Computer Physics Communications 217 (2017) 149–161

Contents lists available at ScienceDirect

Computer Physics Communications

journal homepage: www.elsevier.com/locate/cpc

LB3D: A parallel implementation of the Lattice-Boltzmann method for simulation of interacting amphiphilic fluids☆

S. Schmieschek [a], L. Shamardin [b,a], S. Frijters [c], T. Krüger [d,a], U.D. Schiller [e,a], J. Harting [f,c], P.V. Coveney [a,*]

[a] Centre for Computational Science, Department of Chemistry, University College London, 20 Gordon St., London, WC1H 0AJ, United Kingdom
[b] Google UK Ltd, 6 Pancras Square, London N1C 4AG, United Kingdom
[c] Department of Applied Physics, Eindhoven University of Technology, P.O. Box 513, 5600 MB Eindhoven, The Netherlands
[d] School of Engineering, University of Edinburgh, The King's Buildings, Mayfield Road, EH9 3JL Edinburgh, Scotland, United Kingdom
[e] Department of Materials Science and Engineering, Clemson University, 161 Sirrine Hall, Clemson, SC 29634, USA
[f] Forschungszentrum Jülich GmbH, Helmholtz-Institut Erlangen–Nürnberg for Renewable Energy, Fürther Str. 248, 90429 Nürnberg, Germany

ARTICLE INFO

Article history:
Received 10 June 2016
Received in revised form 25 October 2016
Accepted 29 March 2017
Available online 18 April 2017

Keywords:
Lattice-Boltzmann method
High performance computing
Multiphase flow
LBM
LB3D

ABSTRACT

We introduce the lattice-Boltzmann code LB3D, version 7.1. Building on a parallel program and supporting tools which have enabled research utilising high performance computing resources for nearly two decades, LB3D version 7 provides a subset of the research code functionality as an open source project. Here, we describe the theoretical basis of the algorithm as well as computational aspects of the implementation. The software package is validated against simulations of meso-phases resulting from self-assembly in ternary fluid mixtures comprising immiscible and amphiphilic components such as water–oil–surfactant systems. The impact of the surfactant species on the dynamics of spinodal decomposition are tested and quantitative measurement of the permeability of a body centred cubic (BCC) model porous medium for a simple binary mixture is described. Single-core performance and scaling behaviour of the code are reported for simulations on current supercomputer architectures.

Program summary

Program Title: LB3D
Program Files doi: http://dx.doi.org/10.17632/9g9x2wr8z8.1
Licensing provisions: BSD 3-clause
Programming language: FORTRAN90, Python, C
Nature of problem: Solution of the hydrodynamics of single phase, binary immiscible and ternary amphiphilic fluids. Simulation of fluid mixtures comprising miscible and immiscible fluid components as well as amphiphilic species on the mesoscopic scale. Observable phenomena include self-organisation of mesoscopic complex fluid phases and fluid transport in porous media.
Solution method: Lattice-Boltzmann (lattice-Bhatnagar–Gross–Krook, LBGK) [1, 2, 3] method describing fluid dynamics in terms of the single particle velocity distribution function in a 3-dimensional discrete phase space (D3Q19) [4, 5, 6]. Multiphase interactions are modelled using a phenomenological pseudo-potential approach [7, 8] with amphiphilic interactions utilising an additional dipole field [9, 10]. Solid boundaries are modelled using simple bounce-back boundary conditions and additional pseudo-potential wetting interactions [11].
Additional comments including Restrictions and Unusual features: The purpose of the release is the provision of a refactored minimal version of LB3D suitable as a starting point for the integration of additional features building on the parallel computation and IO functionality.

[1] S. Succi, The Lattice Boltzmann Equation: For Fluid Dynamics and Beyond, Oxford University Press, 2001.
[2] B. Dünweg, A. Ladd, Lattice Boltzmann simulations of soft matter systems, Adv. Poly. Sci. 221 (2009) 89–166

☆ This paper and its associated computer program are available via the Computer Physics Communication homepage on ScienceDirect (http://www.sciencedirect.com/science/journal/00104655).

* Corresponding author.
E-mail address: p.v.coveney@ucl.ac.uk (P.V. Coveney).

http://dx.doi.org/10.1016/j.cpc.2017.03.013
0010-4655/© 2017 The Authors. Published by Elsevier B.V. This is an open access article under the CC BY license (http://creativecommons.org/licenses/by/4.0/).

[3] C. K. Aidun, J. R. Clausen, Lattice-Boltzmann Method for Complex Flows, Annual Review of Fluid Mechanics 42 (2010) 439.
[4] X. He, L.-S. Luo, A priori derivation of the lattice-Boltzmann equation, Phys. Rev. E 55 (1997) R6333.
[5] X. He, L.-S. Luo, Theory of the lattice Boltzmann method: from the Boltzmann equation to the lattice Boltzmann equation, Phys. Rev. E 56.
[6] Y. H. Qian, D. D'Humiéres, P. Lallemand, Lattice BGK Models for Navier–Stokes Equation, Europhysics Letters 17 (1992) 479.
[7] X. Shan, H. Chen, Lattice-Boltzmann model for simulating flows with multiple phases and components, Physical Review E 47 (1993) 1815.
[8] X. Shan, G. Doolen, Multicomponent lattice-Boltzmann model with interparticle interaction, Journal of Statistical Physics 81 (1995) 379.
[9] H. Chen, B. Boghosian, P.V. Coveney, M. Nekovee, A ternary lattice-Boltzmann model for amphiphilic fluids, Proceedings of the Royal Society of London A 456 (2000) 2043.
[10] M. Nekovee, P. V. Coveney, H. Chen, B. M. Boghosian, Lattice-Boltzmann model for interacting amphiphilic fluids, Phys. Rev. E 62 (2000) 8282.
[11] N. S. Martys, H. Chen, Simulation of multicomponent fluids in complex three-dimensional geometries by the lattice-Boltzmann method, Phys. Rev. E 53 (1996) 743.

© 2017 The Authors. Published by Elsevier B.V. This is an open access article under the CC BY license (http://creativecommons.org/licenses/by/4.0/).

1. Introduction

Since its advent almost 30 years ago, the lattice-Boltzmann method (LBM) has gained increasing popularity as a means for the simulation of fluid dynamics. Driving factors for this development are the relative simplicity and locality of the lattice-Boltzmann algorithm. The latter allows for straightforward parallelisation of the method and its application in high performance computing. Today, a wide range of LBM implementations is available ranging from specialised academic packages such as Ludwig for the simulation of complex fluids [1] and HemeLB for the simulation of flow in blood vessels [2] to very versatile open-source projects such as OpenLB [3] and Palabos [4] as well as commercial applications such as PowerFlow [5] and XFlow [6].

The lattice-Boltzmann (LB) code LB3D provides a number of algorithms and scripts designed for the simulation of binary and ternary amphiphilic complex fluid mixtures in bulk and complex geometries using high performance computing environments. As in the case of Ludwig, LB3D focuses on the simulation of complex fluids. While Ludwig implements a top-down model where a free energy has to be provided by the user, LB3D is a bottom-up code in which interactions are specified between particle distributions. LB3D is the only package which handles amphiphilic fluids in such a manner. Originally implemented in 1999, LB3D has since been in constant use and development. This document starts out with an overview of the history of the development and scientific applications of the code. Following this, the paper considers LB3D in version 7.1, the latest release of a re-factored open-source instance of the program which has been development since 2013 and available in this version under the BSD 3-Clause license.

Following work on lattice-gas models [7,8], the development of LB3D started out in the group of Coveney then at Queen Mary University of London, as a summer project of then third year University of Oxford undergraduate student Chin who implemented the amphiphilic fluid model with assistance of Boghosian and Nekovee [9,10]. During the first decade of development and application, the amphiphilic fluid model was employed for extensive research on the properties of complex fluids. Publications on computational science developments in this period report aspects of distributed computing as an integral part of the design, including real time visualisation and computational steering [11–15]. Scientific contributions included the study of the spinodal decomposition and emulsion formation of binary fluid mixtures [16–19] as well as the self-organisation of mesoscopic phases in general and the cubic gyroid phase in particular [20–22]. Flow of complex fluids in porous media and under shear was investigated [23]. The utilisation of substantial resources like the US TeraGrid and UK HPC facilities in ground breaking federation of national grid infrastructures [12–14] and the highest performance class of national supercomputers for capability computing [24,25], allowed scientists working with LB3D large-scale numerical investigation of rheological properties of the gyroid phase [26–29]. Later work explored the parameters of ternary amphiphilic fluids and their flow properties in detail [30–34].

Subsequent research with version 6 of LB3D, which remains unreleased, has focused on fluid surface interaction and coupling of particle models. Version 7.1 of LB3D, however, which is released in conjunction with this paper, includes the amphiphilic fluid model and has been restructured and optimised to ease future development. First steps have been taken to support hybrid parallel programming models in order to ensure compatibility with projected next generation supercomputer architectures on the path to the exascale.

A newly integrated Python interface for the configuration of initial conditions and interactive execution improves ease of use. Moreover, improved in-code documentation and extension of the user manual are intended to encourage new users and developers to harness the potential of LB3D as well to as enhance it further.

The paper has the following structure. In Section 2 we describe the lattice-Boltzmann method (LBM) used. We do not provide the derivation of this model, but show the key points necessary for understanding the implementation; references are provided to the relevant original papers. In Section 3 we discuss implementation details such as memory organisation, communication and scripting capabilities. In Section 4 we discuss tests used to verify and validate the implementation. We also provide performance benchmarks for the code and compare performance and scaling with other LBM codes. The section closes with a conclusion and outlook on future development directions.

2. Method

In this section we give an overview of the single phase lattice-Boltzmann method as well as its multiphase extension and boundary conditions as implemented in LB3D.

2.1. Single phase lattice-Boltzmann

The lattice-Boltzmann equation is a popular method to simulate flows and hydrodynamic interactions in incompressible fluids [35–37]. It is a mesoscopic approach where the fluid is represented by populations that evolve according to a fully discrete analogue of the Boltzmann kinetic equation [38,39]. We write the lattice-Boltzmann equation in the following form

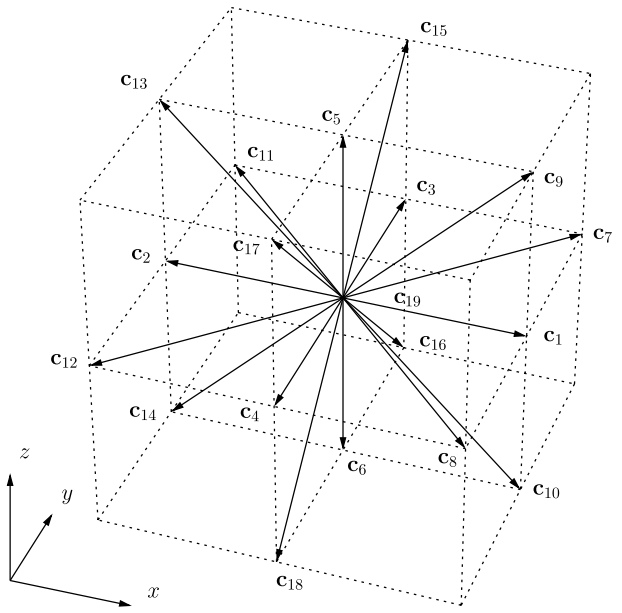

Fig. 1. The geometry of D3Q19 lattice vectors $\mathbf{c}_i$.

$$f^*_{\sigma,i}(\mathbf{x},t) = f_{\sigma,i}(\mathbf{x},t) - \frac{h}{\tau_\sigma}\left[f_{\sigma,i}(\mathbf{x},t) - f_i^{\mathrm{eq}}\left(n_\sigma(\mathbf{x},t), \mathbf{u}(\mathbf{x},t) + \frac{\tau_\sigma}{m_\sigma n_\sigma}\mathbf{F}_\sigma(\mathbf{x},t)\right)\right] \tag{1}$$

where the pre-collisional populations $f_{\sigma,i}$ and post-collisional populations $f^*_{\sigma,i}$ are related through the streaming step

$$f_{\sigma,i}(\mathbf{x},t) = f^*_{\sigma,i}(\mathbf{x} - h\mathbf{c}_i, t-h). \tag{2}$$

Eq. (1) describes a collision step and neglects terms of order $O(h^2/\tau)$ compared to the more common second-order accurate LBE [38–41]; however, this does not affect the density and flow field which are of primary interest. The populations $f_{\sigma,i}(\mathbf{x},t)$ and $f^*_{\sigma,i}(\mathbf{x},t)$ represent number densities of a fluid species σ at discrete grid points $\mathbf{x}$ and discrete time t, moving with discrete velocities $\mathbf{c}_i$. LB3D uses the so-called D3Q19 lattice, a three-dimensional Cartesian lattice with 19 velocities [42] obtained from a projected face centred hyper-cubic (FCHC) lattice. The lattice velocities $\mathbf{c}_i$ can be derived systematically as the abscissae of a Gauss–Hermite quadrature in velocity space [40,43,44]. The D3Q19 lattice is illustrated in Fig. 1 along with the (arbitrary) numbering of the velocities $\mathbf{c}_i$ implemented in LB3D.

Eq. (1) describes the lattice-Bhatnagar–Gross–Krook (LBGK) collision model where the populations relax towards local equilibrium on a single time scale τ_σ [42,45]. The equilibrium distribution $f_i^{\mathrm{eq}}(n,\mathbf{u})$ is a Hermite expansion of the Maxwell–Boltzmann distribution [38,46]. Here we choose an expansion including cubic terms in the velocity

$$f_i^{\mathrm{eq}}(n,\mathbf{u}) = w_i n\left(1 + \frac{c_{i\alpha}u_\alpha}{c_s^2} + \frac{Q_{i\alpha\beta}u_\alpha u_\beta}{2c_s^4} + \frac{W_{i\alpha\beta\gamma}u_\alpha u_\beta u_\gamma}{6c_s^6}\right), \tag{3}$$

where c_s denotes the speed of sound, w_i are lattice weights, and $Q_{i\alpha\beta}$ and $W_{i\alpha\beta\gamma}$ are the second and third rank isotropic tensors

$$Q_{i\alpha\beta} = c_{i\alpha}c_{i\beta} - c_s^2\delta_{\alpha\beta} \tag{4}$$

$$W_{i\alpha\beta\gamma} = c_{i\alpha}c_{i\beta}c_{i\gamma} - c_s^2 c_{i\alpha}\delta_{\beta\gamma} - c_s^2 c_{i\beta}\delta_{\gamma\alpha} - c_s^2 c_{i\gamma}\delta_{\alpha\beta}. \tag{5}$$

Note that the Einstein convention for summation over Greek indices is implied. For the D3Q19 lattice, the speed of sound is $c_s = 1/\sqrt{3}$ and the lattice weights w_i are [42]

$$\begin{aligned} w_i &= 1/3, \quad |\mathbf{c}_i| = 0,\\ w_i &= 1/18, \quad |\mathbf{c}_i| = 1,\\ w_i &= 1/36, \quad |\mathbf{c}_i| = \sqrt{2}. \end{aligned} \tag{6}$$

Hydrodynamic fields are obtained from the populations by calculating velocity moments, *e.g.*, density and momentum density are given by

$$\rho_\sigma(\mathbf{x},t) = m_\sigma \sum_i f^*_{\sigma,i}(\mathbf{x},t), \tag{7}$$

$$\tilde{\mathbf{p}}_\sigma(\mathbf{x},t) = m_\sigma \sum_i f^*_{\sigma,i}(\mathbf{x},t)\mathbf{c}_i, \tag{8}$$

where m_σ is the molecular mass. In the presence of a force density $\mathbf{F}_\sigma(\mathbf{x},t)$, the lattice momentum density $\tilde{\mathbf{p}}_\sigma(\mathbf{x},t)$ has to be corrected for a discretisation effect, such that the hydrodynamic momentum density of the fluid is [36,40,41,47]

$$\mathbf{p}_\sigma(\mathbf{x},t) = \tilde{\mathbf{p}}_\sigma(\mathbf{x},t) - \frac{h}{2}\mathbf{F}_\sigma(\mathbf{x},t). \tag{9}$$

Here the sign stems from the fact that we calculate the moments from the post-collisional distributions. Finally, the hydrodynamic flow field is

$$\mathbf{u}(\mathbf{x},t) = \frac{\mathbf{p}_\sigma(\mathbf{x},t)}{\rho_\sigma(\mathbf{x},t)} \tag{10}$$

which is used in the lattice-Boltzmann equation (1).

Generally the force density $\mathbf{F}_\sigma(\mathbf{x},t)$ may accommodate any kind of contribution. In particular, $\mathbf{F}_\sigma(\mathbf{x},t)$ may include both intermolecular forces (Shan–Chen forces in our model, *cf.* Section 2.2) and external forces such as gravity or an artificial Kolmogorov scaled force [48].

2.2. Shan–Chen multiphase model

The Shan–Chen approach [49] provides a straightforward way to model multi-component (*e.g.* water and oil) and/or multi-phase fluids (*e.g.* water and water vapour). The single phase model is extended by the so-called pseudo-potential method to include multiphase interactions [50,51]. Each fluid component σ is governed by the lattice-Boltzmann equation (1).

The total fluid density is simply

$$\rho(\mathbf{x},t) = \sum_\sigma \rho_\sigma(\mathbf{x},t). \tag{11}$$

It is assumed that there is a common velocity $\tilde{\mathbf{u}}$ for all components. In the absence of forces, it can be shown that total momentum is conserved if the common velocity is chosen to be [49]

$$\tilde{\mathbf{u}}(\mathbf{x},t) = \frac{\sum_\sigma \tilde{\mathbf{p}}_\sigma(\mathbf{x},t)/\tau_\sigma}{\sum_\sigma \rho_\sigma(\mathbf{x},t)/\tau_\sigma}. \tag{12}$$

Note that the common velocity $\tilde{\mathbf{u}}(\mathbf{x},t)$ implies an effective momentum exchange between the species even when no explicit forces are prescribed.

In addition, each fluid component may be subject to an explicit force density $\mathbf{F}_\sigma(\mathbf{x},t)$. Similar to the single phase case, we need to redefine the hydrodynamic momentum to account for discretisation effects. The hydrodynamic flow field for the multiphase fluid is then given by [49]

$$\mathbf{u}(\mathbf{x},t) = \frac{1}{\rho(\mathbf{x},t)}\sum_\sigma\left(\tilde{\mathbf{p}}_\sigma(\mathbf{x},t) - \frac{h}{2}\mathbf{F}_\sigma(\mathbf{x},t)\right). \tag{13}$$

If the fluid as a whole is subject to a force density $\mathbf{F}(\mathbf{x}, t)$, it has to be distributed to each fluid component proportionally to the mass fraction [52]:

$$\mathbf{F}_\sigma(\mathbf{x}, t) = \frac{\rho_\sigma(\mathbf{x}, t)}{\rho(\mathbf{x}, t)}\mathbf{F}(\mathbf{x}, t). \tag{14}$$

Eq. (14) ensures that the total force density obeys $\mathbf{F}(\mathbf{x}, t) = \sum_\sigma \mathbf{F}_\sigma(\mathbf{x}, t)$ and that the acceleration for each component is identical: $\mathbf{a}_\sigma(\mathbf{x}, t) = \mathbf{a}(\mathbf{x}, t) = \mathbf{F}(\mathbf{x}, t)/\rho(\mathbf{x}, t)$.

The basic idea of the Shan–Chen approach for multiphase fluids is to introduce coupling forces

$$\mathbf{F}^{\text{int}}_\sigma(\mathbf{x}, t) = -\psi_\sigma(\mathbf{x}, t)\sum_{\mathbf{x}'}\sum_{\sigma'} G_{\sigma\sigma'}(\mathbf{x}, \mathbf{x}')\psi_{\sigma'}(\mathbf{x}', t)\left(\mathbf{x}' - \mathbf{x}\right) \tag{15}$$

which are non-local and depend on density gradients. Here, $\psi_\sigma(\mathbf{x}, t)$ represents an effective mass of component σ which is realised as a function of component density $\rho_\sigma(\mathbf{x}, t)$ [49]

$$\psi_\sigma(\mathbf{x}, t) = \psi(\rho_\sigma(\mathbf{x}, t)) = \rho_0\left[1 - \exp\left(-\rho_\sigma(\mathbf{x}, t)/\rho_0\right)\right], \tag{16}$$

where ρ_0 is a reference density. The Green's function $G_{\sigma\sigma'}(\mathbf{x}, \mathbf{x}')$ must be symmetric in σ and $\mathbf{x}$ to ensure global momentum conservation

$$G_{\sigma\sigma'}(\mathbf{x}, \mathbf{x}') = G_{\sigma'\sigma}(\mathbf{x}, \mathbf{x}') = G_{\sigma\sigma'}(\mathbf{x}', \mathbf{x}). \tag{17}$$

We choose $G_{\sigma\sigma'}(\mathbf{x}, \mathbf{x}')$ to be short-ranged and allow interaction only between neighbouring lattice sites [52]

$$G_{\sigma\sigma'}(\mathbf{x}, \mathbf{x}') = \begin{cases} 2g_{\sigma\sigma'} & \text{if } |\mathbf{x}' - \mathbf{x}| = 1, \\ g_{\sigma\sigma'} & \text{if } |\mathbf{x}' - \mathbf{x}| = \sqrt{2}, \\ 0 & \text{otherwise.} \end{cases} \tag{18}$$

We can now rewrite Eq. (15) as

$$\mathbf{F}^{\text{int}}_\sigma(\mathbf{x}, t) = -\psi(\rho_\sigma(\mathbf{x}, t))\sum_i\sum_{\sigma'} G_{\sigma\sigma',i}\psi(\rho_{\sigma'}(\mathbf{x} + \mathbf{c}_i, t))\mathbf{c}_i \tag{19}$$

and Eq. (18) as

$$G_{\sigma\sigma',i} = \begin{cases} 0 & |\mathbf{c}_i| = 0 \\ 2g_{\sigma\sigma'}, & |\mathbf{c}_i| = 1, \\ g_{\sigma\sigma'} & |\mathbf{c}_i| = \sqrt{2}. \end{cases} \tag{20}$$

Note that $g_{\sigma\sigma'}$ is the coupling strength of components σ and σ'. The number of components is not limited by the Shan–Chen model. Also note that non-zero values of $g_{\sigma\sigma}$ allow for self-interaction of component σ. The pseudo-potential forces introduce a density-gradient-dependent term into the equation of states which ensures phase separation for a given critical coupling strength and overall density. Masses are determined up to a constant allowing the use of unit mass where convenient. Mass contrasts of components imply a contrast in dynamic viscosity and warrant the introduction of the effective common equilibrium velocity (12). No differences in mass have been used so far in our published work while, in the low Mach number limit, the common velocity equals the single component bulk velocity.

2.3. Amphiphilic fluid components

LB3D supports fluid mixtures of up to three components, one of which is amphiphilic (*oil* or *red*, *water* or *blue* and *surfactant* or *green*). Amphiphilic properties of the surfactant component are modelled by introduction of a dipole field $\mathbf{d}(\mathbf{x}, t)$ representing the average molecule orientation at each lattice site [10]. The orientation of the dipole vector $\mathbf{d}$ is allowed to vary continuously. It is advected with the fluid and thus propagates according to

$$f_\sigma(\mathbf{x}, t + h)\mathbf{d}(\mathbf{x}, t + h) = \sum_i f^*_{\sigma,i}(\mathbf{x} - h\mathbf{c}_i, t)\mathbf{d}^*(\mathbf{x} - h\mathbf{c}_i, t), \tag{21}$$

where $\mathbf{d}^*$ denotes the post-collisional dipole vector which is relaxed through a BGK process on a single timescale

$$\mathbf{d}^*(\mathbf{x}, t) = \mathbf{d}(\mathbf{x}, t) - \frac{h}{\tau_d}\left[\mathbf{d}(\mathbf{x}, t) - \mathbf{d}^{\text{eq}}(\mathbf{x}, t)\right]. \tag{22}$$

The equilibrium orientation is derived from a mean field approach. The mean dipole field in the surrounding fluid can be written as

$$\mathbf{b}(\mathbf{x}, t) = \mathbf{b}^\alpha(\mathbf{x}, t) + \mathbf{b}^\sigma(\mathbf{x}, t), \tag{23}$$

where the contribution from ordinary species is

$$\mathbf{b}^\sigma(\mathbf{x}, t) = \sum_\sigma q_\sigma \sum_i f_{\sigma,i}(\mathbf{x}, t)\mathbf{c}_i. \tag{24}$$

Here q_σ is a colour charge for the ordinary species. The contribution from the amphiphilic species can be written as

$$\mathbf{b}^\alpha(\mathbf{x}, t) = \sum_\alpha \left[\sum_{i\neq 0} f_{\alpha,i}(\mathbf{x} + h\mathbf{c}_i, t)\mathsf{D}_i \cdot \mathbf{d}(\mathbf{x} + h\mathbf{c}_i, t) + f_{\alpha,i}(\mathbf{x}, t)\mathbf{d}(\mathbf{x}, t)\right], \tag{25}$$

where the traceless second rank tensor D_i is given by

$$\mathsf{D}_{i\alpha\beta} = \delta_{\alpha\beta} - 3\frac{c_{i\alpha}c_{i\beta}}{\mathbf{c}_i^2}. \tag{26}$$

The equilibrium dipole configuration can then be derived from a Boltzmann distribution which gives

$$\mathbf{d}^{\text{eq}} = d_0\left[\coth(\beta b) - \frac{1}{\beta b}\right]\hat{\mathbf{b}}, \tag{27}$$

where $b = |\mathbf{b}(\mathbf{x}, t)|$ and $\hat{\mathbf{b}} = \mathbf{b}(\mathbf{x}, t)/b$. β is the inverse temperature, and d_0 is a parameter representing the intrinsic dipole strength of the amphiphiles.

With an amphiphilic species present, additional force components are introduced between the species. In addition to the density dependence of the binary model, the resulting forces do not only depend on the fluid densities alone, but on the dipole moment (with the relative orientation to boundaries and neighbouring dipoles) as well. The force on oil and water components σ is now

$$\mathbf{F}_\sigma(\mathbf{x}, t) = \mathbf{F}^{\text{int}}_\sigma(\mathbf{x}, t) + \mathbf{F}^{\sigma\alpha}(\mathbf{x}, t), \tag{28}$$

where $\mathbf{F}^{\text{int}}_\sigma$ is given in Eq. (19), and the additional force due to the amphiphiles is

$$\mathbf{F}^{\sigma\alpha}(\mathbf{x}, t) = -2g_{\alpha\sigma}\psi^\alpha(\mathbf{x}, t)\sum_{i\neq 0}\mathbf{d}(\mathbf{x} + \mathbf{c}_i h, t) \cdot \mathsf{D}_i\psi^\sigma(\mathbf{x} + \mathbf{c}_i h, t). \tag{29}$$

Similarly, the force on the amphiphilic components is

$$\mathbf{F}^\alpha(\mathbf{x}, t) = \mathbf{F}^{\alpha\sigma}(\mathbf{x}, t) + \mathbf{F}^{\alpha\alpha}(\mathbf{x}, t), \tag{30}$$

where $\mathbf{F}^{\alpha\sigma}$ is the reaction force to $\mathbf{F}^{\sigma\alpha}$ and has the form

$$\mathbf{F}^{\alpha\sigma}(\mathbf{x}, t) = 2\psi^\sigma(\mathbf{x}, t)\mathbf{d}(\mathbf{x}, t) \cdot \sum_\alpha g_{\alpha\sigma}\sum_{i\neq 0}\mathsf{D}_i\psi^\alpha(\mathbf{x} + \mathbf{c}_i h, t). \tag{31}$$

Finally, the force acting between surfactant components on neighbouring sites is given by

$$\mathbf{F}^{\alpha\alpha}(\mathbf{x},t) = -\frac{12}{c^2}\psi^{\alpha}(\mathbf{x},t)\, g_{\alpha\alpha}\sum_{i\neq 0}\psi^{\alpha}(\mathbf{x}+\mathbf{c}_i h,t) \cdot\Big(\mathbf{d}(\mathbf{x}+\mathbf{c}_i h,t)\cdot \mathsf{D}_i\mathbf{d}(\mathbf{x},t)\,\mathbf{c}_i + \big[\mathbf{d}(\mathbf{x}+\mathbf{c}_i h,t)\,\mathbf{d}(\mathbf{x},t) + \mathbf{d}(\mathbf{x},t)\,\mathbf{d}(\mathbf{x}+\mathbf{c}_i h,t)\big]\cdot\mathbf{c}_i\Big). \tag{32}$$

The parameters $g_{\alpha\sigma}$, $g_{\sigma\alpha}$, and $g_{\alpha\alpha}$ in Eqs. (29), (31), and (32) are the coupling strength between water and oil components σ and amphiphilic component α, respectively. Details on the derivation of the force terms can be found in [10,51]. These forces are added in the algorithm in a manner analogous to the pseudo-potential force inducing the phase transition in the binary multi-component system.

2.4. Boundary conditions

In the present lattice-Boltzmann model, each lattice site contains either fluid components or an obstacle, e.g., a boundary wall. LB3D v7.1 supports a number of different boundary closures for the unknown pre-collisional populations on fluid nodes that are adjacent to one or more boundary nodes. In the following we describe the simple bounce-back rule and on-site rules for Dirichlet and Neumann boundary conditions that enable inlet and outlet boundary conditions. Furthermore, we describe wetting boundary conditions for surfaces that have specific affinities with respect to different fluid species.

2.4.1. Bounce-back boundary conditions

The bounce-back boundary condition was originally introduced for lattice-gas models and poses a simple way to implement a no-slip boundary condition located approximately halfway between a boundary node and an adjacent fluid site [53–55]. Simple mid-grid boundary conditions achieve zero velocity on a link connecting fluid and an obstacle by inverting the velocity of the impinging populations. This reflection can be written as a modified streaming step, cf. Eq. (2),

$$f_{\sigma,i}(\mathbf{x},t) = f^{*}_{\sigma,\bar{i}}(\mathbf{x},t-h), \tag{33}$$

where $\mathbf{x}+h\mathbf{c}_i$ is a solid site, and the index $\bar{i}$ indicates the inverse velocity of i, *i.e.*, $\mathbf{c}_{\bar{i}} = -\mathbf{c}_i$. For the simple bounce-back rule, the effective location of the no-slip boundary condition is approximately halfway between fluid and solid. A detailed analysis shows that the exact location depends on the collision model employed, and for the LBGK model the location is viscosity dependent [56–58].

2.4.2. On-site velocity boundary conditions

It is often desirable to specify the exact position where Dirichlet or Neumann boundary conditions are to be satisfied independently of other simulation parameters. To this end, on-site boundary conditions allow us to specify the velocity or fluxes on the boundary and lead to a fully local closure relation. This approach was originally suggested by Zou and He [59] for D2Q9 and D3Q15 models, and later extended to D3Q19 lattices [60–62].

For on-site Neumann (or pressure) boundary conditions, one uses Eqs. (7) and (8) to calculate the unknown velocity component on the boundary from the two known ones and a specified density ρ_0. Similarly, for on-site Dirichlet (or flux) boundary conditions, one can calculate the density ρ from the two known velocity components and the third specified component. In order to determine the unknown populations, one applies the bounce-back condition

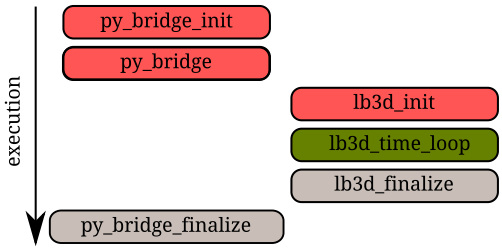

Fig. 2. Overview of the subroutine calls in the program set in standard face LB3D. Principal algorithm execution is divided into three phases, initialisation, main time loop and finalisation. Initialisation comprises the py_bridge portion of the code, which is implemented in C and allows definition of LB3D initialisation parameters via a Python interface. Colour online.

to the non-equilibrium part of the populations, *e.g.*, for a boundary normal to the *xy*-plane,

$$f_{\sigma,5} - f_5^{\mathrm{eq}} = f_{\sigma,6} - f_6^{\mathrm{eq}} \tag{34}$$

which leads to an expression for the unknown population

$$f_{\sigma,5} = f_{\sigma,6} + f_5^{\mathrm{eq}} - f_6^{\mathrm{eq}} = f_{\sigma,6} + \frac{2w_1 a}{c_s^2 h} f_\sigma u_{\sigma,z} + \mathrm{O}(u^3). \tag{35}$$

The equation system for the remaining unknown populations is over-determined, which can be remedied by introducing additional transverse momentum corrections that reflect the stresses introduced by the boundary conditions. The equation system for the remaining populations then reads

$$f_{\sigma,9} = f_{\sigma,14} + \frac{2w_2 a}{c_s^2 h} n_\sigma \left(u_{\sigma,z} + u_{\sigma,x}\right) - N_\sigma^{zx}, \tag{36}$$

$$f_{\sigma,13} = f_{\sigma,10} + \frac{2w_2 a}{c_s^2 h} n_\sigma \left(u_{\sigma,z} - u_{\sigma,x}\right) + N_\sigma^{zx}, \tag{37}$$

$$f_{\sigma,15} = f_{\sigma,18} + \frac{2w_2 a}{c_s^2 h} n_\sigma \left(u_{\sigma,z} + u_{\sigma,y}\right) - N_\sigma^{zy}, \tag{38}$$

$$f_{\sigma,17} = f_{\sigma,16} + \frac{2w_2 a}{c_s^2 h} n_\sigma \left(u_{\sigma,z} - u_{\sigma,y}\right) + N_\sigma^{zy}, \tag{39}$$

where the transverse momentum corrections are given by

$$N_\sigma^{zx} = \frac{1}{2}\left[f_{\sigma,1} + f_{\sigma,7} + f_{\sigma,8} - f_{\sigma,2} - f_{\sigma,11} - f_{\sigma,12}\right] - \frac{4w_2 a}{c_s^2 h} f_\sigma u_{\sigma,x}, \tag{40}$$

$$N_\sigma^{zy} = \frac{1}{2}\left[f_{\sigma,3} + f_{\sigma,7} + f_{\sigma,11} - f_{\sigma,4} - f_{\sigma,8} - f_{\sigma,12}\right] - \frac{4w_2 a}{c_s^2 h} f_\sigma u_{\sigma,y}. \tag{41}$$

These expressions close the boundary equations. More details and the generalisation to arbitrary flow directions can be found in [62].

2.4.3. Wetting boundary conditions

Specific surface interactions can be realised by introducing a *pseudo-density* ψ_W at obstacle sites $\mathbf{x}'$, which is used to calculate a pseudo-potential interaction between the fluid and components and the surface

$$\mathbf{F}_\sigma^{\mathrm{int}}(\mathbf{x},t) = -\psi_\sigma(\mathbf{x},t)\sum_{\mathbf{x}'\in W} G_{\sigma,\mathrm{wall}}(\mathbf{x},\mathbf{x}')\,\psi_W. \tag{42}$$

This strategy was first introduced by Martys and Chen [52]. The wetting boundary conditions are usually augmented with bounce-back or on-site boundary conditions.

3. Implementation

The core of the lattice-Boltzmann code LB3D is written in FORTRAN 90/95. In addition, version 7.1 provides conduits written in

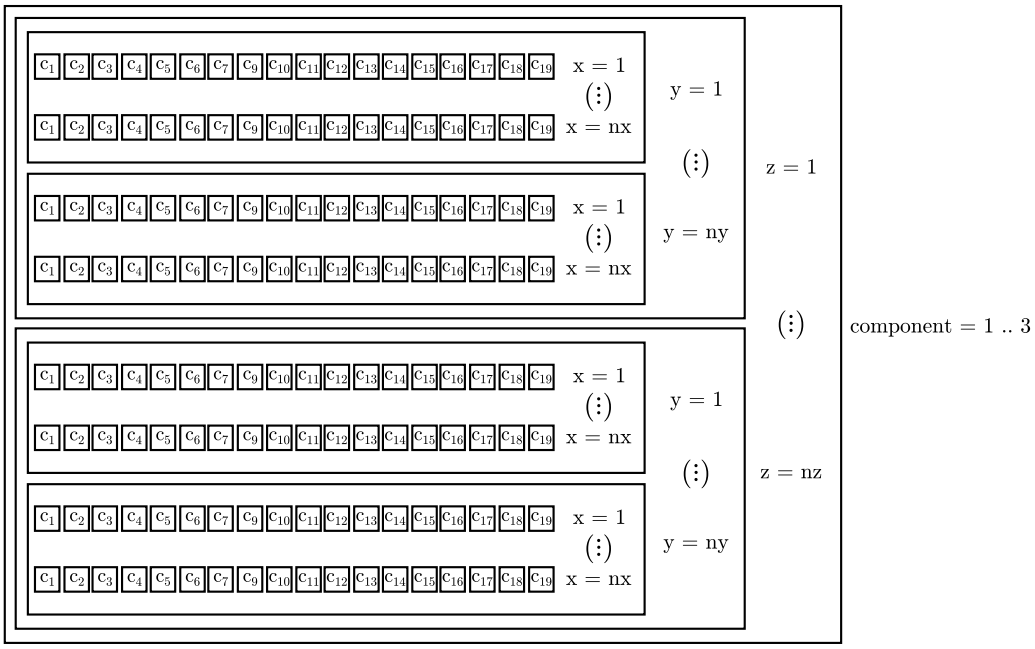

Fig. 3. Illustration of the layout of the 5-dimensional column major order FORTRAN array storing lattice information per component. Lattice index i changes fastest, and component index slowest.

C to facilitate a Python interface that supports scripting during the initialisation stage. The core code is parallelised using MPI-2 for distributed memory and OpenMP for shared memory. It can be compiled and run using arbitrary combinations of MPI and OpenMP threads. LB3D makes use of several external libraries including Python, MPI, OpenMP, HDF5, and XDR. The testing framework and supplementary tools are written in Python.

The execution flow of LB3D is structured in self-contained initialisation, simulation, and finalisation stages. The corresponding subroutine call structure is illustrated in Fig. 2. In the initialisation stage, the code reads an *input file* describing the simulation setup (number of components, initial conditions, simulation time, output options) as well as the physical parameters such as τ_σ, $g_{\sigma\sigma'}$, etc. The input file can also specify Python scripts to be executed during initialisation in order to introduce boundary conditions and to modify the initial fluid state (cf. Section 3.7). The simulation stage evaluates the lattice-Boltzmann equation until the specified simulation time is reached. Each time step involves several substeps for the lattice-Boltzmann algorithm, data caching, parallel communication, and file I/O. After completion of the simulation stage, the finalisation stage shuts down the MPI environment, deallocates memory, and polls and prints the execution timers before the program terminates successfully.

We now outline the essential elements of the implementation in more detail which serves to provide a development guide for future extensions.

3.1. Data structures

The core data structure of the implementation is a five-dimensional array of fluid populations $f_{\sigma,i}(x, y, z)$ for the current time step with index order i, x, y, z and σ. The layout is illustrated in Fig. 3. FORTRAN column major order implies indices varying fastest from left to right. We evaluated different memory layouts for the populations array and found this order to be most effective for the present algorithm implementation. A five-dimensional array is used to store forces $\mathbf{F}_\sigma(x, y, z, t)$ with index order α, x, y, z, σ (α denoting the vector component). Information on solid boundary conditions is stored in a three-dimensional array of obstacle flags with indices x, y, z. In addition, a four-dimensional array for cached fluid densities $f_\sigma(x, y, z)$ with indices x, y, z and σ is used.

3.2. System initialisation

System initialisation can be performed either from scratch using a set of input options or by re-initialisation from checkpoint files in HDF5 format. The initialisation subroutine first reads any command line arguments including the path to a set of input files. The parameter input file is expected in .ini file format and processed by the module fini_parser. It is structured in sections for the *simulation* environment, *common* simulation parameters, *physics* parameters and *output* properties. Settings for special *boundary conditions* and *checkpoints* are given in separate sections. A unique feature of LB3D is the ability to adjust the system initialisation by providing a Python script which can modify density values for the respective fluid components as well as boundary geometries and wetting interaction parameters. Example input files are given in Figs. 4 and 5. Detailed explanations of all available input parameters can be found in the LB3D user manual.

During initialisation, the MPI parallel environment is set up, MPI and HDF5 data structures are instantiated, the random number generators are seeded, and general variable defaults are set. The simulation parameters are then set to the values specified in the input file, where a set of compatibility checks is executed to catch conflicting options and parametrisations. Subsequently, the lattice data structures are allocated and initialised with a velocity field and obstacle site distribution, where the LB3D Python interface is invoked to execute the specified scripts. Moreover, additional properties such as specific interaction forces as well as in- and outflow boundary conditions are initialised. If restoration from a checkpoint is requested, simulation parameters and lattice data are read from the specified HDF5 files. Once the lattice data has been initialised, any derived properties are computed and an initial MPI communication is performed. The initialisation concludes by writing data output for time step zero and performing a sanity check, i.e. validating the numerical stability of the physical properties of the system.

3.3. Algorithmic subroutines

In order to evaluate the lattice-Boltzmann equation (1), the following sequence of subroutine calls is implemented.

```
[boundary conditions]
condition = periodic

[checkpoint]
format = hdf
n = 1000
num_files = 1
safe = T

[output]
folder= ./output
cpfolder = .
sci_rock = 10000
sci_od = 1000
sci_wd = 1000
sci_int = 1000
sci_arrows = 1000
sci_flo = 1000

[common]
n_spec = 2
iterations = 20000
nz = 64
nx = 64
ny = 64
py_ic_module = geometry.py
seed = 0

[simulation]
initial_conditions = random

[physics]
gravity = 0, 0, 1e-5
g_min_int = 0, 0, 0
g_max_int = 64, 64, 64
g_br = 0.01
g_wr = 0.0
```

Fig. 4. Example input file for LB3D v7.1. The file *geometry.py* is part of the *porous* example found in the examples directory. Details of the input parameters as well as a complete list of options is provided in the user manual. This setup is used for the evaluation of permeability in Section 4.1.2.

1. *Streaming step:* Propagate populations according to Eq. (2) and apply bounce-back boundary conditions;
2. Compute the pre-collisional conserved density ρ_σ *via* Eq. (7) and momenta $\mathbf{p}_\sigma$ *via* Eq. (8) for use in influx boundaries, collision calculation and output routines;
3. Calculations for influx and outflux boundary conditions;
4. Compute external and intermolecular (Shan–Chen) forces used in collision calculation and output routines;
5. *Collision step:* Evaluate Eq. (1).

The execution of these steps is performed in the main time loop subroutine of LB3D as illustrated in Fig. 6. The computation steps are interspersed with communication steps as required by the parallelisation, cf. Fig. 7. In addition, optional steps can be executed for input/output, checkpointing, and sanity checks. In contrast to other lattice Boltzmann implementations, the algorithm starts by performing communication followed by an advection step. Subsequently, calculation and caching of data is required by boundary conditions and interaction forces and is performed before the collision step.

In the current implementation the algorithm performs between 5 and 6 complete spatial loops — one for each advection and collision process, one for the calculation of pseudo-potential forces and two more to pre-calculate and cache momentum and forces. A last optional loop is required for the calculation of dipolar interactions in the ternary model. Dividing the algorithm in this fashion provides a clear structure for the integration of future features. Here the focus is not on optimal performance but rather on readability and extensibility.

3.4. Parallelisation

Simulations are run on a three-dimensional rectangular lattice of size tnx × tny × tnz with periodic boundary conditions as default. In order to run parallel jobs, the lattice slab is divided on program start-up into blocks of equal size nx × ny × nz. The exact lattice subdivision may be either specified by the user or chosen automatically by the MPI implementation. The number of lattice sites along each axis must be divisible by the number of processes used along that axis.

The spatial dimensions nx, ny, nz of all arrays on each CPU are extended by a *halo* region which contains copies of lattice blocks from neighbouring CPUs. The algorithms implemented in the current code require a halo depth of one lattice site, but it can be set to any value for more complicated algorithms. MPI-2 array data types are instantiated for the halo regions such that a single pair of MPI send and receive calls can be used for communication between MPI processes.

Shared memory parallelisation using OpenMP is enabled by wrapping the various spatial loops responsible for the calculation of the respective algorithm steps. Advection is performed using a single lattice buffering adjacent lattice site information for one z-layer at the time. Creation of the buffer is multi-threaded separately from the subsequent advection calculation.

3.5. Data output

LB3D writes physical properties of fluids to output files at specified time intervals. The writeable properties include the densities of components and the flow velocity, and the frequency at which data is written can be specified in the input file. Generally, no outputs are written until time step `n_sci_start`, after which data is written every `n_sci_OUTPUT` time steps, where `OUTPUT` stands for a specific property. A value of `n_sci_OUTPUT = 0` disables the output of the respective property. The output file format is the Hierarchical Data Format (HDF) [63] which facilitates portable, platform–independent data. HDF provides the possibility to add meta data to the raw data files and LB3D makes use of this feature by adding specific information on parameters as detailed in the manual [64].

LB3D can be instructed to produce *checkpoint files* at specified time intervals. These files can be used to restart the simulation from a given configuration. When the simulation is being restarted from a checkpoint, it is possible to override any of the simulation parameters and even re-apply initialisation scripts. Therefore checkpoints may be used to create non-trivial initial conditions and *ad hoc* steering of simulations. Both output and checkpoint files are written in HDF5 [65].

3.6. Installation

LB3D version 7.1 supports most Linux environments including Gnu, Cray or Intel compatible compilers.

- MPI $\geq$ 2.0,
- HDF5 $\geq$ 1.8.0 (compiled against MPI with –enable-parallel –enable-fortran),

```
# You can load XDRF files like this:
# lb3d.geometry_from_xdrf("wettingWall.wall.xdr")

# lb3d.geometry_add_box(0, 0, 0,
#                       lb3d.params.tnx-1, lb3d.params.tny-1, lb3d.params.tnz-1)
# offset = lb3d.params.tnx / 4
# lb3d.geometry_add_box(offset, offset, offset,
#                       lb3d.params.tnx-offset, lb3d.params.tny-offset, lb3d.params.tnz-offset)
# lb3d.geometry_fill_region(offset, offset, lb3d.params.tnz-offset,
#                           lb3d.params.tnx-offset, lb3d.params.tny-offset, lb3d.params.tnz-offset,
#                           clear=True)

# lb3d.geometry_add_ellipse(lb3d.params.tnx/2, lb3d.params.tny/2, lb3d.params.tnz/2,
#                           lb3d.params.tnx/4, lb3d.params.tny/5, lb3d.params.tnz/6)

import random

tnx = lb3d.params.tnx-1
tny = lb3d.params.tny-1
tnz = lb3d.params.tnz-1
mx = (lb3d.params.tnx-1)/2
my = (lb3d.params.tny-1)/2
mz = (lb3d.params.tnz-1)/2
scale = (mx+my+mz)/3/5
offset = 0

radius = 28
period = 32

a = radius
b = radius
c = radius

for x in range(0,tnx,period):
  for y in range(0,tny,period):
    for z in range(0,tnz,period):
      if (x%(2*period)==0) and (y%(2*period)==0) and (z%(2*period)==0):
        lb3d.geom_ellipsoid(x,y,z,a,b,c, lb3d.rock_set)
        lb3d.geom_ellipsoid(x,y,z,a,b,c, lb3d.fluid_clear)
      elif (x%(2*period)!=0) and (y%(2*period)!=0) and (z%(2*period)!=0):
        lb3d.geom_ellipsoid(x,y,z,a,b,c, lb3d.rock_set)
        lb3d.geom_ellipsoid(x,y,z,a,b,c, lb3d.fluid_clear)
```

Fig. 5. Example use of the Python bindings for generation of a system setup. The file *geometry.py* is part of the *porous* example found in the examples directory. The functions `rock_set` and `fluid_clear` manipulate the obstacle and density field of the system directly in the LB3D domain. The function `lb3d.geom_ellipsoid` executes respective calls in an ellipsoid volume with centre $\{x, y, z\}$ and axes $\{a, b, c\}$. Algorithm development around atomic get and set functions can make use of the regular Python environment.

- Python $\geq$ 2.6,
- Jinja2.

Configuration of the package should include explicit specification of the HDF5 compiler wrapper `h5pcc` as well as the LB3D Python bindings via the options

```
$ ./configure --with-hdf5=/PATH/TO/h5pcc\
  --with-lb3d-py-path =/PATH/TO/lb3d/py
$ make
```

By default LB3D builds the binary `lb3d` in the `/lb3d/src` directory. Evoking `make install` by default installs to `/usr/local/lb3d`. If configure is provided with an install prefix, the binary is copied to `/PREFIX/bin/lb3d` and Python bindings to `/PREFIX/share/lb3d/py`. More details on the installation process can be found in the user manual [64].

3.7. Workflow

In order to illustrate the typical workflow of a LB3D simulation, we use the porous media example, the results of which are presented in Section 4. The input file is listed in Fig. 4 and the Python script generating the geometry is listed in Fig. 5. The input file specifies simulation parameters like the number of fluid species, system size, and number of time steps to simulate. Further initial parameters can be included through Python bindings. The Python script contains functions to set and unset obstacle and fluid parameters on individual lattice sites or specific geometries. It is also possible to load information from an XDR file which defines applicable boundary conditions for each lattice site (details can be found in the user manual [64]).

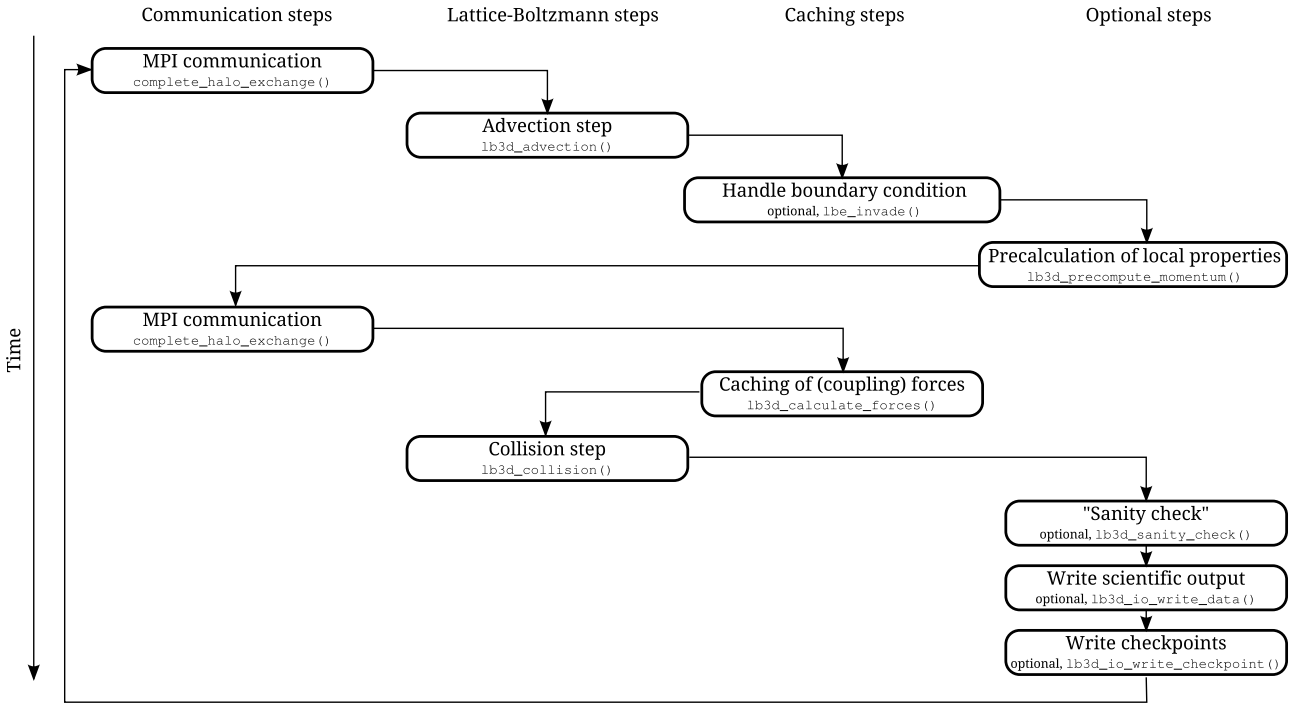

Fig. 6. Algorithm execution in the main time loop of LB3D. As the initialisation step includes a first collision calculation to enforce well-formed initial distributions, the time-loop starts with a communication step and executes advection first. To avoid re-calculation of frequently used properties locally conserved density and momentum as well as interaction forces are cached. Symmetry in the interaction forces permits halving of the computed force components. Every time step or interval can be configured to include checks of system stability and validity, termed sanity checks, as well as scientific and checkpoint output.

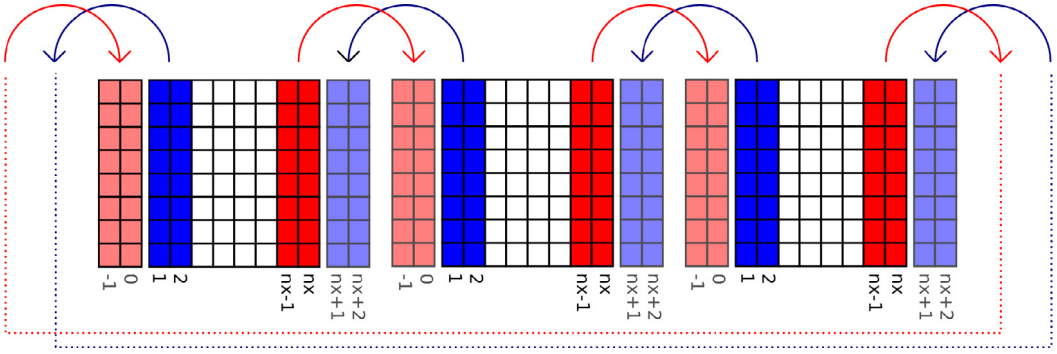

Fig. 7. Illustration of the subarray structure and communication pattern in one dimension for a halo extent of 2. The simulation domain extends from coordinate index 1 to *nx*. The boundary layers of the domain are replicated as halo layers on the neighbouring processes where data is exchanged through MPI communication. Sending and receiving of data is facilitated by MPI subarray types for the halo extent. Periodic boundary conditions are implied by the MPI neighbourhood. Colour online.

All file paths are specified relative to the current execution directory. The actual simulation is started by invoking

```
mpirun -n NUMBER_OF_CORES ./lb3d -f input.in
```

where the `mpirun` command is dependent on the specific system configuration and MPI library. By default LB3D writes output to the directory `./output/` relative to the execution directory. Data analysis can be performed using a range of available HDF5 tools for different languages including Python, C/C++, and FORTRAN. For example, the package h5utils provides a number of converters, i.a., to ASCII and VTK formats for visualisation. More details on the available configuration and evaluation options can be found in the user manual [64].

4. Case studies

The development of LB3D version 7.1 involved substantial refactoring of an earlier version of the code. Besides integrating new features and cleaning up the input/output routines, the parallelisation structure was extended to make use of shared memory strategies within nodes that comprise more and more cores. In this section we report the results of physics validation as well as performance benchmarks. This is done to confirm the accuracy of the added and altered functionality and probe the parametrisation of the hybrid parallel approach on the current UK National Supercomputer ARCHER.

4.1. Validation

To ensure that the re-factoring has preserved all model properties, exemplary mesophase simulations originally performed with prior versions of LB3D have been reproduced. Due to the deterministic nature of the lattice Boltzmann method and the absence of random initial and boundary conditions the obtained results reproduce values calculated by earlier versions exactly. Here we describe simulation parametrisation and results for three test cases. For the

spinodal decomposition of an amphiphilic mixture we report the time behaviour of the average fluid domain size as a function of time and evaluate the power law observed. In a second test case we make use of the Python bindings and measure permeability in a model porous medium. Finally we investigate the qualitative formation of amphiphilic mesophases.

4.1.1. Spinodal decomposition of an amphiphilic mixture

Spinodal decomposition is the process of rapid demixing of immiscible fluids, e.g. water and oil. The phase separation is governed by surface tension effects and characteristically exhibits an exponential increase in domain volumes of the demixing components [16,66]. The addition of an amphiphilic component reduces the surface tension between the components and slows the spinodal decomposition process down. The resulting process still behaves exponentially, but the exponent of the observed dynamics of the domain sizes is reduced [20,67–69].

In order to evaluate the dynamics of the ternary model implemented in LB3D, we validate simulation results published by some of us previously [20,69]. Keeping the overall fluid density constant at $\rho_{tot} = 0.8\, m/a^3$ overall, the density of the amphiphilic component is increased in steps of $\Delta\rho_\alpha = 0.05\, m/a^3$ from $\rho_\alpha = 0.0\, m/a^3$ to $\rho_\alpha = 0.30\, m/a^3$. Simulations are performed on a lattice of $256 \times 256 \times 256$ sites. The coupling between immiscible components is $g_{br} = 0.08\, m/(ah^2)$, between immiscible components and surfactant is $g_{bs} = -0.06\, m/(ah)^2$ and between surfactant components $g_{ss} = -0.03\, m/(ah)^2$.

Fig. 8 shows the dependence on time of the average lateral domain size for varying amphiphilic concentration. The exponent of the decomposition dynamics is determined by fitting an exponential to the relevant portion of the curves (portions linear in the log–log view). Initial non-exponential behaviour is due to the homogeneous initialisation of the fluids and discretisation on the lattice, respectively. The right part of the figure shows the resulting time exponents as a function of amphiphilic concentration. The impact of added surfactant becomes more pronounced approaching the critical micelle concentration at approximately 22%.

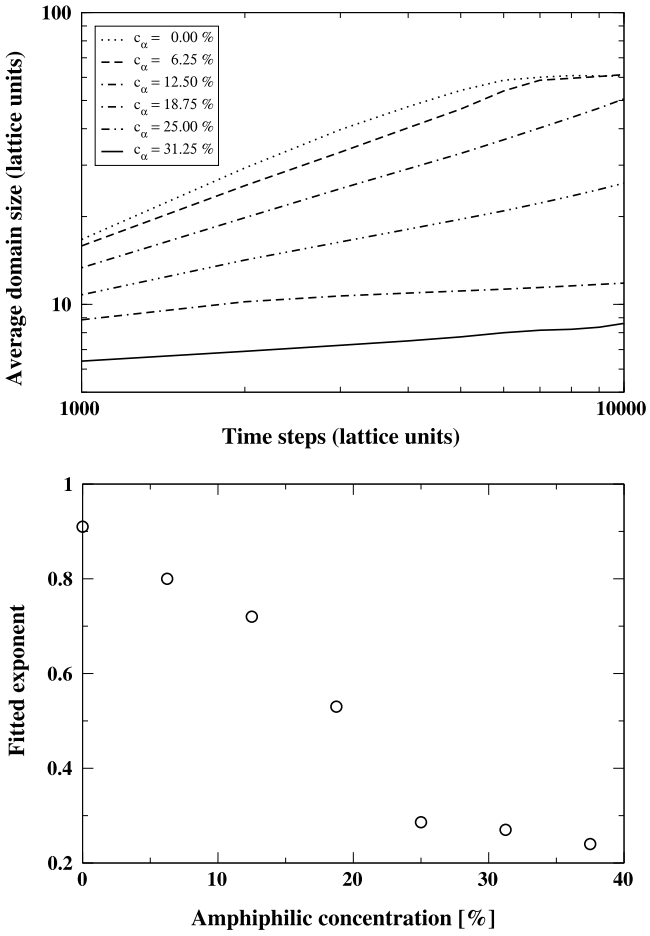

Fig. 8. Log–log plot of the dynamics of the average lateral domain size as a function of amphiphilic fluid concentration. Exponents of the fitted curves are plotted as function of the amphiphilic fluid concentration on the right. When exceeding a concentration of approximately 35%, the critical micelle concentration, spinodal decomposition is arrested. The impact of added surfactant concentration increases significantly towards the critical concentration.

4.1.2. Permeability in a model porous medium

Another area of application of LB3D is the modelling of the flow of fluid mixtures over solid surfaces and in porous media. A phenomenological property to classify porous media is the permeability κ which quantifies the relation of a pressure gradient applied on a medium and the resulting flow through it. For a regular lattice of spheres the permeability is accessible semi-analytically [70,71] and has been found to follow

$$\kappa = \frac{R^2}{6\pi\chi\zeta}, \tag{43}$$

where R is the sphere radius, χ is the ratio between radius and separation, assumed to equal one in our case as spheres created by the initialisation algorithm do touch (see Fig. 9). Finally, ζ is a dimensionless drag coefficient which can be approximated by a series expansion in χ. With an approximate value of $\chi \cdot \zeta \approx 160.0$ the theoretical permeability in lattice units for our system is $\kappa \approx 0.075$. Using the input file shown in Fig. 4, a system of $64 \times 64 \times 64$ lattice sites is initialised with the default *red* and *blue* concentrations $\rho_r = \rho_b = 1.0\, m/a^3$. The coupling $g_{br} = 0.01\, m/(ah)^2$ keeps the liquids miscible. A gravity force of $10^{-5}\, m/(ah)^2$ equivalent to a homogeneous pressure gradient is applied in the z-direction. In order to measure the permeability, we evaluate the average velocity $\langle u \rangle$ and density $\langle \rho \rangle$ as well as the mean pressure gradient $\langle \nabla P \rangle$, and calculate

$$\kappa = -\rho\nu \frac{\langle u \rangle}{\langle \nabla P \rangle}. \tag{44}$$

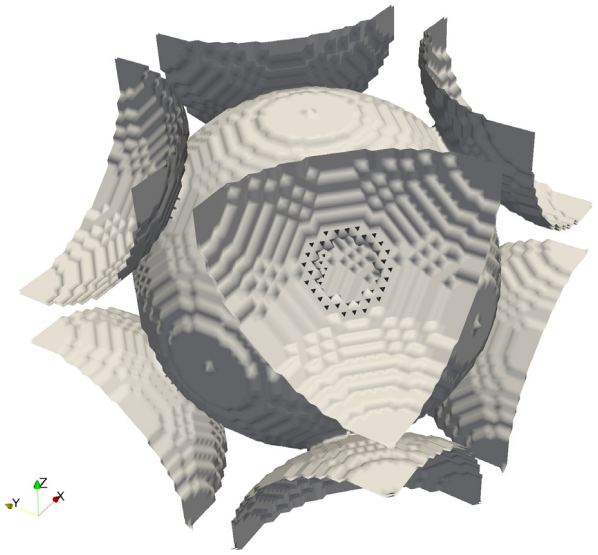

Fig. 9. Volume rendering of the model porous geometry as generated by the `geometry.py` file listed in Section 3. The geometry is used here in the simulations for permeability measurements. Spheres of a radius of 15 lattice sites are positioned on a body centred cubic lattice. The system shown has $64 \times 64 \times 64$ lattice sites.

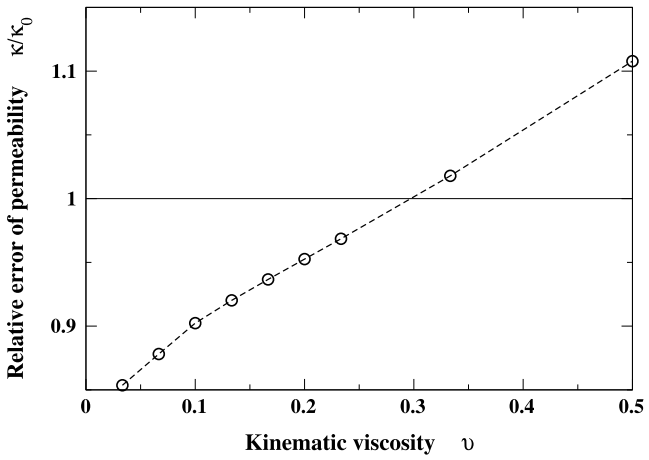

Fig. 10. Relative error of permeability measured in the simulation domain as compared to the theoretical results. For LBGK models fixed relaxation parameters exist that minimise the error. Other ways to decrease deviation include simulate at higher resolutions and use more accurate collision models such as multi-relaxation time collision schemes.

Table 1
Simulation parameters for the self-assembly of representative primitive, hexagonal, diamond and gyroid mesophases used in the validation simulations. The character of the phase is determined by the densities $\rho_{r,b}$ of two partially miscible or immiscible fluid components and the density ρ_s of the amphiphilic component as well as the parameters defining the coupling strength between the immiscible components, g_{br}, the (im)miscible and amphiphilic components g_{bs} and the amphiphilic self-interaction g_{ss}. See text for further details.

Mesophase	ρ_r	ρ_b	ρ_s	g_{br}	g_{bs}	g_{ss}
Primitive	0.3	0.3	0.9	0.01	−0.006	−0.0005
Hexagonal	0.3	0.3	0.9	0.01	−0.006	−0.001
Diamond	0.05	0.05	0.7	0.01	−0.006	0.00035
Gyroid	0.7	0.7	0.9	0.08	−0.006	−0.003

Fig. 10 shows the error of permeability measurements as a function of the simulated fluid viscosity. The deviation is a well known artefact of LBGK models when combined with bounce-back boundary conditions. The effect can be reduced by increasing the resolution as it is directly proportional to the surface to volume ratio of the simulation. Another way to correct the error is the integration of a multi-relaxation time collision scheme within LB3D, as is planned in the future.

4.1.3. Formation of amphiphilic mesophases

The ternary fluid model implemented in LB3D can be applied to investigate the self-assembly of amphiphilic mesophases [20–22,30–34]. Table 1 gives an overview of the parametrisation for simulations of self-assembly of mesophases using LB3D v7.1. Simulations were performed on a lattice of 256 × 256 × 256 sites. Simulation domains are initialised with fluid of homogeneous densities of $\rho_{r,b}$ for the two components and a density ρ_s for the amphiphilic component respectively. The subsequent self-assembly of the mesophases is driven by the coupling forces, cf. Eq. (15).

In pressure jump experiments the density of the fluid components is increased, which is equivalent to an increase in overall system pressure. We observe the formation of primitive and hexagonal mesophases, respectively, for a reduction in the surfactant self-interaction strength. In our simulations the parameter was changed from $g_{ss} = -0.0005\, m/(ah)^2$ to $g_{ss} = -0.001\, m/(ah)^2$. Systems at lower and higher pressure exhibit cubic diamond and gyroid mesophases, respectively. Fig. 11 shows a volume rendering of the observed morphologies. The images display the zero isosurface of the order parameter defining the boundary of immiscible components (cf. [31] for more details).

4.2. Performance

During the refactoring of LB3D, we have focused on creating more clearly structured and well documented code. Moreover, the new implementation exhibits a significantly improved single core performance. While OpenMP shared memory parallelisation has been implemented in addition to the prior MPI parallelisation, we focus here on the single core performance and scaling of the MPI based parallelisation. Tests of a hybrid parallelisation approach, naïvely introducing OpenMP wrappers around the main loops, result in loss of performance in all cases. More sophisticated strategies introducing nested loops and explicit OpenMP parametrisation allow for performance gains of the order of 10 percent compared to simple MPI. We have, however, found the successful parametrisations to not only vary with the machine, but to depend on the chosen problem as well. While this may change for explicit shared memory machines and future heterogeneous exa-scale configurations, in the current publication the systematic investigation of hybrid parallel performance has been omitted.

Table 2 compares the single core performance of LB3D v7.1 against LB3D v7.0 as measured for simulations of domains of 256 × 256 × 256 lattice sites on 22 nodes of ARCHER comprised of two 12-core CPUs each. The speedup observed between the versions measured in millions of lattice site updates per second (MSUPs) per core is approximately three-fold. Acceleration of computation has been achieved by optimising communication, memory structure and algorithms throughout the code. Most prominently, component information has been separated in memory. Pre-calculation and caching steps have reduced redundant calculations. The change in memory layout has furthermore facilitated optimised MPI array definitions and communication patterns.

Fig. 12 illustrates the strong scaling behaviour for geometries of size 1024 × 1024 × 1536 lattice sites and calculations considering single phase flow, binary mixtures or amphiphilic fluid mixtures. Simulating a system of pure fluid (mixture) for 10,000 time steps on 256, 512, 1024 and 2048 nodes of ARCHER, we observe excellent strong scaling behaviour close to ideal speedup. This is especially true for the more computationally demanding multi-component cases. CPU counts are multiples of 24 corresponding to the ARCHER node size. The scaling remains close to linear for many tens of thousands of cores and for the selected system domain size only just starts to deteriorate on 49,152 cores. The single core performance is comparable to the values given in Table 2.

Table 2
Comparison of performance between LB3D version 7.0 and the new LB3D version 7.1. The performance measured in millions of lattice site updates per second (MSUPs) per core shows significant improvement.

Application	# of lattice sites	Discretisation	Architecture	# of cores	MSUPs/core
LB3D v7.0 1-phase	16,777,216	D3Q19	Intel Xeon E5-2697 v2 (ARCHER)	512	0.369
LB3D v7.0 2-phase	16,777,216	D3Q19	Intel Xeon E5-2697 v2 (ARCHER)	512	0.203
LB3D v7.0 3-phase	16,777,216	D3Q19	Intel Xeon E5-2697 v2 (ARCHER)	512	0.098
new LB3D v7.1 1-phase	16,777,216	D3Q19	Intel Xeon E5-2697 v2 (ARCHER)	512	1.360
new LB3D v7.1 2-phase	16,777,216	D3Q19	Intel Xeon E5-2697 v2 (ARCHER)	512	0.648
new LB3D v7.1 3-phase	16,777,216	D3Q19	Intel Xeon E5-2697 v2 (ARCHER)	512	0.277

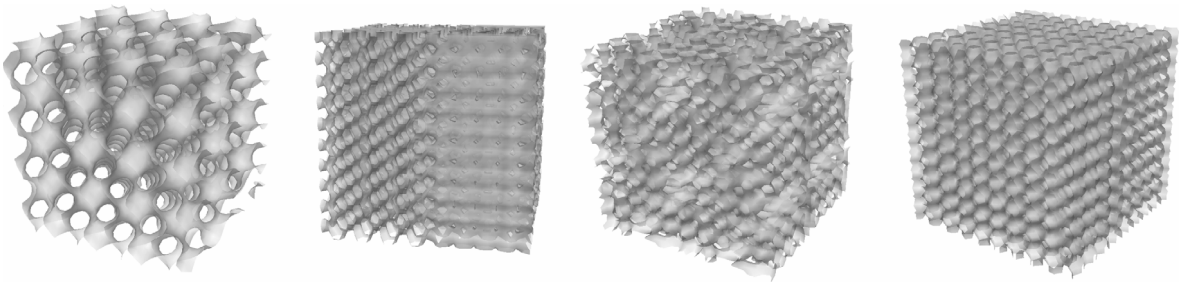

Fig. 11. Volume rendering of simulation snapshots of zero order parameter iso surfaces for the respective mesophases obtained in simulation. Simulations were run in a volume of 32^3 lattice sites for 30,000 time steps. The order parameter is given by the difference of local densities of immiscible components. The density and interaction parameters control the pressure in the model (see [31]).

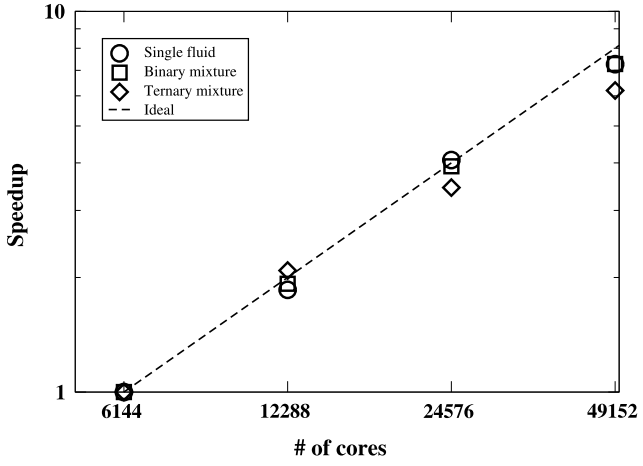

Fig. 12. Strong scaling behaviour of single component, binary immiscible and ternary amphiphilic fluid mixtures in simulations domains of $1024 \times 1024 \times 1536$ lattice sites. Data was obtained from runs of 10,000 time steps performed on ARCHER. The scaling remains close to linear for many tens of thousands of cores and only just starts to deteriorate on 49,152 cores. The single core performance is here comparable to the values given in Table 2.

5. Summary

The LB3D simulation code has enabled a substantial amount of scientific research in diverse contexts. Starting from the investigation of ternary amphiphilic fluid mixtures the code has been extended to include a variety of boundary effects and coupled models. Cleaning and refactoring of the code have led to improved readability and extensibility. To ensure scientific accuracy and model applicability of the refactored version, we have reproduced earlier findings of ternary amphiphilic mixtures on the current UK National Supercomputer. While maintaining flexibility and extensibility, the single-core performance has more than doubled. Exploiting the excellent parallelisation behaviour of the lattice-Boltzmann algorithm, the strong scaling behaviour of the code remains close to linear. With LB3D version 7.1, we release an open source version of the code that provides functionality for simulation of amphiphilic mixtures in complex geometries. The documentation and new Python scripting options enhance the ease of use of existing features, while at the same time facilitating continued development and extension. By releasing this code under a BSD 3-clause license, we hope to inspire independent developers to contribute new features to LB3D.

Acknowledgements

Support from Fujitsu Laboratories Europe, from the UK Consortium on Mesoscale Engineering Sciences (UKCOMES) under EPSRC Grant No. EP/L00030X/1 and the EU H2020 ComPat project No. 223979 is acknowledged. Our work also made use of the ARCHER UK National Supercomputing Service (http://www.archer.ac.uk).

References

[1] J. Desplat, I. Pagonabarraga, P. Bladon, Comput. Phys. Comm. 134 (2001) 273.
[2] M.D. Mazzeo, P.V. Coveney, Comput. Phys. Comm. 178 (2008) 894–914.
[3] OpenLB - Open Source Lattice Boltzmann Code (http://www.optilb.org).
[4] Palabos - CFD Complex Physics (http://www.palabos.org).
[5] Exa Simulation Software Solutions | Exa Corporation (http://exa.com).
[6] XFlow is a next generation CFD software system from Next Limit Technologies that uses a proprietary, particle-based, meshless approach which can easily handle traditionally complex problems (http://www.xflowcfd.com).
[7] B.M. Boghosian, P.V. Coveney, P.J. Love, Proc. R. Soc. A 456 (2000) 1431.
[8] P.J. Love, J.B. Maillet, P.V. Coveney, Phys. Rev. E 64 (2001) 061302.
[9] J. Chin, P.V. Coveney, Phys. Rev. E 66 (2002) 016303.
[10] H. Chen, B. Boghosian, P. Coveney, M. Nekovee, Proc. R. Soc. A 456 (2000) 2043.
[11] J. Chin, J. Harting, S. Jha, P.V. Coveney, A.R. Porter, S.M. Pickles, Contemp. Phys. 44 (2003) 417–434.
[12] J. Chin, P.V. Coveney, J. Harting, Proceedings of the UK E-Science All Hands Meeting, 2004.
[13] S.M. Pickles, R.J. Blake, B.M. Boghosian, J.M. Brooke, J. Chin, P.E.L. Clarke, P.V. Coveney, N. González-Segredo, R. Haines, J. Harting, M. Harvey, S. Jha, M.A.S. Jones, M. McKeown, R.K. Pinning, A.R. Porter, K. Roy, M. Riding, Proceedings of the Workshop on Case Studies on Grid Applications at GGF 10.
[14] R.J. Blake, P.V. Coveney, P. Clarke, S.M. Pickles, Sci. Program. 13 (2005) 1–17.
[15] J. Harting, J. Chin, M. Venturoli, P.V. Coveney, Phil. Trans. R. Soc. A 363 (2005) 1895.
[16] N. González-Segredo, M. Nekovee, P.V. Coveney, Phys. Rev. E 7 (2003) 046304.
[17] N.J. González Segredo, P.V. Coveney, Lattice-Boltzmann and lattice-gas simulations of binary immiscible and ternary amphiphilic fluids in two and three dimensions.
[18] J. Harting, G. Giupponi, P.V. Coveney, Phys. Rev. E 75 (2007) 041504.
[19] J. Harting, G. Giupponi, High Performance Computing in Science and Engineering, vol. 07, 2008, p. 457.
[20] N. González-Segredo, P.V. Coveney, Phys. Rev. E 6 (2004) 9.
[21] N. González-Segredo, P.V. Coveney, Europhys. Lett. 6 (2004) 5.
[22] N. Gonzalez-Segredo, P.V. Coveney, Phys. Rev. E 69 (2004) 060701.
[23] J. Harting, M. Venturoli, P.V. Coveney, Phil. Trans. R. Soc. A 362 (2004) 1703–1722.
[24] R.S. Saksena, B. Boghosian, L. Fazendeiro, O.A. Kenway, S. Manos, M.D. Mazzeo, S.K. Sadiq, J.L. Suter, D. Wright, P.V. Coveney, Phil. Trans. R. Soc. A 367 (2009) 2557–2571.
[25] D. Groen, O. Henrich, F. Janoschek, P.V. Coveney, J. Harting, Lattice-Boltzmann Methods in Fluid Dynamics: Turbulence and Complex Colloidal Fluids, W.F. Bernd Mohr (Ed.), Jülich Blue Gene/P Extreme Scaling Workshop 2011, Jülich Supercomputing Centre, 52425 Jülich, Germany, 2011, fZJ-JSC-IB-2011-02.
[26] J. Harting, M.J. Harvey, J. Chin, P.V. Coveney, Comput. Phys. Comm. 165 (2005) 97–109.
[27] J. Chin, P.V. Coveney, Phil. Trans. R. Soc. A 462 (2006) 3575–3600.
[28] G. Giupponi, P.V. Coveney, Math. Comput. Simulation 72 (2006) 124–127.
[29] N. González-Segredo, J. Harting, G. Giupponi, P.V. Coveney, Phys. Rev. E 73 (2006) 031503.
[30] R.S. Saksena, P.V. Coveney, Parallel Lattice-Boltzmann Simulations for the Investigation of Cubic Phase Rheology at the Petascale. Proceedings of the Teragrid conference 2008 (2008) 92.
[31] R.S. Saksena, P.V. Coveney, J. Phys. Chem. B 112 (2008) 2950–2957.
[32] R.S. Saksena, P.V. Coveney, Proc. R. Soc. A 465 (2009) 1977–2002.
[33] R.S. Saksena, P.V. Coveney, Soft Matter 5 (2009) 4446–4463.

[34] P.V. Coveney, R.S. Saksena, Abstracts of Papers of the American Chemical Society 239 (2010) 473.
[35] S. Succi, The Lattice Boltzmann Equation: For Fluid Dynamics and Beyond, Oxford University Press, 2001.
[36] B. Dünweg, A. Ladd, Adv. Poly. Sci. 221 (2009) 89–166.
[37] C.K. Aidun, J.R. Clausen, Annu. Rev. Fluid Mech. 42 (2010) 439–472.
[38] X. He, L.-S. Luo, Phys. Rev. E 55 (1997) R6333–R6336.
[39] X. He, L.-S. Luo, Phys. Rev. E 56 (1997) 6811.
[40] X. He, X. Shan, G.D. Doolen, Phys. Rev. E 57 (1998) R13–R16.
[41] A.J.C. Ladd, R. Verberg, J. Stat. Phys. 104 (2001) 1197–1251.
[42] Y.H. Qian, D. D'Humières, P. Lallemand, Europhys. Lett. 17 (1992) 479.
[43] X. Shan, X. He, Phys. Rev. Lett. 80 (1998) 65–68.
[44] X. Shan, X.-F. Yuan, H. Chen, J. Fluid Mech. 550 (2006) 413–441.
[45] P.L. Bhatnagar, E.P. Gross, M. Krook, Phys. Rev. 94 (1954) 511–525.
[46] D.A. Wolf-Gladrow, Lattice-Gas Cellular Automata and Lattice Boltzmann Models, Springer, 2000.
[47] Z. Guo, C. Zheng, B. Shi, Phys. Rev. E 65 (2002) 046308.
[48] A. Chekhlov, V. Yakhot, Phys. Rev. E 51 (1995) R2739–R2742.
[49] X. Shan, G. Doolen, J. Stat. Phys 81 (1995) 379.
[50] X. Shan, H. Chen, Phys. Rev. E 47 (1993) 1815–1819.
[51] M. Nekovee, P.V. Coveney, H. Chen, B.M. Boghosian, Phys. Rev. E 62 (2000) 8282–8294.
[52] N.S. Martys, H. Chen, Phys. Rev. E 53 (1996) 743.
[53] R. Cornubert, D. d'Humières, D. Levermore, Physica D 47 (1991) 241–259.
[54] P. Lavallée, J.P. Boon, A. Noullez, Physica D 47 (1991) 233–240.
[55] D.P. Ziegler, J. Stat. Phys. 71 (1993) 1171–1177.
[56] X. He, Q. Zou, L.-S. Luo, M. Dembo, J. Stat. Phys. 87 (1997) 115–136.
[57] I. Ginzburg, D. D'Humières, Phys. Rev. E 68 (2003) 066614.
[58] B. Chun, A.J.C. Ladd, Phys. Rev. E 75 (2007) 066705–066712.
[59] Q. Zou, X. He, Phys. Fluids 9 (1997) 1591–1598.
[60] M.E. Kutay, A.H. Aydilek, E. Masad, Comput. Geotech. 33 (2006) 381–395.
[61] K. Mattila, J. Hyväluoma, T. Rossi, J. Stat. Mech. Theory Exp. 2009 (2009) P06015.
[62] M. Hecht, J. Harting, J. Stat. Mech. Theory Exp. 13 (2010) 1.
[63] The HDF Group, Hierarchical data format version 5 (2000-2010).
[64] E. Breitmoser, J. Chin, C. Dan, F. Dörfler, S. Frijters, G. Giupponi, N. González-Segredo, F. Günther, J. Harting, M. Harvey, M. Hecht, S. Jha, F. Janoschek, F. Jansen, C. Kunert, M. Lujan, I. Murray, A. Narváez, M. Nekovee, A. Porter, F. Raischel, R. Saksena, S. Schmieschek, D. Sinz, M. Venturoli, T. Zauner, LB3D user manual, UCL (http://ccs.chem.ucl.ac.uk/lb3d).
[65] M. Folk, G. Heber, Q. Koziol, E. Pourmal, D. Robinson, Proceedings of the EDBT/ICDT 2011 Workshop on Array Databases, AD'11, ACM, New York, NY, USA, 2011, pp. 36–47.
[66] J.W. Cahn, Acta Metall. 9 (1961) 795–801.
[67] J.W. Cahn, Acta Metall. 10 (1962) 179–183.
[68] P.G.d. Gennes, J. Chem. Phys. 72 (1980) 4756–4763.
[69] S. Schmieschek, A. Narváez, J. Harting, in: High Performance Computing in Science and Engineering, vol. 12, Springer, Berlin, Heidelberg, 2013, pp. 39–49.
[70] H. Hasimoto, J. Fluid Mech. 5 (1959) 317–328.
[71] A. Sangani, A. Acrivos, Int. J. Multiph. Flow 8 (1982) 343–360.