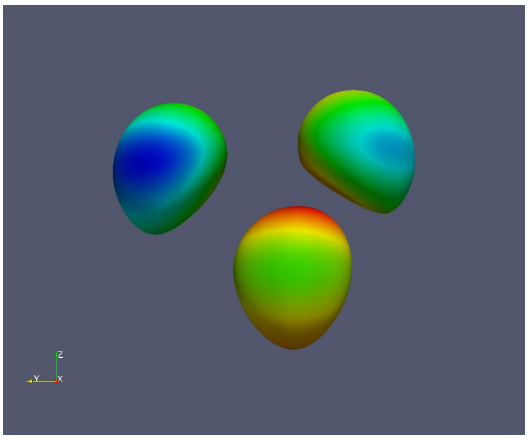
# MP-LABS v1.5

**MultiPhase Lattice Boltzmann Suite User Guide**
**Document Revision 1.0**
**September 2, 2014**




Carlos Rosales Fernández
carlos@tacc.utexas.edu

High Performance Computing
Texas Advanced Computing Center
The University of Texas at Austin

# Preface

MP-LABS is a suite of numerical simulation tools for multiphase flows based on the free energy Lattice Boltzmann Method (LBM). The code allows for the simulation of quasi-incompressible two-phase flows, and uses multiphase models that allow for large density ratios. MP-LABS provides implementations that use periodic boundary conditions, but it is written in a way that allows for easy inclusion of different boundary conditions. The output from MP-LABS is in plain ASCII and VTK format, and can be analyzed using other Open Source tools such as Gnuplot [1] and Paraview [2].

Revision 1.5 is a bugfix for the lbs3d-mpi optimized code. Details regarding this optimized version can be found in [3]

The objective of the MP-LABS project is to provide a core set of routines that are well documented, highly portable, and have proven to perform well in a variety of systems. The source code is written in Fortran 90, OpenMP and MPI and uses separate subroutines for most tasks in order to make modifications easier. An examples of research work performed using this code are references [4] and [5].

We believe in open source software for pushing the frontiers of science and engineering, and we encourage everyone to publish open source software under the GPL License. You are free to use, copy, modify and distribute any element of MP-LABS under the GNU General Public License version 3. Finally, although the GPL License does not require you to contact us, the authors would appreciate if you could send an email telling us that you are using MP-LABS and what application you are using the code for.

The complete package can be downloaded from code.google.com/p/mplabs.

Carlos Rosales Fernández
September 2, 2014

# Contents

# Chapter 1

# Installation

`MP-LABS` is distributed as a series of Fortran 90 subroutine files, and a series of Make files. The complete list of functions is collected in Appendix A.

## 1.1 Quick Install

Download `mplabs-1.2.tar.gz` from `code.google.com/p/mplabs`. Unzip the `mplabs-1.2.tar.gz` file in a suitable directory:

```
$ tar -zxvf mplabs-1.3.tar.gz
```

Once the file has been unpackaged move into the `mplabs` directory and type:

```
$ make && make install
```

The compilation uses by default GNU's gfortran for the serial code and the wrapper mpif90 for the parallel code, with the flag -O3. If for some reason you don't like this, or you would like to add an architecture-related flag read Section 1.2.2 for details.

Assuming you meet all the installation requirements, the binary files are now in the directory `mplabs/bin` and the documentation is in `mplabs/docs`. Enjoy!

## 1.2 Installation

Download `mplabs-1.2.tar.gz` from `code.google.com/p/mplabs`. Unzip the `mplabs-1.2.tar.gz` file in a suitable directory. This can be done in a linux system by typing on the command line:

```
$ tar -zxvf mplabs-1.2.tar.gz
```

or, if tar does not accept the "z" option in your system:

```
$ gunzip mplabs-1.2.tar.gz
$ tar -xvf mplabs-1.2.tar
```

The following directories will be created:

| | |
|---|---|
| `mplabs` | : Main program directory, license and readme files |
| `mplabs/bin` | : Binary files are stored here after compilation |
| `mplabs/docs` | : Code documentation |
| `mplabs/examples` | : Examples to test the installation |
| `mplabs/src` | : Fortran 90 source code |
| `mplabs/src/devel` | : Source code for development implementations |
| `mplabs/src/opt` | : Source code for optimized implementations |
| `mplabs/src/std` | : Source code for standard implementations |

### 1.2.1 Requirements

In order to install and run `MP-LABS` all the `.F90` files listed in Appendix A are necessary. Make sure the uncompressed files exist in their corresponding directories within `mplabs/src`.

Besides the source code for `MP-LABS` you will also need working versions of:

- A Fortran compiler such as gfortran,
- Make,
- An MPI library such as OpenMPI [6] (only for the parallel version).

### 1.2.2 Compilation

A total of ten executables can be compiled in `MP-LABS`. The first eight executables correspond to 2D serial and parallel (MPI) versions of standard and dual grid implementations of the Zheng-Shu-Chew multiphase LBM [7] and the Lee-Lin multiphase LBM [8]. The ninth version is the 3D implementation of the Zheng-Shu-Chew multiphase LBM. The tenth binary is an optimized version of the later. For ease of reference in the rest of this document the two models will be called ZSC-LBM and LL-LBM. The name codes of the executables are self-explanatory, and the following table can be used as a reference (source code directories are within `mplabs/src/`):

| Name | Model | Type | OMP | MPI | Dual Grid | Source Dir |
|---|---|---|---|---|---|---|
| `LL-2D-DGR` | Lee & Lin | D2Q9 | | | * | `std/ll-dgr-seq` |
| `LL-2D` | Lee & Lin | D2Q9 | | | | `std/ll-seq` |
| `LL-2D-DGR-MPI` | Lee & Lin | D2Q9 | | * | * | `std/ll-dgr-mpi` |
| `LL-2D-MPI` | Lee & Lin | D2Q9 | | * | | `std/ll-mpi` |
| `ZSC-2D-DGR` | Zheng *et al* | D2Q9 | | | * | `std/zsc-dgr-seq` |
| `ZSC-2D` | Zheng *et al* | D2Q9 | | | | `std/zsc-seq` |
| `ZSC-2D-DGR-MPI` | Zheng *et al* | D2Q9 | | * | * | `std/zsc-dgr-mpi` |
| `ZSC-2D-MPI` | Zheng *et al* | D2Q9 | | * | | `std/zsc-mpi` |
| `ZSC-3D-MPI` | Zheng *et al* | D3Q19 | | * | | `std/zsc-3d-mpi` |
| `LBS3D` | Zheng *et al* | D3Q19 | * | | | `opt/lbs3d` |
| `LBS3D-MPI` | Zheng *et al* | D3Q19 | * | * | | `opt/lbs3d-mpi` |

In addition to the baseline codes in `mplabs/src/std` and `mplabs/src/opt` a number of intermediate development codes have been provided to illustrate the process of improving performance of a code, starting from a naïve serial implementation and progressing to a code equivalent to LBS3D. These codes follow the Zheng *et al* 3D model, and are located in the `mplabs/src/devel` directory. An example of offload execution to the Intel Xeon Phi while also using the host CPU is provided. The following table provides additioanl details:

| Name | OMP | Offload | Source Dir |
|---|---|---|---|
| `LBS3D-DEVEL-SEQ` | | | `devel/sequential` |
| `LBS3D-DEVEL-OMPv1` | Collision Only | | `devel/omp_v1` |
| `LBS3D-DEVEL-OMPv2` | * | | `devel/omp_v2` |
| `LBS3D-DEVEL-SOA` | * | | `devel/soa` |
| `LBS3D-DEVEL-SOA_AL` | * | | `devel/soa_aligned` |
| `LBS3D-DEVEL-OFFLOAD` | * | * | `devel/offload` |

Edit the `mplabs/Makefile.in` file in order to define the compiler, compiler options, and installation directory. This is the only file where you should have to make changes in order to build the software.

The provided Makefile can be used to build the sequential codes only:

```
$ make seq
$ make seq-install
```

The MPI parallel codes only:

```
$ make par
$ make par-install
```

The optimized code only:

```
$ make opt
$ make opt-install
```

or everything:

```
$ make
$ make install
```

The development codes must be built and installed separately:

```
$ make devel
$ make devel-install
```

This will compile the relevant source code and move the binary files to `INSTALL_DIR/bin`, with the `INSTALL_DIR` defined in Makefile.in.

### 1.2.3 Multiple Versions

Use the provided variable `BIN_SUFFIX` in order to add an informative suffix to your binary files. This could be an indication of the compiler flags used such as `.dbg` or `.opt`, or an indication of the architecture such as `.cpu` or `.mic`.

This variable can be defined in the command line when invoking Make, exported tot he environment, or defined in `Makefile.in`. The definition in Makefile.in will

take precedence.

# Chapter 2

# Input Files

There are two input files required to run any simulation, one called `properties.in` which contains all the simulation parameters, and another one called `discrete.in` which contains a description of the discrete phase.

The input file format is identical for standard and dual grid implementations, but slightly different for the Lee-Lin and Zheng-Shu-Chew versions as they require different parameters. This chapter describes the input files in detail.

## 2.1 Lee-Lin Properties File

The properties file `properties.in` for the Lee-Lin model contains the following variables:

| | |
|---|---|
| *MaxStep* : | Maximum number of iterations to run. |
| *tStat* : | Iterations between statistical data writes to `stats.out`. |
| *tDump* : | Iterations between hydrodynamics data writes to VTK files. |
| *xmin* : | Minimum x value in the geometry. |
| *xmax* : | Maximum x value in the geometry. |
| *ymin* : | Minimum y value in the geometry. |
| *ymax* : | Maximum y value in the geometry. |
| *rhoL* : | Density of the lighter fluid. |
| *rhoH* : | Density of the heavier fluid. |
| *tauL* : | Relaxation time for the lighter fluid. |

| | |
|---|---|
| *tauH* : | Relaxation time for the heavier fluid. |
| *IntWidth* : | Interface width in lattice units. |
| *sigma* : | Interfacial tension. |
| *pConv* : | Maximum relative error for pressure convergence. |
| *mpi_xdim*: | [OPT: MPI only] Number of partitions in x. |
| *mpi_ydim*: | [OPT: MPI only] Number of partitions in y. |

This file typically takes the following form:

```
MaxStep
50000
tStat, tDump
100, 2000
xmin, xmax, ymin, ymax
1, 300, 1, 100
rhoL, rhoH
0.001D0, 1.D0
tauL, tauH
1.0D0, 0.1D0
IntWidth, sigma
5.0D0, 0.001D0
pConv
0.001D0
mpi_xdim, mpi_ydim
1, 2
```

where the last two lines are only required for the parallel versions of the code, and will be ignored by the serial code.

If this input file is provided a regular cartesian grid of size $300 \times 100$ will be used for the simulation, which will run for a maximum of 50000 iterations or until the relative pressure error is below `pConv` $= 0.001$.

The average velocity of the discrete phase, the mas conservation factor, the effective bubble radius, the pressure difference between the interior of the bubble and the continuous phase, and the error with respect to Laplace's Law ($P_{\text{in}} - P_{rmout} = \sigma/R$) will be saved to the file `stats.out` every 100 iterations.

The order parameter, pressure and velocity values for the full computational do-

main will be saved in VTK format every 2000 iterations to files DATA_0002000.vtk, DATA_0004000.vtk, etc...

Notice that for the parallel cases the total number of processors used is given by the product `nprocs = mpi_xdim`×`mpi_ydim` and not by the sum.

## 2.2 Zheng-Shu-Chew Properties File

The properties file `properties.in` for the Zheng-Shu-Chew model contains the following variables:

| | |
|---|---|
| *MaxStep* : | Maximum number of iterations to run. |
| *RelaxStep*: | [OPT: 3D only] Maximum number of steps in the relaxation run. |
| *tStat* : | Iterations between statistical data writes to `stats.out`. |
| *tDump* : | Iterations between hydrodynamics data writes to VTK files. |
| *xjump* : | [OPT: 3D Only] Save every *xjump* nodes to the VTK otuput files. |
| *xmin* : | Minimum x value in the geometry. |
| *xmax* : | Maximum x value in the geometry. |
| *ymin* : | Minimum y value in the geometry. |
| *ymax* : | Maximum y value in the geometry. |
| *zmin* : | [OPT: 3D only] Minimum z value in the geometry. |
| *zmax* : | [OPT: 3D only] Maximum z value in the geometry. |
| *rhoL* : | Density of the lighter fluid. |
| *rhoH* : | Density of the heavier fluid. |
| *tauRho* : | Relaxation time. |
| *tauPhi* : | Relaxation parameter for the phase. |
| *IntWidth* : | Interface width in lattice units. |
| *sigma* : | Interfacial tension. |
| *Gamma* : | Interface mobility. |
| *pConv* : | Maximum relative error for pressure convergence. |
| *mpi_xdim* : | [OPT: MPI only] Number of partitions in x. |
| *mpi_ydim* : | [OPT: MPI only] Number of partitions in y. |
| *mpi_zdim* : | [OPT: 3D MPI only] Number of partitions in z. |

This file typically takes the following form for a 2D simulation:

```
MaxStep
50000
tStat, tDump
100, 2000
xmin, xmax, ymin, ymax
1, 300, 1, 100
rhoL, rhoH
1.0D0, 1000.D0
tauRho, tauPhi
0.6D0, 0.7D0
IntWidth, sigma, Gamma
5.0D0, 0.01D0, 200.D0
pConv
0.001D0
mpi_xdim, mpi_ydim
1, 2
```

where the last two lines are only required for the parallel versions of the code, and will be ignored by the serial code.

If this input file is provided a regular cartesian grid of size $300 \times 100$ will be used for the simulation, which will run for a maximum of 50000 iterations or until the relative pressure error is below `pConv` $= 0.001$.

The average velocity of the discrete phase, the mas conservation factor, the effective bubble radius, the pressure difference between the interior of the bubble and the continuous phase, and the error with respect to Laplace's Law ($P_{\text{in}} - P_{rmout} = \sigma/R$) will be saved to the file `stats.out` every 100 iterations.

The order parameter, pressure and velocity values for the full computational domain will be saved in VTK format every 2000 iterations to files DATA_0002000.vtk, DATA_0004000.vtk, etc...

Notice that for the parallel cases the total number of processors used is given by the product `nprocs = mpi_xdim`$\times$`mpi_ydim` and not by the sum.

## 2.3 LBS3D Properties File

The properties file `properties.in` for the optimized code contains the following variables:

| | |
|---|---|
| *MaxStep* : | Maximum number of iterations to run. |
| *RelaxStep*: | Maximum number of steps in the relaxation run. |
| *tStat* : | Iterations between statistical data writes to `stats.out`. |
| *tSave* : | Iterations between hydrodynamics data writes to VTK files. |
| *xmin* : | Minimum x value in the geometry. |
| *xmax* : | Maximum x value in the geometry. |
| *ymin* : | Minimum y value in the geometry. |
| *ymax* : | Maximum y value in the geometry. |
| *zmin* : | Minimum z value in the geometry. |
| *zmax* : | Maximum z value in the geometry. |
| *rhoL* : | Density of the lighter fluid. |
| *rhoH* : | Density of the heavier fluid. |
| *tauRho* : | Relaxation time. |
| *tauPhi* : | Relaxation parameter for the phase. |
| *IntWidth* : | Interface width in lattice units. |
| *sigma* : | Interfacial tension. |
| *Gamma* : | Interface mobility. |
| *Eo* : | Eotvos number, used to determine external gravity force. |
| *pConv* : | Maximum relative error for pressure convergence. |
| *mpi_xdim* : | [OPT: MPI only] Number of partitions in x. |
| *mpi_ydim* : | [OPT: MPI only] Number of partitions in y. |
| *mpi_zdim* : | [OPT: MPI only] Number of partitions in z. |

This file typically takes the following form:

```
MaxStep, RelaxStep
50000, 1000
tStat, tSave
100, 2000
xmin, xmax, ymin, ymax, zmin, zmax
1, 300, 1, 100, 1, 100
rhoL, rhoH
1.0D0, 1000.D0
tauRho, tauPhi
0.6D0, 0.7D0
IntWidth, sigma, Gamma
5.0D0, 0.01D0, 200.D0
Eo, pConv
1.D0, 0.001D0
mpi_xdim, mpi_ydim, mpi_zdim
1, 1, 2
```

where the last two lines are only required for the parallel versions of the code, and will be ignored by the serial code.

If this input file is provided a regular cartesian grid of size $300 \times 100 \times 100$ will be used for the simulation, which will run without application of the external force for a maximum of 1000 iterations or until the relative pressure error is below `pConv` $= 0.001$. This relaxation loop is designed to relax the domain and minimize spurious velocities at the fluid-fluid interface before starting the 50000 step evolution loop that includes external forcing.

The average velocity of the discrete phase, the mas conservation factor, the effective bubble radius, the pressure difference between the interior of the bubble and the continuous phase, and the error with respect to Laplace's Law ($P_{\text{in}} - P_{rmout} = \sigma/R$) will be saved to the file `stats.out` every 100 iterations.

The order parameter, pressure and velocity values for the full computational domain will be saved in VTK format every 2000 iterations to files EVOL_0002000.vtk, EVOL_0004000.vtk, etc...

## 2.4 Discrete Phase File

The discrete phase file `discrete.in` contains the following variables:

| | |
|---|---|
| *DISTRO* : | [OPT: 3D only] Random (==1) or given (/=1) bubble positions. |
| *xb* : | [OPT: 3D, DISTRO=1 only] Number of different positions in x |
| *yb* : | [OPT: 3D, DISTRO=1 only] Number of different positions in y |
| *zb* : | [OPT: 3D, DISTRO=1 only] Number of different positions in z |
| *rb* : | [OPT: 3D, DISTRO=1 only] Radius of the bubbles. |
| *nBubbles* : | Number of bubbles to include. |
| *bubbles(i,1)* : | x coordinate of the i*th* bubble. |
| *bubbles(i,2)* : | y coordinate of the i*th* bubble. |
| *bubbles(i,3)* : | radius of the i*th* bubble. |

This file typically takes the following form for a 2D simulation:

```
nBubbles
2
xi, yi, ri
40, 51, 20
120, 51, 30
```

This example will generate two bubbles, the first at $(40, 51)$ with radius 20 lattice units, and the second at $(120, 51)$ with radius 30 lattice units.

**WARNING:** There is no limit to the number of bubbles/drops that can be included in the simulation, but one should be aware that the code does not check for overlap. Overlapping bubbles will be initialized as if they had merged and may produce unexpected simulation results.

# Chapter 3

# Multiphase Models

This chapter describes the two multphase models available in `MP-LABS`. In both cases the physical equations that are simulated are the Navier-Stokes equation together, the mass conservation equation, and a convective Cahn-Hilliard equation to track the interface evolution:

$$\frac{\partial \rho}{\partial t} + \nabla \cdot (\rho \mathbf{u}) = 0, \tag{3.1}$$

$$\frac{\partial \rho \mathbf{u}}{\partial t} + \nabla \cdot (\rho \mathbf{u}\mathbf{u}) = -\nabla \cdot P + \mu \nabla^2 \mathbf{u} + \mathbf{F_b}, \tag{3.2}$$

$$\frac{\partial \phi}{\partial t} + \nabla \cdot (\phi \mathbf{u}) = \theta_M \nabla^2 \mu_\phi. \tag{3.3}$$

where $\mu_\phi$ is the chemical potential, $\theta_M$ is the mobility of the interface (molecular diffusion mobility), $P$ is the pressure tensor, $\mathbf{F_b}$ is the body force, $\rho$ is the density, $\mu$ is the viscosity, and $\phi$ is the order parameter.

## 3.1 Lee-Lin

The Lee & Lin [8] formulation allows for the simulation of tw-phase flows with arbitrary density and viscosity ratios. The integration of the relaxation and the

forcing terms is done using a trapezoidal rule which requires three steps in the collision-stream procedure:

(1) Pre-streaming Step

$$\begin{aligned}
\tilde{g}_i(\mathbf{r},t) &= g_i(\mathbf{r},t) + \left.\frac{g_i^{\rm eq}-g_i}{2\tau}\right|_{(\mathbf{r},t)} + \left.\frac{\delta t}{2}(\mathbf{c}_i-\mathbf{u})\nabla\rho\left[\Gamma_i(\mathbf{u})-\Gamma_i(0)\right]\right|_{(\mathbf{r},t)} \\
&+ \left.\frac{\delta t}{2}\frac{(c_{i\alpha}-u_\alpha)\left[\kappa\partial_\alpha(\partial_\gamma\rho\partial_\gamma\rho)-\kappa\partial_\beta(\partial_\alpha\rho\partial_\beta\rho)\right]}{c_s^2}\Gamma_i(\mathbf{u})\right|_{(\mathbf{r},t)}, \qquad (3.4)\\
\tilde{f}_i(\mathbf{r},t) &= f_i(\mathbf{r},t) + \left.\frac{f_i^{\rm eq}-f_i}{2\tau}\right|_{(\mathbf{r},t)} \\
&+ \left.\frac{\delta t}{2}\frac{(\mathbf{c}_i-\mathbf{u})\left[\nabla\rho c_s^2-\rho\nabla(\varphi-\kappa\nabla^2\rho)\right]}{c_s^2}\Gamma_i(\mathbf{u})\right|_{(\mathbf{r},t)}, \qquad (3.5)
\end{aligned}$$

(2) Streaming Step

$$\begin{aligned}
\tilde{g}_i(\mathbf{r}+\mathbf{c}_i\delta t,t+\delta t) &= \tilde{g}_i(\mathbf{r},t)\,, \qquad (3.6)\\
\tilde{f}_i(\mathbf{r}+\mathbf{c}_i\delta t,t+\delta t) &= \tilde{f}_i(\mathbf{r},t)\,, \qquad (3.7)
\end{aligned}$$

(3) Post-Streaming Step

$$\begin{aligned}
g_i(\mathbf{r}+\mathbf{c}_i\delta t,t+\delta t) &= \tilde{g}_i(\mathbf{r}+\mathbf{c}_i\delta t,t+\delta t) + \left.\frac{g_i^{\rm eq}-\tilde{g}_i}{2\tau+1}\right|_{(\mathbf{r}+\mathbf{c}_i\delta t,t+\delta t)} \\
&+ \left.\frac{2\tau}{2\tau+1}\frac{\delta t}{2}(\mathbf{c}_i-\mathbf{u})\nabla\rho\left[\Gamma_i(\mathbf{u})-\Gamma_i(0)\right]\right|_{(\mathbf{r}+\mathbf{c}_i\delta t,t+\delta t)} \\
&+ \left.\frac{2\tau}{2\tau+1}\frac{\delta t}{2}\frac{(c_{i\alpha}-u_\alpha)\left[\kappa\partial_\alpha(\partial_\gamma\rho\partial_\gamma\rho)-\kappa\partial_\beta(\partial_\alpha\rho\partial_\beta\rho)\right]}{c_s^2}\Gamma_i(\mathbf{u})\right|_{(\mathbf{r}+\mathbf{c}_i\delta t,t+\delta t)} \qquad (3.8)
\end{aligned}$$

$$\begin{aligned}
f_i(\mathbf{r}+\mathbf{c}_i\delta t,t+\delta t) &= \tilde{f}_i(\mathbf{r}+\mathbf{c}_i\delta t,t+\delta t) + \left.\frac{f_i^{\rm eq}-\tilde{f}_i}{2\tau+1}\right|_{(\mathbf{r}+\mathbf{c}_i\delta t,t+\delta t)} \\
&+ \left.\frac{2\tau}{2\tau+1}\frac{\delta t}{2}\frac{(\mathbf{c}_i-\mathbf{u})\left[\nabla\rho c_s^2-\rho\nabla(\varphi-\kappa\nabla^2\rho)\right]}{c_s^2}\Gamma_i(\mathbf{u})\right|_{(\mathbf{r}+\mathbf{c}_i\delta t,t+\delta t)} \qquad (3.9)
\end{aligned}$$

The chemical potential is defined as,

$$\varphi = 4\beta(\rho - \rho_H)(\rho - \rho_L)(\rho - \rho_M) \ , \tag{3.10}$$

with $\rho_M = (\rho_L + \rho_H)/2$, and the function $\Gamma$ is defined as,

$$\Gamma_i(\mathbf{u}) = w_i \left[ 1 + \frac{\mathbf{c} \cdot \mathbf{u}}{c_s^2} + \frac{(c_{i\alpha}c_{i\beta} - c_s^2\delta_{\alpha\beta})u_\alpha u_\beta}{2c_s^4} \right] \ . \tag{3.11}$$

The parameters $\beta$ and $\kappa$ depend on the sufrace tension, $\sigma$, and the interface width, $W$,

$$\beta = \frac{3\sigma}{W\rho^{*4}}, \tag{3.12}$$

$$\kappa = \frac{1}{2}A(W\rho^*)^2, \tag{3.13}$$

where $\rho^* = (\rho_H - \rho_L)/2$.

The relaxation time $\tau$ is assumed to depend linearly on the density,

$$\tau(\mathbf{r}, t) = \tau_H \left( \frac{\rho(\mathbf{r}, t) - \rho_L}{\rho_H - \rho_L} \right) + \tau_L \left( \frac{\rho_H - \rho(\mathbf{r}, t)}{\rho_H - \rho_L} \right) \ . \tag{3.14}$$

with $\tau_H$ and $\tau_L$ the constant relaxation times corresponding to each of the fluids.

The macroscopic density, pressure and velocity are calculated after the streaming step based on the values of $\tilde{f}_i$ and $\tilde{g}_i$,

$$\rho = \sum_i \tilde{f}_i \ , \tag{3.15}$$

$$\rho u_\alpha = \sum_i c_{\alpha i}\tilde{g}_i + \frac{\delta t}{2}\kappa \left[ \frac{\partial}{\partial r_\alpha} \left( \frac{\partial \rho}{\partial r_\gamma} \frac{\partial \rho}{\partial r_\gamma} \right) - \frac{\partial}{\partial r_\beta} \left( \frac{\partial \rho}{\partial r_\alpha} \frac{\partial \rho}{\partial r_\beta} \right) \right] \ , \tag{3.16}$$

$$p = c_s^2 \sum_i \tilde{g}_i + \frac{\delta t}{2} c_s^2 \mathbf{u} \nabla \rho \ , \tag{3.17}$$

where the pressure is required for the calculation of $g_i^{\mathrm{eq}}$ as shown in Section 3.1.1.

The discretization of the forcing terms uses a second order switching scheme that alternates between a biased differencing scheme and a central differencing scheme, and which is crucial to ensure stability for large density ratios between the fluids.

### 3.1.1 Equilibrium distribution functions

Both distribution functions are discretized using using D2Q9, and take the following equilibrium values,

$$g_i^{\mathrm{eq}} = w_i \left[ \frac{p}{c_s^2} + \rho \left( \frac{\mathbf{c}_i \cdot \mathbf{u}}{c_s^2} + \frac{(c_{i\alpha} c_{i\beta} - c_s^2 \delta_{\alpha\beta}) u_\alpha \beta}{2 c_s^4} \right) \right] , \tag{3.18}$$

$$f_i^{\mathrm{eq}} = w_i \rho \left[ 1 + \frac{\mathbf{c}_i \cdot \mathbf{u}}{c_s^2} + \frac{(c_{i\alpha} c_{i\beta} - c_s^2 \delta_{\alpha\beta}) u_\alpha \beta}{2 c_s^4} \right] , \tag{3.19}$$

with the coefficients $w_i$,

$$w_0 = \frac{4}{9} , \tag{3.20}$$

$$w_i = \frac{1}{9} \qquad (i = 1, \ldots, 4) , \tag{3.21}$$

$$w_i = \frac{1}{36} \qquad (i = 5, \ldots, 8) , \tag{3.22}$$

## 3.2 Zheng-Shu-Chew

The Zheng-Shu-Chew model for two phase flows is limited to fluids with identical viscosity $\mu$, and uses an average density $n = (\rho_{\mathrm{L}} + \rho_{\mathrm{H}})/2$ and an order parameter $\phi$ for the simulation. The advantage of this model is that the interface between the two phases is captured using a convective Cahn-Hilliard equation with second order accuracy. This is achieved by using a standard lattice Boltzmann equation for the

momentum distribution function $g$, but introducing an over-relaxation term in the equation for the order parameter $f$,

$$g_i(\mathbf{x}+\mathbf{c}_i\delta t, t+\delta t) = g_i(\mathbf{x},t) + \Omega_i^g \tag{3.23}$$

$$f_i(\mathbf{x}+\mathbf{c}_i\delta t, t+\delta t) = f_i(\mathbf{x},t) + \Omega_i^f + (1-\eta)[f_i(\mathbf{x}+\mathbf{c}_i\delta t, t) - f_i(\mathbf{x},t)] \tag{3.24}$$

Here $\Omega_i$ is the collision term in the BGK [9] approximation, modified to include the effect of interfacial and external forces:

$$\begin{aligned}\Omega_i^g &= \frac{g_i^{eq}(\mathbf{x},t) - g_i(\mathbf{x},t)}{\tau_n} \\ &+ \left(1 - \frac{1}{2\tau_n}\right)\frac{E_i}{c_s^2}\left[(\mathbf{c}_i - \mathbf{u}) + \frac{(\mathbf{c}_i\cdot\mathbf{u})}{c_s^2}c_i\right]\left(\mu_\phi\vec{\nabla}\phi + \mathbf{F_b}\right)\end{aligned} \tag{3.25}$$

$$\Omega_i^f = \frac{f_i^{eq}(\mathbf{x},t) - f_i(\mathbf{x},t)}{\tau_\phi} \tag{3.26}$$

where $g_i$ and $f_i$ are the distribution functions for the momentum and the phase, $\tau_n$ and $\tau_\phi$ are their respective relaxation times, $\mathbf{c}_i$ is the lattice velocity, and $\eta$ is the over-relaxation constant coefficient. This scheme reduces to the standard lattice Boltzmann equation when $\eta$ is unity.

The macroscopic quantities corresponding to the order parameter $\phi$, the density of the fluid $n$, and its velocity $\mathbf{u}$, are defined in terms of the distribution functions $f$ and $g$,

$$\phi = \sum_i f_i \tag{3.27}$$

$$n = \sum_i g_i \tag{3.28}$$

$$\mathbf{u} = \frac{1}{n}\left[\sum_i \mathbf{c}_i g_i + \frac{1}{2}\left(\mu_\phi\vec{\nabla}\phi + \mathbf{F_b}\right)\right] \tag{3.29}$$

where the body force is $\mathbf{F_b} = 2\phi^*\mathbf{g}$, and the chemical potential is given by:

$$\mu_\phi = 4\alpha(\phi^3 - \phi^{*2}\phi) - \kappa\nabla^2\phi, \tag{3.30}$$

with $\phi^*$ defined as the constant $\phi^* = (\rho_H - \rho_L)/2$. In this expression the parameters $\alpha$ and $\kappa$ depend on the sufrace tension, $\sigma$, and the interface width, $W$ [7],

$$\alpha = \frac{3\sigma}{W\phi^{*4}}, \tag{3.31}$$

$$\kappa = \frac{1}{2}A(W\phi^*)^2. \tag{3.32}$$

The pressure is calculated from the order parameter $\phi$ and the average density $n$ as:

$$p = \alpha\left(3\phi^4 - 2\phi^2\phi^{*2} - \phi^{*4}\right) - \kappa\left[\nabla^2\phi + \frac{1}{2}\left(\nabla\phi\right)^2\right] + c_s^2 n, \tag{3.33}$$

where $c_s$ is the lattice speed of sound.

The expressions used for the relaxation parameter $\eta$ and the mobility $\theta_M$ are chosen so that the Cahn-Hilliard equation (3.3) can be recovered to second order from (3.24) using the Chapman-Enskog expansion, resulting in the following values,

$$\eta = \frac{1}{\tau_\phi + 0.5} \tag{3.34}$$

$$\theta_M = \eta\left(\tau_\phi\eta - \frac{1}{2}\right)\delta\Gamma \tag{3.35}$$

### 3.2.1 Equilibrium distribution functions

Using these discretization schemes the equilibrium values for the distribution functions are given by,

$$g_i^{\mathrm{eq}} = E_i A_i^g + E_i n\left(3c_{i\alpha}u_\alpha + \frac{9}{2}u_\alpha u_\beta c_{i\alpha}c_{i\beta} - \frac{3}{2}u^2\right), \tag{3.36}$$

$$f_i^{\rm eq} = A_i^f + B_i^f \phi + C_i^f \phi \mathbf{c}_i \cdot \mathbf{u} \, . \tag{3.37}$$

**Equilibrium coefficients (2D)**

The $f$ distribution function is discretized using D2Q5, ad its equilibrium coefficients are,

$$A_0^f = -2\Gamma\mu_\phi \, , \tag{3.38}$$
$$A_i^f = \frac{1}{2}\Gamma\mu_\phi \qquad (i = 1, \ldots, 4) \, , \tag{3.39}$$
$$B_0^f = 1 \, , \tag{3.40}$$
$$B_i^f = 0 \qquad (i = 1, \ldots, 4) \, , \tag{3.41}$$
$$C_i^f = \frac{1}{2\eta} \qquad (i = 0, \ldots, 4) \, , \tag{3.42}$$

The momentum distribution function $g$ is discretized using D2Q9, and its equilibrium coefficients are defined as,

$$A_0^g = \frac{1}{4}\left[9n - 15\left(\phi\mu_\phi + \frac{n}{3}\right)\right] \, , \tag{3.43}$$
$$A_i^g = 3\phi\mu_\phi + n \qquad (i = 1, \ldots, 8) \, , \tag{3.44}$$
$$E_0 = \frac{4}{9} \, , \tag{3.45}$$
$$E_i = \frac{1}{9} \qquad (i = 1, \ldots, 4) \, , \tag{3.46}$$
$$E_i = \frac{1}{36} \qquad (i = 5, \ldots, 8) \, , \tag{3.47}$$

**Equilibrium coefficients (3D)**

The $f$ distribution function is discretized using D3Q7, and its equilibrium coefficients are,

$$A_0^f = -2\Gamma\mu_\phi \,, \tag{3.48}$$

$$A_i^f = \frac{1}{2}\Gamma\mu_\phi \qquad (i = 1 \ldots 6) \,, \tag{3.49}$$

$$B_0^f = \,, \tag{3.50}$$

$$B_i^f = 0 \qquad (i = 1 \ldots 6) \,, \tag{3.51}$$

$$C_i^f = \frac{1}{2\eta} \qquad (i = 0 \ldots 6) \,. \tag{3.52}$$

The momentum is discretized using D3Q19, and its equilibrium coefficients are,

$$A_0 = \frac{9}{4}n - \frac{15}{4}(\phi\mu_\phi + \frac{1}{3}n) \,, \tag{3.53}$$

$$A_i = 3(\phi\mu_\phi + \frac{1}{3}n) \qquad (i = 1 \ldots 8) \,, \tag{3.54}$$

$$E_0 = \frac{4}{9} \,, \tag{3.55}$$

$$E_i = \frac{1}{9} \qquad (i = 1 \ldots 4) \,, \tag{3.56}$$

$$E_i = \frac{1}{36} \qquad (i = 5 \ldots 8) \,. \tag{3.57}$$

## 3.3 Velocity discretization directions

| $i$ | 0 | 1 | 2 | 3 | 4 |
|---|---|---|---|---|---|
| $c_{ix}$ | 0 | 1 | 0 | -1 | 0 |
| $c_{iy}$ | 0 | 0 | 1 | 0 | -1 |

Table 3.1: D2Q5 direction vectors

| $i$ | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 |
|---|---|---|---|---|---|---|---|---|---|
| $c_{ix}$ | 0 | 1 | 0 | -1 | 0 | 1 | -1 | -1 | 1 |
| $c_{iy}$ | 0 | 0 | 1 | 0 | -1 | 1 | 1 | -1 | -1 |

Table 3.2: D2Q9 direction vectors

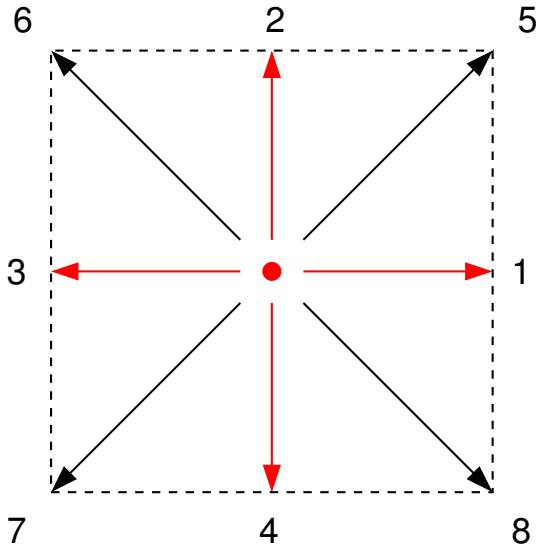



Figure 3.1: D2Q5 (red) and D2Q9 (red and black) velocity directions. Direction 0 corresponds to the central node.

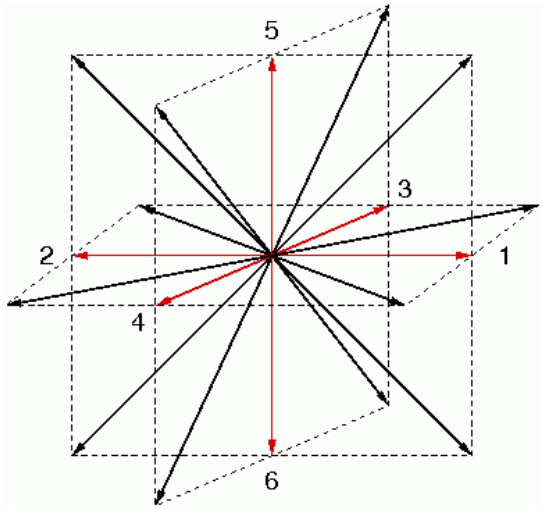



Figure 3.2: D3Q7 (red) and D3Q19 (read and black) velocity directions. Direction 0 corresponds to the central node.

| $i$ | 0 | 1 | 2 | 3 | 4 | 5 | 6 |
|---|---|---|---|---|---|---|---|
| $c_{ix}$ | 0 | 1 | -1 | 0 | 0 | 0 | 0 |
| $c_{iy}$ | 0 | 0 | 0 | 1 | -1 | 0 | 0 |
| $c_{iz}$ | 0 | 0 | 0 | 0 | 0 | 1 | -1 |

Table 3.3: D3Q7 direction vectors

| $i$ | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 | 12 | 13 | 14 | 15 | 16 | 17 | 18 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| $c_{ix}$ | 0 | 1 | -1 | 0 | 0 | 0 | 0 | 1 | -1 | 1 | -1 | 1 | -1 | 1 | -1 | 0 | 0 | 0 | 0 |
| $c_{iy}$ | 0 | 0 | 0 | 1 | -1 | 0 | 0 | 1 | -1 | -1 | 1 | 0 | 0 | 0 | 0 | 1 | -1 | 1 | -1 |
| $c_{iz}$ | 0 | 0 | 0 | 0 | 0 | 1 | -1 | 0 | 0 | 0 | 0 | 1 | -1 | -1 | 1 | 1 | -1 | -1 | 1 |

Table 3.4: D3Q19 direction vectors

# Chapter 4

# Implementation Details

This chapter describes briefly the dual grid implementation as well as some practical implementation details.

## 4.1 Dual Grid

In the dual grid implementation two different grid resolutions are used for the momentum/pressure grid and the order parameter grid. Since all differential terms related to the interfacial force are calculated on the order parameter grid, a finer grid (twice as fine as the momentum/pressure grid) is used for the order parameter. Subsequently, two simulation steps are taken in the order parameter grid for simulation step in the momentum grid in order to maintain the data in the two grid in synchronization.

The velocity and the pressure are transferred from the momentum/pressure grid to the order parameter grid using bilinear interpolation, and all differential terms are calculated in the finer mesh and then copied over to the corresponding (overlapping) nodes in the momentum grid.

The advantage of using a dual grid implementation is that the results are nearly of the same quality than those obtained from a full refinement of both order parameter and momentum/pressure grid, but with considerable savings in computing time ( 60%) and memory use ( 25%).

## 4.2 Interface Initialization

The interface is initialized using the following profile in the direction normal ot the interface:

$$\phi = \phi^* \tanh\left[\frac{2(r-R)}{W}\right] \qquad (ZSC - LBM) \tag{4.1}$$

$$\rho = \rho^* \tanh\left[\frac{2(r-R)}{W}\right] + \frac{1}{2}(\rho_{\rm L} + \rho_{\rm H}) \qquad (ZSC - LBM) \tag{4.2}$$

with $r$ the distance from any point in the fluid to the center a drop of radius $R$.

To be able to initialize several drops using this interface we only extend this between the center of the drop and $R + W$, so that any drops separated by more than $2W$ lattice units will be correctly initialized as individual units.

## 4.3 Pressure Convergence

A simple way to test when the relaxation of the interface between the two fluids has been achieved is to calculate teh deviation from Laplace's Law, which expresses the pressure difference between the inside and the inside of a drop as a function of the surface tension and the radius,

$$P_{\rm in} - P_{\rm out} = \frac{\sigma}{R} \qquad (2D)\ , \tag{4.3}$$

$$P_{\rm in} - P_{\rm out} = \frac{2\sigma}{R} \qquad (3D)\ . \tag{4.4}$$

In the code we calculate the relative pressure difference error as,

$$P_{\rm err} = (\delta P - \frac{\sigma}{R})\frac{R}{\sigma}\ , \tag{4.5}$$

where $\delta P$ is the calculated pressure difference between the inside of the drop and the outside of the drop ignoring any values at the interface itself.

The error in the pressure difference is calculated every `tStat` iterations, and the difference between errors calculated for consecutive iterations is stored in array `Convergence`,

$$Convergence(i) = P_{\mathrm{err},i} - P_{\mathrm{err},i-1} \tag{4.6}$$

This calculation is done in function `Stats`. After ten calls to `Stats` we calculate the average fluctuation in the pressure difference error by adding all the values in the array and dividing by ten,

$$\varepsilon = 0.1 \sum_{i=1}^{10} Convergence(i) \tag{4.7}$$

The relaxation run is stopped when this average fluctuation of the error is less than the given error limit `pConv`.

## 4.4 External Force

The external force due to gravity is included via the Eötvös number, a non-dimensional quantity that characterizes the ratio of gravitational forces to interfacial forces and given by the expression:

$$Eo = \frac{g\,(\rho_H - \rho_L)\,D^2}{\sigma} \tag{4.8}$$

Where $\sigma$ is the interface tension, $D$ is the characteristic bubble diameter and $\rho_H$ and $\rho_L$ anre the densities of the two fluids. In the code $Eo$ is taken as an input, and the acceleration corresponding to gravity in the lattice is obtained from this expression as:

$$g = \frac{\sigma Eo}{(\rho_H - \rho_L)\, D^2} \tag{4.9}$$

The body force is expressed as $\mathbf{F_b} = (\rho_H - \rho_L)\mathbf{g}$ and introduced in the equations following the details in Section 3.2.

## 4.5 Optimized Version

The optimized version of the code is based on the Zheng, Shu and Chew 3D model. The main differences with respect to the standard ZSC-3D version are:

- Change from Array of Structures (AOS) to Structure of Arrays (SOA)
  - Each velocity direction has its own array
  - Array indexing is in 1D using index $m = i + NX \cdot (j + NY \cdot k) + offset$
- OpenMP is used in the outermost loops
- Added `IVDEP` directive to inner loops in `collision`
- Some rearrangement of operations inside `collision`
- No MPI
- Simpler generation of file names (to be adopted in standard code soon)
- All intermediate IO is controlled by pre-processor flags and can be eliminated

These changes improve performance in shared memory systems with wide vector units, and appear to be very effective in the Intel Xeon Phi coprocessors.

In the future two additional optimized version will be released a hybrid MPI/OpenMP version and a CUDA/MPI version. The standard implementations will always be provided for educational purposes.

# Chapter 5

# Benchmarking

This section presents scaling results for the 3D versions of the code in the newest hardware available to the author.

The first case we present is the scaling of a 240x480x480 domain calculation for the ZSC-3D-MPI code in the Stampede supercomputing cluster at TACC. Each compute node in Stampede consists of dual socket Intel Xeon E5-2680 "Sandy Bridge" processors and 32 GB of RAM, as well as an Intel Xeon Phi SE10P coprocessor with 8GB of memory. Nodes are connected with Mellanox FDR Infiniband in a two-level (core/leaf) fat tree setup. The compiler used was Intel 13.1, the MPI library used was Mvapich2 v1.9a2, and the compilation flags for this test case were simply `-O3 -xAVX`. Results for scaling are shown in Figure 5.1 in terms of MLUPS (millions of lattice updates per second).

The benchmark shows that the scalability of the code is high, with the Zheng-Shu-Chew model maintaining 80% parallel efficiency even at 2048 tasks, when each task is only processing a total of 27,000 grid points. A similar behavior is shown by the LBS3D-MPI code – Figure 5.2 – that when using 2 tasks per node and 8 threads per task achieves a parallel efficiency of over 76% at 2048 cores, and has a throughput that is 8% higher than that of the less optimized ZSC-3D-MPI code.

The optimized `LBS3D` code has been benchmarked on TACC's Stampede and Lonestar Linux clusters. Each compute node on Stampede consists of dual socket Intel Xeon E5-2680 "Sandy Bridge" processors and 32 GB of RAM, as well as an Intel Xeon Phi SE10P with 8GB of memory. Each compute node in Lonestar consists of

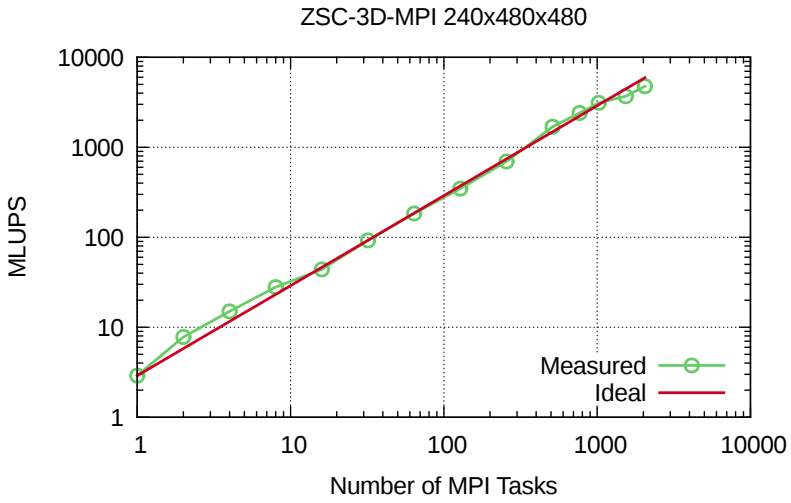



Figure 5.1: Scaling of standard `ZSC-3D-MPI` code.

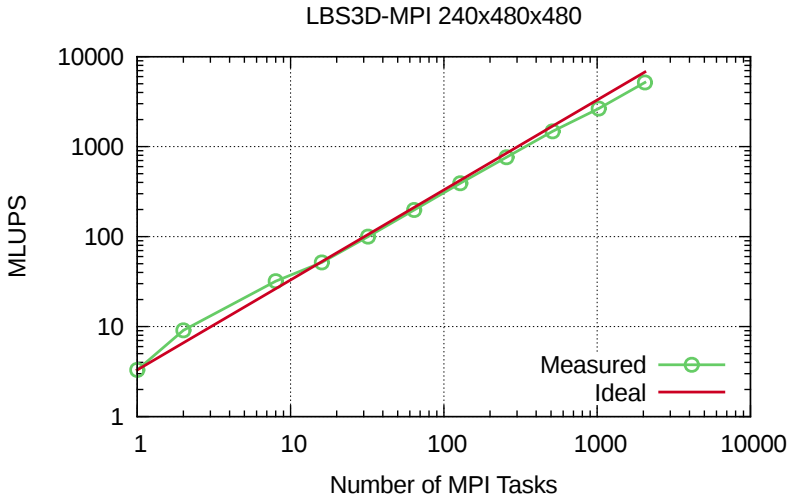



Figure 5.2: Scaling of hybrid `LBS3D-MPI` code.

dual socket Intel Xeon X5680 "Westmere" processors and 24 GB or RAM. In the case of Lonestar the Intel compiler v11.1 and the Mvapich2 1.6 library were used.

The test case used was a 240x240x240 domain, and the the results are shown in Figure 5.3 below.

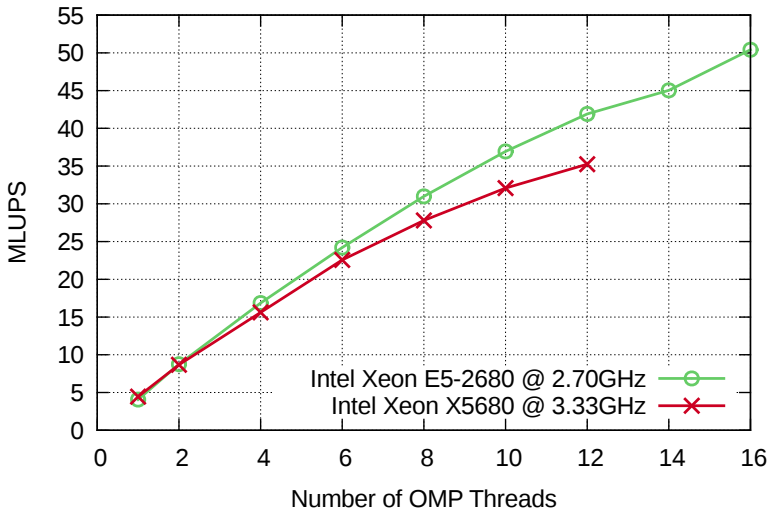



Figure 5.3: Speedup test for LBS3D on two Linux systems.

Tests run on the Intel Xeon Phi coprocessor used the `-openmp -O3 -mmic` compilation flags, and thread binding was specified using the environmental variable `KMP_AFFINITY=balanced,granularity=fine`. The results are shown in Figure 5.4. The rate of performance increase drops when more than one thread per core is used, although the code continues getting faster until all possible threads in the system are in use. Note that the maximum performance achieve on the Xeon Phi coprocessor is more than 2X the maximum performance achieve on two sockets of a regular E5-2680 CPU.

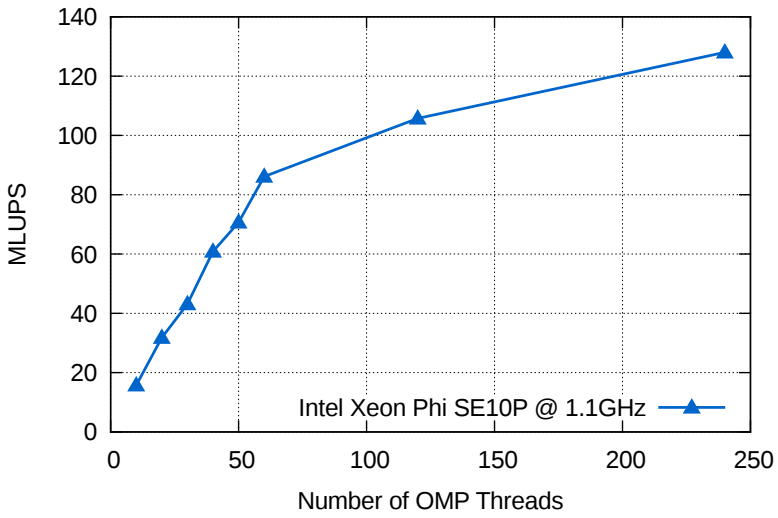



Figure 5.4: Speedup test for LBS3D on the Intel Xeon Phi platform.

# Chapter 6

# Subroutine Listing

| | |
|---|---|
| Directory | : mplabs/src/devel/offload |
| Source Files | : 25 |

| | | |
|---|---|---|
| collision.F90 | packcpu.F90 | relaxstats.F90 |
| collisionmic.F90 | packcpu_f.F90 | stats.F90 |
| common.F90 | packmic.F90 | stream.F90 |
| finalsave.F90 | packmic_f.F90 | streammic.F90 |
| init.F90 | parameters.F90 | updatephi.F90 |
| initmic.F90 | postcollision.F90 | updatephimic.F90 |
| main.F90 | postcollisionmic.F90 | vtksave.F90 |
| Makefile | poststream.F90 | |
| memalloc.F90 | poststreammic.F90 | |

---

| | |
|---|---|
| Directory | : mplabs/src/devel/omp_v1 |
| Source Files | : 14 |

| | | |
|---|---|---|
| collision.F90 | Makefile | relaxstats.F90 |
| common.F90 | memalloc.F90 | stats.F90 |
| finalsave.F90 | parameters.F90 | stream.F90 |
| init.F90 | postcollision.F90 | vtksave.F90 |
| main.F90 | poststream.F90 | |

---

Directory : mplabs/src/devel/omp_v2
Source Files : 14

```
collision.F90  Makefile           relaxstats.F90
common.F90     memalloc.F90       stats.F90
finalsave.F90  parameters.F90     stream.F90
init.F90       postcollision.F90  vtksave.F90
main.F90       poststream.F90
```

---

Directory : mplabs/src/devel/sequential
Source Files : 14

```
collision.F90  Makefile           relaxstats.F90
common.F90     memalloc.F90       stats.F90
finalsave.F90  parameters.F90     stream.F90
init.F90       postcollision.F90  vtksave.F90
main.F90       poststream.F90
```

---

Directory : mplabs/src/devel/soa
Source Files : 14

```
collision.F90  Makefile           relaxstats.F90
common.F90     memalloc.F90       stats.F90
finalsave.F90  parameters.F90     stream.F90
init.F90       postcollision.F90  vtksave.F90
main.F90       poststream.F90
```

---

Directory : mplabs/src/devel/soa_aligned
Source Files : 14

```
collision.F90  Makefile           relaxstats.F90
common.F90     memalloc.F90       stats.F90
finalsave.F90  parameters.F90     stream.F90
init.F90       postcollision.F90  vtksave.F90
main.F90       poststream.F90
```

Directory : mplabs/src/opt/lbs3d
Source Files : 14

```
collision.F90   Makefile            relaxstats.F90
common.F90      memalloc.F90        stats.F90
finalsave.F90   parameters.F90      stream.F90
init.F90        postcollision.F90   vtksave.F90
main.F90        poststream.F90
```

---

Directory : mplabs/src/opt/lbs3d-mpi
Source Files : 15

```
collision.F90   Makefile            relaxstats.F90
common.F90      memalloc.F90        stats.F90
finalsave.F90   parameters.F90      stream.F90
init.F90        postcollision.F90   vgrid.F90
main.F90        poststream.F90      vtksave.F90
```

---

Directory : mplabs/src/std/ll-dgr-mpi
Source Files : 23

```
common.f            mpi_update_f.f      prestream_f.f
finaldump.f         mpi_update_g.f      prestream_g.f
hydrodynamics_f.f   mpi_update_hydro.f  stats.f
hydrodynamics_g.f   mpi_update_rho_f.f  update_f.f
init.f              mpi_update_rho_g.f  update_g.f
main.f              parameters.f        vgrid.f
Makefile            poststream_f.f      vtkplane.f
memalloc.f          poststream_g.f
```

---

Directory : mplabs/src/std/ll-dgr-seq
Source Files : 17

```
common.F90            Makefile          prestream_g.F90
finaldump.F90         memalloc.F90      stats.F90
hydrodynamics_f.F90   parameters.F90    update_f.F90
hydrodynamics_g.F90   poststream_f.F90  update_g.F90
init.F90              poststream_g.F90  vtkplane.F90
main.F90              prestream_f.F90
```

---

Directory : mplabs/src/std/ll-mpi
Source Files : 15

```
common.f          Makefile          poststream.f
finaldump.f       memalloc.f        prestream.f
hydrodynamics.f   mpi_updatefg.f    stats.f
init.f            mpi_updaterho.f   vgrid.f
main.f            parameters.f      vtkplane.f
```

---

Directory : mplabs/src/std/ll-seq
Source Files : 12

```
common.F90          main.F90         poststream.F90
finaldump.F90       Makefile         prestream.F90
hydrodynamics.F90   memalloc.F90     stats.F90
init.F90            parameters.F90   vtkplane.F90
```

---

Directory : mplabs/src/std/zsc-3d-mpi
Source Files : 16

```
collision.f       Makefile          stats.f
collisionrelax.f  memalloc.f        stream.f
common.f          parameters.f      vgrid.f
finaldump.f       postcollision.f   vtkdump.f
init.f            poststream.f
main.f            relaxstats.f
```

Directory : mplabs/src/std/zsc-dgr-mpi
Source Files : 18

| | | |
|---|---|---|
| collision_f.f | main.f | poststream_f.f |
| collision_g.f | Makefile | stats.f |
| common.f | memalloc.f | stream.f |
| differentials.f | parameters.f | update.f |
| finaldump.f | postcollision_f.f | vgrid.f |
| init.f | postcollision_g.f | vtkplane.f |

Directory : mplabs/src/std/zsc-dgr-seq
Source Files : 14

| | | |
|---|---|---|
| collision_f.F90 | init.F90 | stats.F90 |
| collision_g.F90 | main.F90 | stream.F90 |
| common.F90 | Makefile | update.F90 |
| differentials.F90 | memalloc.F90 | vtkplane.F90 |
| finaldump.F90 | parameters.F90 | |

| | |
|---|---|
| Directory | : `mplabs/src/std/zsc-seq` |
| Source Files | : 14 |

```
collision.f   Makefile          stats.f
common.f      memalloc.f        stream.f
finaldump.f   parameters.f      vgrid.f
init.f        postcollision.f   vtkplane.f
main.f        poststream.f
```

---

| | |
|---|---|
| Directory | : mplabs/src/std/zsc-seq |
| Source Files | : 14 |

```
collision.f   Makefile            stats.f
common.f      memalloc.f          stream.f
finaldump.f   mplabs_install.err  vtkplane.f
init.f        mplabs_install.log
main.f        parameters.f
```

---

# Bibliography


[1] Gnuplot is a command-line driven interactive data and function plotting utility distributed as copyrighted but free software. For more information visit the official Gnuplot web site `http://www.gnuplot.info/`.

[2] Paraview is an open source multi-platform visualization application that provides options relevant to the scientific community such as parallel rendering facilities for large data sets. For more information visit the official web site at `http://www.paraview.org/`.

[3] C. Rosales, Porting to the Intel Xeon Phi: Opportunities and Challenges. Extreme Scaling Workshop 2013 (XSCALE13), Boulder, CO, USA (2013).

[4] C. Rosales, D.S. Whyte and M. Cheng, A massively parallel Lattice Boltzmann Method for large density ratios, Proceedings of the 7th Asian CFD Conference, Bangalore, India, 1371 (2007)

[5] C. Rosales and D. S. Whyte, Dual Grid Lattice Boltzmann Method for Multiphase Flows. *International Journal for Numerical Methods in Engineering* 84: 1068–1084 (2010).

[6] The Open MPI project is an open source MPI-2 implementation developed and maintained by a consortium of academic, research, and industry partners. Learn more at `http://www.open-mpi.org/`.

[7] H.W. Zheng, C. Shu and Y.T. Chew, A lattice Boltzmann model for multiphase flows with large density ratio, J. Comput. Phys. 218, 353 (2006)

[8] T. Lee and C.-L. Lin, A stable discretization of the lattice Boltzmann equation for simulation of incompressible two-phase flows at high density ratio, J. Comput. Phys. 206, 16 (2005)

[9] P.L. Bhatnagar, E.P. Gross and M. Krook, A model for collision processes in gases. I. Small amplitude processes in charged and neutral one-component systems, Phys. Rev. 94, 511 (1954)