Ludwig User Manual

July 19, 2022

Contents

1 Introduction

1.1 Overview

This introduction provides a brief overview of what the *Ludwig* code does, how to obtain and build it, and how to run a sample problem. It is assumed that the reader has at least a general knowledge of Navier-Stokes hydrodynamics, complex fluids, and to some extent statistical physics. This knowledge will be required to make sense of the input and output involved in using the code. Those wanting to work on or develop the code itself will need knowledge of ANSI C, and perhaps message passing and CUDA. We assume the reader is using a Unix-based system.

1.1.1 Purpose of the code

The *Ludwig* code has been developed over a number of years to address specific problems in complex fluids. The underlying hydrodynamic model is based on the lattice Boltzmann equation (LBE, or just 'LB'). This itself may be used to study simple (Newtonian) fluids in a number of different scenarios, including porous media and particle suspensions. However, the code is more generally suited to complex fluids, where a number of options are available, among others: symmetric binary fluids and Brazovskii smectics, polar gels, liquid crystals, or charged fluid via a Poisson-Boltzmann equation approach. These features are added in the framework of a free energy approach, where specific compositional or orientational order parameters are evolved according to the appropriate coarse-grained dynamics, but also interact with the fluid in a fully coupled fashion.

A number of other features are catered for is some or all situations. These include fluctuating hydrodynamics, and appropriate fluctuating versions of order parameter dynamics for binary solvents and liquid crystals. Colloidal particles are available for simulation of suspensions and dispersions, and with appropriate boundary conditions binary mixtures, liquid crystals, and charged fluid. Shear flow is available for fluid-only systems implemented via Lees-Edwards sliding periodic boundaries. Not all these features play with all the others: check the specific section of the document for details.

Broadly, the code is intended for complex fluid problems at low Reynolds numbers, so there is no consideration of turbulence, high Mach number flows, high density ratio flows, and so on.

1.1.2 How the code works

We aim to provide a robust and portable code, written in ANSI C, which can be used to perform serial and scalable parallel simulations of complex fluid systems based around hydrodynamics via the lattice Boltzmann method. Time evolution of modelled quantities takes place on a fixed regular discrete lattice. The preferred method of dealing with the corresponding order parameter equations is by using finite difference. However, for the case of a binary fluid, a two-distribution lattice Boltzmann approach is also maintained for historical reference.

Users control the operation of the code via a plain text input file; output for various data are available. These data may be visualised using appropriate third-party software (e.g., Paraview). Specific diagnostic output may require alterations to the code.

Potential users should also note that the complex fluid simulations enabled by *Ludwig* can be time consuming, prone to instability, and provide results which are difficult to interpret. We make no apology for this: that's the way it is.

1.1.3 Who should read what?

This documentation is aimed largely at users, but includes information that will also be relevant for developers. The documentation discusses the underlying features of the

coordinate system, hydrodynamics (LB), and other generic features, and has a separate section for each free energy. You will need to consult the section relevant to your problem.

1.2 Obtaining the code

The code is publicly available from `http://github/ludwig-cf/ludwig/` which provides revision control via git, issue tracking, and so on.

The current stable release is available as the master branch.

2 Quick Start for Users

2.1 Running an Example

2.1.1 Input file

The behaviour of *Ludwig* at run time is controlled by a plain text input file which consists of a series for key value pairs. Each key value pair controls a parameter or parameters within the code, for example the system size and number of time steps, the fluid properties density and viscosity, the free energy type and associated parameters (if required), frequency and type of output, parallel decomposition, and so on.

For most keys, the associated property has some default value which will be used by the code if the relevant key is not present (or commented out) in the input file. While such default values are chosen to be at least sensible, users need to be aware that all necessary keys need to be considered for a given problem. However, many keys are irrelevant for any given problem.

A significant number of example input files are available as part of the test suite, and these can form a useful starting point for a given type of problem. We will consider one which uses some of the common keys.

2.1.2 Example input

We will consider a simple example which computes the motion of a single spherical colloidal particle in an initially stationary fluid of given viscosity. The velocity may be measured as a function of time. Assume we have an executable in the `src` directory.

Make a copy of the input file

```
$ cp ../tests/regression/d3q19-extra/serial-auto-c01.inp .
```

Inspection of this file will reveal the following: blank lines and comments — lines beginning with `#` — may be included, but are ignored at run time; other lines are parsed as key value pairs, each pair on a separate line, and with key and value separated by one or more spaces. Values may be characters or strings, a scalar integer or 3-vector of integers, or a scalar or 3-vector of floating point numbers. Values which are 3-vectors are delineated by an underscore character (not a space), e.g., `1_2_3`, and always correspond to a Cartesian triple (x, y, z). A given value is parsed appropriately in the context of the associated key.

So, the first few lines of the above file are:

```
###############################################################################
#
#  Colloid velocity autocorrelation test (no noise).
#
###############################################################################

N_cycles 40

###############################################################################
#
#  System and MPI
#
###############################################################################

size 64_64_64
grid 2_2_2
```

Here, the first key value pair `N_cycles 40` sets the number of time steps to 40, while the second, `size 64_64_64`, sets the system size to be 64 points in each of the three

coordinate directions. The `grid` key relates to the parallel domain decomposition as is discussed in the following section.

2.1.3 Run time

The executable takes a single argument on the command line which is the name of the input file, which should be in the same directory as the executable. If no input file name is supplied, the executable will look for a default `input`. If no input file is located at all, an error to that effect will be reported.

Serial: If a serial executable is run in the normal way with the copy of the input file as the argument the code should take around 10–20 seconds to execute 40 steps. The fist few lines of output should look like:

```
$ ./Ludwig.exe ./serial-auto-c01.inp
Welcome to Ludwig v0.7.32 (Serial version running on 1 process)

The SVN revision details are: 3209M
Note assertions via standard C assert() are on.

Read 20 user parameters from serial-auto-c01.inp

No free energy selected

System details
--------------
System size:  64 64 64
Decomposition: 1 1 1
Local domain: 64 64 64
Periodic:     1 1 1
Halo nhalo:   1
```

This output shows that the appropriate input file has been read, and the system size set correspondingly with a number of default settings including periodic boundary conditions. "No free energy" tells us we are using a single Newtonian fluid.

Normal termination of execution is accompanied by a report of the time take by various parts of the code, and finally by

```
...
Ludwig finished normally.
```

Parallel: The executable compiled and linked against the MPI library can be run with the MPI launcher for the local system, often `mpirun`. For example, a run on 8 MPI tasks produces a similar output to that seen in the serial case, but reports details of the local domain decomposition:

```
$ mpirun -np 8 ./Ludwig.exe ./serial-auto-c01.inp
Welcome to Ludwig v0.1.26 (MPI version running on 8 processes)
...
System details
--------------
System size:  64 64 64
Decomposition: 2 2 2
Local domain: 32 32 32
...
```

The decomposition is controlled by the `grid` key in the input. Here, `grid 2_2_2` is consistent with the 8 MPI tasks specified on the command line, and the resulting local decomposition is 32 lattice points in each coordinate direction. If no grid is specified in the input, or a request is make for a grid which cannot be met (e.g., the product of

the grid dimensions does not agree with the total number of MPI tasks available) the code will try to determine its own decomposition as best it can. If no valid parallel decomposition is available at all, the code will exit with a message to that effect.

Further details of various input key value pairs are given in relevant sections of the documentation.

2.1.4 Output

The standard output of the running code produces a number of aggregate quantities which allow a broad overview of the progress of the computation to be seen by the user. These include, where relevant, global statistics related to fluid density and momentum, the integrated free energy, particle-related quantities, and so on. The frequency of this information can be adjusted from the input file (see, e.g., `freq_statistics`).

Output for lattice-based quantities (for example, the velocity field $u(\mathbf{r};t)$) is via external file. This output is in serial or in parallel according to how the model is run, and may be in either ASCII or raw binary format as required. Output files are produced with time step encoded in the file, and an extension which describes the parallel output decomposition. Further, each quantity output in this way is accompanied by a metadata description with `meta` extension. For example, the two output files

```
bash-3.2$ ls dist*
dist-00000020.001-001 dist.001-001.meta
```

contain the LB distributions for time step 20 (the file is 1 of a total number of 1 in parallel), and the plain text metadata description, respectively.

To ensure output for based-based quantities is in the correct order for analysis, post-processing may be required (always if the code is run in parallel). This uses a utility provided for the purpose which required both the data and the metadata description to recombine the parallel output. This utility is described ELSEWHERE.

Output for colloidal particle data is again to file with a name encoding the time step and parallel decomposition of the output. For example,

```
bash-3.2$ ls config*
config.cds00000020.001-001
```

contains particle data for time step 20 in a format described in Section **??**. These data may be requested in ASCII or raw binary format.

2.1.5 Errors and run time failures

The code attempts to provide meaningful diagnostic error messages for common problems. Such errors include missing or incorrectly formatted input files, inconsistent input values, and unacceptable feature combinations. The code should also detect other run time problems such as insufficient memory and errors writing output files. These errors will result in termination.

Instability in the computation will often be manifested by numerically absurd values in the statistical output (ultimately `NaN` in many cases). Instability may or may not result in termination, depending on the problem. Such instability is very often related to poor parameter choices, of which there can be many combinations. Check the relevant section of the documentation for advice on reasonable starting parameters for different problems.

2.2 Note on Units

All computation in *Ludwig* is undertaken in "lattice units," fundamentally related to the underlying LB fluid model which expects discrete model space and time steps $\Delta x = \Delta t = 1$. The natural way to approach problems is then to ensure that appropriate dimensionless

quantities are reasonable. However, "reasonable" may not equate to "matching experimental values". For example, typical flows in colloidal suspensions my exhibit Reynolds numbers as low as $O(10^{-6})$ to $O(10^{-8})$. Matching such a value in a computation may imply an impractically small time step; a solution is to let the Reynolds number rise artificially with the constraint that it remains small compared to $O(1)$. Further discussion of the issue of units is provided in, e.g., [**?**]. Again, consult the relevant section of the documentation for comments on specific problems.

3 General Information for Users and Developers

This section contains information on the design of the code, along with details of compilation an testing procedures which may be of interest to both users and developers. *Ludwig* is named for Ludwig Boltzmann (1844–1906) to reflect its background in lattice Boltzmann hydrodynamics.

3.1 Manifest

The top level ludwig directory should contain at least the following:

```
$ ls
bash-3.2$ ls
config LICENSE     mpi_s  src      tests version.h
docs   Makefile.mk README targetDP util
```

The code is released under a standard BSD license; the current version number is found in `version.h`. The main source is found in the `src` directory, with other relevant code in `tests` and `util`. The `targetDP` directory holds code related to the targetDP abstraction layer [**?**]. These documents are found in `docs`.

3.2 Code Overview

The code is ANSI C (1999), and can be built without any external dependencies, with the exception of the message passing interface (MPI), which is used to provide domain-decomposition based parallelism. Note that the code will also compile under NVIDIA `nvcc`, i.e., it also meets the C++ standard.

3.2.1 Design

The *Ludwig* code has evolved and expanded over a number of years to its present state. Its original purpose was to investigate specifically spinodal decomposition in binary fluids (see, e.g., [**?**]). This work used a free-energy based formulation combined with a two-distribution approach to the binary fluid problem (with lattice Boltzmann model D3Q15 at that time). This approach is retained today, albeit in a somewhat updated form. The wetting problem for binary fluid at solid surfaces has also been of consistent interest. The code has always been developed with parallel computing in mind, and has been run on a large number of different parallel machines including the Cray T3E at Edinburgh. These features, developed by J.-C. Desplat and others, were reflected in the early descriptive publication in *Comp. Phys. Comm* in 2001 [**?**].

The expansion of the code to include a number of additional features has occasioned significant alterations over time, and little of the original code remains. However, the fundamental idea that the code should essentially operate for high performance computers and be implemented using ANSI C with message passing via MPI has remained unchanged.

The code has a number of basic building blocks which are encapsulated in individual files.

3.2.2 Parallel environment

The code is designed around message passing using MPI. A stub MPI library is provided for platforms where a real MPI is not available (an increasingly rare occurrence), in which case the code runs in serial. The parallel environment (interface defined in `pe.h`) therefore underpins the entire code and provides basic MPI infrastructure such as `info()`, which provides `printf()` functionality at the root process only. The parallel environment also provides information on version number etc. MPI parallelism is implemented via

standard domain decomposition based on the computational lattice (discussed further in the following section on the coordinate system).

3.2.3 targetDP: threaded parallelism and vector parallelism

To allow various threaded models to be included without duplicating source code, we have developed a lightweight abstraction of the thread level parallelism, This currently supports a standard single-threaded model, OpenMP, or CUDA. targetDP ("target data parallel") allows single kernels to be written which can then be compiled for the different threaded models which may be appropriate on different platforms. targetDP can also be used as a convenient way to express vector parallelism (typically SSE or AVX depending on processor architecture). targetDP allows explicit specification of the vector length at compile time.

targetDP does not automatically identify parallelism; this must still be added appropriately by the developer. It is merely a concise way to include different threaded models. It is only available in the development version.

3.2.4 Array address models

In a related issue, the code allows different data layout to be used to improve performance. The default layout is array of structures (AOS) which is suitable for most CPU architectures. (A blocked version AOSOA is also available which may promote vectorisation.) For GPU machines, a structure of arrays (SOA) format can be selected at compile time.

3.2.5 Coordinate system

It is important to understand the coordinate system used in the computation. This is fundamentally a regular, 3-dimensional Cartesian coordinate system . (Even if a D2Q9 LB model is employed, the code is still fundamentally 3-dimensional, so it is perhaps not as efficient for 2-dimensional problems as it might be.)

The coordinate system is centred around the (LB) lattice, with lattice spacings $\Delta x = \Delta y = \Delta z = 1$. We will refer, in general, to the lattice spacing as Δx throughout, its generalisation to three dimensions x, y, z being understood. Lattice sites in the $x-$direction therefore have unit spacing and are located at $x = 1, 2, \ldots, N_x$. The length of the system $L_x = N_x$, with the limits of the computational domain begin $x = 1/2$ and $x = L_x + 1/2$. This allows us to specify a unit control volume centred on each lattice site x_i being $i - 1/2$ to $i + 1/2$. This will become particularly significant for finite difference (finite volume) approaches discussed later.

Information on the coordinate system, system size and so on is encapsulated in `coords.c`, which also deals with the regular domain decomposition in parallel. Decompositions may be explicitly requested by the user in the input. or computed by the code itself at run time. `coords.h` also provides basic functionality for message passing within a standard MPI Cartesian communicator, specification of periodic boundary conditions, and so on.

Three-dimensional fields are typically stored on the lattice, but are addressed in compressed one-dimensional format. This avoids use of multidimensional arrays in C. This addressing must take account of the width of the halo region required at the edge of each sub-domain required for exchanging information in the domain decomposition. The extent of the halo region varies depending on the application required, and is selected automatically at run time.

3.2.6 Lattice Boltzmann hydrodynamics

The hydrodynamic core of the calculation is supplied by the lattice Boltzmann method, which was central at the time of first development. LB is also the basis for hydrodynamic

solid-fluid interactions at stationary walls and for moving spherical colloids. The LB approach is described in more detail in Section **??**. For general and historical references, the interested reader should consider, e.g., Succi [**?**].

3.2.7 Free energy

For complex fluids, hydrodynamics is augmented by the addition of a free energy for the problem at hand, expressed in terms of an appropriate order parameter. The order parameter may be a three dimensional field, vector field, or tensor field. For each problem type, appropriate tome evolution of the order parameter is supplied.

Coupling between the thermodynamic sector and the hydrodynamics is abstracted so that the hydrodynamic core does not require alteration for the different free energies. This is implemented via a series of call back functions in the free energy which are set appropriately at run time to correspond to that specified by the user in the input.

Different (bulk) fluid free energy choices are complemented by appropriate surface free energy contributions which typically alter the computation of order parameter gradients at solid surface. Specific gradient routines for the calculation of order parameter gradients may be selected or added by the user or developer.

Currently available free energies are:

- Symmetric binary fluid with scalar compositional order parameter $\phi(\mathbf{r})$ and related Cahn-Hilliard equation;
- Brazovskii smectics, again which scalar compositional order parameter $\phi(\mathbf{r})$;
- Polar (active) gels with vector order parameter $P_\alpha(\mathbf{r})$ and related Leslie-Erikson equation;
- Landau-de Gennes liquid crystal free energy with tensor orientational order parameter $Q_{\alpha\beta}(\mathbf{r})$, extended to apolar active fluids and related Beris-Edwards equation;
- a free energy appropriate for electrokinetics for charged fluids and related Nernst-Planck equations (also requiring the solution of the Poisson equation for the potential);
- a coupled electrokinetic binary fluid model;
- a liquid crystal emulsion free energy which couples a binary composition to the liquid crystal.

See the relevant sections on each free energy for further details.

3.3 Compilation

Compilation of the main code is controlled by the `config.mk` in the top-level directory. A number of example `config.mk` files are provided in the `config` directory (which can be copied and adjusted as needed). This section discusses a number of issues which influence compilation.

3.3.1 Dependencies on third-party software:

There is the option to use PETSC to solve the Poisson equation required in the electrokinetic problem. A rather less efficient in-built method can be used if PETSC is not available. We suggest using PETSC v3.4 or later available from Argonne National Laboratory `http://www.mcs.anl.gov/petsc/`.

4 Lattice Boltzmann Hydrodynamics

We review here the lattice Boltzmann method applied to a simple Newtonian fluid with particular emphasis on the relevant implementation in *Ludwig*.

4.1 The Navier Stokes Equation

We seek to solve the isothermal Navier-Stokes equations which, often written in vector form, express mass conservation

$$\partial_t \rho + \boldsymbol{\nabla}.(\rho \mathbf{u}) = 0 \tag{1}$$

and the conservation of momentum

$$\partial_t (\rho \mathbf{u}) + \boldsymbol{\nabla}.(\rho \mathbf{u}\mathbf{u}) = -\boldsymbol{\nabla} p + \eta \nabla^2 \mathbf{u} + \zeta \boldsymbol{\nabla}(\boldsymbol{\nabla}.\mathbf{u}). \tag{2}$$

Equation **??** expresses the local rate of change of the density $\rho(\mathbf{r}; t)$ as the divergence of the flux of mass associated with the velocity field $\mathbf{u}(\mathbf{r}; t)$. Equation **??** expresses Newton's second law for momentum, where the terms on the right hand side represent the force on the fluid.

For this work, it is more convenient to rewrite these equations in tensor notation, where Cartesian coordinates x, y, z are represented by indices α and β, viz

$$\partial_t \rho + \nabla_\alpha (\rho u_\alpha) = 0 \tag{3}$$

and

$$\partial_t (\rho u_\alpha) + \nabla_\beta (\rho u_\alpha u_\beta) = -\nabla_\alpha p + \eta \nabla_\beta (u_\alpha \nabla_\beta + \nabla_\alpha u_\beta) + \zeta \nabla_\alpha (\nabla_\gamma u_\gamma). \tag{4}$$

Here, repeated Greek indices are understood to be summed over. The conservation law is seem better if the forcing terms of the right hand side are combined in the fluid stress $\Pi_{\alpha\beta}$ so that

$$\partial_t (\rho u_\alpha) + \nabla_\beta \Pi_{\alpha\beta} = 0. \tag{5}$$

In this case the expanded expression for the stress tensor is

$$\Pi_{\alpha\beta} = p\delta_{\alpha\beta} + \rho u_\alpha u_\beta + \eta \nabla_\alpha u_\beta + \zeta (\nabla_\gamma u_\gamma)\delta_{\alpha\beta} \tag{6}$$

where $\delta_{\alpha\beta}$ is that of Kroneker. Th Navier-Stokes equations in three dimensions have 10 degrees of freedom (or hydrodynamic modes) being ρ, three components of the mass flux ρu_α, and 6 independent modes from the (symmetric) stress tensor $\Pi_{\alpha\beta}$.

4.2 The Lattice Boltzmann Equation

The Navier-Stokes equation may be approximated in a discrete system by the lattice Boltzmann equation (LBE). A discrete density distribution function $f_i(\mathbf{r}; t)$ at lattice points $\mathbf{r}$ and time t evolves according to

$$f_i(\mathbf{r} + \mathbf{c}_i \Delta t; t + \Delta t) = f_i(\mathbf{r}; t) + \sum_j \mathcal{L}_{ij} \big(f_i(\mathbf{r}; t) - f_i^{\mathrm{eq}}(\mathbf{r}; t) \big) \tag{7}$$

where $\mathbf{c}_i$ is the discrete velocity basis and Δt is the discrete time step. The collision operator $\mathcal{L}_{ij}$ provides the mechanism to compute a discrete update from the non-equilibrium distribution $f_i(\mathbf{r}; t) - f_i^{\mathrm{eq}}(\mathbf{r}; t)$. Additional terms may be added to this equation to represent external body forces, thermal fluctuations, and so on. These additional terms are discussed in the following sections.

4.2.1 The distribution function and its moments

In lattice Boltzmann, the density and velocity of the continuum fluid are complemented by the distribution function $f_i(\mathbf{r};t)$ defined with reference to the discrete velocity space $c_{i\alpha}$. It is possible to relate the hydrodynamic quantities to the distribution function via its moments, that is

$$\rho(\mathbf{r};t) = \sum_i f_i(\mathbf{r};t), \quad \rho u_\alpha(\mathbf{r};t) = \sum_i f_i(\mathbf{r};t)c_{i\alpha}, \quad \Pi_{\alpha\beta}(\mathbf{r};t) = \sum_i f_i(\mathbf{r};t)c_{i\alpha}c_{i\beta}. \tag{8}$$

Here, the index of the summation is over the number of discrete velocities used as the basis, a number which will be denoted N_{vel}. For example, in three dimensions N_{vel} is often 19 and the basis is referred to as D3Q19.

The number of moments, or modes, supported by a velocity set is exactly N_{vel}, and these can be written in general as

$$M^a(\mathbf{r};t) = \sum_i m_i^a f_i(\mathbf{r};t), \tag{9}$$

where the m_i are the eigenvectors of the collision matrix in the LBE. For example, in the case of the density, all the $m_i^a = 1$ and the mode M^a is the density $\rho = \sum_i f_i$. Note that the number of modes supported by a given basis will generally exceed the number of hydrodynamic modes; the excess modes have no direct physical interpretation and are variously referred to as non-hydrodynamic, kinetic, or ghost, modes. The ghost modes take no part in bulk hydrodynamics, but may become important in other contexts, such as thermal fluctuations and near boundaries. The distribution function can be related to the modes $M^a(\mathbf{r};t)$ via

$$f_i(\mathbf{r};t) = w_i \sum_a m_i^a N^a M^a(\mathbf{r};t). \tag{10}$$

In this equation, w_i are the standard LB weights appearing in the equilibrium distribution function, while the N^a are a per-mode normalising factor uniquely determined by the orthogonality condition

$$N^a \sum_i w_i m_i^a m_i^b = \delta_{ab}. \tag{11}$$

Writing the basis this way has the advantage that the equilibrium distribution projects directly into the hydrodynamic modes only. Putting it another way, we may write

$$f_i^{\text{eq}} = w_i\big(\rho + \rho c_{i\alpha}u_\alpha/c_s^2 + (c_{i\alpha}c_{i\beta} - c_s^2\delta_{\alpha\beta})(\Pi_{\alpha\beta}^{\text{eq}} - p\delta_{\alpha\beta})/2c_s^4\big) \tag{12}$$

where only (equilibrium) hydrodynamic quantities appear on the right hand side.

4.2.2 Collision and relaxation times

4.3 Model Basis Descriptions

4.3.1 D2Q9

The D2Q9 model in two dimensions consists one zero vector (0,0), four vectors of length unity being $(\pm 1, 0)$ and $(0, \pm 1)$, and four vectors of length $\sqrt{2}$ being $(\pm 1, \pm 1)$. The eigenvectors of the collision matrix, with associated weights and normalisers are shown in Table **??**. In two dimensions there are six hydrodynamic modes and a total of three kinetic modes, or ghost modes.

4.3.2 D3Q15

The D3Q15 model in three dimensions consists of a set of vectors: one zero vector $(0, 0, 0)$, six vectors of length unity being $(\pm 1, 0, 0)$ cyclically permuted, and 8 vectors of length $\sqrt{3}$ being $(\pm 1, \pm 1, \pm 1)$. The eigenvalues and eigenvectors of the collision matrix used for D3Q15 are given in Table **??**.

M^a	p	m_i^a									N^a	
ρ	-	1	1	1	1	1	1	1	1	1	1	**1**
ρc_{ix}	-	0	1	1	1	0	0	-1	1	-1	3	c_{ix}
ρc_{iy}	-	0	1	0	-1	1	-1	1	0	-1	3	c_{iy}
Q_{xx}	1/3	-1	2	2	2	-1	-1	2	2	2	9/2	$c_{ix}c_{ix} - c_s^2$
Q_{xy}	-	0	1	0	-1	0	0	-1	0	1	9	$c_{ix}c_{iy}$
Q_{yy}	1/3	-1	2	-1	2	2	2	2	-1	2	9/2	$c_{iy}c_{iy} - c_s^2$
χ^1	-	1	4	-2	4	-2	-2	4	-2	4	1/4	χ^1
J_{ix}	-	0	4	-2	4	0	0	-4	-2	-4	3/8	$\chi^1 \rho c_{ix}$
J_{iy}	-	0	4	0	-4	-2	2	4	0	-4	3/8	$\chi^1 \rho c_{iy}$
w_i	1/36	16	1	4	1	4	4	1	4	1		w_i

Table 1: Table showing the details of the basis used for the D2Q9 model in two dimensions. The nine modes M^a include six hydrodynamic modes, one scalar kinetic mode χ^1, and one vector kinetic mode $J_{i\alpha}$. The weights in the equilibrium distribution function are w_i and the normaliser for each mode is N^a. The eigenvectors of the collision matrix are the columns of the transformation matrix m_i^a. The pre-factor p (where present) multiplies all the elements to the right in that row.

4.3.3 D3Q19

The D3Q19 model in three dimensions is constructed with velocities: one zero vector $(0, 0, 0)$, three vectors of length unity being $(\pm 1, 0, 0)$ cyclically permuted, and twelve vectors of length $\sqrt{2}$ being $(\pm 1, \pm 1, 0)$ cyclically permuted. The details of the D3Q19 model are set out in Table **??**.

4.4 Fluctuating LBE

It is possible [**?**] to simulate fluctuating hydrodynamics for an isothermal fluid via the inclusion of a fluctuating stress $\sigma_{\alpha\beta}$:

$$\Pi_{\alpha\beta} = p\delta_{\alpha\beta} + \rho u_\alpha u_\beta + \eta_{\alpha\beta\gamma\delta}\nabla_\gamma u_\delta + \sigma_{\alpha\beta}. \tag{13}$$

The fluctuation-dissipation theorem relates the magnitude of this random stress to the isothermal temperature and the viscosity.

In the LBE, this translates to the addition of a random contribution ξ_i to the distribution at the collision stage, so that

$$\ldots + \xi_i. \tag{14}$$

For the conserved modes $\xi_i = 0$. For all the non-conserved modes, i.e., those with dissipation, the fluctuating part may be written

$$\xi_i(\mathbf{r}; t) = w_i m_i^a \hat{\xi}^a(\mathbf{r}; t) N^a \tag{15}$$

where $\hat{\xi}^a$ is a noise term which has a variance determined by the relaxation time for given mode

$$\left\langle \hat{\xi}^a \hat{\xi}^b \right\rangle = \frac{\tau_a + \tau_b + 1}{\tau_a \tau_b} \left\langle \delta M^a \delta M^b \right\rangle. \tag{16}$$

4.4.1 Fluctuating stress

For the stress, the random contribution to the distributions is

$$\xi_i = w_i \frac{Q_{i\alpha\beta}\hat{\sigma}_{\alpha\beta}}{4c_s^2} \tag{17}$$

M^a	p	m_i^a															N^a	
ρ	-	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	**1**
ρc_{ix}	-	0	1	-1	0	0	0	0	1	-1	1	-1	1	-1	1	-1	3	c_{ix}
ρc_{iy}	-	0	0	0	1	-1	0	0	1	1	-1	-1	1	1	-1	-1	3	c_{iy}
ρc_{iz}	-	0	0	0	0	0	1	-1	1	1	1	1	-1	-1	-1	-1	3	c_{iz}
Q_{xx}	1/3	-1	2	2	-1	-1	-1	-1	2	2	2	2	2	2	2	2	9/2	$c_{ix}c_{ix} - c_s^2$
Q_{yy}	1/3	-1	-1	-1	2	2	-1	-1	2	2	2	2	2	2	2	2	9/2	$c_{iy}c_{iy} - c_s^2$
Q_{zz}	1/3	-1	-1	-1	-1	-1	2	2	2	2	2	2	2	2	2	2	9/2	$c_{iz}c_{iz} - c_s^2$
Q_{xy}	-	0	0	0	0	0	0	0	1	-1	-1	1	1	-1	-1	1	9	$c_{ix}c_{iy}$
Q_{yz}	-	0	0	0	0	0	0	0	1	1	-1	-1	-1	-1	1	1	9	$c_{iy}c_{iz}$
Q_{zx}	-	0	0	0	0	0	0	0	1	-1	1	-1	-1	1	-1	1	9	$c_{iz}c_{ix}$
χ^1	-	-2	1	1	1	1	1	1	-2	-2	-2	-2	-2	-2	-2	-2	1/2	χ^1
J_{ix}	-	0	1	-1	0	0	0	0	-2	2	-2	2	-2	2	-2	2	3/2	$\chi^1 \rho c_{ix}$
J_{iy}	-	0	0	0	1	-1	0	0	-2	-2	2	2	-2	-2	2	2	3/2	$\chi^1 \rho c_{iy}$
J_{iz}	-	0	0	0	0	0	1	-1	-2	-2	-2	-2	2	2	2	2	3/2	$\chi^1 \rho c_{iz}$
χ^3	-	0	0	0	0	0	0	0	1	-1	-1	1	-1	1	1	-1	9	$c_{ix}c_{iy}c_{iz}$
w_i	1/72	16	8	8	8	8	8	8	1	1	1	1	1	1	1	1		w_i

Table 2: Table showing the details of the basis used for the D3Q15 model in three dimensions. The fifteen modes M^a include two scalar kinetic modes χ^1 and χ^3, and one vector kinetic mode $J_{i\alpha}$. The weights in the equilibrium distribution are w_i and the normaliser for each mode is N^a. The eigenvectors of the collision matrix are the columns of the transformation matrix m_i^a. The pre-factor p simply multiplies all elements of m_i^a in that row as a convenience.

where $\hat{\sigma}_{\alpha\beta}$ is a symmetric matrix of random variates drawn from a Gaussian distribution with variance given by equation **??**. In the case that the shear and bulk viscosities are the same, i.e., there is a single relaxation time, then the variances of the six independent components of the matrix are given by

$$\langle \hat{\sigma}_{\alpha\beta}\hat{\sigma}_{\mu\nu}\rangle = \frac{2\tau+1}{\tau^2}(\delta_{\alpha\mu}\delta_{\beta\nu} + \delta_{\alpha\nu}\delta_{\beta\mu}). \tag{18}$$

4.5 Hydrodynamic Boundary Conditions

4.5.1 Bounce-Back on Links

A very general method for the representation of solid objects within the LB approach was put forward by Ladd [**?**, **?**]. Solid objects (of any shape) are defined by a boundary surface which intersects some of the velocity vectors $\mathbf{c}_i$ joining lattice nodes. Sites inside are designated solid, while sites outside remain fluid. The correct boundary condition is defined by identifying *links* between fluid and solid sites, which allows those elements of the distribution which would cross the boundary at the propagation step to be "bounced-back" into the fluid. This bounce-back on links is an efficient method to obtain the correct hydrodynamic interaction between solid and fluid.

4.5.2 Fixed objects

4.5.3 Moving objects

Colloidal particles are assumed to be spherical with a geometrical centre $\mathbf{r}_c$, which is also the centre of mass. The centre is allowed to move continuously across the lattice with velocity $\mathbf{U}$; the particle has an angular velocity $\mathbf{\Omega}$. The surface of the colloid is defined by an input radius, a_0, which determines which lattice nodes are inside or outside the

M^a	m_i^a																			N^a
ρ	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
ρc_{ix}	0	1	-1	0	0	0	0	1	1	-1	-1	1	1	-1	-1	0	0	0	0	3
ρc_{iy}	0	0	0	1	-1	0	0	1	-1	1	-1	0	0	0	0	1	1	-1	-1	3
ρc_{iz}	0	0	0	0	0	1	-1	0	0	0	0	1	-1	1	-1	1	-1	1	-1	3
Q_{ixx}	-1	2	2	-1	-1	-1	-1	2	2	2	2	2	2	2	2	-1	-1	-1	-1	9/2
Q_{iyy}	-1	-1	-1	2	2	-1	-1	2	2	2	2	-1	-1	-1	-1	2	2	2	2	9/2
Q_{izz}	-1	-1	-1	-1	-1	2	2	-1	-1	-1	-1	2	2	2	2	2	2	2	2	9/2
Q_{ixy}	0	0	0	0	0	0	0	1	-1	-1	1	0	0	0	0	0	0	0	0	9
Q_{ixz}	0	0	0	0	0	0	0	0	0	0	0	1	-1	-1	1	0	0	0	0	9
Q_{iyz}	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	-1	-1	1	9
χ^1	0	1	1	1	1	-2	-2	-2	-2	-2	-2	1	1	1	1	1	1	1	1	3/4
$\chi^1 \rho c_{ix}$	0	1	-1	0	0	0	0	-2	-2	2	2	1	1	-1	-1	0	0	0	0	3/2
$\chi^1 \rho c_{iy}$	0	0	0	1	-1	0	0	-2	2	-2	2	0	0	0	0	1	1	-1	-1	3/2
$\chi^1 \rho c_{iz}$	0	0	0	0	0	-2	2	0	0	0	0	1	-1	1	-1	1	-1	1	-1	3/2
χ^2	0	1	1	-1	-1	0	0	0	0	0	0	-1	-1	-1	-1	1	1	1	1	9/4
$\chi^2 \rho c_{ix}$	0	1	-1	0	0	0	0	0	0	0	0	-1	-1	1	1	0	0	0	0	9/2
$\chi^2 \rho c_{iy}$	0	0	0	-1	1	0	0	0	0	0	0	0	0	0	0	1	1	-1	-1	9/2
$\chi^2 \rho c_{iz}$	0	0	0	0	0	0	0	0	0	0	0	-1	1	-1	1	1	-1	1	-1	9/2
χ^3	1	-2	-2	-2	-2	-2	-2	1	1	1	1	1	1	1	1	1	1	1	1	1/2
w_i	12	2	2	2	2	2	2	1	1	1	1	1	1	1	1	1	1	1	1	

Table 3: Table showing the details of the basis used for the D3Q19 model in three dimensions. The nineteen modes M^a include ten hydrodynamic modes, three scalar kinetic modes χ^1, χ^2, and χ^3; there are also two vector kinetic modes $\chi^1 \rho c_{i\alpha}$ and $\chi^2 \rho c_{i\alpha}$. The weights in the equilibrium distribution function are w_i, and the normaliser for each mode is N^a. The eigenvectors of the collision matrix are the columns of the transformation matrix m_i^a.

colloid. (The hydrodynamic properties of the colloid are specified by a different radius a_h — more of this later.) The boundary links are then the set of vectors joining lattice nodes which intersect the spherical surface $\{\mathbf{c}_b\}$. Note that a lattice node exactly at the solid-fluid interface is defined to be outside the colloid.

In the original approach of Ladd, fluid occupied nodes both inside and outside the particle. The effect of the "internal fluid" is known to be restricted to short time scales (compared to the characteristic time a_0^2/ν), on which the fluid inside the particle relaxes to a solid body rotation [**?**]. However, we use fully solid particles via the approach introduced by Nguyen and Ladd [**?**].

A boundary link is defined as joining a node $\mathbf{r}$ inside the particle to one outside at $\mathbf{r} + \mathbf{c}_b \Delta t$. If the post-collision distributions are denoted by f^*, then the distributions must be reflected at the solid surface so that

$$f_{b'}(\mathbf{r}; t + \Delta t) = f_b^*(\mathbf{r}; t) - \frac{2 w_{c_b} \rho_0 \mathbf{u}_b . \mathbf{c}_b}{c_s^2} \tag{19}$$

where the boundary link $\mathbf{c}_{b'} = -\mathbf{c}_b$. Note that the local density at the fluid site $\rho(\mathbf{r}; t)$ is replaced by the mean fluid density ρ_0. in the second term on the right-hand side. The velocity at the boundary is

$$\mathbf{u}_b = \mathbf{U} + \mathbf{\Omega} \times \mathbf{r}_b. \tag{20}$$

The force exerted on a single link is

$$\mathbf{F}_b(\mathbf{r} + \tfrac{1}{2}\mathbf{c}_b \Delta t; t + \tfrac{1}{2}\Delta t) = \frac{\Delta x^3}{\Delta t} \Big[2 f_b^*(\mathbf{r}; t) - \frac{2 w_{c_b} \rho_0 \mathbf{u}_b . \mathbf{c}_b}{c_s^2} \Big] \mathbf{c}_b, \tag{21}$$

with corresponding torque $\mathbf{T}_b = \mathbf{r}_b \times \mathbf{F}_b$. The total hydrodynamic force on the particle is then found by taking the sum of $\mathbf{F}_b$ over all the boundary links defining the particle. There is an associated torque on each link of $\mathbf{r}_b \times \mathbf{F}_b$, which again is summed over all links to give the total torque on the colloid. Colloid dynamics is discussed in more detail in Section **??**.

5 Lees Edwards Sliding Periodic Boundary Conditions

5.1 Background

The idea of introducing a Galilean transformation in a periodic computation to model uniform shear was introduced by Lees and Edwards in 1972 [**?**]. It was first adapted to the lattice Boltzmann picture by Wagner and Pagonabarraga in 2000 [**?**]. The current implementation follows the description of Adhikari et al. [**?**].

5.2 Distributions crossing the LE planes

With the planes conceptually half-way between lattice sites, the propagation moves some of the distributions (namely, those with $c_x = \pm 1$) between different sliding blocks. While the collision stage is unaffected, action must be taken to adjust these distributions when the LE boundary conditions are active. This is done in a two-stage process of reprojection and interpolation implemented between the collision and propagation.

5.2.1 Reprojection

The post-collision distributions with $c_x = \pm 1$ at sites adjacent to a boundary are modified to take account of the velocity jump $\pm u_y^{LE}$ between the sliding blocks. In terms of the hydrodynamic moments we have:

$$\begin{aligned} \rho &\rightarrow \rho', & (22)\\ \rho u_\alpha &\rightarrow (\rho u_\alpha)' \pm \rho' u_\alpha^{LE}, & (23)\\ S_{\alpha\beta} &\rightarrow S'_{\alpha\beta} \pm (\rho u_\alpha)' u_\beta^{LE} \pm (\rho u_\beta)' u_\alpha^{LE} + \rho' u_\alpha^{LE} u_\beta^{LE}. & (24) \end{aligned}$$

If we work with the changes to the moments, then the density is unaffected $\delta\rho = 0$, the velocity is changed by $\delta u_\alpha = \pm u_\alpha^{LE}$, with an analogous expression for the change in the stress

$$\delta S_{\alpha\beta} = \pm(\rho u_\alpha)' u_\beta^{LE} \pm (\rho u_\beta)' u_\alpha^{LE} + \rho' u_\alpha^{LE} u_\beta^{LE}. \quad (25)$$

We can then work out the change in the distributions by reprojecting via Eq.**??**, i.e.,

$$f_i \rightarrow f_i' + w_i \left(\frac{\rho \delta u_\alpha c_{i\alpha}}{c_s^2} + \frac{\delta S_{\alpha\beta} Q_{i\alpha\beta}}{2c_s^4} \right). \quad (26)$$

A similar set of expressions are given for the order parameter distributions in Adhikari *et al.* [**?**].

5.2.2 Interpolation of reprojected distributions

As the sliding blocks are displaced continuously with $\delta y = u_y^{LE} t$, which is not in general a whole number of lattice spacings, an interpolation is also required before propagation can take place. Consider the situation from a stationary frame where the frame above has translated a small distance to the right ($\delta y < \Delta y$). In the stationary frame we must interpolate the distributions with $c_x = 1$ to a 'departure point' from which the propagation will move the distribution exactly to the appropriate lattice site in the moving frame. Schematically,

$$f_i'(x, y + \delta y, z) = (1 - \delta y) f_i^\star(x, y, z) + \delta y f_i^\star(x, y + \Delta y, z), \quad (27)$$

where f_i' here represents the interpolated value and $f_i^\star$ is the reprojected post-collision quantity.

In the case where the displacement $\delta y > 1$, the relevant lattice sites involved in the interpolation are displaced to the right by the integral part of δy, while the relative weight given to the two sites involves the fractional part of δy. To take account of the periodic boundary conditions in the y-direction, all displacements are modulo L_y.

5.3 Order parameter gradients

When the order parameter gradient is computed at a given point near the LE boundaries, the stencil may extend out of the stationary frame. The process of interpolation is slightly different to that for the distributions.

Here, we need to interpolate values in the moving frame to the position which is seen by the rest frame. If a five-point stencil is used to calculate the gradient at the central point (x, y, z) in the rest frame. An interpolation of the ϕ-field of the form

$$\phi'(x + \Delta x, y - \delta y, z) = \delta y \phi(x + \Delta x, y - \Delta y, z) + (1 - \delta y)\phi(x + \Delta x, y, z) \tag{28}$$

must then take place. In practice, the interpolated values are stored in the offset buffer. As the gradient calculation requires ϕ values in the halo regions, interpolated values for the buffer halo region are also required.

In parallel, the interpolation requires data from at most two adjacent processes in the along-plane $y-$direction as long as the ϕ halo regions are up-to-date. The rank of the processes involved is determined by the displacement δy as a function of time.

5.4 Velocities and finite-difference fluxes

For the finite-difference implementation, the velocity field at the cell boundaries is required. At the planes, this means an interpolation which takes the same form as that for ϕ is needed.

In computing the advective fluxes between cells, we must then allow for the presence of the planes. This is done by computing separately the 'east' and 'west' face fluxes in the x-direction (the flux that crosses the plane). Away from the planes, face-flux uniqueness $f^e_{ijk} = f^w_{i+1jk}$ ensures conservation of order parameter. However, at the planes, the non-linear combination of velocity field and order parameter field does not in general give rise to consistent fluxes. In order to restore conservation, we reconcile east and west face fluxes at the plane by taking an average

$$f^{e\star}_{ijk} = \tfrac{1}{2}(f^e_{ijk} + f^w_{i+1j'k}), \quad f^w_{i+1j'k} = \delta y f^w_{i+1j-1k} + (1 - \delta y) f^w_{i+1jk}, \tag{29}$$

$$f^{w\star}_{i+1jk} = \tfrac{1}{2}(f^e_{ij'k} + f^w_{i+1jk}), \quad f^e_{ij'k} = (1 - \delta y) f^e_{ijk} + \delta y f^e_{ij+1k}. \tag{30}$$

In this way, we average the east face flux in the rest frame f^e_{ijk} with the interpolated value of the west face flux in the moving frame $f^w_{i+1j'k}$, and vice-versa. One can then easily show that when integrated over the length of the plane, the average fluxes are consistent, i.e.,

$$\sum_j (f^{e\star}_{ijk} - f^{w\star}_{i+1jk}) = 0 \quad \forall k, \tag{31}$$

thus ensuring order parameter conservation.

A similar correction is required when computing the force on the fluid directly via the divergence of the thermodynamic stress $F_\alpha = \nabla_\beta P^{th}_{\alpha\beta}$. This is implemented by computing fluxes of momentum at cell faces in each direction, and then taking the divergence to compute the force. At the planes, an averaging procedure is again used to ensure conservation of momentum.

6 Colloid Dynamics

6.1 Hydrodynamics

Having computed the total force and torque on an individual colloid by the process of bounce-back-on-links, is is possible to update the the linear velocities

$$m_0\mathbf{U}(t+\Delta t) = m_0\mathbf{U}(t) + \Delta t\mathbf{F}(t), \tag{32}$$

where the mass of the colloid is related to the input radius by $m_0 = (4/3)\pi\rho_0 a_0^3$ and the angular velocity

$$I_0\mathbf{\Omega}(t+\Delta t) = I_0\mathbf{\Omega}(t) + \Delta t\mathbf{T}(t), \tag{33}$$

where the moment of inertia is $I_0 = (2/5)m_0 a_0^2$. However, this explicit update is generally found to have poor stability properties [**?**, **?**]. The alternative is to use a velocity update which is implicit [**?**, **?**].

The total force and torque on a colloid can be split into a velocity-dependent and a velocity-independent part by combining Equations (**??**) and (**??**) to eliminate the boundary velocity $\mathbf{u}_b$. In this way, one can write

$$\begin{pmatrix} \mathbf{F} \\ \mathbf{T} \end{pmatrix} = \begin{pmatrix} \mathbf{F}_0 \\ \mathbf{T}_0 \end{pmatrix} - \begin{pmatrix} \boldsymbol{\zeta}^{FU} & \boldsymbol{\zeta}^{F\Omega} \\ \boldsymbol{\zeta}^{TU} & \boldsymbol{\zeta}^{T\Omega} \end{pmatrix} \begin{pmatrix} \mathbf{U} \\ \mathbf{\Omega} \end{pmatrix}. \tag{34}$$

Here, the velocity independent parts of the force and the torque (appropriate for a colloid at rest) are

$$\mathbf{F}_0(t+\tfrac{1}{2}\Delta t) = \frac{2\Delta x^3}{\Delta t}\sum_b \left[f_b^*(\mathbf{r};t) - f_{b'}^*(\mathbf{r}+\mathbf{c}_i\Delta t;t)\right]\mathbf{c}_b, \tag{35}$$

$$\mathbf{T}_0(t+\tfrac{1}{2}\Delta t) = \frac{2\Delta x^3}{\Delta t}\sum_b \left[f_b^*(\mathbf{r};t) - f_{b'}^*(\mathbf{r}+\mathbf{c}_i\Delta t;t)\right](\mathbf{r}_b\times\mathbf{c}_b), \tag{36}$$

and the matrices $\boldsymbol{\zeta}$ are interpreted as drag coefficients and can be written as

$$\boldsymbol{\zeta}^{FU} = \frac{4\rho_0\Delta x^3}{c_s^2\Delta t}\sum_b w_{c_b}\mathbf{c}_b\mathbf{c}_b, \tag{37}$$

$$\boldsymbol{\zeta}^{F\Omega} = \frac{4\rho_0\Delta x^3}{c_s^2\Delta t}\sum_b w_{c_b}\mathbf{c}_b(\mathbf{r}_b\times\mathbf{c}_b), \tag{38}$$

$$\boldsymbol{\zeta}^{TU} = \frac{4\rho_0\Delta x^3}{c_s^2\Delta t}\sum_b w_{c_b}(\mathbf{r}_b\times\mathbf{c}_b)\mathbf{c}_b, \tag{39}$$

$$\boldsymbol{\zeta}^{T\Omega} = \frac{4\rho_0\Delta x^3}{c_s^2\Delta t}\sum_b w_{c_b}(\mathbf{r}_b\times\mathbf{c}_b)(\mathbf{r}_b\times\mathbf{c}_b). \tag{40}$$

The velocity updates can now be rewritten using an implicit (or self-consistent) approach, where the velocity at the new time step is used in computing the velocity-dependent terms:

$$\begin{pmatrix} m_0\mathbf{U}(t+\Delta t) \\ I_0\mathbf{T}(t+\Delta t) \end{pmatrix} = \begin{pmatrix} m_0\mathbf{U}(t) \\ I_0\mathbf{T}(t) \end{pmatrix} + \Delta t\left[\begin{pmatrix} \mathbf{F}_0 \\ \mathbf{T}_0 \end{pmatrix} - \begin{pmatrix} \boldsymbol{\zeta}^{FU} & \boldsymbol{\zeta}^{F\Omega} \\ \boldsymbol{\zeta}^{TU} & \boldsymbol{\zeta}^{T\Omega} \end{pmatrix} \begin{pmatrix} \mathbf{U}(t+\Delta t) \\ \mathbf{\Omega}(t+\Delta t) \end{pmatrix}\right]. \tag{41}$$

If the particle has a symmetric distribution of boundary links (the special case where the centre of mass is on a symmetry point of the lattice) the $\boldsymbol{\zeta}^{FU}$ and $\boldsymbol{\zeta}^{T\Omega}$ matrices are diagonal, while the $\boldsymbol{\zeta}^{F\Omega}$ and $\boldsymbol{\zeta}^{TU}$ matrices are zero. In general, however, this is not the case, and the full drag matrix must be used to form the update of the dynamic quantities in which case Equation **??** is solved as a 6×6 matrix inversion.

Finally, the position of the particle can be updated: an Euler forward step is taken using the mean of the old and updated velocity

$$\mathbf{r}(t + \Delta t) = \mathbf{r}(t) + (1/2)\Delta t\big[\mathbf{U}(t) + \mathbf{U}(t + \Delta t)\big]. \tag{42}$$

6.1.1 Rotational motions

For a number of applications it is useful to update not only position, as in Equation **??**, but also orientation. This is relevant for magnetic dipoles, swimming directions, and so on. A convenient way to implement the required rotations is by considering quaternions. A quaternion can be thought of as an extended complex number

$$q = w + xi + yj + zk \tag{43}$$

where $i^2 = j^2 = k^2 = -1$ and $ijk = -1$. The norm, or length of a quaternion is

$$N(q) = (w^2 + x^2 + y^2 + z^2)^{1/2}. \tag{44}$$

Addition and subtraction of quaternions proceeds as normal, but care is required for multiplication, which is associative but not commutative. If the quaternion is written as $q = (w, \mathbf{v})$ where $\mathbf{v}$ is a normal 3-vector, then multiplication can be written

$$q_1 q_2 = (w_1 w_2 - \mathbf{v}_1.\mathbf{v}_2, w_1\mathbf{v}_2 + w_2\mathbf{v}_1 + \mathbf{v}_1 \times \mathbf{v}_2), \tag{45}$$

where the usual scalar and vector products are used. Note that the conjugate of a quaternion $q^\star = (w, -\mathbf{v})$ so that $qq^\star = N^2(q)$. It can be shown that a rotation of an arbitrary vector though an angle θ around the unit axis of rotation $\hat{\mathbf{w}}$ can be represented by the quaternion

$$q = (\cos(\theta/2), \hat{\mathbf{w}}\sin(\theta/2)) \tag{46}$$

which ensures that $N(q) = 1$. If the arbitrary vector is (0, $\mathbf{v}$), then the rotated vector $\mathbf{v}'$ can be written as

$$(0, \mathbf{v}') = q(0, \mathbf{v})q^\star \tag{47}$$

which can be expanded as

$$\mathbf{v}' = (1 - \cos\theta)(\mathbf{v}.\hat{\mathbf{w}})\hat{\mathbf{w}} + \mathbf{v}\cos\theta + (\hat{\mathbf{w}} \times \mathbf{v})\sin\theta. \tag{48}$$

This equation is used to update appropriate vector quantities at each time step by a rotation $\mathbf{w} = \Delta t\boldsymbol{\Omega}(t + \Delta t)$, where the angular velocity is that determined by the implicit procedure described in Section **??**.

6.1.2 Changes in particle shape

One consequence is that the discrete shape of the particle fluctuates as the colloid centre moves relative to the lattice. This means that the size of the particle as seen by the fluid changes, despite the fact that the input radius of the colloid is fixed. However, these fluctuations are small for input radii greater than about 5 lattice units [**?**]. Furthermore, the fluctuations are greater for some input radii than others (leading to preferred values of a_0 1.25, 2.33, and so on).

In the original approach of Ladd [**?**], changes in the map of boundary links are accommodated by the internal fluid. If the particle shape changes so that an internal node is exposed, the internal fluid at that site rejoins the fluid proper with the solid body momentum (assuming the internal fluid is relaxed to solid-body rotation). If a fluid node is covered by particle movement, then it simply joins the internal fluid, and will relax to solid-body rotation. Without internal fluid, changes in particle shape in which fluid nodes are either covered or uncovered must be accompanied by the removal or addition

of fluid with appropriate properties at the nodes in question. It is important that this is done in a way which minimises the perturbation to the fluid flow around the particle.

If a fluid node at $\mathbf{r}$ is covered by the movement of a solid particle, the fluid loses a mass $\rho(\mathbf{r})\Delta x^3$, of which an excess $\Delta M_c = (\rho(\mathbf{r}) - \rho_0)\Delta x^3$ must be replaced explicitly so that the overall mean fluid density is unchanged. At the same time, the colloid must assume the momentum lost by the fluid $\Delta x^3 \sum_i f_i(\mathbf{r})\mathbf{c}_i$.

Similarly, when a fluid node is exposed by particle movement, fluid must be replaced with the appropriate distribution, density and velocity. If the new density is ρ (to be determined by some method) then the fluid gains an excess of mass $\Delta M_u = (\rho - \rho_0)\Delta x^3$ which must be balanced elsewhere. If the new fluid is assumed to have velocity $\mathbf{u}$ then the particle must give up momentum $\Delta x^3 \rho \mathbf{u}$.

Following Nguyen and Ladd [**?**], these changes owing to change in particle shape are implemented by adding a small correction to the bounce-back at the following time step. The excess mass from a covered fluid site is redistributed over all the boundary nodes by redefined the momentum transfer at bounce-back

$$f_{b'}(\mathbf{r}; t + \Delta t) = f_b^*(\mathbf{r}; t) - \frac{2w_{c_b}\rho_0 \mathbf{u}_b.\mathbf{c}_b}{c_s^2} + \frac{w_{c_b}\rho_0 \Delta M_c}{A}, \tag{49}$$

where $A = \rho_0 \Delta x^3 \sum_b w_{c_b}$. The accompanying force on the particle owing to the change in shape is then that lost by the fluid plus the contributions from the final term in (**??**) summed over all the boundary links

$$\Delta \mathbf{F}_c = \frac{\Delta x^3}{\Delta t} \sum_i f_i \mathbf{c}_i + \frac{\Delta x^3}{\Delta t} \sum_b \frac{w_{c_b}\rho_0 \Delta M_c}{A} \mathbf{c}_b. \tag{50}$$

Note that for a closed surface, $\sum_b w_{c_b}\mathbf{c}_b = 0$, and the second term on the right-hand side of Eq. (**??**) is zero, i.e., the redistribution of mass does not contribute to the net force and torque on the particle. The corresponding change in torque is

$$\Delta \mathbf{T}_c = \frac{\Delta x^3}{\Delta t} \mathbf{r}_c \times \sum_i f_i \mathbf{c}_i + \frac{\Delta x^3}{\Delta t} \sum_b \frac{w_{c_b}\rho_0 \Delta M_c}{A} \mathbf{r}_b \times \mathbf{c}_b, \tag{51}$$

where $\mathbf{r}_c$ is the boundary vector at the position where the fluid has been removed. Again, for a closed surface, the redistribution of mass does not contribute to the net torque on the particle.

Likewise, the contribution to the bounce-back from newly uncovered nodes is

$$-\frac{w_{c_b}\rho_0 \Delta M_u}{A} \tag{52}$$

leading to a change in force of

$$\Delta \mathbf{F}_u = -\frac{\Delta x^3 \rho(\mathbf{r})\mathbf{u}(\mathbf{r})}{\Delta t} - \frac{\Delta x^3}{\Delta t} \sum_b \frac{w_{c_b}\rho_0 \Delta M_u}{A} \mathbf{c}_b, \tag{53}$$

with a corresponding change in the torque. If more than one lattice node is either covered or uncovered by the movement of the particle, then these contributions add in a simple way. The overall contributions can be added to the velocity-independent terms $\mathbf{F}_0$ and $\mathbf{T}_0$ appearing in Equation **??**.

6.1.3 Particles near contact

Particles near contact may not possess a full set of boundary links. This leads to a potential non-conservation of mass associated with the particle motion which must be corrected.

Again following Nguyen and Ladd [**?**], mass conservation is enforced explicitly using the following procedure. There is a mass transfer associated with the bounce-back at each link of $(2w_{c_b}\rho_0\mathbf{u}_b.\mathbf{c}_b/c_s^2)\Delta x^3$. The net mass transport into the particle is the sum of this quantity over all the links, and can be written

$$\Delta M_s = -\frac{2\Delta x^3\rho_0}{c_s^2}\Big[\mathbf{U}.\sum_b w_{c_b}\mathbf{c}_b + \mathbf{\Omega}.\sum_b w_{c_b}\mathbf{r}_b\times\mathbf{c}_b\Big]. \tag{54}$$

For any particle with a full set of boundary links, both $\sum_b w_{c_b}\mathbf{c}_b$ and $\sum_b w_{c_b}\mathbf{r}_b\times\mathbf{c}_b$ are zero. However, if the set of boundary links is incomplete ΔM_s is not zero and a correction is required. This is again made by redistributing the excess or deficit of mass over the other existing boundary nodes. The additional contribution to the bounce-back here is $-w_{c_b}\rho_0\Delta M_s/A$, where $A=\rho_0\Delta x^3\sum_b w_{c_b}$ (cf. Equation **??**).

This contribution now adds to the velocity-dependent part of the force and torque and leads to a slightly different form of the drag matrices

$$\boldsymbol{\zeta}^{FU} = \frac{-2\rho_0\Delta x^3}{c_s^2\Delta t}\sum_b w_{c_b}(\mathbf{c}_b-\overline{\mathbf{c}_b})\mathbf{c}_b \tag{55}$$

$$\boldsymbol{\zeta}^{F\Omega} = \frac{-2\rho_0\Delta x^3}{c_s^2\Delta t}\sum_b w_{c_b}\mathbf{c}_b(\mathbf{r}_b\times\mathbf{c}_b-\overline{\mathbf{r}_b\times\mathbf{c}_b}), \tag{56}$$

$$\boldsymbol{\zeta}^{TU} = \frac{-2\rho_0\Delta x^3}{c_s^2\Delta t}\sum_b w_{c_b}(\mathbf{r}_b\times\mathbf{c}_b)(\mathbf{c}_b-\overline{\mathbf{c}_b}), \tag{57}$$

$$\boldsymbol{\zeta}^{T\Omega} = \frac{-2\rho_0\Delta x^3}{c_s^2\Delta t}\sum_b w_{c_b}(\mathbf{r}_b\times\mathbf{c}_b)(\mathbf{r}_b\times\mathbf{c}_b-\overline{\mathbf{r}_b\times\mathbf{c}_b}). \tag{58}$$

The mean quantities

$$\overline{\mathbf{c}_b} = \frac{\sum_b w_{c_b}\mathbf{c}_b}{\sum_b w_{c_b}} \tag{59}$$

and

$$\overline{\mathbf{r}_b\times\mathbf{c}_b} = \frac{\sum_b w_{c_b}\mathbf{r}_b\times\mathbf{c}_b}{\sum_b w_{c_b}}. \tag{60}$$

A little algebra will show that both occurrences of $\mathbf{c}_b$ and/or $\mathbf{r}_b\times\mathbf{c}_b$ in the definition of the drag matrices can be replaced by their deviation from the mean. This provides a convenient way to compute the drag matrices (maintaining symmetry) and computing the correct force and torque on the particle when close to contact.

6.2 Colloid-Colloid interactions

Surface-surface separation between two colloids is denoted $h=r_{12}-a_1-a_2$ where r_{12} is the centre-centre separation, and is based on the hydrodynamic radius a_h. For ease of notation all references to a in this section refer to the hydrodynamic radius a_h.

6.2.1 Lubrication corrections

Hydrodynamic lubrication interactions between two particles may not be fully resolved on the lattice if the particles are close to contact. The degree to which this is the case depends on the size of the particles, and their exact discrete position on the lattice. There will be a cut-off separation h_c associated with corrections which must be determined by calibration [**?**]; the cut-off is different for the different contributions to the interaction.

The force and torque between the pair may be corrected by adding back a contribution taken from the known theoretical expression for two spheres. Two contributions are handled:

1. **Normal force**

   The appropriate correction term between colloids with hydrodynamic radii a_1 and a_2, and having separation $h < h_c$, is

$$\mathbf{f}_{12} = -6\pi\eta \frac{a_1^2 a_2^2}{(a_1+a_2)^2} \left[\frac{1}{h} - \frac{1}{h_c}\right] \hat{\mathbf{r}}_{12}.(\mathbf{U}_1 - \mathbf{U}_2)\, \hat{\mathbf{r}}_{12}. \tag{61}$$

2. **Tangential force**

   The corresponding tangential component is a somewhat complex expression:

$$\mathbf{f}_{12} = -(24/15)\pi\eta A\big[(\mathbf{U}_1 - \mathbf{U}_2) - \hat{\mathbf{r}}_{12}.(\mathbf{U}_1 - \mathbf{U}_2)\hat{\mathbf{r}}_{12}\big], \tag{62}$$

   where

$$A = \frac{a_1 a_2 (2a_1^2 + a_1 a_2 + 2a_2^2)}{(a_1+a_2)^3} \left[\ln\left(\frac{a_1+a_2}{2h}\right) - \ln\left(\frac{a_1+a_2}{2h_c}\right)\right].$$

In the presence of isothermal fluctuations, there is an additional consideration related to the dissipation related to the correction itself. This adds an additional random force with a variance $(2k_BT|\mathbf{f}_{12}|)^{1/2}$ to each lubrication component.

We do not implement the collective implicit update required for stability when many particles are in mutual lubrication contact [**?**]. This treats the velocity-dependent corrections as effectively velocity independent and not as part of the implicit update.

This means that mutual contact must be prevented via a repulsive potential of sufficient strength to keep particles from close approach (the update may become unstable if not). For many purposes it may then be preferable to ignore the lubrication corrections completely. Long-range lubrication ($h > h_c$) will still be well represented by the lattice fluid.

6.2.2 Hard sphere interaction

Particles do not overlap, i.e., there is always a hard-sphere interaction which is a function of the separation h:

$$v^{hs}(h) = \begin{cases} \infty & h \le 0, \\ 0 & h > 0. \end{cases} \tag{63}$$

The hard-sphere interaction does not give rise to a force, but gives rise to program termination if $h < 0$.

6.2.3 Soft-sphere interaction

A simple soft-sphere interaction is available, the basic form of which is:

$$v^{ss}(r) = \epsilon(\sigma/r)^\nu, \tag{64}$$

where ϵ sets the energy scale, and σ sets the characteristic width. The steepness of the potential is set by the exponent $\nu(> 0)$.

To prevent the need for calculation of long-range interactions, the soft-sphere potential is truncated in a "cut-and-shift" approach. This is done in such a way as to smoothly match both the potential and the force at the cut-off distance r_c. For $r < r_c$ the potential is then

$$v^{ss}(r) - v^{ss}(r_c) - (r - r_c) \left.\frac{dv^{ss}}{dr}\right|_{r=r_c}. \tag{65}$$

Clearly, this potential is not exactly what we first thought of as soft-sphere. Matching the potential smoothly ensures conservation of energy, while matching the force smoothly prevents potential instabilities in the molecular dynamics update.

The soft-sphere potential can be useful as a mechanism to keep the particles from touching (which risks hard-sphere interactions), in which case is is possible to compute the interaction as a function of the separation h, instead of r. The force between two particles is computed via the derivative of equation **??** with respect to r:

$$\mathbf{F}_{ij}(r) = -\left\{ \frac{dv^{ss}(r)}{dr} - \left.\frac{dv^{ss}(r)}{dr}\right|_{r=r_c} \right\} \hat{\mathbf{r}}_{ij} \tag{66}$$

where $\hat{\mathbf{r}}_{ij}$ is the unit vector joining the centre of particle i to the centre of j.

6.2.4 Lennard-Jones interaction

A cut-and-shifted Lennard-Jones interaction is available based on the centre-centre separation of pairs. The basic form of the potential is

$$v^{lj}(r) = 4\epsilon \left[\left(\frac{\sigma}{r}\right)^{12} - \left(\frac{\sigma}{r}\right)^{6} \right] \tag{67}$$

where ϵ sets the energy scale, and σ is a characteristic width. The cut-and-shift of both the potential and the associated force is as described in the previous section.

If using the Lennard-Jones potential, some care may be required to ensure that no hard-sphere overlaps occur at the level of the hydrodynamic radius a_h.

6.2.5 Dipole-dipole interactions

For particles with magnetic dipoles $\mathbf{m}_i$, long-range dipole-dipole interactions may be computed using a standard Ewald sum for a periodic system (e.g., [**?**], [**?**]). We describe this here only briefly.

We consider particles with uniform dipoles $\mathbf{m}_i = \mu \mathbf{s}_i$ where μ is the dipole strength and $\mathbf{s}_i$ is a unit vector describing the orientation of the dipole. The basic dipole-dipole interaction is then

$$v^{dd}(r) = \frac{\mu_0 \mu^2}{4\pi} \big(\mathbf{s}_i.\mathbf{s}_j - 3(\mathbf{s}_i.\mathbf{r}_{ij})(\mathbf{s}_j.\mathbf{r}_{ij}) \big), \tag{68}$$

where μ_0 is the permeability. The Ewald sum decomposes this into a short-range part contributing forces and torques computed in real space and reciprocal-space part computed as a sum over reciprocal space vectors. The choice of the real-space cut-off r_c determines the number of reciprocal space vectors required in the sum.

A relevant parameter for magnetic colloid systems is the dipolar coupling constant which measures the strength of the dipole-dipole interaction compared with the thermal energy $k_b T$ (which may be introduced via isothermal fluctuations in the fluid). If we take $\mu_0/4\pi$ to be unity in lattice units, the the dipolar coupling constant is

$$\lambda = \mu^2 / 8 k_b T a^3. \tag{69}$$

6.3 External Forces

6.3.1 Gravity

For the purposes of sedimentation and other driven motions, the code allows a uniform gravitational acceleration on colloids which may be written as $\mathbf{g} = (g_x, g_y, g_z)$. We will consider the simple situation where $\mathbf{g} = (0, 0, -g)$, i.e., the negative z-direction is "downwards."

A colloid of density ρ sedimenting in this gravitational field experiences a net buoyancy force density $(\rho - \rho_0)g = \Delta\rho g$, where ρ_0 is the fluid density. (If the colloid and fluid

density are equal, one can still think in the limit that $\Delta\rho \to 0$ and $g \to \infty$, but where the product $\Delta\rho g$ is finite.)

In a fully periodic system, the application of the force on the colloids will cause a continuous and unboundaed acceleration. To prevent this, and to balance the momentum budget, an equal an opposite body force density is applied uniformly to fluid sites at each time step. In a system with a "bottom" (for example, a flat wall perpendicular to the gravitational force) no such correction is required.

6.3.2 Magnetic field

The the presence of a uniform external magnetic field $\mathbf{B}$, a particle with dipole $\mathbf{s}$ experiences a torque $\mathbf{s} \times \mathbf{B}$.

6.3.3 Electric Field

In the presence of a uniform external electric field $\mathbf{E}$, a particle with charge q experiences a body force $q\mathbf{E}$.

7 Binary Fluids

In this section we discuss symmetric two-component fluids, and the closely related subject of Brazovskii smectics. *Ludwig* was first developed to study issues such as spinodal decomposition in binary mixtures, so we will introduce the background in that context (see, e.g., Swift *et al.* [**?**] and Kendon *et al.*, [**?**] for pioneering work).

Early versions of *Ludwig* used, exclusively, a two lattice Boltzmann distribution scheme to introduce the relevant dynamical update for the binary fluid [**?**]. While this approach is retained in the code for historical reasons it is often advantageous, for a number of reasons which are set out in the following, to use a coupled lattice Boltzmann finite-difference approach. We describe both approaches in what follows, but reccmmend using the finite-difference approach for all current applications. First, we consider the common theoretical background for the two-component binary fluid.

7.1 The Symmetric Free Energy

7.1.1 Thermodynamics

For a symmetric fluid of fixed content, we define a single coarse-grained composition variable, or order parameter, $\phi(\mathbf{r})$. At a finer level, one can interpret the order parameter as $\phi(\mathbf{r}) = (n_1 - n_2)/(n_1 + n_2)$ where the n are number densities of the two components. In either case, it is an important feature of these systems that the order parameter is conserved so that

$$\int \mathrm{d}\mathbf{r}(\phi - \phi_0) = 0 \tag{70}$$

at all times, where ϕ_0 is the mean order parameter. In general, if the order parameter values associated with the two components are ϕ_1 and ϕ_2, then the volume fraction of the first component is $v_1 = (\phi_2 - \phi_0)/(\phi_2 - \phi_1)$, and $v_2 = 1 - v_1$.

For a uniform state at constant volume and temperature we can write the density of the (Helmholtz) free energy as $V(\phi)$. In mean field theory, it is predicted that $V(\phi)$ has uniform positive curvature at high temperature, but becomes concave below a critical temperature T_c. This is illustrated in the left panel of Fig. **??**. For temperatures larger than T_c, there is a homogeneous state with $\phi = 0$ everywhere, i.e., the state is fully mixed. For temperatures below T_c, the free energy is minimised by creating two separate domains, each composed of a single constituent. This results in a phase diagram of the sort shown in Fig. 1 (right panel) with mixed and separated phases bounded by a co-existance curve.

In an initially mixed system spinodal decomposition occurs when a temperature change leaves the system below the spinodal line in the phase diagram. This leaves the system in a globally unstable state, and the response is initially small changes in composition everywhere which grow smoothly in time. These small changes ultimately create interfaces between fluid domains; minimisation of energy then translates to minimisation of interfacial curvature which drives the fluid domains to grow. If the quench leaves the system between the coexistance line and the spinoal line, the system is in a metastable state, and the response is local changes in composition which nucleate droplets.

Note that in some real fluid mixtures, the phase diagram is inverted, and a quench into the coexistance region is upwards in temperature. See, e.g., Chaikin and Lubensky [**?**] for an extended discusion of spinodal decomposition.

Quantitatively, fluid domains together with interfaces are described by a free energy functional with two contributions. An appropriate choice is the Ginzburg-Landau functional, sometimes referred to as the squared gradient model:

$$F[\phi] = \int \mathrm{d}\mathbf{r}[V(\phi) + \tfrac{1}{2}\kappa(\partial_\alpha\phi)^2] \tag{71}$$

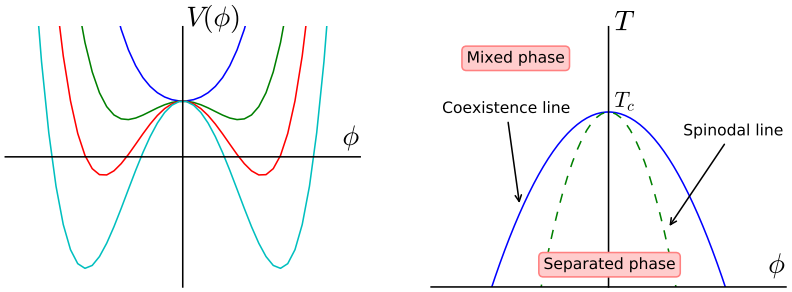


Figure 1: Two panels showing (left) the form of the free energy density $V(\phi)$ for the binary fluid for a number of different temperatures. At temperatures above the critical temperature, there is a single minimum (top curve), while below T_c, two symmetric minima are present (lower curves). The related phase diagram (right) in the temperature-composition plane shows mixed and separated phases the boundary of which is the coexistence line. This (sometimes referred to as the binodal line) is the locus of points which are the minima of $V(\phi)$ at different temperatures. The spinodal line is formed by the locus of points which are the inflections of $V(\phi)$.

where the $V(\phi)$ is as before and the second term penalises gradients in the composition with parameter $\kappa > 0$. The explicit form of $V(\phi)$ is taken here from a general Landau expansion:

$$V(\phi) = \tfrac{1}{2}A\phi^2 + \tfrac{1}{4}B\phi^4. \tag{72}$$

The minima in the potential (cf. Fig. **??**) can then be identified by setting $dV/d\phi = 0$, that is, $A\phi + B\phi^3 = \phi(A + B\phi^2) = 0$. If we take $B > 0$, this condition has two cases: we denote the solutions $\phi^\star$ in both. The first is the mixed case which has $A > 0$ so that we must have $\phi^\star = 0$, and the second is the separated case which has $A < 0$ and so $\phi^\star = \pm(-A/B)^{1/2}$.

The combination of parameters A, B and κ also determine the interfacial tension and the interfacial width. The stationary values of the free energy functional can be obtained from the Euler-Lagrange equation for ϕ. The relevant case is where we have a functional the density of which is written $f(\phi, \partial_\alpha\phi)$ and the corresponding Euler-Lagrange equation is

$$\frac{\partial f}{\partial \phi} - \partial_\alpha \frac{\partial f}{\partial\, \partial_\alpha \phi} = 0. \tag{73}$$

If we consider a one-dimensional case, this is

$$A\phi + B\phi^3 - \kappa\frac{d^2\phi}{dx^2} = 0. \tag{74}$$

Note that the conservation of order parameter means this should be a constrained minimisation, which can be handled by adding the a Lagrange multiplier $\Lambda = \partial f/\partial\phi|_{\phi_0}$. The resulting ordinary differential equation has a solution $\phi(x) = \phi^\star \tanh(x/\xi)$ where ξ is identified as the interfacial width. It can be confirmed that this is a solution if

$$\xi = (-2\kappa/A)^{1/2}. \tag{75}$$

The one-dimensional case can also be used to determine the interfacial tension, which is the excess energy associated with the interface. If we denote the interfacial tension σ, then the excess energy is

$$\sigma = \int_{-\infty}^{+\infty} \mathrm{d}x \Big[\tfrac{1}{2}\kappa(d_x\phi)^2 + V(\phi) - V(\phi^\star)\Big], \tag{76}$$

which can be evaluated from the equilibrium profile with the interface placed at the origin $\phi(x) = \phi^\star \tanh(x/\xi)$, along with a number of standard integrals, to give

$$\sigma = 4\kappa\phi^{\star 2}/3\xi = \left(-8\kappa A^3/9B^2\right)^{1/2}. \tag{77}$$

Further discussion of the practical choice of the free energy parameters, and their impact on interfacial width and tension is deferred until Section **??**.

7.1.2 The free energy density in practice

Writing the free energy density in full, we have

$$f(\phi, \partial_\alpha\phi) = \tfrac{1}{2}A\phi^2 + \tfrac{1}{4}B\phi^4 + \tfrac{1}{2}\kappa(\partial_\alpha\phi)^2. \tag{78}$$

Note that some authors define A to be a positive constant in which case a negtive sign appears in the quadratic term above. We retain A negative to describe the separated case.

7.1.3 Dynamics

In a purely diffusive regime, the time evolution of a conserved order parameter may be expressed as the local divergence of a flux $j_\alpha(\mathbf{r})$, that is, $\partial_t\phi(\mathbf{r}) + \partial_\alpha j_\alpha(\mathbf{r}) = 0$. In this picture, the system responds to small local changes in composition by diffusion, which acts to reduce the energy. Formally, this response of the free energy to a small change in composition is described by the chemical potential

$$\mu = \frac{\delta F}{\delta\phi} = \frac{dV}{d\phi} - \kappa\partial_\alpha^2\phi, \tag{79}$$

being the functional derivative of the free energy with respect to the order parameter. The dynamics is usually described in the context of the model of Cahn and Hilliard [**?**], otherwise known as Model B, where

$$\frac{\partial\phi}{\partial t} = M\partial_\alpha^2\frac{\delta F}{\delta\phi}. \tag{80}$$

Here, the diffusive flux is related to the gradient of chemical potential via $j_\alpha = -M\partial_\alpha\mu$, where M is a mobilty (taken to be uniform).

In the presence of a fluid with velocity field $u_\alpha(\mathbf{r})$, the full equation for the time evolution of the order parameter is

$$\partial_t\phi + \partial_\alpha(\phi u_\alpha + M\partial_\alpha\mu) = 0. \tag{81}$$

This form again stresses the nature of the dynamics as a conservation law, involving the divergence of advective fluxes ϕu_α and diffusive fluxes $j_\alpha = -M\partial_\alpha\mu$. It also admits that the mobility may be a function of position, e.g., via $M = M(\phi)$. Equation **??** will form the basis of the numerical solution of the binary fluid problem, coupled to the Navier-Stokes equation.

The presence of a interface, or non-uniform composition, can lead to thermodynamic stresses on the fluid which we denote $P_{\alpha\beta}$, in additional to the usual viscous stresses. The divergence of this additional stress represents a body force density in the Navier-Stokes equation, which is related to the chemical potential via $f_\alpha = -\phi\partial_\alpha\mu = -\partial_\beta P_{\alpha\beta}$. The full form of the additional stress may be related to the order parameter by

$$P_{\alpha\beta} = p_0\delta_{\alpha\beta} + \kappa\partial_\alpha\phi\partial_\beta\phi \tag{82}$$

with isotropic contribution

$$p_0 = \phi\frac{dV(\phi)}{d\phi} - V(\phi) - \kappa\phi\partial_\alpha^2\phi - \tfrac{1}{2}\kappa(\partial_\alpha\phi)^2. \tag{83}$$

Note that the stress $P_{\alpha\beta}$ is symmetric. At (local) equilibrium, one expects gradients in the chemical potential to vanish, and there will be neither diffusive fluxes ($j_\alpha = 0$) nor force on the fluid ($f_\alpha = 0$).

7.1.4 Surface (or wetting) free energy

In the presence of a solid surface, the two component fluid gives rise to the possiblity of a solid-fluid-fluid contact line. In the symmetric case, the fluid-fluid interface will be at right angles to a uniform flat surface. If one component wets the solid preferentially (that is, it is energetically favourable for that component to be in contact with the solid) the angle may move away from 90 degrees. In general, the equilibrium contact angle θ is given by the Young equation

$$\sigma_1 - \sigma_2 - \sigma\cos\theta = 0, \tag{84}$$

where there are two solid-fluid surface tensions for components 1 and 2, and the fluid-fluid interfacial tension is σ as before.

This may be described in the free energy picture by adding a free energy density per unit area assoicated with the surface:

$$f_s(\phi_s) = \tfrac{1}{2}C\phi_s^2 + H\phi_s, \tag{85}$$

where ϕ_s is the order parameter at the surface, and C and H are constant parameters.

7.2 Implementation

7.2.1 Lattice kinetic approach

A second distribution function is introduced, denoted $g_i(\mathbf{r};t)$, which describes the composition. The moments of this distribution, by analogy with the hydrodynamic case with distribution $f_i(\mathbf{r};t)$ given in Eq. **??**, are written

$$\phi(\mathbf{r};t) = \sum_i g_i(\mathbf{r};t), \quad j_\alpha(\mathbf{r};t) = \sum_i g_i(\mathbf{r};t)c_{i\alpha}, \quad \Phi_{\alpha\beta}(\mathbf{r};t) = \sum_i g_i(\mathbf{r};t)c_{i\alpha}c_{i\beta}. \tag{86}$$

Again, the summation is over index i which counts the number of discrete velocity components for the relevant lattice Boltzmann model N_{vel}. However, for the Cahn-Hilliard equation, the only relevant physical moments are the order parameter $\phi(\mathbf{r};t)$ and the flux $j_\alpha(\mathbf{r};t)$, so the quantity $\Phi_{\alpha\beta}(\mathbf{r};t)$ is counted among the kinetic modes and has no precise physical interpretation. In contrast to the previous section, here $j_\alpha(\mathbf{r};t)$ is used to represent the total flux (advective plus diffusive) rather than the purely diffusive flux.

The lack of direct physical interpretation for the second moment means there are a number of possible choices at the collision stage for the $g_i(\mathbf{r};t)$ distribution.

1. Following Stratford *et al.* [**?**]: relax j_α toward the equilibrium ϕu_α at a rate related to the mobility and fix $\Phi_{\alpha\beta} = \phi u_\alpha u_\beta + \mu\delta_{\alpha\beta}$.
2. Following Kendon *et al.* [**?**]: fix both $j_\alpha = \phi$ and $\Phi = \phi u_\alpha u_\beta + M\mu\delta_{\alpha\beta}$ so the mobilty enters explicitly with the chemical potential.

In both cases a reprojection is used to recover the post-collision distributions $g_i^\star(\mathbf{r};t)$. This explicitly eliminates kinetic modes other than $\Phi_{\alpha\beta}$. Hence

$$g_i = w_i\big(\phi + \phi u_\alpha c_{i\alpha}/c_s^2 + (\mu\delta_{\alpha\beta} + \phi u_\alpha u_\beta)Q_{i\alpha\beta}/2c_s^4\big). \tag{87}$$

In practice, this is not stable, and we use

$$g_i = \phi\delta_{i0} + w_i\big(j_\alpha c_{i\alpha}/c_s^2 + \Phi_{\alpha\beta}Q_{i\alpha\beta}/2c_s^4\big) \tag{88}$$

where the δ_{i0} has the effect of moving most of the order parameter into the non-propagating rest distribution g_0. This reprojection approach has the advantage that it eliminates several spurious anisotropic terms which are introduced by a single relaxation time approach (used by Kendon *et al.*).

In the relaxation approach, the form of the relaxation for the three components of the order parameter flux is

$$j_\alpha^\star = j_\alpha - \tau_\phi^{-1}(j_\alpha - \phi u_\alpha), \tag{89}$$

where $j_\alpha^\star$ is the post-collision flux and the relaxation time τ_ϕ is related to the mobility via

$$\tau = \tfrac{1}{2} + M\rho_0/\Delta t. \tag{90}$$

7.2.2 Finite difference approach

For a number of reasons, it is advantageous to replace the lattice kinetic approach by a conventional finite difference/finite volume treatment of the Cahn-Hilliard equation

$$\partial_t \phi + \partial_\alpha(\phi u_\alpha - M\partial_\alpha \mu) = 0. \tag{91}$$

(Note that the finite difference and finite volume terminology will be used somewhat interchanably here.) Briefly, the advantages are:

1. The ambiguity in interpretation of the kinetic modes is avoided, along with the corresponding computational overhead in memory and memory movement.

2. Additional features may be added to the Cahn-Hilliard equation more easily. These might include a heterogeneous mobility $M = M(\phi)$, and order parameter fluctuations (see below).

8 Liquid Crystals

8.1 The Landau-de Gennes Approach

8.1.1 Tensor order parameter

The liquid crystal order is described by a symmetric traceless tensor $Q_{\alpha\beta}(\mathbf{r};t)$. The largest eigenvalue and associated eigenvector of $Q_{\alpha\beta}$ represent the magnitude and local direction of liquid crystal molecular order. A theory based on the tensor $Q_{\alpha\beta}$ has the advantage of being able to describe disclinations where the local order vanishes and a director is not defined. Readers are referred to, e.g., de Gennes and Prost [**?**], and Wright and Mermin [**?**] for further information.

8.1.2 Free Energy Density

The free energy is a functional whose density $f(Q_{\alpha\beta}, \partial_\gamma Q_{\alpha\beta})$ has bulk contributions depending on $Q_{\alpha\beta}$, and elastic (distortion) contributions dependent on the gradients of the order parameter $\partial_\gamma Q_{\alpha\beta}$.

The bulk contributions are

$$f(Q_{\alpha\beta}) = \tfrac{1}{2}A_0(1-\gamma/3)Q_{\alpha\beta}^2 - \tfrac{1}{3}A_0\gamma Q_{\alpha\beta}Q_{\beta\pi}Q_{\pi\alpha} + \tfrac{1}{4}A_0\gamma(Q_{\alpha\beta}^2)^2. \tag{92}$$

Here, A_0 is a constant which sets the overall energy scale; γ is a temperature-like parameter which controls the position in the phase diagram relative to the isotropic-nematic transition.

The bend, splay and twist distortions making up the gradient free energy are modelled with two elastic constants κ_0 and κ_1 as

$$f(\partial_\gamma Q_{\alpha\beta}) = \tfrac{1}{2}\kappa_0(\partial_\alpha Q_{\alpha\beta})^2 + \tfrac{1}{2}\kappa_1(\epsilon_{\alpha\mu\nu}\partial_\mu Q_{\nu\beta} + 2q_0 Q_{\alpha\beta})^2. \tag{93}$$

Here, $\epsilon_{\alpha\mu\nu}$ is the permutation tensor and $q_0 = 2\pi/p$ is a wavevector related to the pitch p of the cholesteric. Setting $q_0 = 0$ permits a nematic state. The one-constant approximation makes the simplification that $\kappa_0 = \kappa_1 = \kappa$; the distinction between κ_0 and κ_1 will be kept in what follows unless otherwise stated.

8.1.3 Molecular Field

The molecular field $H_{\alpha\beta}$ is defined as the functional derivative of the free energy density with respect to the order parameter which here gives

$$H_{\alpha\beta} = -\left[\frac{\partial f(Q_{\alpha\beta})}{\partial Q_{\alpha\beta}} - \partial_\gamma \frac{\partial f(\partial_\gamma Q_{\alpha\beta})}{\partial Q_{\alpha\beta,\gamma}}\right]. \tag{94}$$

In the absence of flow, the molecular field determines how the order parameter relaxes toward equilibrium.

As the final expression for $H_{\alpha\beta}$ is somewhat complex, it is useful to account for the individual terms here. First, the terms from the bulk free energy density give rise to:

$$-A_0(1-\gamma/3)Q_{\alpha\beta} + A_0\gamma Q_{\alpha\pi}Q_{\pi\beta} - A_0\gamma Q_{\pi\sigma}^2 Q_{\alpha\beta}.$$

The second term in this expression (arising from the cubic term in the free energy) is forced to be traceless by subtracting one third of $\mathrm{Tr}(Q_{\alpha\pi}Q_{\pi\beta}) = Q_{\alpha\beta}^2$ so that we have

$$-A_0(1-\gamma/3)Q_{\alpha\beta} + A_0\gamma(Q_{\alpha\pi}Q_{\pi\beta} - \tfrac{1}{3}Q_{\pi\sigma}^2\delta_{\alpha\beta}) - A_0\gamma Q_{\pi\sigma}^2 Q_{\alpha\beta}.$$

The gradient term in κ_0, along with the first term in κ_1 arising from the expansion of the brackets in Equation **??**, give

$$\kappa_0\partial_\alpha\partial_\gamma Q_{\gamma\beta}. + \kappa_1\partial_\gamma(\partial_\gamma Q_{\alpha\beta} - \partial_\alpha Q_{\gamma\beta}).$$

Note that in the one constant approximation, this term collapses to $\kappa\partial^2 Q_{\alpha\beta}$. The cross term in the expansion of the brackets in **??**, having been symmetrised, contributes

$$-2\kappa_1 q_0(\epsilon_{\alpha\mu\nu}\partial_\mu Q_{\nu\beta} + \epsilon_{\beta\mu\nu}\partial_\mu Q_{\nu\alpha}),$$

which must again be forced to be traceless to give

$$-2\kappa_1 q_0\big\{(\epsilon_{\alpha\mu\nu}\partial_\mu Q_{\nu\beta} + \epsilon_{\beta\mu\nu}\partial_\mu Q_{\nu\alpha}) - \tfrac{1}{3}(\epsilon_{\pi\mu\nu}\partial_\mu Q_{\nu\pi} + \epsilon_{\pi\mu\nu}\partial_\mu Q_{\nu\pi})\delta_{\alpha\beta}\big\}.$$

The final term in is $-4\kappa_1 q_0^2 Q_{\alpha\beta}$. The complete expression for the molecular field is then

$$\begin{aligned} H_{\alpha\beta} \;=\; & -A_0(1-\gamma/3)Q_{\alpha\beta} + A_0\gamma(Q_{\alpha\pi}Q_{\pi\beta} - \tfrac{1}{3}Q_{\pi\sigma}^2\delta_{\alpha\beta}) - A_0\gamma Q_{\pi\sigma}^2 Q_{\alpha\beta} \\ & +\kappa_0\partial_\alpha\partial_\gamma Q_{\gamma\beta} + \kappa_1\partial_\gamma(\partial_\gamma Q_{\alpha\beta} - \partial_\alpha Q_{\gamma\beta}) \\ & -2\kappa_1 q_0\big((\epsilon_{\alpha\mu\nu}\partial_\mu Q_{\nu\beta} + \epsilon_{\beta\mu\nu}\partial_\mu Q_{\nu\alpha}) - \tfrac{1}{3}(\epsilon_{\pi\mu\nu}\partial_\mu Q_{\nu\pi} + \epsilon_{\pi\mu\nu}\partial_\mu Q_{\nu\pi})\delta_{\alpha\beta}\big) \\ & -4\kappa_1 q_0^2 Q_{\alpha\beta}. \end{aligned} \tag{95}$$

8.2 Surface Anchoring

The preferred orientation of the liquid crystal fluid at a solid surface, usually referred to as the surface anchoring, is relevant for both solid walls and colloids.

8.2.1 General boundary condition

To impose a suitable boundary condition for the order parameter tensor at a solid-fluid interface, we consider again the gradient terms in the free energy

$$f(\partial_\gamma Q_{\alpha\beta}) = \tfrac{1}{2}\kappa_0(\partial_\beta Q_{\alpha\beta})^2 + \tfrac{1}{2}\kappa_1(\epsilon_{\alpha\gamma\sigma}\partial_\gamma Q_{\sigma\beta} + 2q_0 Q_{\alpha\beta})^2 \tag{96}$$

where we retain two elastic constants κ_0 and κ_1; the cholesteric pitch $p = 2\pi/q_0$.

Also relevant is a surface free energy (an area density), the exact form of which is determined by the type of anchoring required. In general, we expect this to depend on the order parameter, but not its gradients, so write

$$f_s = f_s(Q_{\alpha\beta}, Q^0_{\alpha\beta}) \tag{97}$$

where $Q^0_{\alpha\beta}$ is some preferred order parameter configuration at the surface (to be either normal or planar as discussed in the following sections).

The boundary condition we wish to apply is derived from the Euler-Lagrange equations, and for the fluid and surface terms gives rise to the equation

$$n_\gamma \frac{\partial f}{\partial Q_{\alpha\beta,\gamma}} + \frac{\partial f_s}{\partial Q_{\alpha\beta}} = 0, \tag{98}$$

where n_γ is the outward unit normal at the surface (pointing into the fluid). Note that the first term in the derivative of the fluid free energy term with respect to $Q_{\alpha\beta,\gamma}$ can be expended as

$$\kappa_0 n_\beta\partial_\gamma Q_{\alpha\gamma} + \kappa_1 n_\gamma(\partial_\gamma Q_{\alpha\beta} - \partial_\alpha Q_{\gamma\beta}) - 2\kappa_1 q_0 n_\gamma\epsilon_{\alpha\gamma\sigma}Q_{\sigma\beta}.$$

However, we should use a symmetric form (the derivative with respect to $Q_{\beta\alpha,\gamma}$ is just as good) so we write this as:

$$\begin{aligned} \tfrac{1}{2}\kappa_0(n_\alpha\partial_\gamma Q_{\beta\gamma} + n_\beta\partial_\gamma Q_{\alpha\gamma}) + \kappa_1 n_\gamma\partial_\gamma Q_{\alpha\beta} - \tfrac{1}{2}\kappa_1 n_\gamma(\partial_\alpha Q_{\gamma\beta} + \partial_\beta Q_{\gamma\alpha}) \\ -\kappa_1 q_0 n_\gamma(\epsilon_{\alpha\gamma\sigma}Q_{\sigma\beta} + \epsilon_{\beta\gamma\sigma}Q_{\sigma\alpha}). \end{aligned}$$

The exact boundary condition will contain this term plus one depending on the exact choice of the surface free energy which describes a given anchoring condition.

8.2.2 Normal or homeotropic anchoring

The preferred direction of the surface order here is, as the name suggests, normal to surface. We can write

$$f_s = \tfrac{1}{2} w (Q_{\alpha\beta} - Q^0_{\alpha\beta})^2 \tag{99}$$

where w is a constant. The preferred orientation $Q^0_{\alpha\beta}$ is based on the unit normal at the surface n_γ, and is computed via the uniaxial approximation:

$$Q^0_{\alpha\beta} = \tfrac{1}{2} A (3 n_\alpha n_\beta - \delta_{\alpha\beta}). \tag{100}$$

The amplitude A is provided by

$$A = \frac{2}{3}\left(\frac{1}{4} + \frac{3}{4}\sqrt{1 - \frac{8}{3\gamma}}\right). \tag{101}$$

The full boundary condition for the gradient of the tensor order parameter at the solid-fluid boundary from Equation **??** is then:

$$\begin{aligned} \tfrac{1}{2}\kappa_0 (n_\alpha \partial_\gamma Q_{\beta\gamma} + n_\beta \partial_\gamma Q_{\alpha\gamma}) + \kappa_1 n_\gamma \partial_\gamma Q_{\alpha\beta} - \tfrac{1}{2}\kappa_1 n_\gamma (\partial_\alpha Q_{\gamma\beta} + \partial_\beta Q_{\gamma\alpha}) \\ -\kappa_1 q_0 n_\gamma (\epsilon_{\alpha\gamma\sigma} Q_{\sigma\beta} + \epsilon_{\beta\gamma\sigma} Q_{\sigma\alpha}) - w(Q_{\alpha\beta} - Q^0_{\alpha\beta}) = 0. \end{aligned} \tag{102}$$

8.2.3 Planar (degenerate) anchoring

For planar anchoring, the preferred orientation is in the local tangent plane at the surface: this is a degenerate case as any orientation in the plane is energetically equivalent. An appropriate boundary condition is described by Fournier and Galatola [**?**] which we write as:

$$f_s = \tfrac{1}{2} w_1 (\tilde{Q}_{\alpha\beta} - \tilde{Q}^\perp_{\alpha\beta})^2 + \tfrac{1}{2} w_2 (\tilde{Q}^2_{\alpha\beta} - A^2)^2. \tag{103}$$

Here again, the amplitude A is provided by Equation **??**. To compute this term we take the local fluid order parameter $Q_{\alpha\beta}$, form the quantity

$$\tilde{Q}_{\alpha\beta} = Q_{\alpha\beta} + \tfrac{1}{2} A \delta_{\alpha\beta}$$

which is then projected onto the tangent plane via $\tilde{Q}^\perp_{\alpha\beta} = P_{\alpha\gamma}\tilde{Q}_{\gamma\sigma}P_{\sigma\beta}$ with the local surface normal entering through $P_{\alpha\beta} = \delta_{\alpha\beta} - n_\alpha n_\beta$. The full boundary condition resulting from Equation **??** is then

$$\begin{aligned} \tfrac{1}{2}\kappa_0 (n_\alpha \partial_\gamma Q_{\beta\gamma} + n_\beta \partial_\gamma Q_{\alpha\gamma}) + \kappa_1 n_\gamma \partial_\gamma Q_{\alpha\beta} - \tfrac{1}{2}\kappa_1 n_\gamma (\partial_\alpha Q_{\gamma\beta} + \partial_\beta Q_{\gamma\alpha}) \\ -\kappa_1 q_0 n_\gamma (\epsilon_{\alpha\gamma\sigma} Q_{\sigma\beta} + \epsilon_{\beta\gamma\sigma} Q_{\sigma\alpha}) - w_1 (\tilde{Q}_{\alpha\beta} - \tilde{Q}^\perp_{\alpha\beta}) - 2 w_2 (\tilde{Q}^2_{\alpha\beta} - A^2)\tilde{Q}_{\alpha\beta} = 0. \end{aligned} \tag{104}$$

8.3 Dynamics

The time evolution of the orientational order parameter $Q_{\alpha\beta}$ in the presence of flow can be described by the Beris-Edwards equation [**?**].

$$\partial_t Q_{\alpha\beta} + \partial_\gamma (u_\gamma Q_{\alpha\beta}) + S_{\alpha\beta}(W_{\alpha\beta}, Q_{\alpha\beta}) = \Gamma H_{\alpha\beta}. \tag{105}$$

This relates the time rate of change of the order parameter to terms involving advection, the response to shear $S_{\alpha\beta}(W_{\alpha\beta}, Q_{\alpha\beta})$, and the molecular field.

The advective term involves the fluid velocity u_α and the shear term involves the velocity gradient tensor $W_{\alpha\beta} = \partial_\beta u_\alpha$. The full definition of the shear term is:

$$\begin{aligned} S_{\alpha\beta}(W_{\alpha\beta}, Q_{\alpha\beta}) \quad &= (\xi D_{\alpha\pi} + \Omega_{\alpha\pi})(Q_{\pi\beta} + \tfrac{1}{3}\delta_{\pi\beta}) + (Q_{\alpha\pi} + \tfrac{1}{3}\delta_{\alpha\pi})(\xi D_{\pi\beta} - \Omega_{\pi\beta}) \\ &\quad -2\xi (Q_{\alpha\beta} + \tfrac{1}{3}\delta_{\alpha\beta}) Q_{\pi\sigma} W_{\sigma\pi} \end{aligned}$$

where $D_{\alpha\beta} = \frac{1}{2}(W_{\alpha\beta} + W_{\beta\alpha})$ and $\Omega_{\alpha\beta} = \frac{1}{2}(W_{\alpha\beta} - W_{\beta\alpha})$ are the symmetric and antisymmetric contributions to the velocity gradient tensor and ξ is a material constant representing an effective molecular aspect ratio.

The thermodynamic contribution to the stress on the fluid can be viewed as the sum of three parts:

$$\Pi_{\alpha\beta} = \sigma_{\alpha\beta} + \tau_{\alpha\beta} - \partial_\alpha Q_{\pi\nu} \frac{\delta \mathcal{F}}{\delta \partial_\beta Q_{\pi\nu}} \tag{106}$$

where $\sigma_{\alpha\beta}$ and $\tau_{\alpha\beta}$ are the symmetric and antisymmetric contributions from the liquid crystal order:

$$\sigma_{\alpha\beta} = -p_0\delta_{\alpha\beta} - \xi H_{\alpha\pi}(Q_{\pi\beta} + \tfrac{1}{3}\delta_{\pi\beta}) - \xi(Q_{\alpha\pi} + \tfrac{1}{3}\delta_{\alpha\pi})H_{\pi\beta} + 2\xi(Q_{\alpha\beta} + \tfrac{1}{3}\delta_{\alpha\beta})Q_{\pi\nu}H_{\pi\nu}, \tag{107}$$

where p_0 is the isotropic pressure, and

$$\tau_{\alpha\beta} = Q_{\alpha\pi}H_{\pi\beta} - H_{\alpha\pi}Q_{\pi\beta}. \tag{108}$$

The final term in Equation **??** is expanded as

$$-\kappa_0 \partial_\alpha Q_{\pi\beta} \partial_\nu Q_{\pi\nu} - \kappa_1 \partial_\alpha Q_{\pi\nu} \left[\partial_\beta Q_{\pi\nu} - \partial_\pi Q_{\nu\beta} + 2q_0 \epsilon_{\gamma\beta\pi} Q_{\gamma\nu}\right].$$

8.3.1 Active stress

An additional term may be added to the stress to model an active apolar fluid. This term is

$$\Pi^a_{\alpha\beta} = -\zeta(Q_{\alpha\beta} - \tfrac{1}{3}\delta_{\alpha\beta}) \tag{109}$$

where ζ is an activity parameter: $\zeta < 0$ gives contractile behaviour and $\zeta > 0$ gives extensile behaviour.

8.4 Implementation

8.4.1 Fluid

The time evolution of the order parameter is computed by a hybrid approach where the hydrodynamics is in the LB sector, and the Beris-Edwards Equation is solved in a finite difference approach. The hydrodynamic quantities and the order parameter share the same regular lattice. The velocity field, computed via LB, supplies advective fluxes and the velocity gradient tensor using an appropriate finite difference stencil.

Coupling to the Navier-Stokes equations is via a body force computed locally at each lattice site via the divergence of the stress in Equation **??**.

It is also possible to compute solely relaxational dynamics where there is no flow, and evolution only depends on the molecular field, and the rotational diffusion constant.

8.4.2 Solid

First, we note that the assignment of solid and fluid lattice nodes for the order parameter follows that for the density: inside and outside are distinguished using the nominal radius of the colloid a_0 and its position. It is useful, in addition, to think about a series of control volumes surrounding each lattice node whose faces are aligned with the lattice (see Figure **??**). A set of these faces constitute the solid-fluid boundary in the hybrid picture. Advective fluxes of order parameter are computed at the faces of the control volumes, and the boundary condition is zero normal flux at solid-fluid interfaces. Note that the colloid is assumed to be stationary in assigning these fluxes.

The net hydrodynamic force on the colloid is computed via BBL in the usual way. As discussed above, the force on the fluid originating from the order parameter is computed

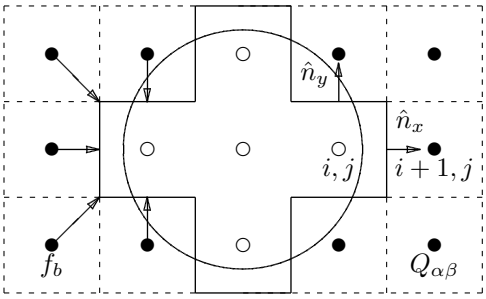


Figure 2: The hybrid picture. In the lattice Boltzmann picture (left) a surface is defined by a set of links f_b, which involve discrete vectors $\mathbf{c}_b \Delta t$ which connect fluid and solid sites. For the order parameter (right), the colloid is represented by the set of faces, e.g., that between sites i, j and $i+1, j$ with unit normal $\hat{n}_x$. This is a rather small colloid.

via the discrete divergence of the stress $\Pi_{\alpha\beta}$. In the fluid, this is implemented by interpolating $\Pi_{\alpha\beta}$ to the control volume faces and taking differences between faces in each direction. This method has the advantage that, with the introduction of solid faces, an extrapolation of $\Pi_{\alpha\beta}$ to the solid-fluid boundary is possible. This allows one to compute the divergence of the stress at fluid nodes adjacent to the colloid in the normal way. At the same time, the discrete equivalent of

$$F_\alpha^{\text{coll}} = \int \Pi_{\alpha\beta} \hat{n}_\beta dS \tag{110}$$

may be found by summing $\Pi_{\alpha\beta}$ over the relevant solid-fluid control volume faces. By construction, this ensures that momentum lost by the fluid is gained by the colloid, i.e., global momentum is conserved.

Finally, movement of the colloid across the lattice is accompanied by changes in its discrete shape. When fluid sites are destroyed, corresponding order parameter information is also lost. If new fluid sites are created, new order parameter information may be added locally either by interpolation from nearby fluid sites, or from geometrical information from the local surface anchoring.

8.4.3 Boundary conditions

In three dimensions, the boundary condition Eq. **??** provides six equations containing (potentially) 18 unknown derivatives $\partial_\gamma Q_{\alpha\beta}$ corresponding to the 6 elements of the order parameter tensor Q_{xx}, Q_{xy}, Q_{xz}, Q_{yy}, Q_{yz}, Q_{zz}. While the equation corresponding to Q_{zz} must appear to retain isotropy, Q_{zz} itself, and its derivatives, may be replaced via the constraint on the trace of $Q_{\alpha\beta}$. We can therefore either solve a fully determined system including Q_{zz}, and then impose tracelessness on the result, or replace Q_{zz} and solve six equations for five unknowns, with the sixth equation acting as the constraint. These methods provide the same answer for cases where the surface normal is along the coordinate directions.

In general, Equation **??** is slightly cumbersome: the coefficients of the various derivatives for each of the six equations are shown in Table **??**. As an concrete example, at a flat surface with normal anchoring and, e.g., $n = (1, 0, 0)$, the number of unknowns (six) are the gradients $\partial_x Q_{\alpha\beta}$ at the boundary — we assume the tangential gradients $\partial_y Q_{\alpha\beta}$ and $\partial_z Q_{\alpha\beta}$ can be approximated using the standard differencing method involving only fluid values of $Q_{\alpha\beta}$. We proceed by computing the constant terms relevant for normal anchoring

$$-\kappa_1 q_0 n_\gamma (\epsilon_{\alpha\gamma\sigma} Q_{\sigma\beta} + \epsilon_{\beta\gamma\sigma} Q_{\sigma\alpha}) - w(Q_{\alpha\beta} - Q^0_{\alpha\beta})$$

	$Q_{xx,x}$	$Q_{xy,x}$	$Q_{xz,x}$	$Q_{yy,x}$	$Q_{yz,x}$	$Q_{zz,x}$
Q_{xx}	$\kappa_0 n_x$	$-\kappa_1 n_y$	$-\kappa_1 n_z$			
Q_{xy}	$\kappa_0 n_y$	$\kappa' n_x$		$-\kappa_1 n_y$	$-\kappa_1 n_z$	
Q_{xz}	$\kappa_0 n_z$		$\kappa' n_x$		$-\kappa_1 n_y$	$-\kappa_1 n_z$
Q_{yy}		$\kappa_0 n_y$		$\kappa_1 n_x$		
Q_{yz}		$\kappa_0 n_z$	$\kappa_0 n_y$		$2\kappa_1 n_x$	
Q_{zz}			$\kappa_0 n_z$			$\kappa_1 n_x$

	$Q_{xx,y}$	$Q_{xy,y}$	$Q_{xz,y}$	$Q_{yy,y}$	$Q_{yz,y}$	$Q_{zz,y}$
Q_{xx}	$\kappa_1 n_y$	$\kappa_0 n_x$				
Q_{xy}	$-\kappa_1 n_x$	$\kappa' n_y$	$-\kappa_1 n_z$	$\kappa_0 n_x$		
Q_{xz}		$\kappa_0 n_z$	$2\kappa_1 n_y$		$\kappa_0 n_x$	
Q_{yy}		$-\kappa_1 n_x$		$\kappa_0 n_y$	$-\kappa_1 n_z$	
Q_{yz}			$-\kappa_1 n_x$	$\kappa_0 n_z$	$\kappa' n_y$	$-\kappa_1 n_z$
Q_{zz}					$\kappa_0 n_z$	$\kappa_1 n_y$

	$Q_{xx,z}$	$Q_{xy,z}$	$Q_{xz,z}$	$Q_{yy,z}$	$Q_{yz,z}$	$Q_{zz,z}$
Q_{xx}	$\kappa_1 n_z$		$\kappa_0 n_x$			
Q_{xy}		$2\kappa_1 n_z$	$\kappa_0 n_y$		$\kappa_0 n_x$	
Q_{xz}	$-\kappa_1 n_x$	$-\kappa_1 n_y$	$\kappa' n_z$			$\kappa_0 n_x$
Q_{yy}				$\kappa_1 n_z$	$\kappa_0 n_y$	
Q_{yz}		$-\kappa_1 n_x$		$-\kappa_1 n_y$	$\kappa' n_z$	$\kappa_0 n_y$
Q_{zz}			$-\kappa_1 n_x$		$-\kappa_1 n_y$	$\kappa_0 n_z$

Table 4: Coefficients of the various derivatives of the order parameter tensor appearing in six equations for the elements of the order parameter (including Q_{zz}). Note $\kappa_0 + \kappa_1 = \kappa'$ and all the coefficients have been multiplied by a factor of 2 in the off-diagonal equations.

using $Q_{\alpha\beta}$ from the adjacent fluid site. To these constant terms are added the tangential gradients. The gradients at the surface are then computed by solving a 6x6 linear algebra problem for $\partial_x Q_{\alpha\beta}$. This allows the full gradient at the adjacent fluid site to be constructed.

At concave edges or corners, where it is not possible to compute the tangential gradients from the usual stencil as for a flat interface, a different approach is required. Here, either a 12×12 or 18×18 system of equations is solved containing the relevant unknown coefficients from Table **??** and the relevant constant terms computed as before.

9 Liquid Crystal Emulsions

This section describes the approach used to model liquid crystal emulsions which might include, for example, droplets of liquid crystal in water. This combines elements of the binary fluid model discussed in Section **??** with the liquid crystal approach of Section **??**.

9.1 Combined Approach

9.1.1 Order parameters

To describe a liquid crystal emulsion, two order parameters are used: the composition $\phi(\mathbf{r};t)$ and the orientational order $Q_{\alpha\beta}(\mathbf{r};t)$.

9.1.2 Free energy densities

The free energy density is here the sum of three contributions:

$$f(\phi, Q_{\alpha\beta}) = f_{\rm c}(\phi) + f_{\rm lc}(Q_{\alpha\beta}) + f_{\rm anchoring}(\phi, Q_{\alpha\beta}). \tag{111}$$

The first contribution relates to the composition using the standard symmetric approach

$$f_{\rm c}(\phi) = \tfrac{1}{2}A\phi^2 + \tfrac{1}{4}B\phi^4 + \tfrac{1}{2}\kappa(\partial_\alpha\phi)^2 \tag{112}$$

(cf. Equation **??**), where we recall that in the separated regime A is a negative constant, B is a positive constant, and κ is a positive constant which determines the interfacial tension. The extremal values of the composition are $\phi^\star = (-A/B)^{1/2}$ while uniformly mixed components in equal proportion have $\phi = 0$.

The second contribution is the liquid crystal bulk free energy density (Equation **??**):

$$\begin{aligned} f_{\rm lc} &= \tfrac{1}{2}A_0\left(1-\gamma(\phi)/3\right)Q_{\alpha\beta}^2 - \tfrac{1}{3}A_0\gamma(\phi)Q_{\alpha\beta}Q_{\beta\pi}Q_{\pi\alpha} + \tfrac{1}{4}A_0\gamma(\phi)(Q_{\alpha\beta})^2 \\ &+ \tfrac{1}{2}\kappa_0(\partial_\alpha Q_{\alpha\beta})^2 + \tfrac{1}{2}\kappa_1(\epsilon_{\alpha\mu\nu}\partial_\mu Q_{\nu\beta} + 2q_0 Q_{\alpha\beta})^2, \end{aligned} \tag{113}$$

where in the first line of Equation **??** the composition enters parametrically through the control parameter $\gamma = \gamma(\phi)$. This will be discussion further below. The distortion terms in the second line of Equation **??** related to the elastic constants κ_0 and κ_1 are unchanged (and are as appear in Equation **??**).

Finally, there is a coupling between the liquid crystal and the composition which determines the anchoring (normal or planar) at the boundary of the droplet:

$$f_{\rm anchoring}(\phi, Q_{\alpha\beta}) = WQ_{\alpha\beta}\partial_\alpha\phi\partial_\beta\phi. \tag{114}$$

For $W > 0$ the anchoring is planar, and for $W < 0$ the anchoring is normal.

9.1.3 Control of the isotropic/nematic transition

The function $\gamma(\phi)$ allows control of the liquid crystal ordering as the composition changes. An appropriate choice of function for the case of $\phi^\star = 1$ is

$$\gamma(\phi) = \gamma_0 + \Delta\,(1+\phi) \tag{115}$$

so that $\gamma(\phi = -1) = \gamma_0$, and $\gamma(\phi = 1) = \gamma_0 + 2\Delta$. We should choose γ_0 and Δ such that $\gamma(\phi = -1)$ is smaller than 2.7 (so that for $\phi = -1$ the fluid is isotropic) and $\gamma(\phi = +1)$ is larger than 2.7 (so that when $\phi = 1$ the liquid crystal is in the ordered phase, nematic or cholesteric etc.). Experience suggests that for example $\gamma_0 = 2.5$ and $\Delta = 0.25$ is a suitable choice.

9.1.4 Chemical potential

The chemical potential $\mu(\phi, Q_{\alpha\beta})$ resulting from the combined free energy density is the functional derivative of the total energy with respect to the composition ϕ,

$$\mu = \mu_{\rm c} + \mu_{\rm lc} + \mu_{\rm anchoring} = \frac{\delta F}{\delta \phi} = \frac{\partial F}{\partial \phi} - \partial_\alpha \frac{\partial F}{\partial(\partial_\alpha \phi)} \tag{116}$$

In the case that $\gamma(\phi) \sim \Delta \cdot \phi$ we generate terms

$$\mu_{\rm c} + \mu_{\rm lc} = A\phi + B\phi^3 - \kappa \partial_\gamma \partial_\gamma \phi - \tfrac{1}{6} A_0 \Delta Q_{\alpha\beta}^2 - \tfrac{1}{3} A_0 \Delta Q_{\alpha\beta} Q_{\beta\pi} Q_{\pi\alpha} + \tfrac{1}{4} A_0 \Delta (Q_{\alpha\beta}^2)^2 \tag{117}$$

from $f_{\rm c}(\phi)$ and $f_{\rm lc}(Q_{\alpha\beta})$. The interfacial or anchoring contribution to the chemical potential is

$$\mu_{\rm anchoring} = -W \left(\partial_\alpha \phi \partial_\beta Q_{\alpha\beta} + \partial_\beta \phi \partial_\alpha Q_{\alpha\beta} \right) - 2W Q_{\alpha\beta} \partial_\alpha \partial_\beta \phi. \tag{118}$$

9.1.5 Molecular field

The molecular field resulting from the combined free energy is the functional derivative of the total free energy with respect to the tensor order parameter $Q_{\alpha\beta}$,

$$H_{\alpha\beta} = H_{\alpha\beta,{\rm lc}} + H_{\alpha\beta,{\rm anchoring}} = \frac{\delta F}{\delta Q_{\alpha\beta}} = \frac{\partial F}{\partial Q_{\alpha\beta}} - \partial_\gamma \frac{\partial F}{\partial(\partial_\gamma Q_{\alpha\beta})} \tag{119}$$

The terms in the molecular field resulting from the bulk free energy density of the liquid crystalline part are:

$$H_{\alpha\beta,{\rm lc}} = -A_0(1 - \gamma(\phi)/3) Q_{\alpha\beta} + A_0 \gamma(\phi) \left[Q_{\alpha\pi} Q_{\pi\beta} - \tfrac{1}{3} \delta_{\alpha\beta} Q_{\pi\sigma}^2 \right] - A_0 \gamma(\phi) Q_{\pi\nu}^2 Q_{\alpha\beta}$$

where the terms related to the elastic contants κ_0 and κ_1 have been omitted (cf. Equation **??**). The interfacial contribution to the molecular field is

$$H_{\alpha\beta,{\rm anchoring}} = -W \left(\partial_\alpha \phi \partial_\beta \phi - \tfrac{1}{3} \partial_\pi \phi \partial_\pi \phi \delta_{\alpha\beta} \right). \tag{120}$$

As before, relevant contributions to the molecular field have been rendered traceless.

9.1.6 Stress and the force on the fluid

The thermodynamic contribution to the pressure tensor is

$$\Pi_{\alpha\beta} = \sigma_{\alpha\beta} + \tau_{\alpha\beta} + P_{\alpha\beta} \tag{121}$$

where $\sigma_{\alpha\beta}$ and $\tau_{\alpha\beta}$ are the symmetric and antisymmetric contributions arising from the liquid crystalline order:

$$\sigma_{\alpha\beta} = -p_0 \delta_{\alpha\beta} - \xi H_{\alpha\pi} \left(Q_{\pi\beta} + \tfrac{1}{3} \delta_{\alpha\beta} \right) - \xi \left(Q_{\alpha\pi} + \tfrac{1}{3} \delta_{\alpha\pi} \right) H_{\pi\beta} + 2\xi \left(Q_{\alpha\beta} + \tfrac{1}{3} \delta_{\alpha\beta} \right) Q_{\pi\nu} H_{\pi\nu}$$

and

$$\tau_{\alpha\beta} = Q_{\alpha\pi} H_{\pi\beta} - H_{\alpha\pi} Q_{\pi\beta}.$$

The term $P_{\alpha\beta}$ includes the binary fluid and coupling (between LC and binary fluid) contributions and is somewhat more complicated than in the bare liquid crystal case:

$$P_{\alpha\beta} = -\left(\phi \frac{\delta F}{\delta \phi} - F \right) \delta_{\alpha\beta} - \frac{\delta F}{\delta \left(\partial_\beta \phi \right)} \partial_\alpha \phi - \frac{\delta F}{\delta \left(\partial_\beta Q_{\gamma\nu} \right)} \partial_\alpha Q_{\gamma\nu}, \tag{122}$$

where we have re-introduced the full free energy functional F. This term is discussed in the following section.

9.2 Implementation

It is convenient [**?**] to rewrite a divergence of the stress $-\partial_\beta P_{\alpha\beta}$ with the stress defined in Equation **??**. This force can be expanded in full as

$$\partial_\alpha \left(\phi\mu - F\right) + \partial_\beta \left(\frac{\delta F}{\delta\left(\partial_\beta\phi\right)}\right)\partial_\alpha\phi + \frac{\delta F}{\delta\left(\partial_\beta\phi\right)}\partial_\beta\left(\partial_\alpha\phi\right) + \partial_\beta\left(\frac{\delta F}{\delta Q_{\pi\nu,\beta}}\right)\partial_\alpha Q_{\pi\nu} + \frac{\delta F}{\delta Q_{\pi\nu,\beta}}\partial_\beta\left(\partial_\alpha Q_{\pi\nu}\right). \tag{123}$$

Applying the chain rule on the first bracket in Equation **??** leads to

$$\partial_\alpha(\phi\mu) - \frac{\partial F}{\partial\phi}\partial_\alpha\phi - \frac{\partial F}{\partial(\partial_\beta\phi)}\partial_\alpha(\partial_\beta\phi) - \frac{\partial F}{\partial Q_{\pi\nu}}\partial_\alpha Q_{\pi\nu} - \frac{\partial F}{\partial Q_{\pi\nu,\beta}}\partial_\alpha Q_{\pi\nu,\beta}.$$

The expansion of the functional derivatives in Equation **??** gives

$$\partial_\beta\left(\frac{\partial F}{\partial(\partial_\beta\phi)} - \partial_\gamma\frac{\partial F}{\partial(\partial_\gamma\partial_\beta\phi)}\right)\partial_\alpha\phi + \left(\frac{\partial F}{\partial(\partial_\beta\phi)} - \partial_\gamma\frac{\partial F}{\partial(\partial_\gamma\partial_\beta\phi)}\right)\partial_\beta\left(\partial_\alpha\phi\right)$$
$$\partial_\beta\left(\frac{\partial F}{\partial Q_{\pi\nu,\beta}} - \partial_\gamma\frac{\partial F}{\partial(\partial_\gamma Q_{\pi\nu,\beta)}}\right)\partial_\alpha Q_{\pi\nu} + \left(\frac{\partial F}{\partial Q_{\pi\nu,\beta}} - \partial_\gamma\frac{\partial F}{\partial(\partial_\gamma Q_{\pi\nu,\beta)}}\right)\partial_\beta Q_{\pi\nu,\alpha}.$$

Using the symmetry of partial derivatives and

$$\frac{\partial F}{\partial(\partial_\gamma\partial_\beta\phi)} = \frac{\partial F}{\partial(\partial_\gamma Q_{\mu\nu,\beta)}} = 0$$

this can be recast with some manipulation as

$$\begin{aligned} -\partial_\beta P_{\alpha\beta} &= -\left(\frac{\partial F}{\partial Q_{\pi\nu}} - \partial_\beta\frac{\partial F}{\partial Q_{\pi\nu,\beta}}\right)\partial_\alpha Q_{\pi\nu} - \left(\frac{\partial F}{\partial\phi} - \partial_\beta\frac{\partial F}{\partial\left(\partial_\beta\phi\right)}\right)\partial_\alpha\phi + \partial_\alpha\left(\phi\mu\right) \\ &= -\left(\frac{\delta F}{\delta Q_{\pi\nu}}\right)\partial_\alpha Q_{\pi\nu} - \left(\frac{\delta F}{\delta\phi}\right)\partial_\alpha\phi + \partial_\alpha\left(\phi\mu\right) \\ &= H_{\pi\nu}\,\partial_\alpha Q_{\pi\nu} + \phi\,\partial_\alpha\mu, \end{aligned} \tag{124}$$

which is a local force on the fluid. The total force on the fluid locally is the divergence of the stress, where the total stress is $\Pi_{\alpha\beta}$ given by equation (**??**). It is sometimes useful to compute the various contributions to the force in different ways. The contribution from the symmetric and antisymmetric stress $\sigma_{\alpha\beta}$ and $\tau_{\alpha\beta}$ can be calculated numerically as a divergence. The contribution from $P_{\alpha\beta}$ can be computed numerically as a divergence, or via the analytical form of the divergence Equation **??**. The last option is at the expense of global conservation of momentum.

10 Electrokinetics

10.1 Sress tensor in the electrosymmetic model

10.1.1 Definitions

The total free energy functional is a function of the composition ϕ, the density of the ionic species ρ_α,

$$\mathcal{F}[\phi, \{\rho_\alpha\}] = \int d\mathbf{r}\; F\left(\phi(\mathbf{r}), \{\rho_\alpha(\mathbf{r})\}\right). \tag{125}$$

The free energy density can be factorised into a contribution from the composition and an ionic contribution according to

$$\begin{aligned} F &= F^{mix} + F^{ion} && (126)\\ F^{mix} &= \frac{1}{2} A\,\phi^2 + \frac{1}{4} B\,\phi^4 + \frac{1}{2}\kappa(\boldsymbol{\nabla}\phi)^2 && (127)\\ F^{ion,ex} &= \sum_{\alpha=\pm} \rho_\alpha \left(V_\alpha^{solv} - \mu_\alpha + \frac{z_\alpha e}{2}\Psi\right). && (128) \end{aligned}$$

In the ionic contribution we have omitted the ideal gas contribution of the ions. The solvation energy is given by

$$V_\alpha^{solv}(\mathbf{r}) = \Delta\mu_\alpha \frac{1+\phi(\mathbf{r})}{2} \tag{129}$$

It is convenient to define the total free energy without electric field F_0, i.e. with vanishing electric potential:

$$F_0 = F^{mix} + F^{ion,ex}|_{\Psi=0} \tag{130}$$

The Poisson equation reads

$$\boldsymbol{\nabla}\left(\varepsilon(\mathbf{r})\boldsymbol{\nabla}\Psi(\mathbf{r})\right) = -\sum_{\alpha=\pm} e\, z_\alpha\, \rho_\alpha(\mathbf{r}) \tag{131}$$

whereas the electric field is given by

$$\boldsymbol{E} = -\boldsymbol{\nabla}\Psi. \tag{132}$$

The permittivity depends on the composition ϕ via

$$\varepsilon(\mathbf{r}) = \bar{\varepsilon}\left(1 - \gamma\,\phi(\mathbf{r})\right) \tag{133}$$

The electric displacement field

$$\boldsymbol{D}(\mathbf{r}) = \varepsilon(\mathbf{r})\boldsymbol{E}(\mathbf{r}) \tag{134}$$

The 'co-energy' density

$$\tilde{F} = F_0 - \frac{\varepsilon}{2}\boldsymbol{E}^2 \tag{135}$$

has the meaning of a Legendre transform of the free energy density F_0.

10.1.2 Electromechanical stress tensor

The electromechanical stress tensor is generally defined as [**?**, **?**].

$$\sigma_{ij} = \varepsilon E_i E_j + \delta_{ij}\left(\tilde{F} - \phi\frac{\delta\tilde{\mathcal{F}}}{\delta\phi} - \sum_{\alpha=\pm}\rho_\alpha\frac{\delta\tilde{\mathcal{F}}}{\delta\rho_\alpha}\right). \tag{136}$$

The first functional derivative yields

$$\begin{aligned}
\phi\frac{\delta\tilde{\mathcal{F}}}{\delta\phi} &= \phi\left(\frac{\delta\mathcal{F}_0}{\delta\phi} - \frac{\boldsymbol{E}^2}{2}\frac{\partial\varepsilon}{\partial\phi}\right) && (137)\\
&= \phi\left(\frac{\partial F_0}{\partial\phi} - \boldsymbol{\nabla}\left(\frac{\partial F_0}{\partial\boldsymbol{\nabla}\phi}\right) - \frac{\boldsymbol{E}^2}{2}\frac{\partial\varepsilon}{\partial\phi}\right) && (138)\\
&= A\,\phi^2 + B\,\phi^4 - \kappa\phi(\boldsymbol{\nabla}^2\phi) + \frac{\phi}{2}\sum_{\alpha=\pm}\rho_\alpha\Delta\mu_\alpha + \phi\,\frac{\bar{\varepsilon}\gamma}{2}\boldsymbol{E}^2 && (139)\\
&= A\,\phi^2 + B\,\phi^4 - \kappa\phi(\boldsymbol{\nabla}^2\phi) + \frac{\phi}{2}\sum_{\alpha=\pm}\rho_\alpha\Delta\mu_\alpha - \frac{\varepsilon-\bar{\varepsilon}}{2}\boldsymbol{E}^2, && (140)
\end{aligned}$$

where we used the above dependence of the permittivity on the composition. The second functional derivative is given by

$$\begin{aligned}
\rho_\alpha\frac{\delta\tilde{\mathcal{F}}}{\delta\rho_\alpha} &= \rho_\alpha\left(\frac{\delta\mathcal{F}_0}{\delta\rho_\alpha}\right) && (141)\\
&= \rho_\alpha(V_\alpha^{solv} - \mu_\alpha) && (142)
\end{aligned}$$

The co-energy density and the functional derivatives with respect to composition ϕ and ionic densities ρ_α add up to

$$\begin{aligned}
&\tilde{F} - \phi\frac{\delta\tilde{F}}{\delta\phi} - \sum_{\alpha=\pm}\rho_\alpha\frac{\delta\tilde{F}}{\delta\rho_\alpha} = \\
&\quad= -\frac{\varepsilon}{2}\boldsymbol{E}^2 + \frac{A}{2}\phi^2 + \frac{B}{4}\phi^4 + \frac{\kappa}{2}(\boldsymbol{\nabla}\phi)^2 \\
&\qquad -A\phi^2 - B\phi^4 + \kappa\phi(\boldsymbol{\nabla}^2\phi) - \frac{\phi}{2}\sum_{\alpha=\pm}\rho_\alpha\Delta\mu_\alpha \\
&\qquad -\phi\,\frac{\bar{\varepsilon}\gamma}{2}\boldsymbol{E}^2 - \sum_{\alpha=\pm}\rho_\alpha(V_\alpha^{solv} - \mu_\alpha) && (143)\\
&\quad= -\frac{\varepsilon}{2}\boldsymbol{E}^2 - \frac{A}{2}\phi^2 - \frac{3B}{4}\phi^4 + \frac{\kappa}{2}(\boldsymbol{\nabla}\phi)^2 + \kappa\phi(\boldsymbol{\nabla}^2\phi) \\
&\qquad -\frac{\phi}{2}\sum_{\alpha=\pm}\rho_\alpha\Delta\mu_\alpha - \phi\,\frac{\bar{\varepsilon}\gamma}{2}\boldsymbol{E}^2 && (144)
\end{aligned}$$

This yields

$$\begin{aligned}
\sigma_{ij} &= \varepsilon E_i E_j - \delta_{ij}\left(\frac{\varepsilon}{2}\boldsymbol{E}^2 + \frac{A}{2}\phi^2 + \frac{3\,B}{4}\phi^4 - \frac{\kappa}{2}(\boldsymbol{\nabla}\phi)^2\right. \\
&\qquad \left. -\kappa\phi(\boldsymbol{\nabla}^2\phi) + \frac{\phi}{2}\sum_{\alpha=\pm}\rho_\alpha\Delta\mu_\alpha + \phi\,\frac{\bar{\varepsilon}\gamma}{2}\boldsymbol{E}^2\right).
\end{aligned} \tag{145}$$

In order to satisfy the condition of mechanical equilibrium $\nabla_j \sigma_{ij} = 0$ a symmetric contribution has to be added to the above stress tensor. The following lines try to elucidate this.

Taking the divergence of the stress tensor gives

$$\begin{aligned}
\nabla_j \sigma_{ij} &= (\nabla_j \varepsilon E_j) E_i + \varepsilon E_j \nabla_j E_i + \nabla_i \left(\tilde{F} - \phi \frac{\delta \tilde{\mathcal{F}}}{\delta \phi} - \sum_{\alpha=\pm} \rho_\alpha \frac{\delta \tilde{\mathcal{F}}}{\delta \rho_\alpha} \right) && (146) \\
&= (\boldsymbol{\nabla} \cdot \boldsymbol{D}) E_i + D_j \nabla_j E_i + \nabla_i F_0 - D_j \nabla_i E_j \\
&\quad -\nabla_i \left(\phi \frac{\delta \tilde{\mathcal{F}}}{\delta \phi} + \sum_{\alpha=\pm} \rho_\alpha \frac{\delta \tilde{\mathcal{F}}}{\delta \rho_\alpha} \right) && (147) \\
&= (\boldsymbol{\nabla} \cdot \boldsymbol{D}) E_i + D_j \left(\nabla_j E_i - \nabla_i E_j \right) + \nabla_i F_0 \\
&\quad -\nabla_i \left(\phi \mu_\phi + \sum_{\alpha=\pm} \rho_\alpha \mu_{\rho_\alpha} \right) && (148)
\end{aligned}$$

According to Gauss' law the first term is

$$(\boldsymbol{\nabla} \cdot \boldsymbol{D}) E_i = \rho_f E_i = \sum_{\alpha=\pm} z_\alpha e \rho_\alpha E_i, \tag{149}$$

whereas ρ_f is the free charge density. Moreover, due to Faraday's law the curl in the second bracket vanishes. This leads us to the following intermediate result:

$$\begin{aligned}
\nabla_j \sigma_{ij} &= \sum_{\alpha=\pm} z_\alpha e \rho_\alpha E_i + \nabla_i F_0 - \nabla_i \left(\phi \mu_\phi + \sum_{\alpha=\pm} \rho_\alpha \mu_{\rho_\alpha} \right) && (150) \\
&= \sum_{\alpha=\pm} z_\alpha e \rho_\alpha E_i + \frac{\partial F_0}{\partial \phi} \nabla_i \phi + \frac{\partial F_0}{\partial \nabla_j \phi} \nabla_j \nabla_i \phi + \sum_{\alpha=\pm} \left\{ \frac{\partial F_0}{\partial \rho_\alpha} \right\} \nabla_i \rho_\alpha \\
&\quad -\nabla_i \left(\phi \mu_\phi + \sum_{\alpha=\pm} \rho_\alpha \mu_{\rho_\alpha} \right) && (151) \\
&= \sum_{\alpha=\pm} z_\alpha e \rho_\alpha E_i + \left\{ \frac{\partial F_0}{\partial \phi} - \nabla_j \left(\frac{\partial F_0}{\partial \nabla_j \phi} \right) \right\} \nabla_i \phi + \nabla_j \left(\frac{\partial F_0}{\partial \nabla_j \phi} \nabla_i \phi \right) \\
&\quad + \sum_{\alpha=\pm} \left\{ \frac{\partial F_0}{\partial \rho_\alpha} \right\} \nabla_i \rho_\alpha - \nabla_i \left(\phi \mu_\phi + \sum_{\alpha=\pm} \rho_\alpha \mu_{\rho_\alpha} \right). && (152)
\end{aligned}$$

If we replace the terms in curly brackets the chemical potentials μ_ϕ and μ_{ρ_α} and use the fact that gradients of the chemical potential vanish in equilibrium, we end up with

$$\nabla_j \sigma_{ij} = \sum_{\alpha=\pm} z_\alpha e \rho_\alpha E_i + \nabla_j \left(\frac{\partial F_0}{\partial \nabla_j \phi} \nabla_i \phi \right) \tag{153}$$

The first term is the external force on the charges, which is zero if we have no counterions and an equal amount of positve and negative ions. Therefore, we have to add a term

$$-\left(\frac{\partial F_0}{\partial \nabla_j \phi} \nabla_i \phi \right) \tag{154}$$

to the stress tensor to fulfill the equilibrium condition. This is equivalent to similar derivations in [**?**].

The total stress tensor reads

$$\begin{aligned}\sigma_{ij} &= \varepsilon E_i E_j - \kappa(\nabla_i\phi)(\nabla_j\phi) - \delta_{ij}\left(\frac{\varepsilon}{2}\boldsymbol{E}^2 + \frac{A}{2}\phi^2 + \frac{3\,B}{4}\phi^4\right. \\ &\left. -\frac{\kappa}{2}(\boldsymbol{\nabla}\phi)^2 - \kappa\phi(\boldsymbol{\nabla}^2\phi) + \frac{\phi}{2}\sum_{\alpha=\pm}\rho_\alpha\Delta\mu_\alpha + \phi\,\frac{\bar{\varepsilon}\gamma}{2}\boldsymbol{E}^2\right). \end{aligned} \tag{155}$$

10.2 Examples

A number of regression tests verify the implemention of the Nernst-Planck equation in combination with the SOR solver. They can be found in

`trunk/tests/regression`

The original tests we carried out during the first implementation comprise the Gouy-Chapman theory for electric double layers in front of a charged wall [**?**], the liquid junction potential emerging between two electrolytes of slighty different concentration and diffusivity of the charged species [**?**], electro-osmotic flow in a slit pore [**?**, **?**] and Debye-Hückel theory for charged colloidal particles and small enough potentials [**?**]. They are described in the following paragraphs.

10.2.1 Gouy-Chapman

This validation test is a flat surface carrying a surface charge σ with counterions and symmetic electrolyte. This is a one-dimensional, electroneutral problem of a diffusive electric double layer which has an analytical solution [**?**]. The approximation for low electrostatic potentials reads

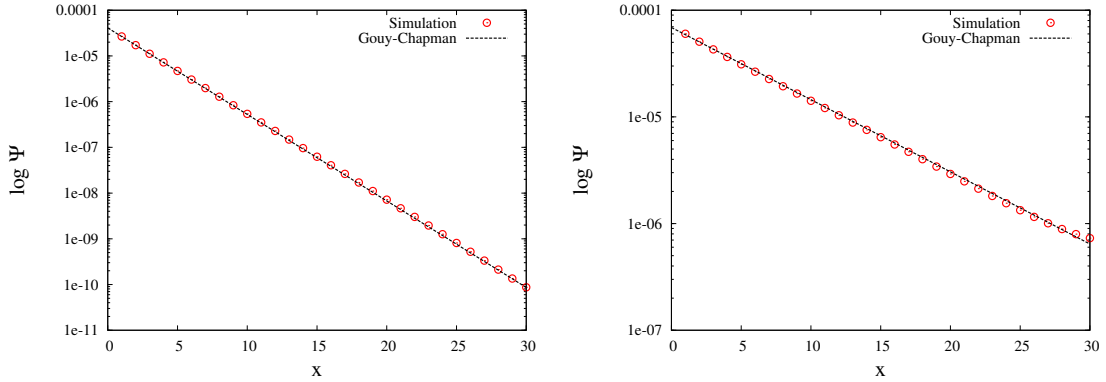


Figure 3: Gouy-Chapman theory for electric double layers: The plots compare the simulation results for the electric potential with the analytical solution for $\rho_{0,\pm} = 1 \cdot 10^{-2}$ (left) and $\rho_{0,\pm} = 1 \cdot 10^{-3}$ (right).

$$\Psi(x) = \Psi_D \exp(-\kappa\, x) \tag{156}$$

with κ as inverse Debye length

$$\kappa = l_D^{-1} = \sqrt{8\pi\, l_B\, I}. \tag{157}$$

The Bjerrum length l_B is given by

$$l_B = \frac{\beta\, e^2}{4\pi\, \varepsilon} \tag{158}$$

with $\beta^{-1} = k_B T$, e as unit charge and $\varepsilon = \varepsilon_0 \varepsilon_r$ as dielectric permittivity. The parameter

$$I = \frac{1}{2} \sum_k z_k^2\, \rho_{B,k} \tag{159}$$

is the ionic strength of the electrolyte with z_k as valencies of species k ($z_\pm = \pm 1$ for simple symmetric electrolyte) and $\rho_{B,k}$ as bulk charge density of species k far away from the wall. The Stern potential Ψ_D at the surface of the wall is related to the surface charge σ via

$$\Psi_D = \frac{2}{\beta\, e} \sinh^{-1}(-\sigma\, p) \tag{160}$$

$$\Psi_D \simeq \frac{2}{\beta\, e} \ln\left(-\sigma\, p + \sqrt{(\sigma\, p)^2 + 1}\right) \tag{161}$$

$$p = \frac{1}{\sqrt{8\, \varepsilon\, \beta^{-1}\, \rho_B}}. \tag{162}$$

The quantity $\rho_B = \rho_{B,+} = \rho_{B,-}$ is the average bulk charge density of the electrolyte.

We solved the Gouy-Chapman problem for a system consisting of $L_x \times L_y \times L_z = 64 \times 4 \times 4$ lattice sites. Periodic boundary conditions were used at all sides and a no-flux boundary conditions was set at $L_x = 1$ and $L_x = 64$.

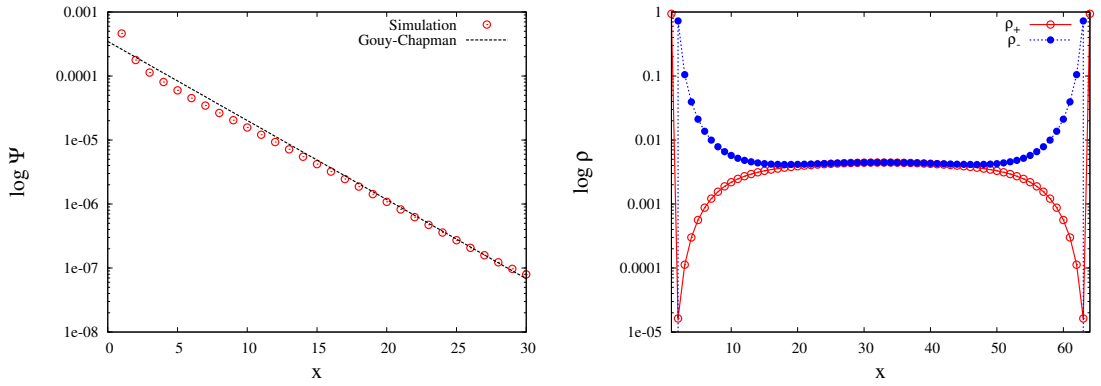


Figure 4: Nonlinear regime: For larger surface charges the solution deviates from the low-potential approximation Eq. **??** to the Gouy-Chapman theory. The righthand picture shows the charge distribution across the gap.

The parameters were chosen as follows (all in simulation units): unit charge $e = 1$, temperature $k_B T = \beta^{-1} = 3.333 \cdot 10^{-5}$, valency $z_\pm = \pm 1$, dielectric permittivity $\varepsilon = 3.3 \cdot 10^3$, diffusivities of the species $D_\pm = 1 \cdot 10^{-2}$ and initial densities $\rho_{0,\pm} = 1 \cdot 10^{-2}$. The entire system was electro-neutral and had a Bjerrum length $l_B = 0.723$.

At first we tested the linear case applying a small positive surface charge $\sigma = 3.125 \cdot 10^{-2}$, which led to bulk charge density $\rho_{B,+} = \rho_{B,-} = 1.044 \cdot 10^{-2}$, Debye length $l_D = 2.295$ and surface potential $\Psi_D = 2.136 \cdot 10^{-5}$. The potential was initialised with zero and had a value $\Psi_c = -2.364 \cdot 10^{-6}$ in the centre of the system after equilibration which we subtracted in the following analysis.

We reduced the initial density of the electrolyte to $\rho_{0,\pm} = 1 \cdot 10^{-3}$, which resulted in bulk charge densities $\rho_{B,+} = 1.298 \cdot 10^{-3}$, $\rho_{B,-} = 1.370 \cdot 10^{-3}$, Debye length $l_D = 6.420$, surface

potential $\Psi_D = 5.451 \cdot 10^{-5}$ and centre potential $\Psi_c = -1.256 \cdot 10^{-5}$. The comparison with the approximate solution Eq. **??** is shown in Fig. **??**.

For larger surface charges the low-potential assumption which led to Eq. **??** is no longer valid and the nonlinear nature of the Poisson-Boltzmann equation becomes evident. Fig. **??** shows the results for a surface charge $\sigma = 9.375 \cdot 10^{-1}$ and electrolyte density $\rho_{0,\pm} = 3 \cdot 10^{-3}$. We obtained for bulk charge densities $\rho_{B,+} = 4.443 \cdot 10^{-3}$ and $\rho_{B,-} = 4.461 \cdot 10^{-3}$, Debye length $l_D = 3.514$, the surface potential $\Psi_D = 2.267 \cdot 10^{-4}$ and the centre potential $\Psi_c = -3.395 \cdot 10^{-5}$.

10.2.2 Liquid-junction potential

The liquid junction potential is a charge separation process that occurs when electrolytes with slightly different concentrations whose species have different diffusivities are brought into contact. Charges from the regions of higher concentration diffuse into the parts with lower concentration. Due to the difference in diffusivity they migrate at different speeds, leaving parts of the system charged. This leads to a build-up of a potential which balances the diffusive flux.

After the initial build-up phase the potential decreases slowly again until the charge concentration has become homogeneous throughout the system. Both timescales of emergence and decay of the potential can be separated by chosing a sufficiently large system size.

This problem allowed us to verify the correct temporal behaviour of the Nernst-Planck equation solver by resolving the transient dynamics without having to account for advective terms.

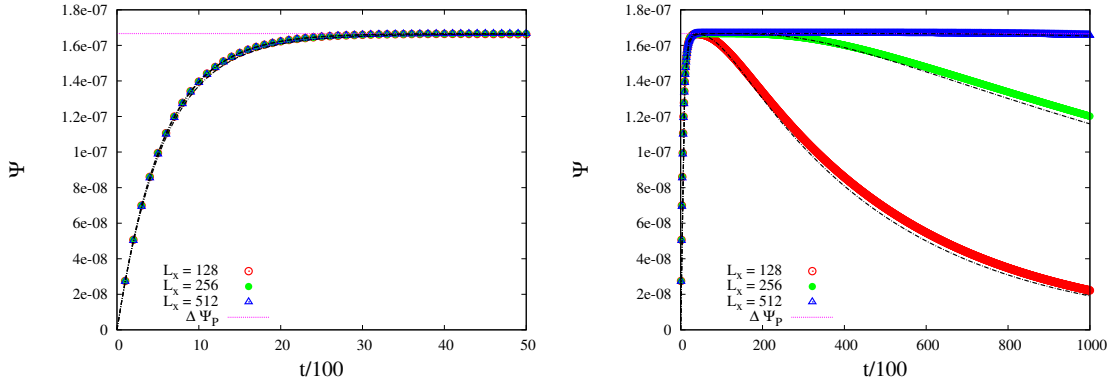


Figure 5: Time evolution of the liquid junction potential for $L_x = 128$ (blue), $L_x = 256$ (green) and $L_x = 512$ (blue). The dashed black curves represent the approximate solution in the limit $l_D/L_x << 1$. Deviations can be seen which are presumably due to the approximate nature of the analytical solution and the fact that we have a different asymptotic flux close to the boundary - the theoretical analysis assumes no flux at a boundary which is infinitely far away from the diffusive region.

For simplicity we considered systems of size $L_x \times L_y \times Lz = 128 \times 4 \times 4$ and $L_x \times L_y \times Lz = 256 \times 4 \times 4$ with periodic boundary conditions at either end. The two halfs were electroneutral and had ionic concentrations $\rho_{L,\pm} = \rho_{0,\pm} + \delta\rho$ and $\rho_{R,\pm} = \rho_{0,\pm} - \delta\rho$ with $\rho_{0,\pm} = 1 \cdot 10^{-2}$ and $\delta\rho = 0.01$.

The potential difference between both sides during the build-up obeys approximately

$$\Delta\Psi(t) \simeq \Delta\Psi_P \left\{ 1 - \exp\left(-\frac{t}{\tau_e} \right) \right\} \tag{163}$$

with

$$\Delta\Psi_P = \frac{(D_+ D_-)}{\beta e (D_+ + D_-)} \frac{\delta\rho}{\rho_0} \tag{164}$$

as saturation value of the potential difference. The saturation time scale is given by

$$\tau_e = \frac{\varepsilon}{\beta\, e^2 (D_+ + D_-)\rho_0}. \tag{165}$$

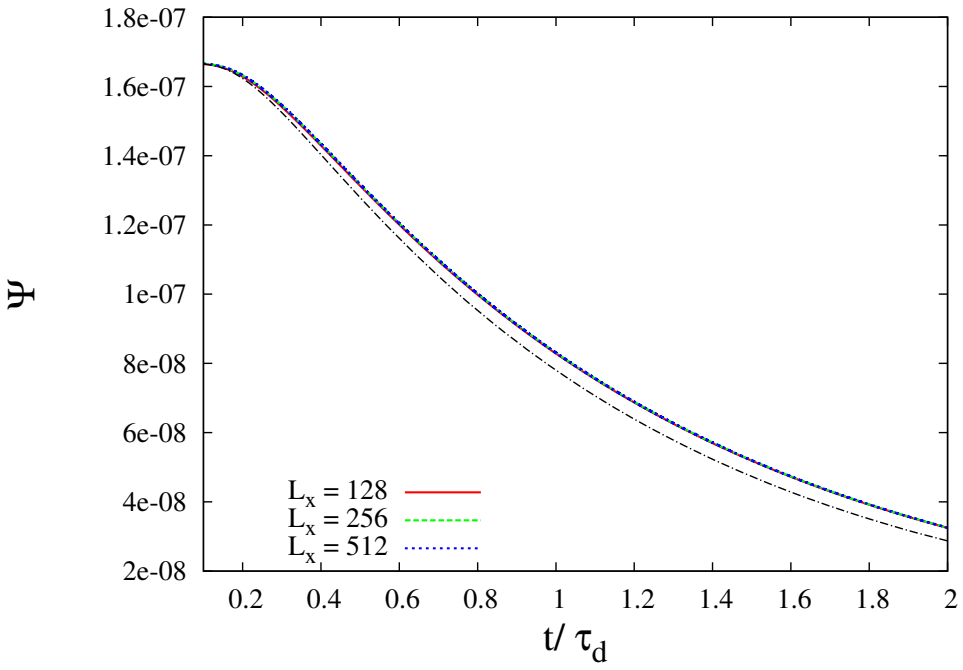


Figure 6: Rescaled plot of the decay: Times have been Time evolution of the liquid junction potential for $L_x = 128$ (blue), $L_x = 256$ (green) and $L_x = 512$ (blue). The dashed black curves represent the approximate solution in the limit $l_D/L_x << 1$. Deviations can be seen which are due to the approximate nature of the analytical solution and the diffusive fluxes close to the boundary.

A more exact solution can be derived in the limit $l_D/L_x << 1, N \to \infty$:

$$\Delta\Psi(t) = \Delta\Psi_P \left\{1 - \exp\left(-\frac{t}{\tau_e}\right)\right\} \frac{4}{\pi} \left\{\sum_{m=1}^{N} \frac{\sin^3(m\pi/2)}{m} \exp\left(-\frac{m^2\, t}{\tau_d}\right)\right\} \tag{166}$$

$$\tau_d = \frac{L^2}{2\pi^2 (D_+ + D_-)}. \tag{167}$$

It contains as well the dependence on the decay timescale τ_d. Only odd indices m contribute to the sum:

$$\sum_{m=1}^{N} \frac{\sin^3(m\pi/2)}{m} \exp\left(-\frac{m^2\, t}{\tau_d}\right) = \exp\left(-\frac{t}{\tau_d}\right) - \frac{1}{3}\exp\left(-\frac{9t}{\tau_d}\right) + \frac{1}{5}\exp\left(-\frac{25t}{\tau_d}\right) - \frac{1}{9}\exp\left(-\frac{81t}{\tau_d}\right) + . \tag{168}$$

A complete discussion of the solution can be found in [**?**]. There, the upper limit of significant modes has been also estimated as $N_{max} = L/\pi l_D$. Note the factor 2 difference between Eq. **??** and the corresponding expression in [**?**].

The following parameters were used: dielectric permittivity $\epsilon = 3.3 \cdot 10^3$, temperature $\beta^{-1} = 3.333 \cdot 10^{-5}$, unit charge $e = 1$, valency $z_\pm = \pm 1$, diffusivities $D_+ = 0.0125$ and $D_- = 0.0075$. We obtained $\Delta\Psi_P = 1.6667 \cdot 10^{-7}$, $\tau_e = 550$, $\tau_d = 41501.2\,(L_x = 128)$, $\tau_d = 166004.6\,(L_x = 256)$ and $\tau_d = 664018.5 (L_x = 512)$. The results for the potential difference over time are shown in Fig. **??**.

Fig. **??** shows results with times rescaled to the decay time scale τ_d (cf. Eq. **??**). Obviously the deviations we observe are not due to the limited system size and have a more systematic origin.

The curves coincide if the theoretic limit for τ_d is rescaled by a factor 1.067, suggesting the effective system length for this sort of setup is actually about 3% larger than the numerical value.

A reason for this might be the approximate nature of the analytical solution and the fact that it was gained for an infinitely large system with constant charge concentrations, vanishing currents at both ends and finite diffusive zone of size L_x. In our situation the entire system is within the diffusive zone. This may lead to smaller effective diffusivities or larger effective system sizes. Interestingly, all runs with solid walls at both ends resulted in oscillatory behaviour and an effective system size of $2L_x$.

10.2.3 Electroosmotic Flow

To test the implementation with all couplings to external and internal forces we consider a forced charged fluid in a slit of size L_z. An electrostatic field $E_{||}$ is allied parallel to the walls. The entire system is electroneutral with each wall having the surface charge density σ and compensationg counterions with total charge $2\sigma A_{wall}$ in the fluid.

In equilibrium the charge density at a distance x from the wall obeys

$$\rho(x) = \frac{\rho_0}{\cos^2(K\,x)} \tag{169}$$

with

$$\rho_0 = K^2/2\pi l_B \tag{170}$$

and

$$K\,L_x \tan\left(\frac{K\,L_x}{2}\right) = \pi\,l_B\,L_x\,2\sigma \tag{171}$$

$$K\,L_x \simeq \sqrt{4\pi\,l_B\,L_x\,2\sigma}. \tag{172}$$

The liniarised version Eq. **??** has only a limited range of applicabilty. We solved Eq. **??** numerically and found solutions $K = 0.01959$ ($\sigma = 0.003125$) and $K = 0.03311$ ($\sigma = 0.00125$), which is reasonably far away from the theoretical limit $K_{max} = \pi/L_x$ set by the tangent.

Note the factor 2 difference on the lhs of Eq. **??** with respect to [**?**, **?**]. There is also a factor L_x missing on the rhs of Eq. **??**.

The steady state velocity of the fluid can be derived from the force balance of the gradient of the stresses and the electrostatic forces:

$$v_y(x) = \hat{v} \ln\left(\frac{\cos(K\,x)}{\cos(K\,L_x/2)}\right) \tag{173}$$

$$\hat{v} = \frac{e\,E_{||}\rho_o}{\eta\,K^2} = \frac{e\,E_{||}}{2\pi\eta l_B} \tag{174}$$

The result for two different charge densities is shown in Fig. **??**. The accuracy is acceptable with deviations for high surface charged potentially being caused by the chosen discretisation or by the numerical solution of Eq. **??** approaching the limit of $\pi/L_z \simeq 0.049$.

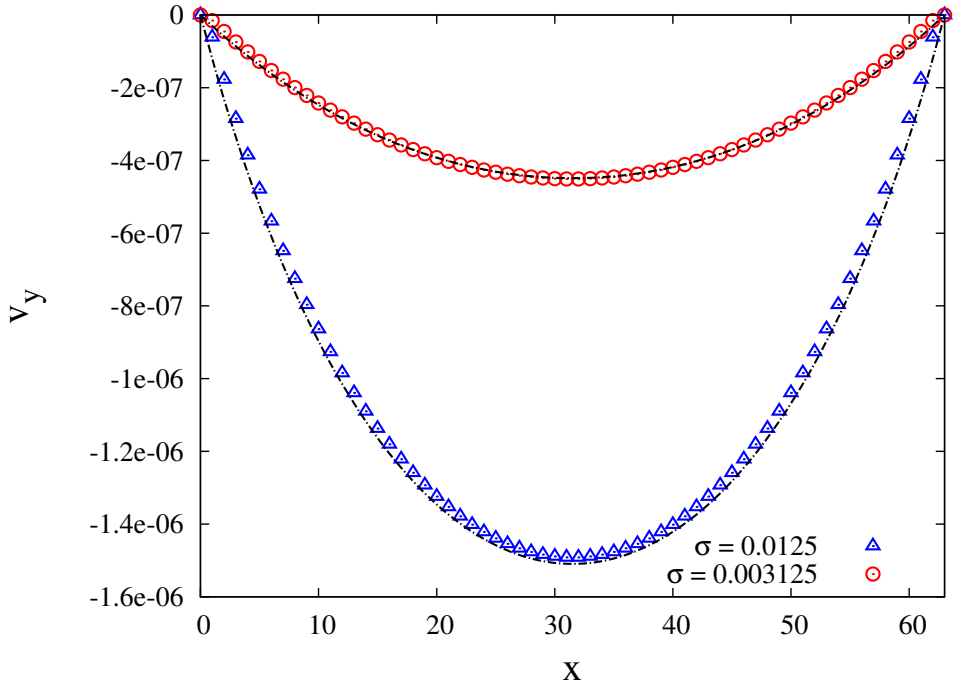


Figure 7: Steady state flow profile across a slit of width $L_x = 63$ for an applied field of magnitude $e\beta E L_x = 1.89$ for two different charge densities $\sigma = 0.003125$ (red) and $\sigma = 0.0125$ (blue), Bjerrum length $l_B = 0.7234$, viscosity $\eta = 0.1$ and unit charge $e = 1$. The dashed black lines correspond to theoretical prediction according to Eqs. **??** and **??**.

10.2.4 Debye-Hückel Theory

We have tested the implementation for a single fixed colloid and compared the result with Debye-Hückel theory. A result is shown in Fig. **??**. We used the following parametrisation:

$L_x \times L_y \times L_z = 64 \times 64 \times 64$, $D_+ = D_- = 0.01$, $e = 1$, $z_\pm = 1$, $\beta^{-1} = 3.333 \cdot 10^{-5}$, $\varepsilon = 3.3 \cdot 10^3$, $l_B = 0.723$.

For a central and fixed colloid of radius $R_c = 7.5$ carrying a positive unit charge $q_{c,+} = 1.0, q_{c,-} = 0$ we obtained $\rho_{c,+} = 5.58 \cdot 10^{-4}$, $\rho_{el} = \rho_{B,\pm} = 1 \cdot 10^{-2}$, $\Psi_D = 8.836 \cdot 10^{-7}$ for $2\,R_c = 16$.

For a higher larger positive charge $q_{c,+} = 4.0$ we got $\rho_{c,+} = 2.23 \cdot 10^{-3}$ $\rho_{el} = \rho_{B,\pm} = 5 \cdot 10^{-3}$ $\Psi_D = 4.993 \cdot 10^{-6}$ for $2\,R_c = 16$.

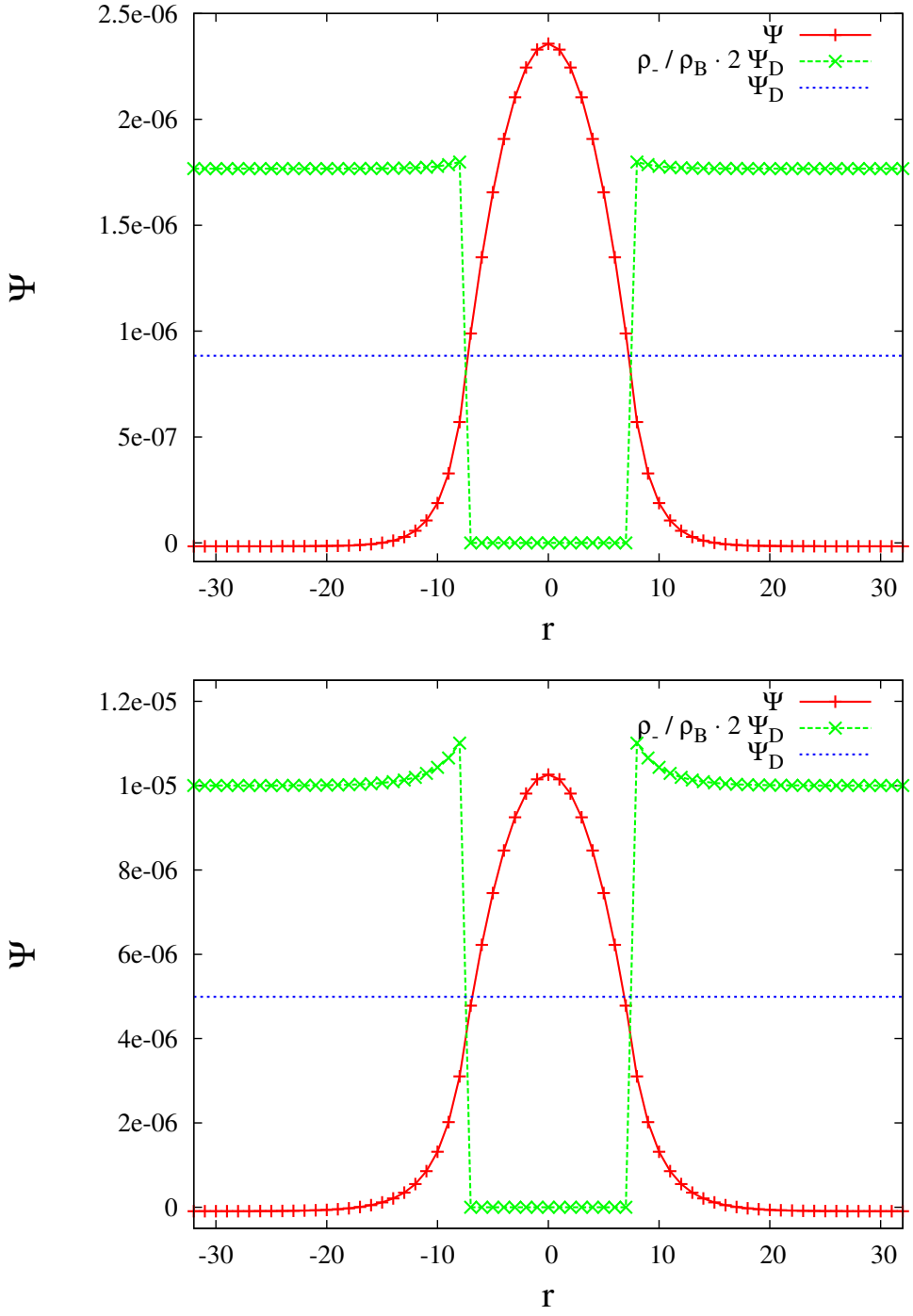


Figure 8: Debye-Hückel theory for positively charged colloid: the picture shows a cut through the center of a single colloid in a peridoic system. The potential is shown in red, the Stern potential in blue, and the negative charge density in green. The colloid is in the center.

11 The Input File

11.1 General

By default, the run time expects to find user input in a file `input` in the current working directory. If a different file name is required, its name should be provided as the sole command line argument, e.g.,

```
./Ludwig.exe input_file_name
```

If the input file is not located in the current working directory the code will terminate immediately with an error message.

When an input file is located, its content is read by a single MPI task, and its contents then broadcast to all MPI relevant tasks. The format of the file is plain ASCII text, and its contents are parsed on a line by line basis. Lines may contain the following:

- comments introduced by `#`.
- key value pairs separated by white space.

Blank lines are treated as comments. The behaviour of the code is determined by a set of key value pairs. Any given key may appear only once in the input file; unused key value pairs are not reported. If the key value pairs are not correctly formed, the code will terminate with an error message and indicate the offending input line.

Key value pairs may be present in the input file, but have no effect for any given run: they are merely ignored. Relevant control parameters for given input are reported in the standard output.

11.1.1 Key value pairs

Key value pairs are made up of a key — an alphanumeric string with no white space — and corresponding value following white space. Values may take on the follow forms:

```
key_string         value_string

key_integer_scalar 1
key_integer_vector 1_2_3

key_double_scalar  1.0
key_double_vector  1.0_2.0_3.0
```

Values which are strings should contain no white space. Scalar parameters may be integer values, or floating point values with a decimal point (scientific notation is also allowed). Vector parameters are introduced by a set of three values (to be interpreted as x, y, z components of the vector in Cartesian coordinates) separated by an underscore. The identity of the key will specify what type of value is expected. Keys and (string) values are case sensitive.

Most keys have an associated default value which will be used if the key is not present. Some keys must be specified: an error will occur if they are missing. The remainder of this part of the guide details the various choices for key value pairs, along with any default values, and any relevant constraints.

11.2 The Free Energy

The choice of free energy is determined as follows:

```
free_energy  none
```

The default value is `none`, i.e., a simple Newtonian fluid is used. Possible values of the `free_energy` key are:

```
#  none                   Newtonian fluid [DEFAULT]
#  symmetric              Symmetric binary fluid (finite difference)
#  symmetric_lb           Symmetric binary fluid (two distributions)
#  brazovskii             Brazovskii smectics
#  surfactant             Surfactants
#  polar_active           Polar active gels
#  lc_blue_phase          Liquid crystal (nematics, cholesterics, BPs)
#  lc_droplet             Liquid crystal emulsions
#  fe_electro             Single fluid electrokinetics
#  fe_electro_symmetric   Binary fluid electrokinetics
```

The choice of free energy will control automatically a number of factors related to choice of order parameter, the degree of parallel communication required, and so on. Each free energy has a number of associated parameters discussed in the following sections.

Details of general (Newtonian) fluid parameters, such as viscosity, are discussed in Section **??**.

11.2.1 Symmetric Binary Fluids

We recall that the free energy density is, as a function of compositional order ϕ:

$$\tfrac{1}{2}A\phi^2 + \tfrac{1}{4}B\phi^4 + \tfrac{1}{2}\kappa(\nabla\phi)^2.$$

Parameters are introduced by (with default values):

```
free_energy symmetric
A           -0.0625                    # Default: -0.003125
B           +0.0625                    # Default: +0.003125
K           +0.04                      # Default: +0.002
```

Common usage has $A < 0$ and $B = -A$ so that $\phi^\star = \pm 1$. The parameter κ (key `K`) controls the interfacial energy penalty and is usually positive.

11.2.2 Brazovskii smectics

The free energy density is:

$$\tfrac{1}{2}A\phi^2 + \tfrac{1}{4}B\phi^4 + \tfrac{1}{2}\kappa(\nabla\phi)^2 + \tfrac{1}{2}C(\nabla^2\phi)^2$$

Parameters are introduced via the keys:

```
free_energy brazovskii
A            -0.0005                   # Default: 0.0
B            +0.0005                   # Default: 0.0
K            -0.0006                   # Default: 0.0
C            +0.00076                  # Default: 0.0
```

For $A < 0$, phase separation occurs with a result depending on κ: one gets two symmetric phases for $\kappa > 0$ (cf. the symmetric case) or a lamellar phase for $\kappa < 0$. Typically, $B = -A$ and the parameter in the highest derivative $C > 0$.

11.2.3 Surfactants

The surfactant free energy should not be used at the present time.

11.2.4 Polar active gels

The free energy density is a function of vector order parameter P_α:

$$\tfrac{1}{2}AP_\alpha P_\alpha + \tfrac{1}{4}B(P_\alpha P_\alpha)^2 + \tfrac{1}{2}\kappa(\partial_\alpha P_\beta)^2$$

There are no default parameters:

```
free_energy      polar_active
polar_active_a  -0.1                    # Default: 0.0
polar_active_b  +0.1                    # Default: 0.0
polar_active_k   0.01                   # Default: 0.0
```

It is usual to choose $B > 0$, in which case $A > 0$ gives an isotropic phase, whereas $A < 0$ gives a polar nematic phase. The elastic constant κ (key `polar_active_k`) is positive.

11.2.5 Liquid crystal

The free energy density is a function of tensor order parameter $Q_{\alpha\beta}$:

$$\begin{aligned}\tfrac{1}{2}A_0(1-\gamma/3)Q_{\alpha\beta}^2 - \tfrac{1}{3}A_0\gamma Q_{\alpha\beta}Q_{\beta\delta}Q_{\delta\alpha} + \tfrac{1}{4}A_0\gamma(Q_{\alpha\beta}^2)^2 \\ +\tfrac{1}{2}\Big(\kappa_0(\epsilon_{\alpha\delta\sigma}\partial_\delta Q_{\sigma\beta} + 2q_0Q_{\alpha\beta})^2 + \kappa_1(\partial_\alpha Q_{\alpha\beta})^2\Big)\end{aligned}$$

The corresponding `free_energy` value, despite its name, is suitable for nematics and cholesterics, and not just blue phases:

```
free_energy     lc_blue_phase
lc_a0           0.01                    # Deafult: 0.0
lc_gamma        3.0                     # Default: 0.0
lc_q0           0.19635                 # Default: 0.0
lc_kappa0       0.00648456              # Default: 0.0
lc_kappa1       0.00648456              # Default: 0.0
```

The bulk free energy parameter A_0 is positive and controls the energy scale (key `lc_a0`); γ is positive and influences the position in the phase diagram relative to the isotropic/nematic transition (key `lc_gamma`). The two elastic constants must be equal, i.e., we enforce the single elastic constant approximation (both keys `lc_kappa0` and `lc_kappa1` must be specified).

Other important parameters in the liquid crystal picture are:

```
lc_xi           0.7                     # Default: 0.0
lc_Gamma        0.5                     # Default: 0.0
lc_active_zeta  0.0                     # Default: 0.0
```

The first is ξ (key `lc_xi`) is the effective molecular aspect ratio and should be in the range $0 < \xi < 1$. The rotational diffusion constant is Γ (key `lc_Gamma`; not to be confused with `lc_gamma`). The (optional) apolar activity parameter is ζ (key `lc_active_zeta`).

11.2.6 Liquid crystal anchoring

Different types of anchoring are available at solid surfaces, with one or two related free energy parameters depending on the type. The type of anchoring may be set independently for stationary boundaries (walls) and colloids.

```
lc_anchoring_strength   0.01            # free energy parameter w1
lc_anchoring_strength_2 0.0             # free energy parameter w2
lc_wall_anchoring       normal          # ``normal'' or ``planar''
lc_coll_anchoring       normal          # ``normal'' or ``planar''
```

See section **??** for details of surface anchoring.

11.2.7 Liquid crystal emulsion

This an interaction free energy which combines the symmetric and liquid crystal free energies. The liquid crystal free energy constant γ becomes a function of composition via $\gamma(\phi) = \gamma_0 + \delta(1 + \phi)$, and a coupling term is added to the free energy density:

$$WQ_{\alpha\beta}\partial_\alpha\phi\partial_\beta\phi.$$

Typically, we might choose γ_0 and δ so that $\gamma(-\phi^\star) < 2.7$ and the $-\phi^\star$ phase is isotropic, while $\gamma(+\phi^\star) > 2.7$ and the $+\phi^\star$ phase is ordered (nematic, cholesteric, or blue phase). Experience suggests that a suitable choice is $\gamma_0 = 2.5$ and $\delta = 0.25$.

For anchoring constant $W > 0$, the liquid crystal anchoring at the interface is planar, while for $W < 0$ the anchoring is normal. This is set via key `lc_droplet_W`.

Relevant keys (with default values) are:

```
free_energy          lc_droplet

A                    -0.0625
B                    +0.0625
K                    +0.053

lc_a0                 0.1
lc_q0                 0.19635
lc_kappa0             0.007
lc_kappa1             0.007

lc_droplet_gamma      2.586             # Default: 0.0
lc_droplet_delta      0.25              # Default: 0.0
lc_droplet_W         -0.05              # Default: 0.0
```

Note that key `lc_gamma` is not set in this case.

11.3 System Parameters

Basic parameters controlling the number of time steps and the system size are:

```
N_start     0                          # Default: 0
N_cycles    100                        # Default: 0
size        128_128_1                  # Default: 64_64_64
```

A typical simulation will start from time zero (key `N_start`) and run for a certain number of time steps (key `N_cycles`). The system size (key `size`) specifies the total number of lattice sites in each dimension. If a two-dimensional system is required, the extent in the z-direction must be set to unity, as in the above example.

If a restart from a previous run is required, the choice of parameters may be as follows:

```
N_start     100
N_cycles    400
```

This will restart from data previously saved at time step 100, and run a further 400 cycles, i.e., to time step 500.

11.3.1 Parallel decomposition

In parallel, the domain decomposition is closely related to the system size, and is specified as follows:

```
size        64_64_64
grid        4_2_1
```

The `grid` key specifies the number of MPI tasks required in each coordinate direction. In the above example, the decomposition is into 4 in the x-direction, into 2 in the y-direction, while the z-direction is not decomposed. In this example, the local domain size per MPI task would then be $16 \times 32 \times 64$. The total number of MPI tasks available must match the total implied by `grid` (8 in the example).

The `grid` specifications must exactly divide the system size; if no decomposition is possible, the code will terminate with an error message. If the requested decomposition is not valid, or `grid` is omitted, the code will try to supply a decomposition based on the number of MPI tasks available and `MPI_Dims_create()`; this may be implementation dependent.

11.4 Fluid Parameters

Control parameters for a Newtonian fluid include:

```
fluid_rho0              1.0
viscosity               0.166666666666666
viscosity_bulk          0.166666666666666
isothermal_fluctuations off
temperature             0.0
```

The mean fluid density is ρ_0 (key `fluid_rho0`) which defaults to unity in lattice units; it is not usually necessary to change this. The shear viscosity is `viscosity` and as default value 1/6 to correspond to unit relaxation time in the lattice Boltzmann picture. Reasonable values of the shear viscosity are $0.2 > \eta > 0.0001$ in lattice units. Higher values move further into the over-relaxation region, and can result in poor behaviour. Lower values increase the Reynolds number and tend to cause problems with stability. The bulk viscosity has a default value which is equal to whatever shear viscosity has been selected. Higher values of the bulk viscosity may be set independently and can help to suppress large deviations from incompressibility and maintain numerical stability in certain situations.

If fluctuating hydrodynamics is wanted, set the value of `isothermal_fluctuations` to `on`. The associated temperature is in lattice units: reasonable values (at $\rho_0 = 1$) are $0 < kT < 0.0001$. If the temperature is too high, local velocities will rapidly exceed the Mach number constraint and the simulation will be unstable.

11.5 Lees Edwards Planes

Constant uniform shear may be introduced via a number of Lees Edwards planes with given speed. This is appropriate for fluid-only systems with periodic boundaries.

```
N_LE_plane              2         # Number of planes (default: 0)
LE_plane_vel            0.05      # Constant plane speed
```

The placing of the planes in the system is as follows. The number of planes N must divide evenly the lattice size in the x-direction to give an integer δx. Planes are then placed at $\delta x/2, 3\delta x/2, \ldots$. All planes have the same, constant, velocity jump associated with them: this is positive in the positive x-direction. (This jump is usually referred to as the plane speed.) The uniform shear rate will be $\dot{\gamma} = NU_{LE}/L_x$ where U_{LE} is the plane speed (which is always in the y-direction).

The velocity gradient or shear direction is x, the flow direction is y and the vorticity direction is z.

The spacing between planes must not be less than twice the halo size in lattice units; 8–16 lattice units may be the practical limit in many cases. In additional, the speed of the planes must not cause a violation of the Mach number constraint in the fluid velocity on the lattice, which will match the plane speed in magnitude directly either side of the

planes. A value of around 0.05 should be regarded as a maximum safe limit for practical purposes.

Additional keys associated with the Lees Edwards planes are:

```
LE_init_profile        1          # Initialise u(x) (off/on)
LE_time_offset         10000      # Offset time (default 0)
```

If `LE_init_profile` is set, the fluid velocity is initialised to reflect a steady state shear flow appropriate for the number of planes at the given speed (ie., shear rate). If set to zero (the default), the fluid is initialised with zero velocity.

The code works out the current displacement of the planes by computing $U_{LE}t$, where t is the current time step. A shear run should then start from $t = 0$, i.e. zero plane displacement. It is often convenient to run an equilibration with no shear, and then to start an experiment after some number of steps. This key allows you to offset the start of the Lees-Edwards motion. It should then take the value of the start time (in time steps) corresponding to the restart at the end of the equilibration period.

There are a couple of additional constraints to use the Lees-Edwards planes in parallel. In particular, the planes cannot fall at a processor boundary in the x-direction. This means you should arrange an integer number of planes per process in the x-direction. (For example, use one plane per process; this will also ensure the number of planes still evenly divides the total system size.) This will interleave the planes with the processor decomposition. The y-direction and z-direction may be decomposed without further constraint.

Note that this means a simulation with one plane will only work if there is one process in the x decomposition.

11.6 Colloids

If no relevant key words are present, no colloids will be expected at run time. The simulation will progress in the usual fashion with fluid only.

If colloids are required, the `colloid_init` key word must be present to allow the code to determine where colloid information will come from. The options for the `colloid_init` key word are summarised below:

```
colloid_init           none

#  none                no colloids [DEFAULT]
#  input_one           one colloid from input file
#  input_two           two colloids from input file
#  input_three         three colloids from input file
#  input_random        Small number at random
#  from_file           Read a separate file (including all restarts)
```

For idealised simulations which require 1, 2, or 3 colloids, relevant initial state information for each particle can be included in the input file. In principle, most of the colloid state as defined in the colloid state structure in `colloid.h` may be specified. (Some state is reserved for internal use only and cannot be set from the input file.) Furthermore, not all the state is relevant in all simulations — quantities such as charge and wetting parameters may not be required, in which case they a simply ignored.

A minimal initialisation is shown in the following example:

```
colloid_init            input_one

colloid_one_a0          1.25
colloid_one_ah          1.25
colloid_one_r           12.0_12.0_12.0
```

This initialises a single colloid with an input radius $a_0 = 1.25$, and a hydrodynamic radius $a_h = 1.25$; in general both are required, but they do not have to be equal. A valid position is required within the extent of the system $0.5 < x, y, z < L + 0.5$ as specified by the `size` key word. State which is not explicitly defined is initialised to zero.

11.6.1 General Initialisation

A full list of colloid state-related key words is as follows. All the quantities are floating point numbers unless explicitly stated to be otherwise:

```
# colloid_one_nbonds    (integer) number of bonds
#   colloid_one_bond1   (integer) index of bond partner 1
#   colloid_one_bond2   (integer) index of bond partner 2
# colloid_one_isfixedr  colloid has fixed position (integer 0 or 1)
# colloid_one_isfixedv  colloid has fixed velocity (integer 0 or 1)
# colloid_one_isfixedw  colloid has fixed angular velocity (0 or 1)
# colloid_one_isfixeds  colloid has fixed spin (0 or 1)
# colloid_one_type      ``default'' COLLOID_TYPE_DEFAULT
#                       ``active'' COLLOID_TYPE_ACTIVE
#                       ``subgrid'' COLLOID_TYPE_SUBGRID
# colloid_one_a0        input radius
# colloid_one_ah        hydrodynamic radius
# colloid_one_r         position vector
# colloid_one_v         velocity (vector)
# colloid_one_w         angular velocity (vector)
# colloid_one_s         spin (unit vector)
# colloid_one_m         direction of motion (unit) vector for swimmers
```

Note that for magnetic particles, the appropriate initialisation involves the spin key word `colloid_one_s` which relates to the dipole moment $\mu \mathbf{s}_i$, while `colloid_one_m` relates to the direction of motion vector. Do not confuse the two. It is possible in principle to have magnetic active particles, in which case the dipole direction or spin ($\mathbf{s}_i$) and the direction of swimming motion $\mathbf{m}_i$ are allowed to be distinct.

```
# colloid_one_b1        Squirmer parameter B_1
# colloid_one_b2        Squirmer parameter B_2
# colloid_one_rng       (integer) random number generator state
# colloid_one_q0        charge (charge species 0)
# colloid_one_q1        charge (charge species 1)
# colloid_one_epsilon   Permeativity
# colloid_one_c         Wetting parameter C
# colloid_one_h         Wetting parameter H
```

Example: Single active (squirmer) particle

The following example shows a single active particle with initial swimming direction along the x-axis.

```
colloid_init            input_one

colloid_one_type        active
colloid_one_a0          7.25
colloid_one_ah          7.25
colloid_one_r           32.0_32.0_32.0
colloid_one_v           0.0_0.0_0.0
colloid_one_m           1.0_0.0_0.0
colloid_one_b1          0.05
colloid_one_b2          0.05
```

11.6.2 Random initialisation

For suspensions with more than few colloids, but still at relatively low volume fraction (10–20% by volume), it is possible to request initialisation at random positions.

The additional key word value pair `colloid_random_no` determines the total number of particles to be placed in the system. To prevent particles being initialised very close together, which can cause problems in the first few time steps if strong potential interactions are present, a "grace" distance or minimum surface-surface separation may also be specified (`colloid_random_dh`).

The following example asks for 100 colloids to be initialised at random positions, with a minimum separation of 0.5 lattice spacing.

```
colloid_init            input_random

colloid_random_no       100           # Total number of colloids
colloid_random_dh       0.5           # ''Grace'' distance

colloid_random_a0       2.30
colloid_random_ah       2.40
```

An input radius and hydrodynamic radius must be provided: these are the same for all colloids. If specific initialisations of the colloid state (excepting the position) other than the radii are wanted, values should be provided as for the single particle case in the preceding section, but using `colloid_random_a0` in place of `colloid_one_a0` and so on.

The code will try to initialise the requested number in the current system size, but only makes a finite number of attempts to place particles at random with no overlaps. (The initialisation will also take into account the presence of any solid walls, using the same grace distance.) If the the number of particles is too large, the code will halt with a message to that effect.

In general, colloid information for a arbitrary configuration with many particles should be read from a pre-prepared file. See the section on File I/O for further information on reading files.

11.6.3 Interactions

Note that two-body pair-potential interactions are defined uniformly for all colloids in a simulation. The same is true for lubrication corrections. There are a number of constraints related to the computation of interactions discussed below.

Boundary-colloid lubrication correction. Lubrication corrections (here the normal force) between a flat wall (see section XREF) are required to prevent overlap between colloid and the wall. The cutoff distance is set via the key word value pair

```
boundary_lubrication_rcnormal 0.5
```

It is recommended that this is used in all cases involving walls. A reasonable value is in the range $0.1 < r_c < 0.5$ in lattice units, and should be calibrated for particle hydrodynamic radius and fluid viscosity if exact results are important.

Colloid-colloid lubrication corrections. The key words to activate the calculation of lubrication corrections are:

```
lubrication_on                1
lubrication_normal_cutoff     0.5
lubrication_tangential_cutoff 0.05
```

Soft sphere potential. A cut-and-shifted soft-sphere potential of the form $v \sim \epsilon(\sigma/r)^\nu$ is available. Some trial-and-error with the parameters may be required in any given

situation to ensure simulation stability in the long run. The following gives an example of the relevant input key words:

```
soft_sphere_on          1               # integer 0/1 for off/on
soft_sphere_epsilon     0.0004          # energy units
soft_sphere_sigma       1.0             # a length
soft_sphere_nu          1.0             # exponent is positive
soft_sphere_cutoff      2.25            # a surface-surface separation
```

See Section **??** for a detailed description.

Lennard-Jones potential. The Lennard-Jones potential (Section **??**) is controlled by the following key words:

```
lennard_jones_on        1               # integer 0/1 off/on
lj_epsilon              0.1             # energy units
lj_sigma                2.6             # potential length scale
lj_cutoff               8.0             # a centre-centre separation
```

Yukawa potential. A cut-and-shifted Yukawa potential of the form $v \sim \epsilon \exp(-\kappa r)/r$ is available using the following key word value pairs:

```
yukawa_on               1               # integer 0/1 off/on
yukawa_epsilon          1.330           # energy units
yukawa_kappa            0.725           # an inverse length
yukawa_cutoff           16.0            # a centre-centre cutoff
```

Dipole-dipole interactions (Ewald sum). The Ewald sum is completely specified in the input file by the uniform dipole strength μ and the real-space cut off r_c.

```
ewald_sum               1               # integer 0/1 off/on
ewald_mu                0.285           # dipole strength mu
ewald_rc                16.0            # real space cut off
```

If short range interactions are required, particle information is stored in a cell list, which allows efficient computation of the potentially N^2 interactions present. This gives rise to a constraint that the width of the cells must be large enough that all relevant interactions are included. This generally means that the cells must be at least $2a_h + h_c$ where h_c is the largest relevant cut off distance.

The requirement for at least two cells per local domain in parallel means that there is a associated minimum local domain size. This is computed at run time on the basis of the input. If the local domain is too small, the code will terminate with an error message. The local domain size should be increased.

11.6.4 External forces

External body force (gravity). The following example requests a uniform body force in the negative z-direction on all particles.

```
colloid_gravity         0.0_0.0_-0.001  # vector
```

The counterbalancing body force on the fluid which enforces the constraint of momentum conservation for the system as a whole is computed automatically by the code at each time step.

11.7 Order parameter initialisations

Free energy choices requiring one or more fluid order parameters can make use of the following initial coniditions.

11.7.1 Composition ϕ

The following initialisations are available.

```
phi_initialisation       spinodal   # spinodal
phi0                     0.0        # mean
noise                    0.05       # noise amplitude
random_seed              102839     # +ve integer
```

Suitable for initialising isothermal spinodal decomposition, the order parameter may be set at random at each position via $\phi = \phi_0 + A(\hat{\phi} - 1/2)$ with the random variate $\hat{\phi}$ selected uniformly on the interval $[0, 1)$. For symmetric quenches (mean order parameter $\phi_0 = 0$ and $\phi^\star = \pm 1$), a value of A in the range 0.05-0.1 is usually appropriate.

For off-symmetric quenches, larger patches of fluid may be required to initiate decomposition:

```
phi_initialisation       patches    # patches of phi = +/- 1
phi_init_patch_size      2          # patch size
phi_init_patch_vol       0.1        # volume fraction phi = -1 phase
random_seed              13         # +ve integer
```

The initialises cubics patches of fluid of given size with $\phi = \pm 1$ at random. The requested overall volume fractions may be met approximately.

A spherical drop can be initialised at the centre of the system.

```
phi_initialisation       drop       # spherical droplet
phi_init_drop_radius     16.0       # radius
phi_init_drop_amplitude -1.0        # phi value inside
```

The drop is initialised with a $\tanh(r/\xi)$ profile where the interfacial width ξ is computed via the current free energy parameters.

For restarted simulations, the default position is to read order parameter information from file

```
phi_initialisation       from_file
```

in which case a file or files for the appropriate time step should be present in the working directory.

11.7.2 Tensor order $Q_{\alpha\beta}$

A number of different initialisations are available for the liquid crystal order parameter $Q_{\alpha\beta}$. Some care may be required to ensure consistency between the choice and the free energy parameters, the system size, and so on (particularly for the blue phases).

A summmary of choices is:

```
lc_q_initialisation nematic        # uniform nematic...
lc_init_nematic     1.0_0.0_0.0    # ...with given director

lc_q_initialisation cholesteric_x  # cholesteric with helical axis x
lc_q_initialisation cholesteric_y  # cholesteric with helical axis y
lc_q_initialisation cholesteric_z  # cholesteric with helical axis z

lc_q_initialisation o8m            # BPI high chirality limit
lc_q_initialisation o2             # BPII high chirality limit
lc_q_initialisation o5
lc_q_initialisation h2d            # 2d hexagonal
lc_q_initialisation h3da           # 3d hexagonal BP A
lc_q_initialisation h3db           # 3d hexagonal BP B
lc_q_initialisation dtc            # double twist cylinders
```

```
lc_q_initialisation bp3

lc_q_initialisation cf1_x          # cholesteric ‘‘finger’’ axis x
lc_q_initialisation cf1_y          # cholesteric ‘‘finger’’ axis y
lc_q_initialisation cf1_z          # cholesteric ‘‘finger’’ axis z

lc_q_initialisation cf1_fluc_x     # as cf1_x with random perterbations
lc_q_initialisation cf1_fluc_y     # as cf1_y with random perturbations
lc_q_initialisation cf1_flux_z     # as cf1_z with random perturbations

lc_q_initialistion  random         # with randomly chosen unit director
```

Note many of the initialiations require an initial amplitude of order, which should be set via

```
lc_q_init_amplitude 0.01           # initial amplitude of order A
```

For example, if an initial uniform nematic is requested with unit director n_α, the corresponding initial tensor will be

$$Q_{\alpha\beta} = \tfrac{1}{2} A(3 n_\alpha n_\beta - \delta_{\alpha\beta}).$$

12 Examples

12.1 Permeability calculations in simple porous media

Single fluid permeability calculations for a given porous structure can be undertaken by driving a flow via the fluid body force. Note that the structure must be periodic in the direction of the force (this may mean duplicating a 'reflected' version of a given sample to create the correct input). The body force can be specified so that, e.g., to drive a flow in the positive $x-$direction

```
force 0.001_0.000_0.000
```

The force should not be so large that the maximum velocity generated threatens the Mach number constraint. To get a measurement of the flow at equilibrium, the calculation should be run at least the momentum diffusion time for the system (L^2/η in LB time steps).

The net flow can be measured by combining the statistics for the total momemnum and the total density (which is equal to the volume with $\rho_0 = 1$). We will see an example of this later in the section.

Note that in the case of porous media with narrow channels at the grid scale, the wall velocity can be dependent on the viscosity of the fluid. This is an artefact of the bounce-back on links and will result in a viscosity-dependent permeability (see, e.g., [**?**]). To minimise this effect, e.g., Ginzburg and d'Humières [**?**] corrects the viscosity-dependence of the apparent boundary position using a three relaxation time scheme.

Before considering a concrete example, it's worth reviewing the basic calculation of the conductance in ideal geometries.

12.1.1 Circular and rectangular capillaries

For an infinite capillary of circular cross section radius a, the conductance is known analytically[**?**]. The flow per unit area J is given by

$$J = -\frac{1}{8\eta}\frac{\partial p}{\partial x}a^2. \tag{175}$$

If the pressure gradient is replaced by a uniform body force $-\partial p/\partial x = \rho g$, then one can define a viscosity-independent conductance C via $J = C\rho g/\eta$, i.e., $C = a^2/8$. So, by measuring the volume flux at steady state for a fixed applied body force, one can compute an estimate of the conductance to compare with this result.

An analytical expression is also available for the conductance of a square or rectangular capillary. For an infinite capillary of square cross section width $w \times h$ (where h is the longer), the volume flux per unit area J is expected to be [**?**, **?**]

$$J = -\frac{1}{3\eta}\frac{\partial p}{\partial x}(h/2)^2\left[1 - 6(h/w)\sum_{k=1}^{\infty}\frac{\tanh(\alpha_k w/h)}{\alpha_k^5}\right] \tag{176}$$

where $\alpha_k = (2k-1)\pi/2$. The pressure gradient $-\partial p/\partial x$ is replaced by the body force ρg and one can difine the viscosity-independent conductance C via $J = C\rho g/\eta$. The calculation proceeds as above.

12.1.2 Setting up a structure

We will work out the conductance of a simple one-dimensional capillary of circular cross section. We can set up the structure using the utility program found in `util/capillary.c`.

We set the system size to be $(10, 10, 32)$ in the $x-$, $y-$, and $z-$directions, respectively. The cross section in $x - y$ will be a discrete approximation to a circle. The nominal

radius is $a = 4$ lattice units (allowing for solid sites at each edge). In `capillary.c` we set

```
const int xmax = 10;
const int ymax = 10;
const int zmax = 32;
```

Note that the conductance should be independent of the length of the capillary in the $z-$direction. We will check this later.

We choose a circular cross section via:

```
enum {CIRCLE, SQUARE, XWALL, YWALL, ZWALL, XWALL_OBSTACLES};
const int xsection = CIRCLE;
```

For this problem, the wetting parameters are irrelevant, and can be ignored. In addition we set

```
enum {STATUS_ONLY, STATUS_WITH_C_H, STATUS_WITH_SIGMA};
const int output_type = STATUS_ONLY;
```

to specify that the output will contain only the structural information. The output filename containing the structure is set via

```
const char * filename = "capillary.001-001";
```

The program may be compiled and run via

```
$ make capillary.c -lm
$ ./capillary
```

The output provides a simple representation of the cross section, and a count of the number of solid sites, the number of fluid sites, and the total. Again, the wetting parameters are irrelevant. For this case we have

```
Cross section (0 = fluid, 1 = solid)
 1 1 1 1 1 1 1 1 1 1
 1 1 1 0 0 0 0 1 1 1
 1 1 0 0 0 0 0 0 1 1
 1 0 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 0 1
 1 0 0 0 0 0 0 0 0 1
 1 1 0 0 0 0 0 0 1 1
 1 1 1 0 0 0 0 1 1 1
 1 1 1 1 1 1 1 1 1 1
n = 3200 nsolid = 1536 nfluid = 1664
```

Note that the outermost sites in each direction here are solid. There should be a file `capillary.001-001` in the current directory. This should be moved to the executable directory.

12.1.3 Setting up the input

Assuming we have compiled the serial code, we must now set up the input file to be consistent with the capillary structure we want to use. The `src/input.ref` file can be used as a template. We should set the system size

```
size 10_10_32
```

The location of the porous media file is specified using

```
porous_media_format BINARY
porous_media_file capillary
porous_media_type status_only
```

Note that the extension `.001-001` of the filename has been removed in the input file (it gets added back automatically by the main code at run time).

As this is a single fluid calculation with no free energy involved we must set

```
free_energy none
```

What are the fluid parameters? We will set the viscosity to 1/6 in lattice units (and the bulk viscosity to the default value by commenting it out; the default value is the same as the shear viscosity):

```
viscosity 0.16666666666666666
#bulk_viscosity 0.1
```

We expect that the number of time steps to reach a steady state will be of order $t \approx a^2/\eta$, which we estimate using $a = 4$ so $t \approx 100$ LB time steps. We will try

```
N_cycles 200
```

Note that we will need output on the total momentum and the volume flux of the system as a function of time, so we set

```
freq_statistics 50
```

Finally, we need to set a force in the z-direction to drive the flow. Again, the final conductance should be independent of this force providing the force is small enough that both the Mach number and the Reynolds number are small compared to unity in steady stead. We will set

```
force 0.00_0.00_0.0000001
```

12.1.4 Extracting the conductance

With these parameters specified, we can run the code. The output should reflect the parameters in the input file, and the time step loop should start if the parameters, and the `capillary.001-001` file has been read successfully. The code reports various fluid properties. The density is `[rho]`, and we see that the total is the same as the number of fluid sites reported by the `capillary.c` program:

```
Scalars - total mean variance min max
[rho]   1664.00 1.00000000000-2.2204460e-16 1.00000000000 1.00000000000
```

Note that the total density (ie., mass) should remain exactly unchanged for the duration of the run; if not, there is something wrong.

The integrated volume flux of the fluid in each coordinate direction is reported, and we should see, as a function of time

```
Velocity - x y z
...
[vol flux] 3.1086245e-15 2.4868996e-14 1.9303426e-03
[vol flux] -6.8278716e-14 4.8849813e-15 2.0251488e-03
[vol flux] 8.4265928e-14 5.5178084e-14 2.0299067e-03
[vol flux] 1.6209256e-14 1.2212453e-14 2.0301454e-03
```

Note as we have applied an external force in the $z-$direction, the z-flux increases with time while the volume flux in the other two directions is constant (zero to machine accuracy, which is around 10^{-16}). Further, the z-volume flux is still changing slightly after 200 time steps, so we have not run long enough to reach a steady state. If we run for 400 steps, we should see that the z-flux is unchanged over the last 100 time steps.

```
Velocity - x y z
...
[vol flux] 6.6391337e-14 -2.2204460e-15 2.0301580e-03
[vol flux] 3.4638958e-14 3.6415315e-14 2.0301580e-03
```

Finally, note the maximum velocity in the flow direction is small compared with unity, and so both Mach number and Reynolds number are also small in this case.

The figure we are interested in is the volume flux in the z-direction which is 2.030×10^{-3}. As the mean density is unity, this is also the integrated volume flux J_v. The volume flux per unit area $J = J_v/a^2L_z$, where a^2L_z is the volume of the discrete system (1664 in lattice units). Following section **??**, we can write a viscosity independent conductance $C = J\eta/\rho g$, with g the force in the $z-$direction. Putting all this together, we have $C = (2.030 \times 10^{-3}/1664) \times (1/6)/(1.0 \times 1 \times 10^{-7}) = 2.033$. The theoretical figure is $C = 2$, so the simulation is correct to within a discretisation error of about 2%.

12.1.5 In parallel

We can run the same calculation using the parallel code (recompile with `make mpi`). In the input file we set

```
size 10_10_32
grid 1_1_2
```

to set the parallel decomposition explicitly to two MPI tasks in the z-direction. We also set

```
porous_media_format BINARY_SERIAL
porous_media_file capillary
porous_media_type status_only
```

to ensure the structure file is read correctly in parallel. We should run this on 2 MPI tasks, e.g.,

```
$ mpirun -np 2 ./Ludwig.exe input
```

The final result should be exactly the same as the serial version to machine accuracy. You may also be able to spot that the execution time is approximately half that for the serial version (although this problem is a little small for efficient parallelisation).

12.1.6 General porous media

For a general porous media, we can make use of Darcy's Law which defines a permeability k (with dimensions of area) via

$$J_v = -\frac{kA}{\eta}\frac{\partial P}{\partial z} \tag{177}$$

where J_v is the volume flux, A is the cross-sectional area of the sample, and $\partial P/\partial z$ is the pressure gradient as before.

12.1.7 Matching lattice and real units in porous media

Following Succi [**?**] (Chapter 8) the following argument can be made to match lattice quantities and real physical quantities for a given system of interest. Dimensionless lattice Boltzmann units set the lattice spacing $\Delta x = 1$, the lattice time step $\Delta t = 1$ and the mean density of the fluid to $\rho_0 = 1$. The speed of sound in lattice units is $c_s = 1/\sqrt{3}$. How do we match these units to physical ones?

Suppose we have discretised an X-ray tomography image with a resolution of 1 voxel equal to 1 μm. If we represent one voxel by one cubic lattice with width Δx, then we can match $\Delta x = 1\mu$m. A large data set might provide 1000 voxels on a side, giving 1000 lattice sites, which would represent 1 mm. Similarly, if our real system is water at room temperature with density $\rho = 1000$kg m^{-3}, then we may equate the lattice density $\rho_0 = 1$ to be equivalent to 1000 kg m^{-3}. (This means the mass of fluid at one lattice site of volume $\Delta x^3 = 1\mu$m^3 is then 10^{-15} kg, although this is not a very useful number.)

This leaves time. This is matched via the speed of sound. For the real system, we take the speed of sound in water at 20^oC to be 1480 m s^{-1}. So the speed of sound on the lattice $(1/\sqrt{3})\Delta x$ per Δt matches 1480 m s^{-1}, or $\Delta t = 1/\sqrt{3} \times 10^{-6}/1480 \approx 3.9 \times 10^{-10}$s. Note that this is a small unit, i.e., it would require very many time steps to reach a 'macroscopic' time in any simulation.

We should also consider the Reynolds number $Re = \rho UL/\eta$. Suppose our real structure has a characteristic pore size which is just at the resolution of the X-ray tomography image: $L = h = 1\mu$m. If the fluid (water) has viscosity $\eta = 10^{-3}$ Pa s, and a typical flow is $U \sim 10^{-3}$ m s^{-1}, then the pore scale Reynolds number is $Re_h \sim 10^{-3}$. In the LB, if we choose a lattice viscosity $\eta = 1/6$, and generate a typical flow in lattice units of 10^{-4}, then the Reynolds number in the simulation (with $\rho_0 = 1$ and $L = \Delta x = 1$) is $Re \sim 6 \times 10^{-4}$, which is similar.

There is one potential problem if we wish to study transport phenomena where we might want to run a simulation long enough for the flow to cross the whole sample. With $U \sim 10^{-4}$ in lattice units and a large sample size of, say, 1024^3 the flow would take around 10^7 simulation time steps to propagate information across the system. This is computationally expensive. One solution is to artificially raise the flow speed (and hence the Reynolds number). This is acceptable as long as the Reynolds number remains small compared with unity (all Reynolds numbers < 0.1 can be regarded as negligible [**?**].) Here, for example, if we raise the flow by two orders of magnitude to 10^{-2} in lattice units, the Reynolds number is still acceptable at $Re = 6 \times 10^{-2}$, and the number of simulation time steps is reduced to a more managable 10^5 (see also [**?**]).

13 Further Information for Developers

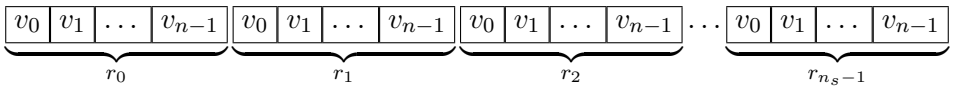
13.1 Address models for 3-dimensional fields

The code allows for different order of storage associated with 3-dimensional fields (scalars, vectors, and so on). These storage schemes are usually referred to as array of structures (AOS), structure of arrays (SOA), and for the blocked version array of structures of (short) arrays (AOSOA). For historical reasons these are also sometimes referred to as 'Model' and 'Model R' options, which correspond to array-of-structures and structure-of-arrays layout, respectively. Early versions of the code for CPU were written to favour summation of LB distributions on a per lattice site basis in operations such as $\rho(\mathbf{r}) = \sum_i f_i(\mathbf{r})$. This is array-of-structures, where the f_i are stored contiguously per site. Introduction of GPU versions, where memory coalescing favours the opposite memory order, were then referred to as 'Model R', the 'R' standing for 'reverse'.

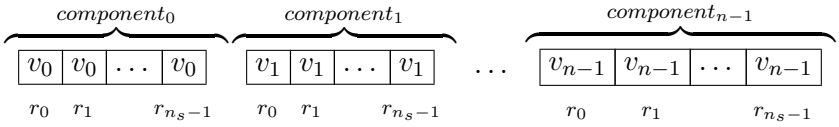
The memory layouts are discussed below. In all cases, a 3-d field occupies lattice sites with a unique spatial index determine by position, and computed via `coords_index()`. These position indices will be denoted $r_0, r_1, \ldots, r_{n_s-1}$ where n_s is the total number of lattice sites (including halo points).

13.1.1 Rank 1 objects (to include scalar fields)

ADDR_AOS: The array-of-structures order for an $n-$vector field with components $v_\alpha = (v_0, v_1, \ldots, v_{n-1})$ is, schematically:



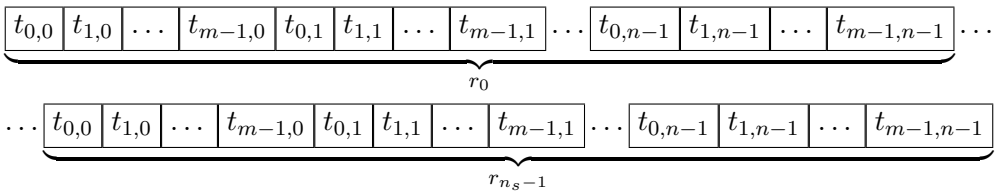
ADDR_SOA: The structure-of-arrays version is:



A scalar field has $n = 1$.

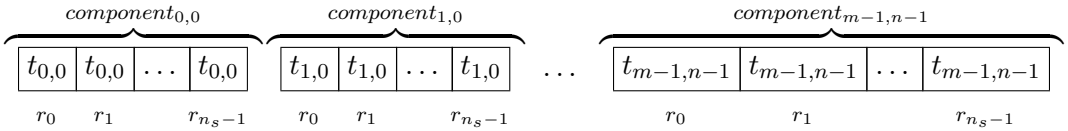
13.1.2 Rank 2 objects (to include dyadic tensor fields)

ADDR_AOS: A general rank 2 tensor $t_{\alpha\beta}$ with components $(t_{0,0}, \ldots, t_{m-1,n-1})$ is stored as:



Dyadic tensors, for example the gradient of a vector field $\partial_\alpha v_\beta$ in three dimensions, are stored in corresponding fashion with $m = 3$ and $\partial_\alpha = (\partial_x, \partial_y, \partial_z)$.

ADDR_SOA: The structure-of-arrays version is



13.1.3 Compressed rank 2 objects

A symmetric tensor $S_{\alpha\beta}$ in three dimensions has six independent components. It may be convenient to store this in compressed form as a rank 1 vector $(S_{xx}, S_{xy}, S_{xz}, S_{yy}, S_{yz}, S_{zz})$ to eliminate redundent storage.

A symmetric traceless rank 2 tensor — for example, the Landau-de Gennes liquid crystal order parameter $Q_{\alpha\beta}$ — has five independent components. This is stored as a rank 1 vector with five components $(Q_{xx}, Q_{xy}, Q_{xz}, Q_{yy}, Q_{yz})$ to eliminate redundant storage. API calls are provided to expand the compressed format to the full rank-2 representation $Q_{\alpha\beta}$ and, conversely, to compress the full representation to five components.

13.1.4 Rank 3 objects (to include triadic tensor fields)

The general rank 3 object $t_{\alpha\beta\gamma}$ with components $(t_{0,0,0}, \ldots, t_{m-1,n-1,p-1})$ is stored in a manner which generalises from the above, i.e., with the rightmost index running fastest. Diagrams are omitted, but AOS and SOA storage patterns follow the same form as seen above.

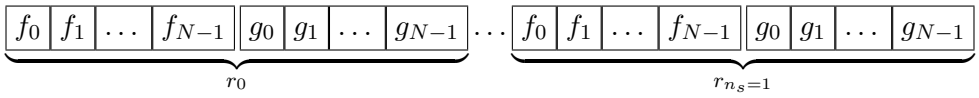
A triadic tensor, for example the general second derivative of a vector field $\partial_\alpha \partial_\beta v_\gamma$ may be stored as a rank 3 object.

13.1.5 Compressed rank 3 objects

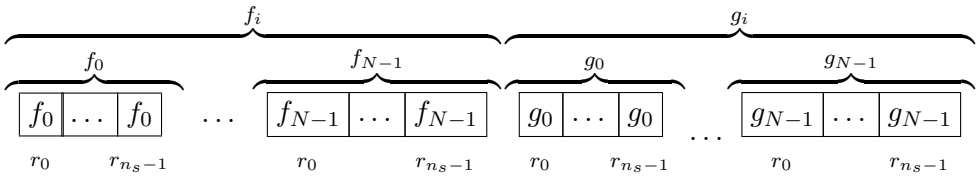
Symmetry properties may be used to reduce the storage requirement associated with rank 3 tensors. For example, the gradient of the liquid crystal order parameter $\partial_\gamma Q_{\alpha\beta}$ may be stored as a rank 2 object. The exact requirement will depend on the construction of the tensor.

13.1.6 LB distribution data

ADDR_AOS: For the LB distributions, up to two distinct distributions can be accommodated to allow for a binary fluid implementation, although the usual situation is to have only one. The AOS order for two N-velocity distributions f and g is:



ADDR_SOA: The strucuture-of-arrays order is:



For the single-distribution case, this is equivalent to the rank 1 vector field with $n = N$.

13.2 Generalised model for SIMD vectorisation

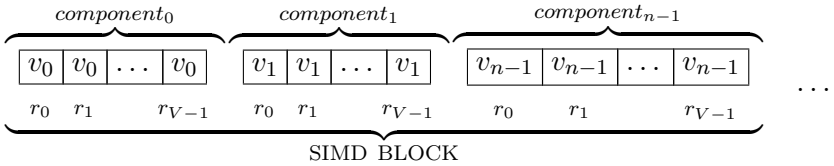
To promote parallelism at the instruction level, it is necessary to insure the innermost loop of any loop construct has an extent which is the vector length for the target architecture. The memory layout should therefore be adjusted accordingly. This means the MODEL format is generalised to a (rather clumsily named) array of structures of short vectors. The length of the short vectors is the SIMD vector length.

The outermost loop in this context is always to be the loop over lattice sites, which is adjusted according to the vector length; see Section **??** below for practical examples.

The SOA picture, which targets coalescence, can be viewed as naturally supporting SIMD vectorisation by virtue of allowing contiguous access to individual quantities as a function of lattice index. (If SIMD vectorisation is wanted at all on GPU architecture, it can be as a means to relieve register pressure.) SOA therefore remains unaltered and we concentrate on the AOS picture.

13.2.1 Rank 1 objects

ADDR_AOSOA: The structure of short vectors is based on the SIMD vector length which we here denote V:



Subsequent SIMD blocks involve lattice sites $r_V \ldots r_{2V-1}$, $r_{2V} \ldots r_{3V-1}$, and so on. If the SIMD vector length is unity, this collapses to the standard Model picture shown in Section **??**.

The generalisation of this approach to rank 2 and rank 3 objects follows the corresponding Model implementation.

13.3 Addressing: How to?

13.3.1 Compilation

A number of options exist to control at compilation time the choice of the address model. If no options are specified the default addressing model is AOS. Two additional C preprocessor directives exist:

`-DADDR_SOA` selects structure-of-arrays order (e.g., for coallescing on GPU platforms).

`-DADDR_AOSOA` block SOA pattern (usually combined with a non-unit vector length).

For development purposes, the address functions allow the array indices to be checked against the correct range. For production runs, the relevant functions are replaced by macros if `-DNDEBUG` is specified.

The targetDP vector length is specified via, e.g., `-DVVL=4`.

13.3.2 Kernels

To provide a transparent interface for addressing vector fields, a single API is provided which implements the addressing order selected at compile time. This interface is always based on a fixed order of indices which may include the vector index of the innermost SIMD loop if present. This allows both vectorised and non-vectorised loops to be constructed in a consistent manner.

The usual model for constructing a loop involving all lattice sites (in this case without haloes) would be, in the absence of vectorisation:

```
for (ic = 1; ic <= nlocal[X]; ic++) {
  for (jc = 1; jc <= nlocal[Y]; jc++) {
    for (kc = 1; kc <= nlocal[Z]; kc++) {

      index = coords_index(ic, jc, kc);
      /* Perform operation per lattice site ... */
    }
  }
}
```

where final array indexing is based on the coordinate index and is specific to the object at hand. To allow transparent switching between different addressing models, the indexing must be abstracted. The abstraction is provided in `memory.h`, where functions (or function-like macros) are provided to make the appropriate address computation. As a concrete example, the following considers a rank 1 3-vector:

```
for (ic = 1; ic <= nlocal[X]; ic++) {
  for (jc = 1; jc <= nlocal[Y]; jc++) {
    for (kc = 1; kc <= nlocal[Z]; kc++) {

      index = coords_index(ic, jc, kc);
      for (ia = 0; ia < 3; ia++) {
        iref = addr_rank1(nsites, 3, index, ia);
        array[iref] = ...;
      }
    }
  }
}
```

Here, the function `addr_rank1()` performs the appropriate arithmetic to reference the correct 1-d array element in either address model. We note that this can be implemented to operate appropriately when the SIMD vector length is an arbitrary number.

This will usually be carried out in computational kernels, where the `kernel.h` interface can be used to help to manage the one-dimensional loop structure required the the targetDP threading model. In this way, the kernel launch stage will specify the limits of the kernel in terms of local x-coordinates, y-coordinates, and z-coordinates in the usual way. The kernel itself is then written in terms of a one-dimensions kernel index.

In the following schematic, the launch stage defines a kernel context to be passed to the kernel along with the other actual arguments:

```
  kernel_info_t limits;
  kernel_ctxt_t * ctxt;

  limits.xmin = 1; limits.xmax = nlocal[X];
  limits.ymin = 1; limits.ymax = nlocal[Y];
  limits.zmin = 1; limits,zmax = nlocal[Z];

  kernel_ctxt_create(NSIMDVL, limits, &ctxt);

  __host_launch(kernel_function, nblk, ntpb, ktx, ...);
```

An suitable vectorised kernel may be constructed as follows with the appropriate base index for the loop computed via the utility `kernel_baseindex()`:

```
for (kindex = 0; kindex < kiterations; kindex += SIMDVL) {
  index = kernel_baseindex(ctxt, kindex);
  for (ia = 0; ia < 3; ia++) {
    for (iv = 0; iv < SIMDVL; iv++) {
      iref = addr_rank1(nsites, 3, index + iv, ia);
      array[iref] = ...;
    }
  }
}
```

Note that the same function `addr_rank1()` with the argument which is the index of the vector loop is used. Note also that vectorised versions must take into account the extent that the kernel extends into the halo region, which involves logic to avoid lattice sites which are not required (this type of masking operation is not shown in this example).

References

[1] Adhikari, R. J.-C. Desplat, and K. Stratford, Sliding periodic boundary conditions for lattice Boltzmann and lattice kinetic equations, `arXiv:cond-mat/0503175v1` (2005).

[2] Apache Portable Runtime Project "APR's Version Numbering". See, e.g., `https://apr.apache.org/versioning.html)` (accessed November 2015).

[3] Adhikari, R., K. Stratford. M.E. Cates, and A.J. Wagner, Fluctuating Lattice Boltzmann, *Europhys. Lett.*, **71**, 473 2005.

[4] Aidun, C.K., Y. Lu, and E.-J. Ding, Direct analysis of particulate suspensions with inertia using the discrete Boltzmann equation, *J. Fluid Mech.*, **373**, 287, 1998.

[5] M.P. Allen and D.J. Tildesley, *Computer simulation of liquids*, Oxford University Press (1987).

[6] G.K. Batchelor *An Introduction to Fluid Mechanics*, Cambridge University Press (1967).

[7] Beris, A.N., and B.J. Edwards, *Thermodynamics of flowing systems with internal microstructure*, Oxford University Press (1994).

[8] J.R. Blake, A spherical envelope approach to ciliary propusion, *J. Fluid Mech.*, **46**, 199, 1971.

[9] J.W. Cahn and J.E. Hilliard, Free energy of a nonuniform system. I. Interfacial free energy, *J. Chem. Phys.* **28** 258–267 (1958).

[10] M. E. Cates, J.-C. Desplat, P. Stansell, A.J. Wagner, K. Stratford, R. Adhikari, and I. Pagonabarraga, Physical and Computational Scaling Issues in Lattice Boltzmann Simulations of Binary Fluid Mixtures, *Phil. Trans. Roy. Soc. A*, **363**, 1917 (2005).

[11] P.M. Chaikin and T.C. Lubensky, *Principles of condensed matter physics*, Cambridge University Press (1995).

[12] B. Chun and A.J.C. Ladd, Interpolated boundary condition for lattice Boltzmann simulations in narrow gaps, *Phys. Rev. E*, **75**, 066705, 2007.

[13] Cichocki, B., and R.B. Jones, Image representation of a spherical particle near a hard wall, *Physica A*, **258**, 273, 1998.

[14] C.-H. Chang and E.I. Franses, Adsorption dynamics of surfactants at the air/water interface: a critical review of mathematical models, data, and mechanisms, *Colloids and Surfaces A*, **100** 1, 1995.

[15] de Gennes, J.-G., and J. Prost, *The physics of liquid crystals*, Oxford University Press (2002).

[16] Desplat, J.-C., I. Pagonabarraga, and P. Bladon, LUDWIG: A parallel lattice-Boltzmann code for complex fluids. *Comput. Phys. Comms.*, **134**, 273, 2001.

[17] H. Diamant and D. Andelman, Kinetics of surfactant adsorption at fluid/fluid interfaces: non-ionic surfactants, *Europhys. Lett.* **34**, 575 (1996).

[18] H. Diamant and D. Andelman, Kinetics of surfactant adsorption at fluid-fluid interfaces, *J. Phys. Chem.*, **100** 13732, 1996.

[19] J.-B. Fournier and P. Galatola, Modeling planar degenerate wetting and anchoring in nematic liquid crystals, *Europhys. Lett.*, **72** 403–409 (2005).

[20] J. Eastoe and J.S. Dalton, Dynamic surface tension and adsorption mechanims of surfactants at the air-water interface, *Advances in Colloid and Interface Science*, **85** 13, 2000.

[21] I. Ginzburg and D. d'Humières, Multireflection boundary conditions for lattice Boltzmann models, *Phys. Rev. E*, **68**, 066614, 2003.

[22] A. Gray and K. Stratford, TargetDP reference.

[23] Hasimoto, H., On the periodic fundamental solutions of the Stokes equation and their application to viscous flow past a cubic array of spheres. *J. Fluid Mech.*, **5**, 317.

[24] Heemels, M.W., M.H.J. Hagen, and C.P. Lowe, Simulating solid colloidal particles using the lattice-Boltzmann method, *J. Comp. Phys.*, **164**, 48, 2000.

[25] IEEE Standard 208-2005, IEEE Standard for Software Configuration Management Plans. See `http://ieeexplore.ieee.org/xpl/standards.jsp` (accessed November 2015).

[26] IEEE Standard 828-2012. IEEE Standard for Configuration Management in Systems and Software Engineering. See `http://ieeexplore.ieee.org/xpl/standards.jsp` (accessed November 2015).

[27] Jeffrey, D.J., and Y. Onishi, Calculation of the resistance and mobility functions for the two unequal rigid spheres in low-Reynolds-number flow, *J. Fluid Mech.*, **139**, 261, 1984.

[28] Kendon, V.M., M.E. Cates, I. Pagonabarraga, J.-C. Desplat, and P. Bladon, Inertial effects in three dimensional spinodal decomposition of a symmetric binary fluid mixture: A lattice Boltzmann study, *J. Fluid Mech.*, **440**, 147 (2001).

[29] Ladd, A.J.C., Numerical simulations of particulate suspensions via a discretised Boltzmann equation. Part 1. Theoretical foundation, *J. Fluid. Mech.*, **271**, 285, 1994.

[30] Ladd, A.J.C., Numerical simulations of particulate suspensions via a discretised Boltzmann equation. Part 2. Numerical results, *J. Fluid. Mech.*, **271**, 311, 1994.

[31] Ladd, A.J.C., Sedimentation of homogenous suspensions of non-Brownian spheres, *Phys. Fluids*, **9**, 491. 1996.

[32] Ladd, A.J.C., Hydrodynamic screening in sedimentating suspensions of non-Brownian spheres, *Phys. Rev. Lett.*, **76**, 1392, 1996.

[33] Ladd, A.J.C., and R. Verberg, Lattice-Boltzmann simulations of particle-fluid suspensions, *J. Stat. Phys.*, **104**, 1191, 2001.

[34] A.W. Lees and S.F. Edwards, The computer study of transport processes under extreme conditions, *J. Phys C* **5** 1921–1929 (1972).

[35] H. Li, C. Pan, and C.T. Miller, Pore-scale investigation of viscous coupling effects for two-phase flow in porous media, *Phys. Rev. E*, **72**, 026705, 2005.

[36] M.J. Lighthill, On the squirming motion of nearly spherical deformable bodies through liquid at very small Reynolds numbers, *Comm. Pure Appl. Math.*, **5**, 109, 1952.

[37] I. Llopis Fusté, *Hydrodynamic cooperativity in micro-swimmer suspensions*, Ph.D. Thesis, University of Barcelona, 2008.

[38] Message Passing Interface Forum. MPI: A Message Passing Interface Standard Version 1.3 (2008).

[39] Nguyen, N.-Q., and A.J.C. Ladd, Lubrication corrections for lattice-Boltzmann simulations of particle suspensions, *Phys. Rev. E*, **66**, 046708, 2002.

[40] T. paapnastasiou, G. Georgiou, and A. Alexandrou, *Viscous Fluid Flow*, CRC Press, Boca Raton, Florida, 2000.

[41] Paraview. See `http://www.paraview.org/`. Accessed 2011.

[42] Rapaport, D.C., *The Art of Molecular Dynamics Simulation*, Cambridge University Press, 1995.

[43] K. Stratford, R. Adhikari, I. Pagonabarraga, and J.-C. Desplat, Lattice Boltzmann for binary fluids with suspended colloids, *J. Stat. Phys.* **121** 163–178 (2005).

[44] S. Succi, *The lattice Boltzmann equation and beyond*, Oxford University Press, Oxford, 2001.

[45] N. Sulaiman, D. Marenduzzo, and J.M. Yeomans, Lattice Boltzmann algorithm to simulate isotropic-nematic emulsions, *Phys. Rev. E* **74** 041708 (2006).

[46] M.R. Swift, E. Orlandini, W.R. Osborn, and J.M. Yeomans, Lattice Boltzmann simulation of liquid-gas and binary fluid systems, *Phys. Rev. E*, **54** 5041 (1996).

[47] M. Venuroli and E.S. Boek, Two-dimensional lattice-Boltzmann simulations of single phase flow in a pseudo two-dimensional micromodel, *Physica A*, **362**, 23, 2006.

[48] R.G.M. van der Sman and S. van der Graaf, Diffuse interface model of surfactant adsorption onto flat and droplet interfaces, *Rheol. Acta* **46** 3 (2006).

[49] O. Theissen and G. Gompper, Lattice Boltzmann study of spontaneous emulsification, *Eur. Phys. J. B*, **11** 91 (1999).

[50] A.J. Wagner and I. Pagonabarraga, Lees-Edwards boundary conditions of lattice Boltzmann, *J. Stat. Phys.* **107** 521–537 (2002).

[51] A.F.H. Ward and L. Tordai, *J. Chem. Phys.* **14** 453, 1946.

[52] M. Skarabot, M. Ravnik, S. Zumer, U. Tkalec, I. Poberaj, D. Babic, N. Osterman and I. Musevic, *Phys. Rev. E* **76**, 051406 (2007).

[53] Wright, D.C¿ and N.D. Mermin, *Rev. Mod. Phys.* **61**, 385–432 (1989).

[54] J. Lyklema *Fundamentals of Interface and Colloid Science* Academic Press (1995).

[55] S. Mafé, J.A. Manzanares, J. Pellicer, *J.* Electroanal. Chem. **2**41, 57-77 (1988).

[56] F. Capuani, I. Pagonabarraga, D. Frenkel, *J.* Chem. Phys. **1**21, 973-986 (2004).

[57] B. Rotenberg, I. Pagonabarraga, D. Frenkel, *F*arad. Discuss. **1**44, 223-243 (2010).

[58] L.D. Landau, E.M. Lifshitz, *E*lectrodynamics of Continuous Media, §15, 2nd ed., Pergamon Press, Oxford, UK (1984).

[59] J.R. Melcher, *C*ontinuum Electromechanics, §3.10, MIT Press, Cambridge, MA, USA (1981).
downloadable from:
`http://ocw.mit.edu/ans7870/resources/melcher/resized/cem_811.pdf`

[60] L.D. Landau, E.M. Lifshitz, *T*heory of Elasticity, §3 & §16, 3rd ed., Butterworth-Heinemann, Oxford, UK (1986).