tau_{f,bulk} - \frac{1}{2}\right)\frac{(\Delta x)^2}{\Delta t}.$$

Guo forcing can be applied using the following central moment transformations of the associated source terms:

$$\vec{S}^m = \begin{pmatrix} 0 \\ F_x \\ F_y \\ F_z \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ \left(\frac{1}{3} - v_y^2\right)F_x - 2v_xv_yF_y \\ \left(\frac{1}{3} - v_z^2\right)F_x - 2v_xv_zF_z \\ \left(\frac{1}{3} - v_x^2\right)F_y - 2v_xv_yF_x \\ \left(\frac{1}{3} - v_x^2\right)F_z - 2v_xv_zF_x \\ \left(\frac{1}{3} - v_z^2\right)F_y - 2v_yv_zF_z \\ \left(\frac{1}{3} - v_y^2\right)F_z - 2v_yv_zF_y \\ 4v_xv_y\left(v_yF_x + v_xF_y\right) - \frac{1}{3}v_zF_z \\ 4v_xv_z\left(v_zF_x + v_xF_z\right) - \frac{1}{3}v_yF_y \\ 4v_yv_z\left(v_zF_y + v_yF_z\right) - \frac{1}{3}v_xF_x \end{pmatrix},$$

and He forcing can be applied using these central moment terms:

$$\vec{S}^m = \begin{pmatrix} 0 \\ F_x \\ F_y \\ F_z \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ \frac{1}{3}F_x \\ \frac{1}{3}F_x \\ \frac{1}{3}F_y \\ \frac{1}{3}F_z \\ \frac{1}{3}F_y \\ \frac{1}{3}F_z \\ 0 \\ 0 \\ 0 \end{pmatrix}.$$

D.4 D3Q27

Speed vectors

i	$e_{i,x}$	$e_{i,y}$	$e_{i,z}$
0	0	0	0
1	-1	0	0
2	0	-1	0
3	0	0	-1
4	-1	-1	0
5	-1	1	0
6	-1	0	-1
7	-1	0	1
8	0	-1	-1
9	0	-1	1
10	-1	-1	-1
11	-1	-1	1
12	-1	1	-1
13	-1	1	1
14	1	0	0
15	0	1	0
16	0	0	1
17	1	1	0
18	1	-1	0
19	1	0	1
20	1	0	-1
21	0	1	1
22	0	1	-1
23	1	1	1
24	1	1	-1
25	1	-1	1
26	1	-1	-1

Weight factors

i	w_i
0	$\frac{8}{27}$
1–3, 14–16	$\frac{2}{27}$
4–9, 17–22	$\frac{1}{54}$
10–13, 23–26	$\frac{1}{216}$

Multiple relaxation time scheme

Definition of transformation matrix based on [124]:

$$
\mathbf{T} = \left[\begin{array}{cccccccccccccccccccccccccccccccc}
1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 \\
0 & -1 & 0 & -1 & 0 & 0 & -1 & -1 & -1 & -1 & -1 & -1 & 0 & 0 & 1 & 1 & 1 & 1 & 0 & 0 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 \\
0 & 0 & -1 & 0 & -1 & 1 & 0 & -1 & -1 & -1 & 1 & 1 & 0 & 1 & 0 & 0 & -1 & 0 & 0 & 1 & 1 & 1 & 1 & -1 & -1 & -1 & -1 & -1 & -1 & -1 \\
0 & 0 & -1 & 0 & 0 & 0 & -1 & 1 & -1 & 1 & 1 & 0 & 0 & 1 & 0 & 0 & 1 & -1 & 0 & 0 & 0 & 1 & 1 & -1 & 1 & 1 & 1 & 1 & -1 & -1 \\
-2 & -1 & -1 & -1 & 0 & 0 & 0 & 0 & 1 & 1 & 1 & -1 & -1 & 0 & 0 & 0 & 0 & 1 & 1 & 1 & -2 & -2 & 0 & 0 & 1 & 1 & 1 & 1 & 1 & 1 \\
0 & 2 & -1 & -1 & 1 & 1 & 1 & -2 & -2 & 0 & 0 & 0 & 2 & -1 & -1 & 1 & 1 & 1 & -1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 1 & -1 & 1 & 1 & -1 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & -1 & 1 & 1 & 1 & -1 & -1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 1 & -1 & 0 & 0 & 0 & 0 & 1 & -1 & 0 & 0 & 0 & 1 & -1 & 0 & 0 & 0 & 0 & 1 & 1 & 1 & -1 & -1 & -1 & -1 & -1 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & -1 & 1 & 1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & -1 & -1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 4 & 0 & 0 & 0 & 1 & 1 & 0 & 0 & -2 & -2 & -4 & 0 & 0 & -1 & -1 & -1 & -1 & 0 & 0 & -1 & -1 & 0 & 0 & 2 & 2 & 2 & 2 & 2 & 2 \\
0 & 0 & 0 & 4 & 0 & 1 & -1 & 0 & 0 & 1 & -2 & -2 & 2 & 0 & -4 & 0 & -1 & 1 & 0 & 0 & -1 & -1 & 2 & 2 & 2 & -2 & -2 & -2 & -2 & -2 \\
0 & 0 & 0 & 0 & 4 & 0 & 0 & 1 & -1 & 1 & -1 & -2 & 2 & 0 & 0 & -4 & 0 & 0 & -1 & 1 & -1 & 1 & 2 & 2 & -2 & 2 & 2 & 2 & 2 & 2 \\
0 & -4 & 0 & 0 & 0 & 2 & 2 & 2 & 0 & 0 & -1 & -1 & -1 & 4 & 0 & 0 & -2 & -2 & -2 & 0 & 0 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 \\
0 & 0 & -4 & 0 & 0 & 2 & -2 & 0 & 0 & 2 & 2 & -1 & -1 & 1 & 1 & 0 & 4 & 0 & -2 & 2 & 0 & 0 & -2 & -2 & 1 & 1 & -1 & -1 & -1 & -1 \\
0 & 0 & 0 & -4 & 0 & 0 & 2 & -2 & 2 & -2 & -2 & -1 & 1 & -1 & 1 & 0 & 0 & 4 & 0 & 0 & -2 & 2 & -2 & 2 & 1 & -1 & 1 & -1 & -1 & -1 \\
4 & 0 & 0 & 0 & -1 & -1 & -1 & -1 & -1 & 1 & 1 & 1 & 1 & 1 & 0 & 0 & 0 & 4 & 4 & -2 & -2 & -2 & -1 & -1 & -1 & 1 & 1 & 1 & 1 & 1 \\
-8 & 4 & 4 & 4 & -2 & -2 & -2 & -2 & -2 & 1 & 1 & 1 & 1 & 1 & 4 & 4 & 4 & -2 & -2 & -2 & -2 & -2 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 \\
0 & -4 & 2 & 2 & 1 & 1 & 1 & 1 & -2 & -2 & 0 & 0 & 0 & 0 & -4 & 2 & 2 & 1 & 1 & 1 & 1 & -2 & -2 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & -2 & 2 & 1 & 1 & -1 & -1 & 0 & 0 & 0 & 0 & 1 & 1 & 0 & 0 & -2 & 2 & 1 & 1 & -1 & -1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & -2 & 2 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & -1 & -1 & 1 & 0 & -2 & 2 & 0 & 0 & 0 & -2 & 2 & 0 & 0 & 1 & 1 & -1 & -1 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & -2 & 2 & 0 & 0 & 0 & 1 & 1 & -1 & -1 & 1 & 0 & 0 & 0 & 0 & 0 & -2 & 2 & 1 & -1 & -1 & 1 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & -2 & 2 & 0 & 0 & 0 & 1 & -1 & -1 & 1 & 1 & -1 & 0 & 0 & -2 & 2 & 0 & 0 & 1 & -1 & 1 & -1 & -1 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & -1 & -1 & 1 & 1 & 1 & 0 & 0 & 0 & 0 & 0 & 1 & 1 & -1 & -1 & 0 & 0 & -1 & -1 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & -1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 1 & 1 & 1 & 0 & 0 & 1 & 1 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0
\end{array}\right]
$$

For the standard local equilibrium distribution functions, the equilibrium moments are expressed for incompressible fluids as:

$$\vec{M}^{eq} = \begin{pmatrix} \rho \\ j_x \\ j_y \\ j_z \\ e^{eq} \\ 3p_{xx}^{eq} \\ p_{ww}^{eq} \\ p_{xy}^{eq} \\ p_{yz}^{eq} \\ p_{zx}^{eq} \\ q_x^{eq} \\ q_y^{eq} \\ q_z^{eq} \\ \kappa_x^{eq} \\ \kappa_y^{eq} \\ \kappa_z^{eq} \\ \epsilon^{eq} \\ e_3^{eq} \\ 3\pi_{xx}^{eq} \\ \pi_{ww}^{eq} \\ \pi_{xy}^{eq} \\ \pi_{yz}^{eq} \\ \pi_{zx}^{eq} \\ \tau_x^{eq} \\ \tau_y^{eq} \\ \tau_z^{eq} \\ q_{xyz}^{eq} \end{pmatrix} = \begin{pmatrix} \rho \\ j_x \\ j_y \\ j_z \\ -\rho + \frac{1}{\rho_0}(j_x^2 + j_y^2 + j_z^2) \\ \frac{2j_x^2 - j_y^2 - j_z^2}{\rho_0} \\ \frac{j_y^2 - j_z^2}{\rho_0} \\ \frac{j_x j_y}{\rho_0} \\ \frac{j_y j_z}{\rho_0} \\ \frac{j_z j_x}{\rho_0} \\ -2j_x \\ -2j_y \\ -2j_z \\ j_x \\ j_y \\ j_z \\ \rho - \frac{2}{\rho_0}(j_x^2 + j_y^2 + j_z^2) \\ -\rho + \frac{1}{\rho_0}(j_x^2 + j_y^2 + j_z^2) \\ -\frac{2j_x^2 - j_y^2 - j_z^2}{\rho_0} \\ -\frac{j_y^2 - j_z^2}{\rho_0} \\ -\frac{j_x j_y}{\rho_0} \\ -\frac{j_y j_z}{\rho_0} \\ -\frac{j_z j_x}{\rho_0} \\ 0 \\ 0 \\ 0 \\ 0 \end{pmatrix}$$

where ρ is used in place of ρ_0 for mildly compressible fluids, $j_x = \rho_0 u_x$, $j_y = \rho_0 u_y$ and $j_z = \rho_0 u_z$.

The relaxation frequencies for the above moments can be expressed by the following diagonal matrix:

$$\vec{s} = \text{diag}\left(1, 1, 1, 1, \tau_{f,bulk}^{-1}, \tau_f^{-1}, \tau_f^{-1}, \tau_f^{-1}, \tau_f^{-1}, \tau_f^{-1}, \tau_f^{-1}, s_{10}, s_{10}, s_{10}, s_{13}, s_{13}, s_{13}, s_{16}, s_{17}, s_{18}, s_{18}, s_{20}, s_{20}, s_{20}, s_{23}, s_{23}, s_{23}, s_{26}\right)$$

where the bulk viscosity can be related to the associated relaxation time by:

$$\nu' = \frac{2}{9}\left(\tau_{f,bulk} - \frac{1}{2}\right)\frac{(\Delta x)^2}{\Delta t}.$$

Recommended default values for the eight variable relaxation frequencies are: $s_{10} = 1.5$, $s_{13} = 1.83$, $s_{16} = 1.4$, $s_{17} = 1.61$, $s_{18} = s_{20} = 1.98$ and $s_{23} = s_{26} = 1.74$.

Guo forcing can be applied using the following moment transformations of the associated source terms:

$$\vec{S}^m = \begin{pmatrix} 0 \\ F_x \\ F_y \\ F_z \\ 2(v_x F_x + v_y F_y + v_z F_z) \\ 2(2v_x F_x - v_y F_y - v_z F_z) \\ 2(v_y F_y - v_z F_z) \\ v_y F_x + v_x F_y \\ v_z F_y + v_y F_z \\ v_x F_z + v_z F_x \\ -2F_x \\ -2F_y \\ -2F_z \\ F_x \\ F_y \\ F_z \\ -4(v_x F_x + v_y F_y + v_z F_z) \\ 6(v_x F_x + v_y F_y + v_z F_z) \\ -2(2v_x F_x - v_y F_y - v_z F_z) \\ -2(v_y F_y - v_z F_z) \\ -(v_y F_x + v_x F_y) \\ -(v_z F_y + v_y F_z) \\ -(v_x F_z + v_z F_x) \\ 0 \\ 0 \\ 0 \\ 0 \end{pmatrix},$$

and He forcing can be applied using these moment terms:

$$\vec{S}^m = \begin{pmatrix} 0 \\ F_x \\ F_y \\ F_z \\ 2(v_x F_x + v_y F_y + v_z F_z) \\ 2(2v_x F_x - v_y F_y - v_z F_z) \\ 2(v_y F_y - v_z F_z) \\ v_y F_x + v_x F_y \\ v_z F_y + v_y F_z \\ v_x F_z + v_z F_x \\ \left(-2 + 3v_y^2 + 3v_z^2\right) F_x + 6v_x v_y F_y + 6v_x v_z F_z \\ \left(-2 + 3v_x^2 + 3v_z^2\right) F_y + 6v_x v_y F_x + 6v_y v_z F_z \\ \left(-2 + 3v_x^2 + 3v_y^2\right) F_z + 6v_x v_y F_x + 6v_y v_z F_y \\ F_x \left(1 - 3v_y^2 - 3v_z^2\right) - +6v_x v_y F_y - 6v_x v_z F_z \\ F_y \left(1 - 3v_x^2 - 3v_z^2\right) - 6v_x v_y F_x - 6v_y v_z F_z \\ F_z \left(1 - 3v_x^2 - 3v_y^2\right) - 6v_x v_y F_x - 6v_y v_z F_y \\ -4(v_x F_x + v_y F_y + v_z F_z) \\ 6(v_x F_x + v_y F_y + v_z F_z) \\ -2(2v_x F_x - v_y F_y - v_z F_z) \\ -2(v_y F_y - v_z F_z) \\ -(v_y F_x + v_x F_y) \\ -(v_z F_y + v_y F_z) \\ -(v_x F_z + v_z F_x) \\ \left(v_y^2 - v_z^2\right) F_x + 2v_x v_y F_y - 2v_x v_z F_z \\ \left(v_z^2 - v_x^2\right) F_y + 2v_y v_z F_z - 2v_x v_y F_x \\ \left(v_x^2 - v_y^2\right) F_z + 2v_x v_z F_x - 2v_y v_z F_y \\ v_y v_z F_x + v_x v_z F_y + v_x v_y F_z \end{pmatrix}.$$

Cascaded LBE scheme

Definitions of transformation and shift matrices based on [30]:

$$
\mathbf{T} = \left[\begin{array}{ccccccccccccccccccccccccccc}
1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 \\
0 & -1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & -1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & -1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & -1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & -1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & -1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & -1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & -1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & -1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & -1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & -1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & -1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & -1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & -1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & -1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & -1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & -1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & -1 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & -1 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & -1 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & -1 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & -1 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & -1 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & -1
\end{array}\right]
$$

$$
\mathbf{M} = \left[\begin{array}{cccccccccccccccccccccccccccccccccccc}
1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
-u_x & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
-u_y & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
-u_z & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
u_x u_y & -u_y & -u_x & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
u_x u_z & -u_z & 0 & -u_x & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
u_y u_z & 0 & -u_z & -u_y & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
u_x^2 & -2u_x & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
u_y^2 & 0 & -2u_y & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
u_z^2 & 0 & 0 & -2u_z & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
-u_x u_y^2 & u_y^2 & 2u_x u_y & 0 & -2u_y & 0 & 0 & 0 & -u_x & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
-u_x u_z^2 & u_z^2 & 2u_x u_z & 0 & -2u_x & 0 & 0 & -u_y & 0 & -u_x & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
-u_x^2 u_y & 2u_x u_y & u_x^2 & 0 & 0 & -2u_x & 0 & -u_z & 0 & 0 & -u_y & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
-u_x^2 u_z & 2u_x u_z & 0 & u_x^2 & 0 & 0 & -2u_x & 0 & -2u_y & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
-u_y u_z^2 & 0 & u_z^2 & 2u_y u_z & 0 & 0 & -2u_y & 0 & -u_x & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
-u_y^2 u_z & 0 & 2u_y u_z & u_y^2 & 0 & 0 & -u_z & 0 & 0 & -u_z & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
-u_x u_y u_z & u_y u_z & u_x u_z & u_x u_y & -u_z & -u_y & -u_x & 0 & 0 & 0 & 0 & 0 & -u_z & -u_y & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
u_x^2 u_y^2 & -2u_x^2 u_y^2 & -2u_x^2 u_y & 0 & 4u_x u_y & 0 & 0 & u_y^2 & 0 & u_x^2 & 0 & -2u_x & 0 & -2u_y & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
u_x^2 u_z^2 & -2u_x^2 u_z^2 & 0 & -2u_x^2 u_z & 0 & 4u_x u_z & 0 & u_z^2 & 0 & 0 & -2u_x & 0 & 0 & 0 & -2u_z & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
u_y^2 u_z^2 & 0 & -2u_y^2 u_z^2 & -2u_y^2 u_z & 0 & 0 & 4u_y u_z & 0 & u_y^2 & u_y u_z & 0 & 0 & 0 & u_y^2 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
u_x^2 u_y u_z & -2u_x u_y^2 u_z & -u_x^2 u_z & -u_x^2 u_y & 2u_x u_z & 2u_x u_y & u_x^2 & 0 & 0 & u_x u_z & 0 & 0 & 0 & 0 & 2u_x u_y & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
u_x u_y^2 u_z & -u_y^2 u_z & -2u_x u_y u_z & -u_x u_y^2 & 2u_y u_z & u_y^2 & 2u_x u_y & 0 & u_x u_z & 0 & u_x u_y & 0 & 0 & 0 & u_y^2 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
u_x u_y u_z^2 & -u_y u_z^2 & -u_x u_z^2 & -2u_x u_y u_z & u_z^2 & 2u_y u_z & 2u_x u_z & 0 & -u_x u_z^2 & -u_x u_y u_z & 0 & u_z^2 & u_y^2 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
-u_x u_y^2 u_z^2 & u_y^2 u_z^2 & 2u_x u_y u_z^2 & 2u_x u_y^2 u_z & -2u_y u_z^2 & -2u_y^2 u_z & -4u_x u_y u_z & 0 & -u_y u_z^2 & -u_y^2 u_z & 0 & 0 & 0 & u_z^2 & 2u_y u_z & u_y^2 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
-u_x^2 u_y^2 u_z^2 & 2u_x^2 u_y^2 u_z & u_x^2 u_z^2 & 2u_x^2 u_y u_z & -2u_x^2 u_z & -2u_x^2 u_y & -2u_x^2 u_y & 0 & -u_x^2 u_z & -u_x^2 u_y & 0 & 2u_x u_z & 0 & 2u_y u_z & u_y^2 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
-u_x^2 u_y^2 u_z & 2u_x^2 u_y^2 u_z & 2u_x^2 u_y u_z & u_x^2 u_y^2 & -2u_x^2 u_y & -4u_x u_y u_z & -2u_x^2 u_y & 0 & -u_x^2 u_z & -u_x^2 u_y & 0 & 2u_x u_z & 0 & 2u_y u_z & u_y^2 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
u_z^2 u_y^2 u_z^2 & -2u_x u_y^2 u_z^2 & -2u_x^2 u_y u_z^2 & -2u_x^2 u_y^2 u_z & 4u_x u_y^2 u_z^2 & 4u_x u_y^2 u_z & 4u_x^2 u_y^2 u_z & 0 & u_x^2 u_z^2 & u_x^2 u_y^2 & 0 & -2u_x u_z^2 & -2u_x^2 u_y^2 u_z^2 & -2u_y^2 u_z^2 & -2u_x^2 u_y & -2u_x^2 u_z & -8u_x u_y u_z & u_z^2 & u_y^2 & u_x^2 & 4u_y u_z & 4u_x u_z & 4u_x u_y & -2u_x & -2u_y & -2u_z & 1
\end{array}\right]
$$

The equilibrium central moments are expressed as follows:

$$\vec{\tilde{M}}^{eq} = \begin{pmatrix} \tilde{M}^{eq}_{000} \\ \tilde{M}^{eq}_{100} \\ \tilde{M}^{eq}_{010} \\ \tilde{M}^{eq}_{001} \\ \tilde{M}^{eq}_{110} \\ \tilde{M}^{eq}_{101} \\ \tilde{M}^{eq}_{011} \\ \tilde{M}^{eq}_{200} \\ \tilde{M}^{eq}_{020} \\ \tilde{M}^{eq}_{002} \\ \tilde{M}^{eq}_{120} \\ \tilde{M}^{eq}_{102} \\ \tilde{M}^{eq}_{210} \\ \tilde{M}^{eq}_{201} \\ \tilde{M}^{eq}_{012} \\ \tilde{M}^{eq}_{021} \\ \tilde{M}^{eq}_{111} \\ \tilde{M}^{eq}_{220} \\ \tilde{M}^{eq}_{202} \\ \tilde{M}^{eq}_{022} \\ \tilde{M}^{eq}_{211} \\ \tilde{M}^{eq}_{121} \\ \tilde{M}^{eq}_{112} \\ \tilde{M}^{eq}_{122} \\ \tilde{M}^{eq}_{212} \\ \tilde{M}^{eq}_{221} \\ \tilde{M}^{eq}_{222} \end{pmatrix} = \begin{pmatrix} \rho \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ \frac{1}{3}\rho \\ \frac{1}{3}\rho \\ \frac{1}{3}\rho \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ \frac{1}{9}\rho \\ \frac{1}{9}\rho \\ \frac{1}{9}\rho \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ \frac{1}{27}\rho \end{pmatrix}$$

and transformation of the above leads to the following expressions for the local equilibrium distribution functions:

$$
\begin{aligned}
f_0^{eq} &= \tfrac{8}{27}\rho - \tfrac{4}{9}\rho u_x^2 - \tfrac{4}{9}\rho u_y^2 - \tfrac{4}{9}\rho u_z^2 + \tfrac{2}{3}\rho u_x^2 u_y^2 + \tfrac{2}{3}\rho u_x^2 u_z^2 + \tfrac{2}{3}\rho u_y^2 u_z^2 - \rho u_x^2 u_y^2 u_z^2 \\
f_1^{eq} &= \tfrac{2}{27}\rho - \tfrac{2}{9}\rho u_x + \tfrac{2}{9}\rho u_x^2 - \tfrac{1}{9}\rho u_y^2 - \tfrac{1}{9}\rho u_z^2 + \tfrac{1}{3}\rho u_x u_y^2 + \tfrac{1}{3}\rho u_x u_z^2 - \tfrac{1}{3}\rho u_x^2 u_y^2 - \tfrac{1}{3}\rho u_x^2 u_z^2 + \tfrac{1}{6}\rho u_y^2 u_z^2 - \tfrac{1}{2}\rho u_x u_y^2 u_z^2 + \tfrac{1}{2}\rho u_x^2 u_y^2 u_z^2 \\
f_2^{eq} &= \tfrac{2}{27}\rho - \tfrac{2}{9}\rho u_y - \tfrac{1}{9}\rho u_x^2 + \tfrac{2}{9}\rho u_y^2 - \tfrac{1}{9}\rho u_z^2 + \tfrac{1}{3}\rho u_x^2 u_y + \tfrac{1}{3}\rho u_y u_z^2 - \tfrac{1}{3}\rho u_x^2 u_y^2 + \tfrac{1}{6}\rho u_x^2 u_z^2 - \tfrac{1}{3}\rho u_y^2 u_z^2 - \tfrac{1}{2}\rho u_x^2 u_y u_z^2 + \tfrac{1}{2}\rho u_x^2 u_y^2 u_z^2 \\
f_3^{eq} &= \tfrac{2}{27}\rho - \tfrac{2}{9}\rho u_z - \tfrac{1}{9}\rho u_x^2 - \tfrac{1}{9}\rho u_y^2 + \tfrac{2}{9}\rho u_z^2 + \tfrac{1}{3}\rho u_x^2 u_z + \tfrac{1}{3}\rho u_y^2 u_z + \tfrac{1}{6}\rho u_x^2 u_y^2 - \tfrac{1}{3}\rho u_x^2 u_z^2 - \tfrac{1}{3}\rho u_y^2 u_z^2 - \tfrac{1}{2}\rho u_x^2 u_y^2 u_z + \tfrac{1}{2}\rho u_x^2 u_y^2 u_z^2 \\
f_4^{eq} &= \tfrac{1}{54}\rho - \tfrac{1}{18}\rho u_x - \tfrac{1}{18}\rho u_y + \tfrac{1}{18}\rho u_x^2 + \tfrac{1}{18}\rho u_y^2 - \tfrac{1}{36}\rho u_z^2 + \tfrac{1}{6}\rho u_x u_y - \tfrac{1}{6}\rho u_x u_y^2 - \tfrac{1}{6}\rho u_x^2 u_y + \tfrac{1}{12}\rho u_x u_z^2 + \tfrac{1}{12}\rho u_y u_z^2 \\
&+ \tfrac{1}{6}\rho u_x^2 u_y^2 - \tfrac{1}{12}\rho u_x^2 u_z^2 - \tfrac{1}{12}\rho u_y^2 u_z^2 - \tfrac{1}{4}\rho u_x u_y u_z^2 + \tfrac{1}{4}\rho u_x u_y^2 u_z^2 + \tfrac{1}{4}\rho u_x^2 u_y u_z^2 - \tfrac{1}{4}\rho u_x^2 u_y^2 u_z^2 \\
f_5^{eq} &= \tfrac{1}{54}\rho - \tfrac{1}{18}\rho u_x + \tfrac{1}{18}\rho u_y + \tfrac{1}{18}\rho u_x^2 + \tfrac{1}{18}\rho u_y^2 - \tfrac{1}{36}\rho u_z^2 - \tfrac{1}{6}\rho u_x u_y - \tfrac{1}{6}\rho u_x u_y^2 + \tfrac{1}{6}\rho u_x^2 u_y + \tfrac{1}{12}\rho u_x u_z^2 - \tfrac{1}{12}\rho u_y u_z^2 \\
&+ \tfrac{1}{6}\rho u_x^2 u_y^2 - \tfrac{1}{12}\rho u_x^2 u_z^2 - \tfrac{1}{12}\rho u_y^2 u_z^2 + \tfrac{1}{4}\rho u_x u_y u_z^2 + \tfrac{1}{4}\rho u_x u_y^2 u_z^2 - \tfrac{1}{4}\rho u_x^2 u_y u_z^2 - \tfrac{1}{4}\rho u_x^2 u_y^2 u_z^2 \\
f_6^{eq} &= \tfrac{1}{54}\rho - \tfrac{1}{18}\rho u_x - \tfrac{1}{18}\rho u_z + \tfrac{1}{18}\rho u_x^2 - \tfrac{1}{36}\rho u_y^2 + \tfrac{1}{18}\rho u_z^2 + \tfrac{1}{6}\rho u_x u_z - \tfrac{1}{6}\rho u_x u_z^2 - \tfrac{1}{6}\rho u_x^2 u_z + \tfrac{1}{12}\rho u_x u_y^2 + \tfrac{1}{12}\rho u_y^2 u_z \\
&- \tfrac{1}{12}\rho u_x^2 u_y^2 + \tfrac{1}{6}\rho u_x^2 u_z^2 - \tfrac{1}{12}\rho u_y^2 u_z^2 - \tfrac{1}{4}\rho u_x u_y^2 u_z + \tfrac{1}{4}\rho u_x^2 u_y^2 u_z + \tfrac{1}{4}\rho u_x u_y^2 u_z^2 - \tfrac{1}{4}\rho u_x^2 u_y^2 u_z^2 \\
f_7^{eq} &= \tfrac{1}{54}\rho - \tfrac{1}{18}\rho u_x + \tfrac{1}{18}\rho u_z + \tfrac{1}{18}\rho u_x^2 - \tfrac{1}{36}\rho u_y^2 + \tfrac{1}{18}\rho u_z^2 - \tfrac{1}{6}\rho u_x u_z - \tfrac{1}{6}\rho u_x u_z^2 + \tfrac{1}{6}\rho u_x^2 u_z + \tfrac{1}{12}\rho u_x u_y^2 - \tfrac{1}{12}\rho u_y^2 u_z \\
&- \tfrac{1}{12}\rho u_x^2 u_y^2 + \tfrac{1}{6}\rho u_x^2 u_z^2 - \tfrac{1}{12}\rho u_y^2 u_z^2 + \tfrac{1}{4}\rho u_x u_y^2 u_z - \tfrac{1}{4}\rho u_x^2 u_y^2 u_z + \tfrac{1}{4}\rho u_x u_y^2 u_z^2 - \tfrac{1}{4}\rho u_x^2 u_y^2 u_z^2 \\
f_8^{eq} &= \tfrac{1}{54}\rho - \tfrac{1}{18}\rho u_y - \tfrac{1}{18}\rho u_z - \tfrac{1}{36}\rho u_x^2 + \tfrac{1}{18}\rho u_y^2 + \tfrac{1}{18}\rho u_z^2 + \tfrac{1}{6}\rho u_y u_z - \tfrac{1}{6}\rho u_y^2 u_z - \tfrac{1}{6}\rho u_y u_z^2 + \tfrac{1}{12}\rho u_x^2 u_y + \tfrac{1}{12}\rho u_x^2 u_z \\
&- \tfrac{1}{12}\rho u_x^2 u_y^2 - \tfrac{1}{12}\rho u_x^2 u_z^2 + \tfrac{1}{6}\rho u_y^2 u_z^2 - \tfrac{1}{4}\rho u_x^2 u_y u_z + \tfrac{1}{4}\rho u_x^2 u_y^2 u_z + \tfrac{1}{4}\rho u_x^2 u_y u_z^2 - \tfrac{1}{4}\rho u_x^2 u_y^2 u_z^2 \\
f_9^{eq} &= \tfrac{1}{54}\rho - \tfrac{1}{18}\rho u_y + \tfrac{1}{18}\rho u_z - \tfrac{1}{36}\rho u_x^2 + \tfrac{1}{18}\rho u_y^2 + \tfrac{1}{18}\rho u_z^2 - \tfrac{1}{6}\rho u_y u_z + \tfrac{1}{6}\rho u_y^2 u_z - \tfrac{1}{6}\rho u_y u_z^2 + \tfrac{1}{12}\rho u_x^2 u_y - \tfrac{1}{12}\rho u_x^2 u_z \\
&- \tfrac{1}{12}\rho u_x^2 u_y^2 - \tfrac{1}{12}\rho u_x^2 u_z^2 + \tfrac{1}{6}\rho u_y^2 u_z^2 + \tfrac{1}{4}\rho u_x^2 u_y u_z - \tfrac{1}{4}\rho u_x^2 u_y^2 u_z + \tfrac{1}{4}\rho u_x^2 u_y u_z^2 - \tfrac{1}{4}\rho u_x^2 u_y^2 u_z^2 \\
f_{10}^{eq} &= \tfrac{1}{216}\rho - \tfrac{1}{72}\rho u_x - \tfrac{1}{72}\rho u_y - \tfrac{1}{72}\rho u_z + \tfrac{1}{72}\rho u_x^2 + \tfrac{1}{72}\rho u_y^2 + \tfrac{1}{72}\rho u_z^2 + \tfrac{1}{24}\rho u_x u_y + \tfrac{1}{24}\rho u_x u_z + \tfrac{1}{24}\rho u_y u_z \\
&- \tfrac{1}{24}\rho u_x^2 u_y - \tfrac{1}{24}\rho u_x^2 u_z - \tfrac{1}{24}\rho u_x u_y^2 - \tfrac{1}{24}\rho u_y^2 u_z - \tfrac{1}{24}\rho u_x u_z^2 - \tfrac{1}{24}\rho u_y u_z^2 - \tfrac{1}{8}\rho u_x u_y u_z \\
&+ \tfrac{1}{24}\rho u_x^2 u_y^2 + \tfrac{1}{24}\rho u_x^2 u_z^2 + \tfrac{1}{24}\rho u_y^2 u_z^2 + \tfrac{1}{8}\rho u_x^2 u_y u_z + \tfrac{1}{8}\rho u_x u_y^2 u_z + \tfrac{1}{8}\rho u_x u_y u_z^2 \\
&- \tfrac{1}{8}\rho u_x^2 u_y^2 u_z - \tfrac{1}{8}\rho u_x^2 u_y u_z^2 - \tfrac{1}{8}\rho u_x u_y^2 u_z^2 + \tfrac{1}{8}\rho u_x^2 u_y^2 u_z^2 \\
f_{11}^{eq} &= \tfrac{1}{216}\rho - \tfrac{1}{72}\rho u_x - \tfrac{1}{72}\rho u_y + \tfrac{1}{72}\rho u_z + \tfrac{1}{72}\rho u_x^2 + \tfrac{1}{72}\rho u_y^2 + \tfrac{1}{72}\rho u_z^2 + \tfrac{1}{24}\rho u_x u_y - \tfrac{1}{24}\rho u_x u_z - \tfrac{1}{24}\rho u_y u_z \\
&- \tfrac{1}{24}\rho u_x^2 u_y + \tfrac{1}{24}\rho u_x^2 u_z - \tfrac{1}{24}\rho u_x u_y^2 + \tfrac{1}{24}\rho u_y^2 u_z - \tfrac{1}{24}\rho u_x u_z^2 - \tfrac{1}{24}\rho u_y u_z^2 + \tfrac{1}{8}\rho u_x u_y u_z \\
&+ \tfrac{1}{24}\rho u_x^2 u_y^2 + \tfrac{1}{24}\rho u_x^2 u_z^2 + \tfrac{1}{24}\rho u_y^2 u_z^2 - \tfrac{1}{8}\rho u_x^2 u_y u_z - \tfrac{1}{8}\rho u_x u_y^2 u_z + \tfrac{1}{8}\rho u_x u_y u_z^2 \\
&+ \tfrac{1}{8}\rho u_x^2 u_y^2 u_z - \tfrac{1}{8}\rho u_x^2 u_y u_z^2 - \tfrac{1}{8}\rho u_x u_y^2 u_z^2 + \tfrac{1}{8}\rho u_x^2 u_y^2 u_z^2
\end{aligned}
$$

$$
\begin{aligned}
f_{12}^{eq} &= \tfrac{1}{216}\rho - \tfrac{1}{72}\rho u_x + \tfrac{1}{72}\rho u_y - \tfrac{1}{72}\rho u_z + \tfrac{1}{72}\rho u_x^2 + \tfrac{1}{72}\rho u_y^2 + \tfrac{1}{72}\rho u_z^2 - \tfrac{1}{24}\rho u_x u_y + \tfrac{1}{24}\rho u_x u_z - \tfrac{1}{24}\rho u_y u_z \\
&+ \tfrac{1}{24}\rho u_x^2 u_y - \tfrac{1}{24}\rho u_x^2 u_z - \tfrac{1}{24}\rho u_x u_y^2 - \tfrac{1}{24}\rho u_y^2 u_z - \tfrac{1}{24}\rho u_x u_z^2 + \tfrac{1}{24}\rho u_y u_z^2 + \tfrac{1}{8}\rho u_x u_y u_z \\
&+ \tfrac{1}{24}\rho u_x^2 u_y^2 + \tfrac{1}{24}\rho u_x^2 u_z^2 + \tfrac{1}{24}\rho u_y^2 u_z^2 - \tfrac{1}{8}\rho u_x^2 u_y u_z + \tfrac{1}{8}\rho u_x u_y^2 u_z - \tfrac{1}{8}\rho u_x u_y u_z^2 \\
&- \tfrac{1}{8}\rho u_x^2 u_y^2 u_z + \tfrac{1}{8}\rho u_x^2 u_y u_z^2 - \tfrac{1}{8}\rho u_x u_y^2 u_z^2 + \tfrac{1}{8}\rho u_x^2 u_y^2 u_z^2 \\
f_{13}^{eq} &= \tfrac{1}{216}\rho - \tfrac{1}{72}\rho u_x + \tfrac{1}{72}\rho u_y + \tfrac{1}{72}\rho u_z + \tfrac{1}{72}\rho u_x^2 + \tfrac{1}{72}\rho u_y^2 + \tfrac{1}{72}\rho u_z^2 - \tfrac{1}{24}\rho u_x u_y - \tfrac{1}{24}\rho u_x u_z + \tfrac{1}{24}\rho u_y u_z \\
&+ \tfrac{1}{24}\rho u_x^2 u_y + \tfrac{1}{24}\rho u_x^2 u_z - \tfrac{1}{24}\rho u_x u_y^2 + \tfrac{1}{24}\rho u_y^2 u_z - \tfrac{1}{24}\rho u_x u_z^2 + \tfrac{1}{24}\rho u_y u_z^2 - \tfrac{1}{8}\rho u_x u_y u_z \\
&+ \tfrac{1}{24}\rho u_x^2 u_y^2 + \tfrac{1}{24}\rho u_x^2 u_z^2 + \tfrac{1}{24}\rho u_y^2 u_z^2 + \tfrac{1}{8}\rho u_x^2 u_y u_z - \tfrac{1}{8}\rho u_x u_y^2 u_z - \tfrac{1}{8}\rho u_x u_y u_z^2 \\
&+ \tfrac{1}{8}\rho u_x^2 u_y^2 u_z + \tfrac{1}{8}\rho u_x^2 u_y u_z^2 - \tfrac{1}{8}\rho u_x u_y^2 u_z^2 + \tfrac{1}{8}\rho u_x^2 u_y^2 u_z^2 \\
f_{14}^{eq} &= \tfrac{2}{27}\rho + \tfrac{2}{9}\rho u_x + \tfrac{2}{9}\rho u_x^2 - \tfrac{1}{9}\rho u_y^2 - \tfrac{1}{9}\rho u_z^2 - \tfrac{1}{3}\rho u_x u_y^2 - \tfrac{1}{3}\rho u_x u_z^2 - \tfrac{1}{3}\rho u_x^2 u_y^2 - \tfrac{1}{3}\rho u_x^2 u_z^2 + \tfrac{1}{6}\rho u_y^2 u_z^2 + \tfrac{1}{2}\rho u_x u_y^2 u_z^2 + \tfrac{1}{2}\rho u_x^2 u_y^2 u_z^2 \\
f_{15}^{eq} &= \tfrac{2}{27}\rho + \tfrac{2}{9}\rho u_y - \tfrac{1}{9}\rho u_x^2 + \tfrac{2}{9}\rho u_y^2 - \tfrac{1}{9}\rho u_z^2 - \tfrac{1}{3}\rho u_x^2 u_y - \tfrac{1}{3}\rho u_y u_z^2 - \tfrac{1}{3}\rho u_x^2 u_y^2 + \tfrac{1}{6}\rho u_x^2 u_z^2 - \tfrac{1}{3}\rho u_y^2 u_z^2 + \tfrac{1}{2}\rho u_x^2 u_y u_z^2 + \tfrac{1}{2}\rho u_x^2 u_y^2 u_z^2 \\
f_{16}^{eq} &= \tfrac{2}{27}\rho + \tfrac{2}{9}\rho u_z - \tfrac{1}{9}\rho u_x^2 - \tfrac{1}{9}\rho u_y^2 + \tfrac{2}{9}\rho u_z^2 - \tfrac{1}{3}\rho u_x^2 u_z - \tfrac{1}{3}\rho u_y^2 u_z + \tfrac{1}{6}\rho u_x^2 u_y^2 - \tfrac{1}{3}\rho u_x^2 u_z^2 - \tfrac{1}{3}\rho u_y^2 u_z^2 + \tfrac{1}{2}\rho u_x^2 u_y^2 u_z + \tfrac{1}{2}\rho u_x^2 u_y^2 u_z^2 \\
f_{17}^{eq} &= \tfrac{1}{54}\rho + \tfrac{1}{18}\rho u_x + \tfrac{1}{18}\rho u_y + \tfrac{1}{18}\rho u_x^2 + \tfrac{1}{18}\rho u_y^2 - \tfrac{1}{36}\rho u_z^2 + \tfrac{1}{6}\rho u_x u_y + \tfrac{1}{6}\rho u_x u_y^2 + \tfrac{1}{6}\rho u_x^2 u_y - \tfrac{1}{12}\rho u_x u_z^2 - \tfrac{1}{12}\rho u_y u_z^2 \\
&+ \tfrac{1}{6}\rho u_x^2 u_y^2 - \tfrac{1}{12}\rho u_x^2 u_z^2 - \tfrac{1}{12}\rho u_y^2 u_z^2 - \tfrac{1}{4}\rho u_x u_y u_z^2 - \tfrac{1}{4}\rho u_x u_y^2 u_z^2 - \tfrac{1}{4}\rho u_x^2 u_y u_z^2 - \tfrac{1}{4}\rho u_x^2 u_y^2 u_z^2 \\
f_{18}^{eq} &= \tfrac{1}{54}\rho + \tfrac{1}{18}\rho u_x - \tfrac{1}{18}\rho u_y + \tfrac{1}{18}\rho u_x^2 + \tfrac{1}{18}\rho u_y^2 - \tfrac{1}{36}\rho u_z^2 - \tfrac{1}{6}\rho u_x u_y + \tfrac{1}{6}\rho u_x u_y^2 - \tfrac{1}{6}\rho u_x^2 u_y - \tfrac{1}{12}\rho u_x u_z^2 + \tfrac{1}{12}\rho u_y u_z^2 \\
&+ \tfrac{1}{6}\rho u_x^2 u_y^2 - \tfrac{1}{12}\rho u_x^2 u_z^2 - \tfrac{1}{12}\rho u_y^2 u_z^2 + \tfrac{1}{4}\rho u_x u_y u_z^2 - \tfrac{1}{4}\rho u_x u_y^2 u_z^2 + \tfrac{1}{4}\rho u_x^2 u_y u_z^2 - \tfrac{1}{4}\rho u_x^2 u_y^2 u_z^2 \\
f_{19}^{eq} &= \tfrac{1}{54}\rho + \tfrac{1}{18}\rho u_x + \tfrac{1}{18}\rho u_z + \tfrac{1}{18}\rho u_x^2 - \tfrac{1}{36}\rho u_y^2 + \tfrac{1}{18}\rho u_z^2 + \tfrac{1}{6}\rho u_x u_z + \tfrac{1}{6}\rho u_x u_z^2 + \tfrac{1}{6}\rho u_x^2 u_z - \tfrac{1}{12}\rho u_x u_y^2 - \tfrac{1}{12}\rho u_y^2 u_z \\
&- \tfrac{1}{12}\rho u_x^2 u_y^2 + \tfrac{1}{6}\rho u_x^2 u_z^2 - \tfrac{1}{12}\rho u_y^2 u_z^2 - \tfrac{1}{4}\rho u_x u_y^2 u_z - \tfrac{1}{4}\rho u_x^2 u_y^2 u_z - \tfrac{1}{4}\rho u_x u_y^2 u_z^2 - \tfrac{1}{4}\rho u_x^2 u_y^2 u_z^2 \\
f_{20}^{eq} &= \tfrac{1}{54}\rho + \tfrac{1}{18}\rho u_x - \tfrac{1}{18}\rho u_z + \tfrac{1}{18}\rho u_x^2 - \tfrac{1}{36}\rho u_y^2 + \tfrac{1}{18}\rho u_z^2 - \tfrac{1}{6}\rho u_x u_z + \tfrac{1}{6}\rho u_x u_z^2 - \tfrac{1}{6}\rho u_x^2 u_z - \tfrac{1}{12}\rho u_x u_y^2 + \tfrac{1}{12}\rho u_y^2 u_z \\
&- \tfrac{1}{12}\rho u_x^2 u_y^2 + \tfrac{1}{6}\rho u_x^2 u_z^2 - \tfrac{1}{12}\rho u_y^2 u_z^2 + \tfrac{1}{4}\rho u_x u_y^2 u_z + \tfrac{1}{4}\rho u_x^2 u_y^2 u_z - \tfrac{1}{4}\rho u_x u_y^2 u_z^2 - \tfrac{1}{4}\rho u_x^2 u_y^2 u_z^2 \\
f_{21}^{eq} &= \tfrac{1}{54}\rho + \tfrac{1}{18}\rho u_y + \tfrac{1}{18}\rho u_z - \tfrac{1}{36}\rho u_x^2 + \tfrac{1}{18}\rho u_y^2 + \tfrac{1}{18}\rho u_z^2 + \tfrac{1}{6}\rho u_y u_z + \tfrac{1}{6}\rho u_y^2 u_z + \tfrac{1}{6}\rho u_y u_z^2 - \tfrac{1}{12}\rho u_x^2 u_y - \tfrac{1}{12}\rho u_x^2 u_z \\
&- \tfrac{1}{12}\rho u_x^2 u_y^2 - \tfrac{1}{12}\rho u_x^2 u_z^2 + \tfrac{1}{6}\rho u_y^2 u_z^2 - \tfrac{1}{4}\rho u_x^2 u_y u_z - \tfrac{1}{4}\rho u_x^2 u_y^2 u_z - \tfrac{1}{4}\rho u_x^2 u_y u_z^2 - \tfrac{1}{4}\rho u_x^2 u_y^2 u_z^2 \\
f_{22}^{eq} &= \tfrac{1}{54}\rho + \tfrac{1}{18}\rho u_y - \tfrac{1}{18}\rho u_z - \tfrac{1}{36}\rho u_x^2 + \tfrac{1}{18}\rho u_y^2 + \tfrac{1}{18}\rho u_z^2 - \tfrac{1}{6}\rho u_y u_z - \tfrac{1}{6}\rho u_y^2 u_z + \tfrac{1}{6}\rho u_y u_z^2 - \tfrac{1}{12}\rho u_x^2 u_y + \tfrac{1}{12}\rho u_x^2 u_z \\
&- \tfrac{1}{12}\rho u_x^2 u_y^2 - \tfrac{1}{12}\rho u_x^2 u_z^2 + \tfrac{1}{6}\rho u_y^2 u_z^2 + \tfrac{1}{4}\rho u_x^2 u_y u_z + \tfrac{1}{4}\rho u_x^2 u_y^2 u_z - \tfrac{1}{4}\rho u_x^2 u_y u_z^2 - \tfrac{1}{4}\rho u_x^2 u_y^2 u_z^2 \\
f_{23}^{eq} &= \tfrac{1}{216}\rho + \tfrac{1}{72}\rho u_x + \tfrac{1}{72}\rho u_y + \tfrac{1}{72}\rho u_z + \tfrac{1}{72}\rho u_x^2 + \tfrac{1}{72}\rho u_y^2 + \tfrac{1}{72}\rho u_z^2 + \tfrac{1}{24}\rho u_x u_y + \tfrac{1}{24}\rho u_x u_z + \tfrac{1}{24}\rho u_y u_z \\
&+ \tfrac{1}{24}\rho u_x^2 u_y + \tfrac{1}{24}\rho u_x^2 u_z + \tfrac{1}{24}\rho u_x u_y^2 + \tfrac{1}{24}\rho u_y^2 u_z + \tfrac{1}{24}\rho u_x u_z^2 + \tfrac{1}{24}\rho u_y u_z^2 + \tfrac{1}{8}\rho u_x u_y u_z \\
&+ \tfrac{1}{24}\rho u_x^2 u_y^2 + \tfrac{1}{24}\rho u_x^2 u_z^2 + \tfrac{1}{24}\rho u_y^2 u_z^2 + \tfrac{1}{8}\rho u_x^2 u_y u_z + \tfrac{1}{8}\rho u_x u_y^2 u_z + \tfrac{1}{8}\rho u_x u_y u_z^2 \\
&+ \tfrac{1}{8}\rho u_x^2 u_y^2 u_z + \tfrac{1}{8}\rho u_x^2 u_y u_z^2 + \tfrac{1}{8}\rho u_x u_y^2 u_z^2 + \tfrac{1}{8}\rho u_x^2 u_y^2 u_z^2 \\
f_{24}^{eq} &= \tfrac{1}{216}\rho + \tfrac{1}{72}\rho u_x + \tfrac{1}{72}\rho u_y - \tfrac{1}{72}\rho u_z + \tfrac{1}{72}\rho u_x^2 + \tfrac{1}{72}\rho u_y^2 + \tfrac{1}{72}\rho u_z^2 + \tfrac{1}{24}\rho u_x u_y - \tfrac{1}{24}\rho u_x u_z - \tfrac{1}{24}\rho u_y u_z \\
&+ \tfrac{1}{24}\rho u_x^2 u_y - \tfrac{1}{24}\rho u_x^2 u_z + \tfrac{1}{24}\rho u_x u_y^2 - \tfrac{1}{24}\rho u_y^2 u_z + \tfrac{1}{24}\rho u_x u_z^2 + \tfrac{1}{24}\rho u_y u_z^2 - \tfrac{1}{8}\rho u_x u_y u_z \\
&+ \tfrac{1}{24}\rho u_x^2 u_y^2 + \tfrac{1}{24}\rho u_x^2 u_z^2 + \tfrac{1}{24}\rho u_y^2 u_z^2 - \tfrac{1}{8}\rho u_x^2 u_y u_z - \tfrac{1}{8}\rho u_x u_y^2 u_z + \tfrac{1}{8}\rho u_x u_y u_z^2 \\
&- \tfrac{1}{8}\rho u_x^2 u_y^2 u_z + \tfrac{1}{8}\rho u_x^2 u_y u_z^2 + \tfrac{1}{8}\rho u_x u_y^2 u_z^2 + \tfrac{1}{8}\rho u_x^2 u_y^2 u_z^2 \\
f_{25}^{eq} &= \tfrac{1}{216}\rho + \tfrac{1}{72}\rho u_x - \tfrac{1}{72}\rho u_y + \tfrac{1}{72}\rho u_z + \tfrac{1}{72}\rho u_x^2 + \tfrac{1}{72}\rho u_y^2 + \tfrac{1}{72}\rho u_z^2 - \tfrac{1}{24}\rho u_x u_y + \tfrac{1}{24}\rho u_x u_z - \tfrac{1}{24}\rho u_y u_z \\
&- \tfrac{1}{24}\rho u_x^2 u_y + \tfrac{1}{24}\rho u_x^2 u_z + \tfrac{1}{24}\rho u_x u_y^2 + \tfrac{1}{24}\rho u_y^2 u_z + \tfrac{1}{24}\rho u_x u_z^2 - \tfrac{1}{24}\rho u_y u_z^2 - \tfrac{1}{8}\rho u_x u_y u_z \\
&+ \tfrac{1}{24}\rho u_x^2 u_y^2 + \tfrac{1}{24}\rho u_x^2 u_z^2 + \tfrac{1}{24}\rho u_y^2 u_z^2 - \tfrac{1}{8}\rho u_x^2 u_y u_z + \tfrac{1}{8}\rho u_x u_y^2 u_z - \tfrac{1}{8}\rho u_x u_y u_z^2 \\
&+ \tfrac{1}{8}\rho u_x^2 u_y^2 u_z - \tfrac{1}{8}\rho u_x^2 u_y u_z^2 + \tfrac{1}{8}\rho u_x u_y^2 u_z^2 + \tfrac{1}{8}\rho u_x^2 u_y^2 u_z^2 \\
f_{26}^{eq} &= \tfrac{1}{216}\rho + \tfrac{1}{72}\rho u_x - \tfrac{1}{72}\rho u_y - \tfrac{1}{72}\rho u_z + \tfrac{1}{72}\rho u_x^2 + \tfrac{1}{72}\rho u_y^2 + \tfrac{1}{72}\rho u_z^2 - \tfrac{1}{24}\rho u_x u_y - \tfrac{1}{24}\rho u_x u_z + \tfrac{1}{24}\rho u_y u_z \\
&- \tfrac{1}{24}\rho u_x^2 u_y - \tfrac{1}{24}\rho u_x^2 u_z + \tfrac{1}{24}\rho u_x u_y^2 - \tfrac{1}{24}\rho u_y^2 u_z + \tfrac{1}{24}\rho u_x u_z^2 - \tfrac{1}{24}\rho u_y u_z^2 + \tfrac{1}{8}\rho u_x u_y u_z \\
&+ \tfrac{1}{24}\rho u_x^2 u_y^2 + \tfrac{1}{24}\rho u_x^2 u_z^2 + \tfrac{1}{24}\rho u_y^2 u_z^2 + \tfrac{1}{8}\rho u_x^2 u_y u_z - \tfrac{1}{8}\rho u_x u_y^2 u_z - \tfrac{1}{8}\rho u_x u_y u_z^2 \\
&- \tfrac{1}{8}\rho u_x^2 u_y^2 u_z - \tfrac{1}{8}\rho u_x^2 u_y u_z^2 + \tfrac{1}{8}\rho u_x u_y^2 u_z^2 + \tfrac{1}{8}\rho u_x^2 u_y^2 u_z^2
\end{aligned}
$$

The relaxation frequencies can be expressed by the following block diagonal matrix:

$$\mathbf{\Lambda} = \text{diag}\left(1,1,1,1,\tau_f^{-1},\tau_f^{-1},\tau_f^{-1},\begin{bmatrix} s_+ & s_- & s_- \\ s_- & s_+ & s_- \\ s_- & s_- & s_+ \end{bmatrix}\omega_3,\omega_3,\omega_3,\omega_3,\omega_3,\omega_3,1,\omega_4,\omega_4,\omega_4,1,1,1,1,1,1,1\right)$$

where $s_+ = \frac{1}{3}\left(\tau_{f,bulk}^{-1} + 2\tau_f^{-1}\right)$ and $s_- = \frac{1}{3}\left(\tau_{f,bulk}^{-1} - \tau_f^{-1}\right)$. The bulk viscosity can be related to the associated relaxation time by:

$$\nu' = \frac{2}{9}\left(\tau_{f,bulk} - \frac{1}{2}\right)\frac{(\Delta x)^2}{\Delta t}.$$

Guo forcing can be applied using the following central moment transformations of the associated source terms:

$$\vec{S}^m = \begin{pmatrix} 0 \\ F_x \\ F_y \\ F_z \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ \left(\frac{1}{3} - v_y^2\right)F_x - 2v_xv_yF_y \\ \left(\frac{1}{3} - v_z^2\right)F_x - 2v_xv_zF_z \\ \left(\frac{1}{3} - v_x^2\right)F_y - 2v_xv_yF_x \\ \left(\frac{1}{3} - v_x^2\right)F_z - 2v_xv_zF_x \\ \left(\frac{1}{3} - v_z^2\right)F_y - 2v_yv_zF_z \\ \left(\frac{1}{3} - v_y^2\right)F_z - 2v_yv_zF_y \\ -v_yv_zF_x + v_xv_zF_y + v_xv_yF_z \\ 4v_xv_y\left(v_yF_x + v_xF_y\right) \\ 4v_xv_z\left(v_zF_x + v_xF_z\right) \\ 4v_yv_z\left(v_zF_y + v_yF_z\right) \\ 2v_x\left(2v_yv_zF_x + v_xv_zF_y + v_xv_yF_z\right) \\ 2v_y\left(v_yv_zF_x + 2v_xv_zF_y + v_xv_yF_z\right) \\ 2v_z\left(v_yv_zF_x + v_xv_zF_y + 2v_xv_yF_z\right) \\ F_x\left(\frac{1}{9} - \frac{1}{3}v_y^2 - \frac{1}{3}v_z^2 - 3v_y^2v_z^2\right) - \left(v_zF_y + v_yF_z\right)\left(\frac{2}{3}v_x + 6v_xv_yv_z\right) \\ F_y\left(\frac{1}{9} - \frac{1}{3}v_x^2 - \frac{1}{3}v_z^2 - 3v_x^2v_z^2\right) - \left(v_zF_x + v_xF_z\right)\left(\frac{2}{3}v_y + 6v_xv_yv_z\right) \\ F_z\left(\frac{1}{9} - \frac{1}{3}v_x^2 - \frac{1}{3}v_y^2 - 3v_x^2v_y^2\right) - \left(v_yF_x + v_xF_y\right)\left(\frac{2}{3}v_z + 6v_xv_yv_z\right) \\ \frac{4}{3}v_xv_y\left(v_yF_x + v_xF_y\right) + \frac{4}{3}v_xv_z\left(v_zF_x + v_xF_z\right) + \frac{4}{3}v_yv_z\left(v_zF_y + v_yF_z\right) + 8v_xv_yv_z\left(v_yv_zF_x + v_xv_zF_y + v_xv_yF_z\right) \end{pmatrix},$$

and He forcing can be applied using these central moment terms:

$$\vec{S}^m = \begin{pmatrix} 0 \\ F_x \\ F_y \\ F_z \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ \frac{1}{3}F_x \\ \frac{1}{3}F_x \\ \frac{1}{3}F_y \\ \frac{1}{3}F_z \\ \frac{1}{3}F_y \\ \frac{1}{3}F_z \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ \frac{1}{9}F_x \\ \frac{1}{9}F_y \\ \frac{1}{9}F_z \\ 0 \end{pmatrix}.$$

Appendix E

DL_MESO_DPD Error Messages

This appendix documents the error and warning messages currently available in the DPD code in DL_MESO, DL_MESO_DPD, and recommendations for users to try and overcome the errors. Users may contact the authors of DL_MESO *after* attempting the recommended actions.

Message 1: cutoff radius value not set

A valid cutoff radius (either $r_{c,max}$ or r_c) for all (non-electrostatic) interactions cannot be found in the `CONTROL` file: this is a compulsory parameter for DPD simulations.

Action: Look in the `CONTROL` file and make sure either the **cut**off directive or the **therm**ostat **cut**off directive is included with a non-zero value.

Message 2: temperature not set

A valid system temperature (k_BT) cannot be found in the `CONTROL` file: this is a compulsory parameter for DPD simulations.

Action: Look in the `CONTROL` file and make sure the **temp**erature directive is included with a non-zero value.

Message 3: time step size not set

A valid simulation timestep (Δt) cannot be found in the `CONTROL` file: this is a compulsory parameter for DPD simulations.

Action: Look in the `CONTROL` file and make sure the **timestep** directive is included with a non-zero value.

Message 4: boundary halo size larger than half subdomain size

The size of the boundary halo (either specified by the user or determined from required interaction and bond lengths) exceeds half the length of at least one dimension of the subdomain volume assigned to each processor. The DPD simulation may therefore run less efficiently.

Action: None required to ensure the simulation runs as this is a warning message, but the user may wish to reduce the specified boundary halo size or use global bond calculations for future calculations.

Message 5: too many beads per node

The number of particles likely to be assigned to each processor is greater than the calculated maximum value `maxdim`.

Action: This error is unlikely to happen as `maxdim` is calculated according to the numbers of particles and processors available, but the user may wish to use the **densvar** directive in the `CONTROL` file to increase this value.

Message 6: system is not charge neutral

The overall charge for the system is non-zero.

Action: None immediately required to ensure the simulation runs as this is a warning message, but the user may wish to adjust the numbers of charged particles to balance out positive and negative charges. (Ewald sums do not tend to work for periodic systems with overall charges.)

Message 7: at least one interaction length larger than cutoff radius

The interaction length for at least one interacting pair of species ($r_{c,ij}$) specified in the `FIELD` file exceeds the (global) cutoff radius (r_c) specified in the `CONTROL` file: any overly-long interactions will be truncated at the global cutoff radius.

Action: None required to ensure the simulation runs as this is a warning message, but the user may wish to increase the specified cutoff radius to prevent truncation of interactions at larger separations.

Message 8: boundary halo size larger than subdomain size - cannot apply SPME

The size of the boundary halo (either specified by the user or determined from required interaction and bond lengths) exceeds the length of at least one dimension of the subdomain volume assigned to each processor. This prevents correct charge grid assignment for Smooth Particle Mesh Ewald calculations when using distributed charge grids.

Action: The user should either increase the size of the maximum reciprocal vector or otherwise reduce the size of the boundary halo.

Message 9: volume not specified in CONTROL or CONFIG file

The system volume cannot be found in either the CONTROL file (for systems initialized from scratch) or the supplied CONFIG file.

Action: The user should check that either the CONTROL file includes a **vol**ume directive or the CONFIG file includes lines specifying the system volume.

Message 10: cannot read CONFIG file

The supplied `CONFIG` file cannot be read by DL_MESO_DPD: it might have been corrupted.

Action: Check the `CONFIG` file to ensure it is complete and in ANSI (text) format.

Message 20: missing CONTROL file

No input file named `CONTROL` can be found.

Action: Make sure there is a `CONTROL` file in the same directory as the DL_MESO_DPD executable.

Message 21: cannot read CONTROL file

The supplied `CONTROL` file cannot be read by DL_MESO_DPD: it might have been corrupted.

Action: Check the `CONTROL` file to ensure it is complete and in ANSI (text) format.

Message 22: no charge smearing length defined in CONTROL file

The supplied `CONTROL` file has specified a charge smearing type but no charge smearing length (or reciprocal).

Action: Check the `CONTROL` file includes a **smear len**gth or **smear beta** directive with a non-zero value. If the `CONTROL` file was originally created for earlier versions of DL_MESO_DPD, it will need modification to specify both charge smearing type and lengthscale in separate lines: see Appendix A for further details.

Message 23: no pressure specified in CONTROL file for barostat

The supplied `CONTROL` file has specified an ensemble involving a barostat (constant pressure, surface area or surface tension) but no pressure.

Action: Check the `CONTROL` file includes a **pres**sure directive.

Message 30: missing FIELD file

No input file named `FIELD` can be found.

Action: Make sure there is a `FIELD` file in the same directory as the DL_MESO_DPD executable.

Message 31: cannot read FIELD file

The supplied `FIELD` file cannot be read by DL_MESO_DPD: it might have been corrupted.

Action: Check the `FIELD` file to ensure it is complete and in ANSI (text) format.

Message 32: unrecognised bond type defined in FIELD file

A bond type not included in Table 10.3 has been found in the `FIELD` file.

Action: Check the `FIELD` file to ensure all bond types are valid; if adding a new bond type to DL_MESO_DPD, the `scan_field` and `read_field` routines in `config_module` need to be modified.

Message 33: unrecognised bond angle type defined in FIELD file

A bond angle type not included in Table 10.4 has been found in the `FIELD` file.

Action: Check the `FIELD` file to ensure all bond angle types are valid; if adding a new bond angle type to DL_MESO_DPD, the `scan_field` and `read_field` routines in `config_module` need to be modified.

Message 34: unrecognised bond dihedral type defined in FIELD file

A bond dihedral type not included in Table 10.5 has been found in the `FIELD` file.

Action: Check the `FIELD` file to ensure all bond dihedral types are valid; if adding a new bond dihedral type to DL_MESO_DPD, the `scan_field` and `read_field` routines in `config_module` need to be modified.

Message 35: name for species number i in FIELD file truncated to 8 characters

The given name for the i-th species in the `FIELD` file exceeds 8 characters and has had to be truncated.

Action: None required to ensure the simulation runs as this is a warning message, but the user may wish to check the `FIELD` file to ensure this species cannot be confused with another.

Message 36: name for molecule number i in FIELD file truncated to 8 characters

The given name for the i-th molecule type in the `FIELD` file exceeds 8 characters and has had to be truncated.

Action: None required to ensure the simulation runs as this is a warning message, but the user may wish to check the `FIELD` file to ensure this molecule type cannot be confused with another.

Message 37: missing finish directive in FIELD file for molecule i

No **finish** directive can be found for the i-th molecule type in the `FIELD` file.

Action: Check the `FIELD` file, particularly the i-molecule type, and ensure each molecule type ends with a **finish** directive.

Message 38: non-existent species given in FIELD file for molecule i

An undefined species has been found in the definition for the i-th molecule type in the `FIELD` file.

Action: Check the `FIELD` file, particularly the i-th molecule type and the species definitions, to ensure the species in the molecule are defined.

Message 39: unrecognised bond definition in FIELD file for molecule i

A bond definition has been found in the `FIELD` file for the i-th molecule that was not detected during the initial scan of the input file.

Action: This error should never occur! If it does, please contact the authors of DL_MESO.

Message 40: out-of-range bead number for bond in FIELD file for molecule i

At least one of the bond definitions for the i-th molecule type refers to a bead number out of range (either below 1 or above the number of beads for the molecule).

Action: Check the `FIELD` file, particularly the i-th molecule type, and check that the bond definitions refer to bead numbers within range.

Message 41: unrecognised bond angle definition in FIELD file for molecule i

A bond angle definition has been found in the `FIELD` file for the i-th molecule that was not detected during the initial scan of the input file.

Action: This error should never occur! If it does, please contact the authors of DL_MESO.

Message 42: out-of-range bead number for angle in FIELD file for molecule i

At least one of the angle definitions for the i-th molecule type refers to a bead number out of range (either below 1 or above the number of beads for the molecule).

Action: Check the `FIELD` file, particularly the i-th molecule type, and check that the angle definitions refer to bead numbers within range.

Message 43: unrecognised dihedral angle definition in FIELD file for molecule i

A bond dihedral definition has been found in the `FIELD` file for the i-th molecule that was not detected during the initial scan of the input file.

Action: This error should never occur! If it does, please contact the authors of DL_MESO.

Message 44: out-of-range bead number for dihedral in FIELD file for molecule i

At least one of the dihedral definitions for the i-th molecule type refers to a bead number out of range (either below 1 or above the number of beads for the molecule).

Action: Check the `FIELD` file, particularly the i-th molecule type, and check that the dihedral definitions refer to bead numbers within range.

Message 45: non-existent species given in FIELD file for unbonded interaction i

An undefined species has been found in the i-th (unbonded) interaction definition in the `FIELD` file.

Action: Check the `FIELD` file, particularly the i-th interaction type and the species definitions, to ensure the species in the interaction are defined.

Message 46: non-existent species given in FIELD file for surface interaction i

An undefined species has been found in the i-th (hard) surface interaction definition in the `FIELD` file.

Action: Check the `FIELD` file, particularly the i-th surface interaction and species definitions, to ensure the species in the interaction are defined.

Message 47: non-existent species given in FIELD file for frozen wall interaction

An undefined species has been found in the definition for frozen particle walls in the `FIELD` file.

Action: Check the `FIELD` file, particularly the frozen particle wall and species definitions, to ensure the required frozen particle species is defined.

Message 48: incomplete many-body DPD interaction data in FIELD file

Not all species pairs have defined interaction parameters in the `FIELD` file: this is vital for systems with any many-body DPD interactions as universal mixing rules are unavailable for many-body DPD parameters.

Action: Check the `FIELD` file to ensure that unbonded interactions between every possible species pair is defined.

Message 49: no interaction data in FIELD file for single species i

Unbonded interaction data between particle pairs of the same species i are unavailable in the `FIELD` file: mixing rules to determine any missing interaction data thus cannot be applied.

Action: Check the `FIELD` file to ensure that unbonded interactions exist for same-species pairs.

Message 50: zero reciprocal vector range for ewald sum

The maximum reciprocal vector, $\vec{k}_{max}$, has not been defined for systems requiring Ewald sum or SPME electrostatics.

Action: Look in the `CONTROL` file and make sure the **ewald** or **spme** directive includes the convergence parameter α and the extents of the maximum reciprocal vector, k_1, k_2 and k_3.

Message 51: cannot read restart file named export

No valid file called `export` can be found or read to restart a simulation.

Action: Check for the existence of the `export` file in the working directory; if no file exists or is needed, remove the **restart** directive from the `CONTROL` file.

Message 52: too many beads needed per node for export file

The total number of particles from the `export` file for a particular process exceeds the predicted maximum number of particles per node (`maxdim`).

Action: This error is unlikely to happen as `maxdim` is calculated according to the numbers of particles and processors available, but the user may wish to use the **densvar** directive in the `CONTROL` file to increase this value.

Message 53: cannot read REVIVE file - no previous statistical data available

No valid `REVIVE` file with statistical accumulators, barostat properties and random number generator states can be found or read.

Action: No immediate action is necessary to get DL_MESO to run as this message is a warning, but the user may need to consider either ensuring there is a `REVIVE` file available in the working directory or using the

restart noscale directive in the CONTROL file (which starts a new simulation based on the configuration given in the export file).

Message 54: at least one molecule type incorrectly defined in FIELD file

The FIELD file contains a molecule definition that is incorrectly formatted, preventing other definitions from being read.

Action: Check the molecule definitions in the FIELD file to ensure each of them includes the number of molecules to be used in the simulation, the number of beads per molecule and finishes with the **finish** directive (which indicates the end of the molecule definition).

Message 55: insufficient number of beads per node allocated for required CONFIG file

The total number of particles from the CONFIG file for a particular process exceeds the predicted maximum number of particles per node (maxdim).

Action: This error is unlikely to happen as maxdim is calculated according to the numbers of particles and processors available, but the user may wish to use the **densvar** directive in the CONTROL file to increase this value.

Message 56: non-existent species given in CONFIG file

No valid species can be found for an entry in the CONFIG file: this can happen either when the file is being scanned to find the start of a particle entry or when the particles are being read (in the latter case, additional messages will be produced stating which particles have invalid species identifiers).

Action: Check the species definitions in the FIELD file and the i-th particle in the CONFIG file (if given as an additional message) to ensure that species is defined.

Message 57: out-of-range particle index given in CONFIG file

At least one of the particle indices given in the CONFIG file is out of range compared with the contents of the FIELD file.

Action: Check the particle numbering in the CONFIG file and the contents of the FIELD file, and adjust where necessary. If the contents of the CONFIG file match up with the FIELD file, consider using the **no index** option in the CONTROL file to ignore the supplied particle numbering.

Message 58: mismatch in species particle counts between FIELD and CONFIG files

The numbers of particles for a given species in the CONFIG file does not match those given in the FIELD file: a table itemising the total numbers for each species and for all species in both files is given before this message.

Action: Check the CONFIG and FIELD files to ensure the species compositions of both files match up.

Message 59: out-of-range particle index given in export file

At least one of the particle indices given in the export file is out of range compared with the contents of the FIELD file.

Action: Check the contents of the FIELD file to make sure it matches up with that used when the export file was created, and adjust where necessary. The export_config utility might be useful here to determine the contents of the export file.

Message 60: out-of-range species type given in export file

At least one of the species types given in the export file is out of range compared with the contents of the FIELD file.

Action: Check the contents of the FIELD file to make sure it matches up with that used when the export file was created, and adjust where necessary. The export_config utility might be useful here to determine the contents of the export file.

Message 61: out-of-range molecule type given in export file

At least one of the molecule types given in the export file is out of range compared with the contents of the FIELD file.

Action: Check the contents of the FIELD file to make sure it matches up with that used when the export file was created, and adjust where necessary. The export_config utility might be useful here to determine the contents of the export file, although the resulting CONFIG file would not directly show which particles belong to particular molecules.

Message 62: mismatch in species particle counts between FIELD and export files

The numbers of particles for a given species in the export file does not match those given in the FIELD file (with any cell duplication from the previous simulation taken into account): a table itemising the total numbers for each species and for all species in both files is given before this message.

Action: Check the export and FIELD files to ensure the species compositions of both files match up. If the **nfold** option was used to create for the simulation that created the export file, make sure the relevant values are specified in the CONTROL file.

Message 65: insufficient number of beads per node allocated for required initialization

The value of maxdim is not large enough to include the unbonded particles assigned to each processor for a new simulation (without a CONFIG file).

Action: This error is unlikely to happen as maxdim is calculated according to the numbers of particles and processors available, but the user may wish to use the **densvar** directive in the CONTROL file to increase this value.

Message 66: discrepancy in total number of starting beads - i too many/few

The total number of particles assigned to all processors for a new simulation does not match up with the numbers specified in the FIELD file (taking nfold duplication into account if a CONFIG file is used).

Action: For new simulations without CONFIG files or restarted simulations without frozen bead walls, this error should never occur and the authors of DL_MESO should be contacted if it does. If using a CONFIG file, check the FIELD file to ensure that the number of particles for each species and numbers of molecules match up with those in the CONFIG file. If restarting a simulation that used frozen bead walls, remove the directives for frozen bead walls in the CONTROL and FIELD files and include the number of frozen beads used in the species totals in the FIELD file.

Message 67: molecule i bigger than domain - cannot insert into system

The maximum extent of molecule i, which is represented as a cuboid, is larger than the defined size of the system in all three dimensions. This message will only appear for systems with hard surfaces or frozen walls, and the molecule cannot therefore be inserted into the system without crossing these surfaces or walls.

Action: Either the system size must be increased to accommodate the defined molecule or the molecule needs to be made smaller to fit inside the system dimensions (which are reduced by the specified positions of hard surfaces or the thicknesses of frozen walls).

Message 68: molecule i bigger than domain in at least one dimension

The maximum extent of molecule i, which is represented as a cuboid, is larger than the defined size of the system in at least one dimension.

Action: No immediate action is necessary as this is a warning message, but the user may wish to either increase the system size or reduce the molecule size in future simulations. In the case of systems with hard surfaces or frozen walls, the molecule size may restrict how it can be randomly inserted into the system volume and it may take longer to initialize such systems.

Message 71: deport coordinate buffers exceeded

The amount of particle data received during deport is greater than the current processor can accommodate.

Action: This error message suggests non-constant particle densities across the system and poor load-balancing. The user may wish to use the **densvar** directive in the `CONTROL` file to increase the value of `maxdim` and thus accommodate larger numbers of particles.

Message 72: deport coordinate buffers exceeded for lees-edwards shear

The amount of particle data received during deport with Lees-Edwards shearing is greater than the current processor can accommodate.

Action: This error message suggests non-constant particle densities across the system and poor load-balancing. The user may wish to use the **densvar** directive in the `CONTROL` file to increase the value of `maxdim` and thus accommodate larger numbers of particles.

Message 81: import coordinate buffers exceeded

The number of additional particles created during import of particle forces is greater than the current processor can accommodate.

Action: This error message suggests non-constant particle densities across the system and poor load-balancing. The user may wish to use the **densvar** directive in the `CONTROL` file to increase the value of `maxdim` and thus accommodate larger numbers of particles.

Message 82: import coordinate buffers exceeded for lees-edwards shear

The number of additional particles created during import of particle forces with Lees-Edwards shearing is greater than the current processor can accommodate.

Action: This error message suggests non-constant particle densities across the system and poor load-balancing. The user may wish to use the **densvar** directive in the `CONTROL` file to increase the value of `maxdim` and thus accommodate larger numbers of particles.

Message 91: export coordinate buffers exceeded

The number of additional particles created during export of particles into boundary halos is greater than the current processor can accommodate.

Action: This error message suggests non-constant particle densities across the system and poor load-balancing. The user may wish to use the **densvar** directive in the `CONTROL` file to increase the value of `maxdim` and thus accommodate larger numbers of particles.

Message 92: export coordinate buffers exceeded for lees-edwards shear

The number of additional particles created during export of particles into boundary halos with Lees-Edwards shearing is greater than the current processor can accommodate.

Action: This error message suggests non-constant particle densities across the system and poor load-balancing. The user may wish to use the **densvar** directive in the `CONTROL` file to increase the value of `maxdim` and thus accommodate larger numbers of particles.

Message 93: cannot correctly export velocities to boundary halos

Particle velocities (for DPD Velocity Verlet integration) cannot be exported correctly to particles already in the boundary halos. (This error message can only be invoked when running the serial version of DL_MESO.)

Action: This error should never occur! If it does, please contact the authors of DL_MESO.

Message 94: cannot correctly export data to boundary halos for density calculations

Particle data for calculating localised densities (needed for many-body DPD) cannot be exported to the boundary halos as the number of particles created would be greater than that accounted for in memory. (This error message can only be invoked when running the serial version of DL_MESO.)

Action: This error message suggests non-constant particle densities across the system and poor load-balancing. The user may wish to use the **densvar** directive in the `CONTROL` file to increase the value of `maxdim` and thus accommodate larger numbers of particles.

Message 95: cannot correctly export data to boundary halos for density calculations with shear

Particle data for calculating localised densities (needed for many-body DPD) cannot be exported to the boundary halos with Lees-Edwards shearing as the number of particles created would be greater than that accounted for in memory. (This error message can only be invoked when running the serial version of DL_MESO.)

Action: This error message suggests non-constant particle densities across the system. The user may wish to use the **densvar** directive in the `CONTROL` file to increase the value of `maxdim` and thus accommodate larger numbers of particles.

Message 100: wrong bead total after compression - i too many/few

The total number of particles after the first Velocity Verlet integration stage (including dealing with boundary conditions etc.) does not equal the specified total number of particles for the system.

Action: This error should never occur! If it does, please contact the authors of DL_MESO.

Message 200: bond too long or cannot be found

At least one bond between specified particles is too long (e.g. longer than the maximum specified length for the potential) or cannot be calculated due to lack of available information for both particles. The bond(s) identified as overly long or lost is/are printed either in the `OUTPUT` file or in the standard output (e.g. screen).

Action: If calculating bonds locally, increasing the size of boundary halos may reduce the likelihood of bonds being 'broken'. Alternatively, global bond calculations can ensure all data is available at the cost of replication over all processors. Adjusting the parameters for the bond potential may also help ensure bonds do not get too long.

Message 201: too many interacting pairs

The number of interacting pairs for non-DPD thermostats (Lowe-Andersen, Peters, Stoyanov-Groot) exceeds the maximum number calculated from the number of particles in the system, `maxpair`.

Action: The user may wish to use the **densvar** directive in the `CONTROL` file to increase the values of `maxdim` and `maxpair`, thus accommodating larger numbers of interacting pairs.

Message 301: incorrect endianness in HISTORY file - cannot append additional data

The endianness used for a pre-existing `HISTORY` file does not match the endianness currently used for the calculation: data appended to the end of the file will therefore not be readable.

Action: The endianness used in DL_MESO can be swapped by re-compiling with appropriate compiler flags in the Makefile.

Message 302: incorrect real number size in HISTORY file - cannot append additional data

The number of bytes per real number used in the `HISTORY` does not match the number of bytes per real number used by DL_MESO: there is a mismatch in real number types.

Action: The kind of real number used in DL_MESO can be modified by changing the definition of `dp` in the `constants` module.

Message 303: data corruption in HISTORY file

The beginning of a trajectory frame (the header containing the number of particles and time) in the `HISTORY` file cannot be read due to a previous interruption in writing it.

Action: Greater care must be taken to ensure there is enough closing time to complete writing the last trajectory frame to the `HISTORY` file at the end of a simulation.

Message 304: HISTORY file missing i frames of trajectories

The `HISTORY` file ends before the timestep at which the simulation is being restarted.

Action: None immediately required as this message is a warning and does not prevent the simulation from being restarted, but the user should be aware there will be a gap in the sequence of trajectories in the `HISTORY` file when it is visualised or analysed with post-processing utilities.

Messages 1001–1256: allocation/deallocation errors

Allocation or deallocation of arrays for DPD calculations (including reading of input data, transfer buffers for communications between processors, global arrays for Lowe-Andersen/Peters/Stoyanov-Groot thermostats etc.) has failed. This may be due to a lack of addressable memory required for the DPD calculations. These messages identify which allocation/deallocation has failed by module and routine names.

Action: Increase the amount of memory available for running DL_MESO_DPD by closing any other running applications, running the simulation on a larger number of processors (to reduce the memory required per processor to hold particle data), under-populating multicore processors (i.e. using fewer cores per processor than the maximum available) or upgrading your machine. Alternatively, try running a smaller simulation.

Appendix F

DL_MESO Licence Agreement (Academic Purposes)

1. DEFINITIONS AND INTERPRETATION

1.1 In this Licence Agreement the following expressions have the meanings set opposite:

Academic Purposes	fundamental or basic research or academic teaching, including any fundamental research that is funded by any public or charitable body, but not any purpose that generates revenue (as opposed to grant income) for the Licensee or any third party. Any research that is wholly or partially sponsored by any profit making organisation or that is carried out for the benefit of any profit-making organisation is not an Academic Purpose;
a Derived Work	any modification of, or enhancement or improvement to, any of the DL_MESO Software and any software or other work developed or derived from any of the DL_MESO Software;
the DL_MESO Software	the release and version of the DL_MESO Software downloaded by the Licensee from the DL_MESO Website immediately after the Licensee agrees to the terms and conditions of this Licence Agreement;
the DL_MESO Website	the website with the URL http://www.ccp5.ac.uk/DL_MESO, and any website that from time to time replaces that website;
a Harmful Element	any virus, worm, time bomb, time lock, drop dead device, trap and access code or anything else that might disrupt, disable, harm or impede the operation of any information

system, or that might corrupt, damage, destroy or render inaccessible any software, data or file on, or that may allow any unauthorised person to gain access to, any information system or any software, data or file on it;

Intellectual Property — patents, trade marks, service marks, registered designs, copyrights, database rights, design rights, know-how, confidential information, applications for any of the above, and any similar right recognised from time to time in any jurisdiction, together with all rights of action in relation to the infringement of any of the above;

the Licence Period — the period beginning when the Licensee agrees to the terms and conditions of this Licence Agreement and downloads the DL_MESO Software from the DL_MESO Website and ending on the termination of this Licence Agreement under clause 5.2.

2. LICENCE

2.1 UKRI STFC grants the Licensee an indefinite, non-exclusive, non-transferable, royalty free licence to use, copy, modify, and enhance the DL_MESO Software during the Licence Period on the terms and conditions of this Licence Agreement provided that:

2.1.1 the Licensee may not distribute any of the DL_MESO Software or any Derived Work to any third party, or share their use with any third party (whether free of charge or otherwise), and the Licensee may not sub-license the use of any of the DL_MESO Software to any third party unless, in each case, that third party has complied with clause 2.3 below;

2.1.2 the Licensee may not use the DL_MESO Software on behalf of, or for the benefit of, any third party (including, without limitation, using it to provide bureau, outsourcing or application services or facilities management services) party unless that third party has complied with clause 2.3 below; and

2.1.3 the DL_MESO Software and any Derived Work may be used by the Licensee and its employees and registered students for Academic Purposes only.

2.2 If the Licensee wishes to use the DL_MESO Software or any Derived Work in any way except for Academic Purposes, or wishes to distribute or make the DL_MESO Software or any Derived Work available to any third party for non-Academic Purposes, it must

obtain a commercial licence from UKRI STFC. UKRI STFC may refuse to grant the Licensee a commercial licence. If UKRI STFC agrees to grant a commercial licence, that licence will be on such terms and conditions as UKRI STFC sees fit.

2.3 If the Licensee wishes to carry out any collaboration for Academic Purposes with any third party and that third party needs to use the DL_MESO Software in connection with that collaboration, or if the Licensee wishes to make the DL_MESO Software available online to any third party for Academic Purposes, the Licensee must direct that third party to the DL_MESO Website. That third party may use the DL_MESO Software and any Derived Work (whether obtained from UKRI STFC or from the Licensee) only if it has completed the registration process on the DL_MESO Website and agreed to the terms and conditions of the Licence Agreement for the use of the DL_MESO Software for Academic Purposes that then appear on the DL_MESO Website.

2.4 This Licence Agreement allows the Licensee to use only the release or version of the DL_MESO Software downloaded by the Licensee from the DL_MESO Website immediately after the Licensee agrees to the terms and conditions of this Licence Agreement; the Licensee must acquire a new licence for any future release or version of the DL_MESO Software that it wishes to use.

2.5 The Licensee will not tamper with, or remove, any copyright or other proprietary notice or any disclaimer that appears on or in any part of the DL_MESO Software, and will reproduce the same in all copies of any of the DL_MESO Software and in all Derived Works.

3. WARRANTIES AND LIABILITY

3.1 The DL_MESO Software is provided for Academic Purposes free of charge. Therefore UKRI STFC gives no warranty and makes no representation in relation to the DL_MESO Software or any assistance or advice that UKRI STFC may give in connection with the DL_MESO Software. The Licensee will indemnify UKRI STFC against any and all claims arising as a result of the Licensee having made any of the DL_MESO Software or any Derived Work available to any third party.

3.2 Before using any of the DL_MESO Software, the Licensee will check that the DL_MESO Software does not contain any Harmful Element. UKRI STFC does not warrant that the DL_MESO Software will run without interruption or be error free, or be free from any Harmful Element. UKRI STFC is not obliged to provide any support or error correction service, assistance or advice in relation to the DL_MESO Software, but the Licensee may access any error corrections and online assistance that UKRI STFC chooses to make available on the DL_MESO Website from time to time. If UKRI STFC

does provide that sort of service, assistance or advice, subject to clause 3.7, UKRI STFC will not be liable for any loss or damage suffered by the Licensee as a result.

3.3 UKRI STFC will not be liable to the Licensee to the extent that any loss or damage is caused: by the Licensee's failure to implement, or the Licensee's delay in implementing, any correction or advice in relation to the DL_MESO Software that UKRI STFC has made available on the DL_MESO Website; or by the Licensee's failure to acquire a licence of and to implement any new release or version of the DL_MESO Software that would have remedied or mitigated the effects of any error, defect, bug or deficiency in the DL_MESO Software.

3.4 The Licensee acknowledges that proper use of the DL_MESO Software and any Derived Work is dependent on the Licensee, its employees and students exercising proper skill and care in inputting data and interpreting the output provided by the DL_MESO Software or that Derived Work. UKRI STFC will not be liable for the consequences of decisions taken by the Licensee or any other person on the basis of that output. UKRI STFC does not accept any responsibility for any use which may be made by the Licensee of that output, nor for any reliance which may be placed on that output, nor for advice or information given in connection with that output.

3.5 Subject to clause 3.7, UKRI STFC's liability or any breach of this Licence Agreement, any negligence or arising in any other way out of the subject matter of this Licence Agreement or the use of the DL_MESO Software, will not extend to any incidental or consequential damages or losses, or any loss of profits, loss of revenue, loss of data, loss of contracts or opportunity, whether direct or indirect, even if the Licensee has advised UKRI STFC of the possibility of those losses arising or if they were or are within UKRI STFC's contemplation.

3.6 Subject to clause 3.7, the aggregate liability of UKRI STFC for any and all breaches of this Licence Agreement, any negligence or arising in any other way out of the subject matter of this Licence Agreement or the use of the DL_MESO Software will not exceed in total £5000.

3.7 Nothing in this Licence Agreement limits or excludes UKRI STFC's liability for death or personal injury caused by its negligence or for any fraud, or for any sort of liability that, by law, cannot be limited or excluded.

3.8 The express undertakings and given by UKRI STFC in this Licence Agreement and the terms of this Licence Agreement are in lieu of all warranties, conditions, terms, undertakings and obligations on the part of UKRI STFC, whether express or implied by statute,

common law, custom, trade usage, course of dealing or in any other way. All of these are excluded to the fullest extent permitted by law.

4. INTELLECTUAL PROPERTY RIGHTS AND ACKNOWLEDGEMENTS

4.1 Nothing in this Licence Agreement assigns or transfers any Intellectual Property Rights in any of the DL_MESO Software. Those rights are reserved to UKRI STFC.

4.2 The Licensee will ensure that, if any of its employees or students publishes any article or other material resulting from, or relating to, a project or work undertaken with the assistance of any part of the DL_MESO Software, that publication will contain the following acknowledgement:

"DL_MESO is a mesoscale simulation package written by R. Qin, W. Smith and M. A. Seaton and has been obtained from UKRI STFC's Daresbury Laboratory via the website http://www.ccp5.ac.uk/DL_MESO"

and cite the following reference:

"M.A. Seaton, R.L. Anderson, S. Metz & W. Smith, 'Mol. Sim.', 39 (10), 796-821 (2013)".

5. TERMINATION

5.1 This Licence Agreement will take effect and the Licence Period will start when the Licensee has agreed to the terms and conditions of this Licence Agreement and downloaded the DL_MESO Software from the DL_MESO Website.

5.2 This Licence Agreement will terminate immediately and automatically if:

5.2.1 the Licensee is in breach of this Licence Agreement; or

5.2.2 the Licensee becomes insolvent, or if an order is made or a resolution is passed for its winding up (except voluntarily for the purpose of solvent amalgamation or reconstruction), or if an administrator, administrative receiver or receiver is appointed over the whole or any part of its assets, or if it makes any arrangement with its creditors.

5.3 The Licensee's right to use the DL_MESO Software will cease immediately on the termination of this Licence Agreement, and the Licensee will destroy all copies of the DL_MESO Software that it or any of its employees or students then holds.

5.4 Clauses 1, 2.2, 3, 4, 5.3, 5.4, 5.5 and 6 will survive the expiry

of the Licence Period and the termination of this Licence Agreement, and will continue indefinitely.

5.5 UKRI STFC may stop providing any assistance or advice in relation to, or any corrections, new releases or versions of the DL_MESO, and may stop updating or publishing the DL_MESO Website at any time.

6. GENERAL

6.1 Headings: The headings in this Licence Agreement are for ease of reference only; they do not affect its construction or interpretation.

6.2 Assignment etc: The Licensee may not assign or transfer this Licence Agreement as a whole, or any of its rights or obligations under it, without first obtaining the written consent of UKRI STFC.

6.3 Illegal/unenforceable provisions: If the whole or any part of any provision of this Licence Agreement is void or unenforceable in any jurisdiction, the other provisions of this Licence Agreement, and the rest of the void or unenforceable provision, will continue in force in that jurisdiction, and the validity and enforceability of that provision in any other jurisdiction will not be affected.

6.4 Waiver of rights: If UKRI STFC fails to enforce, or delays in enforcing, an obligation of the Licensee, or fails to exercise, or delays in exercising, a right under this Licence Agreement, that failure or delay will not affect its right to enforce that obligation or constitute a waiver of that right. Any waiver by UKRI STFC of any provision of this Licence Agreement will not, unless expressly stated to the contrary, constitute a waiver of that provision on a future occasion.

6.5 Entire agreement: This Licence Agreement constitutes the entire agreement between the parties relating to its subject matter. The Licensee acknowledges that it has not entered into this Licence Agreement on the basis of any warranty, representation, statement, agreement or undertaking except those expressly set out in this Licence Agreement. The Licensee waives any claim for breach of, or any right to rescind this Licence Agreement in respect of, any representation which is not an express provision of this Licence Agreement. However, this clause does not exclude any liability which UKRI STFC may have to the Licensee (or any right which the Licensee may have to rescind this Licence Agreement) in respect of any fraudulent misrepresentation or fraudulent concealment before the signing of this Licence Agreement.

6.6 Amendments: No variation of, or amendment to, this Licence Agreement will be effective unless it is made in writing and signed by each party's representative.

6.7 Third parties: No one who is not a party to this Licence Agreement has any right to prevent the amendment of this Licence Agreement or its termination, and no one except a party to this Licence Agreement may enforce any benefit conferred by this Licence Agreement, unless this Licence Agreement expressly provides otherwise.

6.8 Governing law: This Licence Agreement is governed by, and is to be construed in accordance with, English law. The English Courts will have exclusive jurisdiction to deal with any dispute which has arisen or may arise out of or in connection with this Licence Agreement, except that UKRI STFC may bring proceedings against the Licensee or for an injunction in any jurisdiction.

Bibliography

[1] Hans C. Andersen. Molecular dynamics simulations at constant pressure and/or temperature. *Journal of Chemical Physics*, 72(4):2384–2393, 1980.

[2] Richard L. Anderson, David J. Bray, Annalaura Del Regno, Michael A. Seaton, Andrea S. Ferrante, and Patrick B. Warren. Micelle formation in alkyl sulfate surfactants using dissipative particle dynamics. *Journal of Chemical Theory and Computation*, 14(5):2633–2643, 2018. PMID: 29570296.

[3] Santosh Ansumali and Iliya V. Karlin. Kinetic boundary conditions in the lattice Boltzmann method. *Physical Review E*, 66:026311, August 2002.

[4] Abdel Monim Mohamed Ali Mohamed Hassan Artoli. *Mesoscopic computational haemodynamics.* PhD thesis, University of Amsterdam, October 2003.

[5] Pietro Asinari. Generalized local equilibrium in the cascaded lattice Boltzmann method. *Physical Review E*, 78:016701, July 2008.

[6] H. J. C. Berendsen, J. P. M. Postma, W. F. van Gunsteren, A. DiNola, and J. R. Haak. Molecular dynamics with coupling to an external bath. *Journal of Chemical Physics*, 81(8):3684–3690, 1984.

[7] P. L. Bhatnagar, E. R. Gross, and M. Krook. A model for collision processes in gases. I. Small amplitude processes in charged and neutral one-component systems. *Physical Review*, 94(3):511–525, May 1954.

[8] Eugene C. Bingham. *Fluidity and plasticity.* McGraw-Hill, first edition, 1922.

[9] J. Boyd, J. Buick, and S. Green. A second-order accurate lattice Boltzmann non-Newtonian flow model. *Journal of Physics A: Mathematical and General*, 39(46):14241, 2006.

[10] A. J. Briant, P. Papatzacos, and J. M. Yeomans. Lattice Boltzmann simulations of contact line motion in a liquid-gas system. *Philosophical Transactions of the Royal Society of London A: Mathematical, Physical and Engineering Sciences*, 360(1792):485–495, 2002.

[11] I. J. Bush, I. T. Todorov, and W. Smith. A DAFT DL_POLY distributed memory adaptation of the smoothed particle mesh Ewald method. *Computer Physics Communications*, 175(5):323–329, September 2006.

[12] R. Byron Bird and Pierre J. Carreau. A nonlinear viscoelastic model for polymer solutions and melts–I. *Chemical Engineering Science*, 23(5):427–434, 1968.

[13] Norman F. Carnahan and Kenneth E. Starling. Intermolecular repulsions and the equation of state for fluids. *AIChE Journal*, 18(6):1184–1189, 1972.

[14] Pierre J. Carreau, Ian F. MacDonald, and R. Byron Bird. A nonlinear viscoelastic model for polymer solutions and melts–II. *Chemical Engineering Science*, 23(8):901–911, 1968.

[15] Mauricio Carrillo-Tripp, Humberto Saint-Martin, and Iván Ortega-Blake. A comparative study of the hydration of Na^+ and K^+ with refined polarizable model potentials. *Journal of Chemical Physics*, 118(15):7062–7073, 2003.

[16] N. Casson. Flow equation for pigment oil suspensions of the printing ink type. In C. C. Mill, editor, *Rheology of disperse systems*, pages 84–102. Pergamon Press, 1959.

[17] Zhenhua Chai, Baochang Shi, Zhaoli Guo, and Fumei Rong. Multiple-relaxation-time lattice Boltzmann model for generalized Newtonian fluid flows. *Journal of Non-Newtonian Fluid Mechanics*, 166(5):332–342, 2011.

[18] Cheng Chang, Chih-Hao Liu, and Chao-An Lin. Boundary conditions for lattice Boltzmann simulations with complex geometry flows. *Computers & Mathematics with Applications*, 58(5):940–949, 2009. Mesoscopic Methods in Engineering and Science.

[19] A. Chatterjee. Modification to Lees-Edwards periodic boundary condition for dissipative particle dynamics simulation with high dissipation rates. *Molecular Simulation*, 33(15):1233–1236, 2007.

[20] Shiyi Chen and Gary D. Doolen. Lattice Boltzmann method for fluid flows. *Annual Review of Fluid Mechanics*, 30(1):329–364, 1998.

[21] Daniele Coslovich, Jean-Pierre Hansen, and Gerhard Kahl. Ultrasoft primitive model of polyionic solutions: Structure, aggregation, and dynamics. *Journal of Chemical Physics*, 134(24):244514, 2011.

[22] Paul J. Dellar. Bulk and shear viscosities in lattice Boltzmann equations. *Physical Review E*, 64(3):031203, August 2001.

[23] Dominique d'Humières, Irina Ginzburg, Manfred Krafczyk, Pierre Lallemand, and Li-Shi Luo. Multiple-relaxation-time lattice Boltzmann models in three dimensions. *Philosophical Transactions of the Royal Society of London A*, 360(1792):437–451, 2002.

[24] U. D'Ortona, D. Salin, Marek Cieplak, Renata B. Rybka, and Jayanth R. Banavar. Two-color nonlinear Boltzmann cellular automata: Surface tension and wetting. *Physical Review E*, 51(4):3718–3728, April 1995.

[25] Michael M. Dupin, Ian Halliday, and Chris M. Care. Simulation of a microfluidic flow-focusing device. *Physical Review E*, 73(5):055701, 2006.

[26] A. Dupuis and J. M. Yeomans. Modeling droplets on superhydrophobic surfaces: equilibrium states and transitions. *Langmuir*, 21(6):2624–2629, 2005. PMID: 15752062.

[27] Pep Español. Hydrodynamics from dissipative particle dynamics. *Physical Review E*, 52(2):1734–1742, August 1995.

[28] Ulrich Essmann, Lalith Perera, Max L. Berkowitz, Tom Darden, Hsing Lee, and Lee G. Pedersen. A smooth particle mesh Ewald method. *Journal of Chemical Physics*, 103(19):8577–8593, 1995.

[29] P. P. Ewald. Die Berechnung optischer und elektrostatischer Gitterpotentiale. *Annalen der Physik*, 369(3):253–287, 1921.

[30] Linlin Fei, Kai H. Luo, and Qing Li. Three-dimensional cascaded lattice Boltzmann method: improved implementation and consistent forcing scheme. *Physical Review E*, 97:053309, May 2018.

[31] Linlin Fei and Kai Hong Luo. Consistent forcing scheme in the cascaded lattice Boltzmann method. *Physical Review E*, 96:053307, November 2017.

[32] Linlin Fei, Kai Hong Luo, Chuandong Lin, and Qing Li. Modeling incompressible thermal flows using a central-moments-based lattice Boltzmann method. *International Journal of Heat and Mass Transfer*, 120:624–634, 2018.

[33] Matteo Frigo and Steven G. Johnson. The design and implementation of FFTW3. *Proceedings of the Institute of Electrical and Electronics Engineers (IEEE)*, 93(2):216–231, January 2005.

[34] A. A. Gavrilov, A. V. Chertovich, and E. Yu. Kramarenko. Dissipative particle dynamics for systems with high density of charges: Implementation of electrostatic interactions. *Journal of Chemical Physics*, 145(17):174101, 2016.

[35] Martin Geier, Andreas Greiner, and Jan G. Korvink. Cascaded digital lattice Boltzmann automata for high Reynolds number flow. *Physical Review E*, 73:066705, June 2006.

[36] A. Ghoufi and P. Malfreyt. Mesoscale modeling of the water liquid-vapor interface: A surface tension calculation. *Physical Review E*, 83:051601, May 2011.

[37] J. B. Gibson, K. Chen, and S. Chynoweth. The equilibrium of a velocity-Verlet type algorithm for DPD with finite time steps. *International Journal of Modern Physics C*, 10(1):241–261, February 1999.

[38] Irina Ginzburg, Frederik Verhaeghe, and Dominique d'Humières. Study of simple hydrodynamic solutions with the two-relaxation-times lattice Boltzmann scheme. *Communications in Computational Physics*, 3(3):519–581, March 2008.

[39] Irina Ginzburg, Frederik Verhaeghe, and Dominique d'Humières. Two-relaxation-time lattice Boltzmann scheme: About parametrization, velocity, pressure and mixed boundary conditions. *Communications in Computational Physics*, 3(2):427–478, February 2008.

[40] Shuai Gong and Ping Cheng. Numerical investigation of droplet motion and coalescence by an improved lattice Boltzmann model for phase transitions and multiphase flows. *Computers & Fluids*, 53(0):93–104, 2012.

[41] Minerva González-Melchor, Estela Mayoral, María Eugenia Velázquez, and José Alejandre. Electrostatic interactions in dissipative particle dynamics using the Ewald sums. *Journal of Chemical Physics*, 125(22):224107, 2006.

[42] R. D. Groot. Electrostatic interactions in dissipative particle dynamics—simulation of polyelectrolytes and anionic surfactants. *Journal of Chemical Physics*, 118(24):11265–11277, 2003.

[43] Robert D. Groot and Patrick B. Warren. Dissipative particle dynamics: Bridging the gap between atomistic and mesoscopic simulation. *Journal of Chemical Physics*, 107(11):4423–4435, 1997.

[44] Andrew J. Gunstensen, Daniel H. Rothman, Stéphane Zaleski, and Gianluigi Zanetti. Lattice Boltzmann model of immiscible fluids. *Physical Review A*, 43(8):4320–4327, April 1991.

[45] Zhaoli Guo, Baochang Shi, and Chuguang Zheng. A coupled lattice BGK model for the Boussinesq equations. *International Journal for Numerical Methods in Fluids*, 39(4):325–342, 2002.

[46] Zhaoli Guo, Chuguang Zheng, and Baochang Shi. Discrete lattice effects on the forcing term in the lattice Boltzmann method. *Physical Review E*, 65(4):046308, April 2002.

[47] I. Halliday, A. P. Hollis, and C. M. Care. Lattice Boltzmann algorithm for continuum multicomponent flow. *Physical Review E*, 76(2):026708, 2007.

[48] I. Halliday, R. Law, C. M. Care, and A. Hollis. Improved simulation of drop dynamics in a shear flow at low Reynolds and capillary number. *Physical Review E*, 73(5):056708, 2006.

[49] Xiaoyi He and Li-Shi Luo. Lattice Boltzmann model for the incompressible Navier-Stokes equation. *Journal of Statistical Physics*, 88(3–4):927–944, 1997.

[50] Xinoyi He, Xiaowen Shan, and Gary D. Doolen. Discrete Boltzmann equation model for nonideal gases. *Physical Review E*, 57(1):R13–R16, January 1998.

[51] Winslow H. Herschel and Ronald Bulkley. Konsistenzmessungen von Gummi-Benzollösungen. *Kolloid-Zeitschrift*, 39(4):291–300, 8 1926.

[52] F. J. Higuera and J. Jiménez. Boltzmann approach to lattice gas simulations. *EPL (Europhysics Letters)*, 9(7):663–668, 1989.

[53] R. W. Hockney and J. W. Eastwood. *Computer simulation using particles*. McGraw-Hill International, 1981.

[54] D. J. Holdych, D. Rovas, J. G. Georgiadis, and R. O. Buckius. An improved hydrodynamics formulation for multiphase flow Lattice-Boltzmann models. *International Journal of Modern Physics C*, 9(8):1393–1404, 1998.

[55] Kainan Hu, Jianping Meng, Hongwu Zhang, Xiao-Jun Gu, David R. Emerson, and Yonghao Zhang. A comparative study of boundary conditions for lattice Boltzmann simulations of high Reynolds number flows. *Computers & Fluids*, 156:1–8, 2017.

[56] William Humphrey, Andrew Dalke, and Klaus Schulten. VMD: Visual molecular dynamics. *Journal of Molecular Graphics*, 14(1):33–38, 1996.

[57] M. K. Ikeda, P. R. Rao, and L. A. Schaefer. A thermal multicomponent lattice Boltzmann model. *Computers & Fluids*, 101:250–262, 2014.

[58] Takaji Inamuro, Masato Yoshino, Hiroshi Inoue, Riki Mizuno, and Fumimaru Ogino. A lattice Boltzmann method for a binary miscible fluid mixture and its application to a heat-transfer problem. *Journal of Computational Physics*, 179(1):201–215, 2002.

[59] Takaji Inamuro, Masato Yoshino, and Fumimaru Ogino. A non-slip boundary condition for lattice Boltzmann simulations. *Physics of Fluids*, 7(12):2928–2930, 1995.

[60] Ask F. Jakobsen. Constant-pressure and constant-surface tension simulations in dissipative particle dynamics. *Journal of Chemical Physics*, 122(12):124901, 2005.

[61] Erik Johansson. Simulating fluid flow and heat transfer using dissipative particle dynamics. Project report, Department of Energy Sciences, Faculty of Engineering, Lund University, Department of Energy Sciences, Faculty of Engineering, Lund University, Box 118, 22100 Lund, Sweden, 2012.

[62] J. E. Jones. On the determination of molecular fields. II. From the equation of state of a gas. *Proceedings of the Royal Society of London Series A*, 106(738):463–477, October 1924.

[63] Simon Jury, Peter Bladon, Mike Cates, Sujata Krishna, Maarten Hagen, Noel Ruddock, and Patrick Warren. Simulation of amphiphilic mesophases using dissipative particle dynamics. *Physical Chemistry Chemical Physics*, 1(9):2051–2056, 1999.

[64] J. M. V. A. Koelman and P. J. Hoogerbrugge. Dynamic simulations of hard-sphere suspensions under steady shear. *EPL (Europhysics Letters)*, 21(3):363–368, 1993.

[65] A. L. Kupershtokh and D. A. Medvedev. Lattice boltzmann equation method in electrohydrodynamic problems. *Journal of Electrostatics*, 64(7–9):581–585, 2006. Fifth International Electrohydrodynamics (EHD) Workshop and Fourth Conference of the Société Française d'Electrostatique (SFE).

[66] A. L. Kupershtokh, D. A. Medvedev, and D. I. Karpov. On equations of state in a lattice Boltzmann method. *Computers & Mathematics with Applications*, 58(5):965–974, 2009. Mesoscopic Methods in Engineering and Science.

[67] A. Kuzmin, M. Januszewski, D. Eskin, F. Mostowfi, and J. J. Derksen. Three-dimensional binary-liquid lattice Boltzmann simulation of microchannels with rectangular cross sections. *Chemical Engineering Journal*, 178:306–316, 2011.

[68] Pierre Lallemand and Li-Shi Luo. Theory of the lattice Boltzmann method: Dispersion, dissipation, isotropy, Galilean invariance, and stability. *Physical Review E*, 61(6):6546–6562, June 2000.

[69] Jonas Latt, Bastien Chopard, Orestis Malaspinas, Michel Deville, and Andreas Michler. Straight velocity boundaries in the lattice Boltzmann method. *Physical Review E*, 77(5):056703, May 2008.

[70] Taehun Lee and Ching-Long Lin. A stable discretization of the lattice Boltzmann equation for simulation of incompressible two-phase flows at high density ratio. *Journal of Computational Physics*, 206(1):16–47, June 2005.

[71] A. W. Lees and S. F. Edwards. The computer study of transport processes under extreme conditions. *Journal of Physics C*, 5(15):1921–1928, 1972.

[72] Benedict Leimkuhler and Xiaocheng Shang. Pairwise adaptive thermostats for improved accuracy and stability in dissipative particle dynamics. *Journal of Computational Physics*, 324:174–193, November 2016.

[73] M. Leslie and N. J. Gillan. The energy and elastic dipole tensor of defects in ionic crystals calculated by the supercell method. *Journal of Physics C*, 18(5):973, 1985.

[74] Q. Li, K. H. Luo, and X. J. Li. Forcing scheme in pseudopotential lattice Boltzmann model for multiphase flows. *Physical Review E*, 86:016709, July 2012.

[75] Qing Li, K. H. Luo, Q. J. Kang, and Q. Chen. Contact angles in the pseudopotential lattice Boltzmann modeling of wetting. *Physical Review E*, 90:053301, November 2014.

[76] Martin Lísal, John K. Brennan, and Josep Bonet Avalos. Dissipative particle dynamics at isothermal, isobaric, isoenergetic, and isoenthalpic conditions using Shardlow-like splitting algorithms. *Journal of Chemical Physics*, 135(20):204105, 2011.

[77] S. V. Lishchuk, C. M. Care, and I. Halliday. Lattice Boltzmann algorithm for surface tension with greatly reduced microcurrents. *Physical Review E*, 67(3):036701, March 2003.

[78] Qin Lou, Zhaoli Guo, and Baochang Shi. Evaluation of outflow boundary conditions for two-phase lattice Boltzmann equation. *Physical Review E*, 87:063301, June 2013.

[79] C. P. Lowe. An alternative approach to dissipative particle dynamics. *EPL (Europhysics Letters)*, 47(2):145–151, July 1999.

[80] Daniel Lycett-Brown and Kai H. Luo. Multiphase cascaded lattice Boltzmann method. *Computers & Mathematics with Applications*, 67(2):350–362, 2014. Mesoscopic Methods for Engineering and Science (Proceedings of ICMMES-2012, Taipei, Taiwan, 23–27 July 2012).

[81] Daniel Lycett-Brown, Kai H. Luo, Ronghou Liu, and Pengmei Lv. Binary droplet collision simulations by a multiphase cascaded lattice Boltzmann method. *Physics of Fluids*, 26(2):023303, 2014.

[82] John F. Marko and Eric D. Siggia. Stretching DNA. *Macromolecules*, 28(26):8759–8770, 1995.

[83] C. A. Marsh, G. Backx, and M. H. Ernst. Static and dynamic properties of dissipative particle dynamics. *Physical Review E*, 56(2):1676–1691, August 1997.

[84] Nicos S. Martys and Hudong Chen. Simulation of multicomponent fluids in complex three-dimensional geometries by the lattice Boltzmann method. *Physical Review E*, 53(1):743–750, January 1996.

[85] Keijo Mattila, Jari Hyväluoma, Tuomo Rossi, Mats Aspnäs, and Jan Westerholm. An efficient swap algorithm for the lattice Boltzmann method. *Computer Physics Communications*, 176(3):200–210, February 2007.

[86] Philip M. Morse. Diatomic molecules according to the wave mechanics. II. Vibrational levels. *Physical Review*, 34(1):57–64, July 1929.

[87] David R. Noble, Shiyi Chen, John G. Georgiadis, and Richard O. Buckius. A consistent hydrodynamic boundary condition for the lattice Boltzmann method. *Physics of Fluids*, 7(1):203–209, 1995.

[88] Rafik Ouared and Bastien Chopard. Lattice Boltzmann simulations of blood flow: non-Newtonian rheology and clotting processes. *Journal of Statistical Physics*, 121:209–221, 2005.

[89] I. Pagonabarraga and D. Frenkel. Dissipative particle dynamics for interacting systems. *Journal of Chemical Physics*, 115(11):5015–5026, 2001.

[90] Tasos C. Papanastasiou. Flows of materials with yield. *Journal of Rheology*, 31(5):385–404, 1987.

[91] Ding-Yu Peng and Donald B. Robinson. A new two-constant equation of state. *Industrial & Engineering Chemistry Fundamentals*, 15(1):59–64, 1976.

[92] E. A. J. F. Peters. Elimination of time step effects in DPD. *EPL (Europhysics Letters)*, 66(3):311–317, May 2004.

[93] B. Piaud, M.J. Clifton, S. Blanco, and R. Fournier. Lattice Boltzmann method for colloidal dispersions with phase change. *Progress in Computational Fluid Dynamics*, 8(1–4):129–137, 2008.

[94] C. M. Pooley and K. Furtado. Eliminating spurious velocities in the free-energy lattice Boltzmann method. *Physical Review E*, 77:046702, Apr 2008.

[95] C. M. Pooley, H. Kusumaatmaja, and J. M. Yeomans. Contact line dynamics in binary lattice Boltzmann simulations. *Physical Review E*, 78:056709, Nov 2008.

[96] Kannan N. Premnath and John Abraham. Three-dimensional multi-relaxation time (MRT) lattice-Boltzmann models for multiphase flow. *Journal of Computational Physics*, 224(2):539–559, 2007.

[97] P. Prinsen, P. B. Warren, and M. A. J. Michels. Mesoscale simulations of surfactant dissolution and mesophase formation. *Physical Review Letters*, 89(14):148302, September 2002.

[98] Y. H. Qian, S. Succi, and S. A. Orszag. Recent advances in lattice boltzmann computing. In Dietrich Stauffer, editor, *Annual Reviews of Computational Physics III*, chapter 6, pages 195–242. World Scientific, October 1995.

[99] P. Raiskinmäki, A Koponen, J. Merikoski, and J. Timonen. Spreading dynamics of three-dimensional droplets by the lattice-Boltzmann method. *Computational Materials Science*, 18(1):7–12, 2000.

[100] P. Raiskinmäki, A. Shakib-Manesh, A. Jäsberg, A. Koponen, J. Merikoski, and J. Timonen. Lattice-Boltzmann simulation of capillary rise dynamics. *Journal of Statistical Physics*, 107(1–2):143–158, 2002.

[101] Otto. Redlich and J. N. S. Kwong. On the thermodynamics of solutions. V. An equation of state. Fugacities of gaseous solutions. *Chemical Reviews*, 44(1):233–244, 1949.

[102] M. Revenga, I. Zúñiga, and P. Español. Boundary conditions in dissipative particle dynamics. *Computer Physics Communications*, 121–122:309–311, 1999. Proceedings of the Europhysics Conference on Computational Physics CCP 1998.

[103] A. G. Schlijper, P. J. Hoogerbrugge, and C. W. Manke. Computer simulation of dilute polymer solutions with the dissipative particle dynamics method. *Journal of Rheology*, 39(3):567–579, 1995.

[104] M. A. Seaton, I. Halliday, and A. J. Masters. Application of the multicomponent lattice Boltzmann simulation method to oil/water dispersions. *Journal of Physics A*, 44(10):105502, March 2011.

[105] Takeshi Seta, Roberto Rojas, Kosuke Hayashi, and Akio Tomiyama. Implicit-correction-based immersed boundary-lattice Boltzmann method with two relaxation times. *Physical Review E*, 89:023307, February 2014.

[106] Xiaowen Shan. Analysis and reduction of the spurious current in a class of multiphase lattice Boltzmann models. *Physical Review E*, 73(4):047701, 2006.

[107] Xiaowen Shan. Pressure tensor calculation in a class of nonideal gas lattice Boltzmann models. *Physical Review E*, 77(6):066702, June 2008.

[108] Xiaowen Shan and Hudong Chen. Lattice Boltzmann model for simulating flows with multiple phases and components. *Physical Review E*, 47(3):1815–1819, March 1993.

[109] Xiaowen Shan and Hudong Chen. Simulation of nonideal gases and liquid-gas phase transitions by the lattice Boltzmann equation. *Physical Review E*, 49(4):2941–2948, April 1994.

[110] Tony Shardlow. Splitting for dissipative particle dynamics. *SIAM Journal on Scientific Computing*, 24(4):1267–1282, 2003.

[111] Julian C. Shillcock and Reinhard Lipowsky. Equilibrium structure and lateral stress distribution of amphiphilic bilayers from dissipative particle dynamics simulations. *Journal of Chemical Physics*, 117(10):5048–5061, 2002.

[112] Richard C. Singleton. An algorithm for computing the mixed radix fast Fourier transform. *IEEE Transactions on Audio and Electroacoustics*, 17(2):93–103, July 1969.

[113] W. Smith. Coping with the pressure! How to calculate the virial. *CCP5 Information Quarterly*, 26:43–51, September 1987.

[114] W. Smith. Molecular dynamics on hypercube parallel computers. *Computer Physics Communications*, 62(2-3):229–248, March 1991.

[115] W. Smith. A replicated data molecular dynamics strategy for the parallel Ewald sum. *Computer Physics Communications*, 67(3):392–406, January 1992.

[116] W. Smith. Calculating the pressure. *CCP5 Information Quarterly*, 39:14–20, October 1993.

[117] W. Smith, T. R. Forester, and I.T. Todorov. *The DL_POLY Classic user manual*. STFC, STFC Daresbury Laboratory, Daresbury, Warrington, Cheshire, WA4 4AD, United Kingdom, version 1.0 edition, December 2010.

[118] Giorgio Soave. Equilibrium constants from a modified Redlich-Kwong equation of state. *Chemical Engineering Science*, 27(6):1197–1203, 1972.

[119] T. J. Spencer, I. Halliday, and C. M. Care. Lattice Boltzmann equation method for multiple immiscible continuum fluids. *Physical Review E*, 82(6):066701, December 2010.

[120] Timothy J. Spencer, Ian Halliday, and Chris M. Care. A local lattice Boltzmann method for multiple immiscible fluids and dense suspensions of drops. *Philosophical Transactions of the Royal Society of London A*, 369(1944):2255–2263, 2011.

[121] Simeon D. Stoyanov and Robert D. Groot. From molecular dynamics to hydrodynamics: A novel Galilean invariant thermostat. *Journal of Chemical Physics*, 122(11):114112, 2005.

[122] Alexander Stukowski. Visualization and analysis of atomistic simulation data with OVITO–the open visualization tool. *Modelling and Simulation in Materials Science and Engineering*, 18(1):015012, dec 2009.

[123] Sauro Succi. *The Lattice Boltzmann Equation for Fluid Dynamics and Beyond*. Clarendon Press, Oxford, 2001.

[124] K. Suga, Y. Kuwata, K. Takashima, and R. Chikasue. A D3Q27 multiple-relaxation-time lattice Boltzmann method for turbulent flows. *Computers & Mathematics with Applications*, 69(6):518–529, 2015.

[125] Michael R. Swift, E. Orlandini, W. R. Osborn, and J. M. Yeomans. Lattice Boltzmann simulations of liquid-gas and binary fluid systems. *Physical Review E*, 54(5):5041–5052, November 1996.

[126] Michael R. Swift, W. R. Osborn, and J. M. Yeomans. Lattice Boltzmann simulation of nonideal fluids. *Physical Review Letters*, 75(5):830–833, July 1995.

[127] Ketzasmin A. Terrón-Mejía, Roberto López-Rendón, and Armando Gama Goicochea. Electrostatics in dissipative particle dynamics using Ewald sums with point charges. *Journal of Physics: Condensed Matter*, 28(42):425101, 2016.

[128] B. D. Todd, Denis J. Evans, and Peter J. Daivis. Pressure tensor for inhomogeneous fluids. *Physical Review E*, 52:1627–1638, August 1995.

[129] I. T. Todorov and W. Smith. *The DL_POLY_4 user manual*. STFC, STFC Daresbury Laboratory, Daresbury, Warrington, Cheshire, WA4 4AD, United Kingdom, version 4.01.0 edition, October 2010.

[130] S. Y. Trofimov, E. L. F. Nies, and M. A. J. Michels. Thermodynamic consistency in dissipative particle dynamics simulations of strongly nonideal liquids and liquid mixtures. *Journal of Chemical Physics*, 117(20):9383–9394, 2002.

[131] Loup Verlet. Computer "experiments" on classical fluids. I. Thermodynamical properties of Lennard-Jones molecules. *Physical Review*, 159(1):98–103, July 1967.

[132] P. B. Warren. Vapor-liquid coexistence in many-body dissipative particle dynamics. *Physical Review E*, 68(6):066702, December 2003.

[133] P. B. Warren, P. Prinsen, and M. A. J. Michels. The physics of surfactant dissolution. *Philosophical Transactions of the Royal Society of London A*, 361(1805):665–676, 2003.

[134] Patrick B. Warren. No-go theorem in many-body dissipative particle dynamics. *Physical Review E*, 87:045303, April 2013.

[135] Patrick B. Warren and Andrey Vlasov. Screening properties of four mesoscale smoothed charge models, with application to dissipative particle dynamics. *Journal of Chemical Physics*, 140(8):084904, 2014.

[136] Patrick B. Warren, Andrey Vlasov, Lucian Anton, and Andrew J. Masters. Screening properties of Gaussian electrolyte models, with application to dissipative particle dynamics. *Journal of Chemical Physics*, 138(20):204907, 2013.

[137] John D. Weeks, David Chandler, and Hans C. Andersen. Role of repulsive forces in determining the equilibrium structure of simple liquids. *Journal of Chemical Physics*, 54(12):5237–5247, 1971.

[138] Dean R. Wheeler, Norman G. Fuller, and Richard L. Rowley. Non-equilibrium molecular dynamics simulation of the shear viscosity of liquid methanol: adaptation of the Ewald sum to Lees-Edwards boundary conditions. *Molecular Physics*, 92(1):55–62, 1997.

[139] B. Widom. Some topics in the theory of fluids. *Journal of Chemical Physics*, 39(11):2808–2812, 1963.

[140] B. Widom. Potential-distribution theory and the statistical mechanics of fluids. *Journal of Physical Chemistry*, 86(6):869–872, 1982.

[141] Satoru Yamamoto, Yutaka Maruyama, and Shi-aki Hyodo. Dissipative particle dynamics study of spontaneous vesicle formation of amphiphilic molecules. *Journal of Chemical Physics*, 116(13):5842–5849, 2002.

[142] Kenji Yasuda. *Investigation of the analogies between viscometric and linear viscoelastic properties of polystyrene fluids*. PhD thesis, Massachusetts Institute of Technology, 1979.

[143] In-Chul Yeh and Max L. Berkowitz. Ewald summation for systems with slab geometry. *Journal of Chemical Physics*, 111(7):3155–3162, 1999.

[144] M. Yoshino and T. Inamuro. Lattice Boltzmann simulations for flow and heat/mass transfer problems in a three-dimensional porous structure. *International Journal for Numerical Methods in Fluids*, 43(2):183–198, 2003.

[145] Peng Yuan and Laura Schaefer. Equations of state in a lattice Boltzmann model. *Physics of Fluids*, 18(4):042101, 2006.

[146] Raoyang Zhang and Hudong Chen. Lattice Boltzmann method for simulations of liquid-vapor thermal flows. *Physical Review E*, 67(6):066711, June 2003.

[147] You-Liang Zhu, Hong Liu, Zhan-Wei Li, Hu-Jun Qian, Giuseppe Milano, and Zhong-Yuan Lu. GALAMOST: GPU-accelerated large-scale molecular simulation toolkit. *Journal of Computational Chemistry*, 34(25):2197–2211, 2013.

[148] Qisu Zou and Xiaoyi He. On pressure and velocity boundary conditions for the lattice Boltzmann BGK model. *Physics of Fluids*, 9(6):1591–1598, June 1997.