ARTICLE IN PRESS

Computers and Mathematics with Applications xxx (xxxx) xxx

Contents lists available at ScienceDirect

Computers and Mathematics with Applications

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

OpenLB—Open source lattice Boltzmann code

Mathias J. Krause [a,b,c,*], Adrian Kummerländer [a,c], Samuel J. Avis [e], Halim Kusumaatmaja [e], Davide Dapelo [d], Fabian Klemens [a,c], Maximilian Gaedtke [a,b], Nicolas Hafen [a,b], Albert Mink [a,b], Robin Trunk [a,b], Jan E. Marquardt [a,b], Marie-Luise Maier [a,b], Marc Haussmann [a,b], Stephan Simonis [a,c]

[a] Lattice Boltzmann Research Group, Karlsruhe Institute of Technology, Karlsruhe, Germany
[b] Institute for Mechanical Process Engineering and Mechanics, Karlsruhe Institute of Technology, Karlsruhe, Germany
[c] Institute for Applied and Numerical Mathematics, Karlsruhe Institute of Technology, Karlsruhe, Germany
[d] Faculty of Engineering and Informatics, University of Bradford, Bradford, United Kingdom
[e] Department of Physics, Durham University, Durham, United Kingdom

ARTICLE INFO

Article history:
Available online xxxx

Keywords:
Numerical simulation
Lattice Boltzmann methods
Partial differential equations
Computational fluid dynamics
Transport problems
OpenLB

ABSTRACT

We present the OpenLB package, a C++ library providing a flexible framework for lattice Boltzmann simulations. The code is publicly available and published under GNU GPLv2, which allows for adaption and implementation of additional models. The extensibility benefits from a modular code structure achieved e.g. by utilizing template meta-programming. The package covers various methodical approaches and is applicable to a wide range of transport problems (e.g. fluid, particulate and thermal flows). The built-in processing of the STL file format furthermore allows for the simple setup of simulations in complex geometries. The utilization of MPI as well as OpenMP parallelism enables the user to perform those simulations on large-scale computing clusters. It requires a minimal amount of dependencies and includes several benchmark cases and examples. The package presented here aims at providing an open access platform for both, applicants and developers, from academia as well as industry, which facilitates the extension of previous implementations and results to novel fields of application for lattice Boltzmann methods. OpenLB was tested and validated over several code reviews and publications. This paper summarizes the findings and gives a brief introduction to the underlying concepts as well as the design of the parallel data structure.

1. Introduction

In most engineering disciplines, especially in process and mechanical engineering as well as in biotechnology, numerical fluid flow simulations are considered as an important and cost-saving tool for the design and dimensioning of novel processes and machines. For researching fundamental cause–effect relationships, computational fluid dynamics (CFD) have become indispensable. Due to improved performance in computing, the significance of numerical methods increased quickly over the past decades. Despite the proceeding hardware enhancement, this performance acceleration was only achieved with well-scaling parallel algorithms. The paradigm change towards massively parallel computers consisting of more than one million processor cores, triggered the development of CFD simulation software based on novel

* Corresponding author at: Lattice Boltzmann Research Group, Karlsruhe Institute of Technology, Karlsruhe, Germany.
E-mail address: mathias.krause@kit.edu (M.J. Krause).

https://doi.org/10.1016/j.camwa.2020.04.033
0898-1221/

Please cite this article as: M.J. Krause, A. Kummerländer, S.J. Avis et al., OpenLB—Open source lattice Boltzmann code, Computers and Mathematics with Applications (2020), https://doi.org/10.1016/j.camwa.2020.04.033.

approaches besides the conventional well-established methods such as finite element methods (FEM) or finite volume methods (FVM).

The lattice Boltzmann method (LBM) constitutes one of these promising recently emerged alternatives [1], which can efficiently utilize 10,000 and more processor cores [2,3]. Historically, LBM were widely and successfully applied as flow solvers for viscous, weakly compressible fluids. Due to its simple algorithmic structure and its local computation properties, which enable efficient implementations on high performance computers, LBM have become a popular tool also in other fields of application. Meanwhile, besides OpenLB [4] and several other open source codes (e.g. Palabos [5], waLBerla [6]), commercial codes (e.g. SIMULIA PowerFlow [7], OMNIS/LB [8]) have been released, which indicates the industrial potential of LBM.

In 2007, Latt and Krause released the first version of OpenLB [9]. At that time they observed that the majority of publications on LBM focused on comparing results obtained by particular LBM implementations for standard benchmarks with data gained experimentally or by other numerical schemes. Comparative numerical experiments for the investigation and juxtaposition of distinct LBM and associated boundary methods, were missing. Further, the equally important part of mathematical analysis, remained rather unattended. Even studies on numerical error analysis using e.g. error norms to prove convergence at least experimentally were rare to find. Besides aiming to impact the qualitative research on LBM, the intention of OpenLB was and is, transferring LBM to a state which enables solving more complex application scenarios beyond the researchers' prototypical implementations. The early authors of OpenLB were convinced that in order to obtain a successful reproduction of published results, a well-established theoretical fundamental as well as sustainable software development and open source universal access to prior achievements, are of crucial necessity for the LBM community and the general public alike. To their knowledge OpenLB is the first modular open source platform realizing LBM usage via a sustainably maintained library [10]. Since the initial release in 2007, many more code developers joined the open access project and 96 scientific publications have been issued using OpenLB. Among them, 47 with contributions by major code developers.

This publication aims at broadening the visibility of LBM via introducing OpenLB to potential future developers as well as applicants outside of the community. For this purpose, OpenLB is presented as a general LBM-based solver for partial differential equations (PDE) with particular focus on CFD stressing the basic implementation as well as open source community concepts. Providing a summary of most recent publications, main emphasis is placed on illustrating OpenLB's integrative realization, which enables efficient parallel simulations of transport problems in complex geometries. The presented work is structured as follows. Section 2 generally introduces the lattice Boltzmann (LB) methodology as a bottom-up discretization approach for solving problems governed by PDEs. In Section 3, fundamental concepts of OpenLB's implementation and software design are discussed, including its hybrid parallelization concept and recent performance results. Finally, the application oriented sections are devoted to summarize the scientific findings which were obtained with OpenLB in the fields of multiphase (Section 4), particulate (Section 5), turbulent (Section 6), and thermal flows (Section 7), as well as optimal control of fluid flow (Section 8), and for other transport-related problems (Section 9). Concluding the paper, a summary is given in Section 10.

2. Lattice Boltzmann methods

In general, the LBM resembles a bottom-up approach to numerically approximating the solution of a given PDE. Instead of a conventional top-down discretization of the target equation (TEQ), the method emerges from the discretization in space and time of a discrete velocity Boltzmann-type equation, where the solution to the TEQ is recovered in a limiting process. Different PDEs can be reached by altering certain ingredients of the method. The necessary LBM building blocks to recover exemplary TEQs appearing in the following sections are introduced below. Thorough derivations of specific LBM approaches are summarized more comprehensively for example in [11–13].

2.1. Methodology

A prototypical LBM can be derived from the non-dimensional Boltzmann equation (BE)

$$\partial_t f + \boldsymbol{v} \cdot \nabla_{\boldsymbol{x}} f + \frac{1}{\rho} \boldsymbol{F} \cdot \nabla_{\boldsymbol{\xi}} f = \Omega\left(f, f\right), \tag{1}$$

where the particle distribution function $f : \mathcal{X} \times \mathcal{V} \times \mathcal{I} \to \mathbb{R}_0^+$, $(\boldsymbol{x}, \boldsymbol{\xi}, t) \mapsto f\left(\boldsymbol{x}, \boldsymbol{\xi}, t\right)$ represents the density of mass in position space $\mathcal{X} \subseteq \mathbb{R}^d$ and velocity space $\mathcal{V} \subseteq \mathbb{R}^d$ with respect to time $\mathcal{I} \subseteq \mathbb{R}_0^+$. The specific body force is denoted by $\boldsymbol{F} \in \mathbb{R}^d$, and the density ρ is obtained via moment integration of f over $\mathcal{V}$. The inclusion of possible particle collisions depends on the formulation of the collision operator Ω and determines the system described by (1). Discretizing the velocity space to a finite subset $\mathcal{C} = \left\{\boldsymbol{c}_i \in \mathbb{R}^d \mid i = 0, 1, \ldots, q-1\right\} \subseteq \mathcal{V}$, hereafter referred to as *DdQq*, leads to a discrete velocity version of (1), termed in the following as discrete velocity Boltzmann equation (DVBE)

$$\partial_t f_i + \boldsymbol{c}_i \cdot \nabla_{\boldsymbol{x}} f_i = \Omega_i + \tilde{F}_i, \quad i = 0, 1, \ldots, q-1. \tag{2}$$

Since the velocity dependency of the quantities in the single equation (1) has been traded for a system of q equations, definite arguments are neglected from now on and i-indexed variables are occasionally summarized in vectors in $\mathbb{R}^q$,

Please cite this article as: M.J. Krause, A. Kummerländer, S.J. Avis et al., OpenLB—Open source lattice Boltzmann code, Computers and Mathematics with Applications (2020), https://doi.org/10.1016/j.camwa.2020.04.033.

where $i = 0, \dots, q-1$ holds consistently throughout the current section. Note that the indexing of the variables f_i, Ω_i, and $\tilde{F}_i$, implicitly denotes an approximation of f, Ω, and the force term $\frac{1}{\rho}\boldsymbol{F}\cdot\nabla_{\boldsymbol{\xi}} f$ in (1), by a truncated Hermite expansion [11], respectively. The moment integration can be discretized to a summation over discrete velocity indices [11]. Thus, q (discrete velocity) moments are obtained via

$$m_{\boldsymbol{g}^{(j)}} = \sum_{i=0}^{q-1} g_i^{(j)} f_i, \quad j = 0, \dots, q-1, \tag{3}$$

where $g_i^{(j)} = g^{(j)}(\boldsymbol{c}_i)$ denote polynomials in discrete velocity components and determine the order of $m_{\boldsymbol{g}^{(j)}}$. Let

$$\mathcal{G} = \left\{ \boldsymbol{g}^{(j)} = \left(g_0^{(j)}, g_1^{(j)}, \dots, g_{q-1}^{(j)}\right)^{\mathrm{T}} \in \mathbb{R}^q \;\middle|\; j = 0, 1, \dots, q-1 \right\} \tag{4}$$

define the set of the polynomial representations used in (3). Dependent on the choice of $\mathcal{C}$, $\boldsymbol{\Omega} = \left(\Omega_0, \Omega_1, \dots, \Omega_{q-1}\right)^{\mathrm{T}}$ and its collisional invariants contained in

$$\mathcal{G}^{\mathrm{inv}} = \left\{ \hat{\boldsymbol{g}} \in \mathcal{G} \;\middle|\; \sum_{i=0}^{q-1} \hat{g}_i \Omega_i = 0 \right\} \subseteq \mathcal{G}, \tag{5}$$

the linkage to the desired macroscopic partial differential equation, can be obtained through identifying $m_{\tilde{\boldsymbol{g}}}$, for $\tilde{\boldsymbol{g}} \in \mathcal{G}^{\mathrm{inv}}$, with the conserved macroscopic variables. Up to the discretization error, this relation of mesoscopic and macroscopic quantities is still valid for a completely discretized version of (2). To achieve discretization in time and space, integration along characteristics and a trapezoidal rule [11,14] for approximating the thus appearing time integral of $\Omega_i + \tilde{F}_i$ is employed. Other approaches are covered for example in [15–17]. As a result, the lattice Boltzmann equation (LBE) can generally be written as

$$f_i(\boldsymbol{x} + \boldsymbol{c}_i \triangle t, t + \triangle t) = f_i(\boldsymbol{x}, t) + \triangle t\, \Omega_i(\boldsymbol{x}, t) + \triangle t \hat{F}_i, \tag{6}$$

where $\hat{\boldsymbol{F}} \in \mathbb{R}^q$ denotes a transformation of $\tilde{\boldsymbol{F}} = \left(\tilde{F}_0, \tilde{F}_1, \dots, \tilde{F}_{q-1}\right)^{\mathrm{T}}$. Eq. (6) operates on a fully discrete level, i.e. $\boldsymbol{x} \in \mathcal{X}_h \subseteq \mathcal{X}$ and $t \in \mathcal{I}_{\triangle t} \subseteq \mathcal{I}$ are now discrete points in space and time, respectively, such that $\boldsymbol{x} + \boldsymbol{c}_i \triangle t \in \mathcal{X}_h$, where $h > 0$ is the grid spacing of the regular lattice $\mathcal{X}_h$ and $\triangle t > 0$ denotes the time step size in $\mathcal{I}_{\triangle t}$. The implementation of (6) within the final LB algorithm is typically split into a local collision step comprising the right-hand side of (6),

$$f_i^{\diamond}(\boldsymbol{x}, t) = f_i(\boldsymbol{x}, t) + \triangle t\, \Omega_i(\boldsymbol{x}, t) + \triangle t \hat{F}_i, \tag{7}$$

and a streaming step associated to the population transfer to neighboring nodes determined in the left-hand side of (6), i.e.

$$f_i(\boldsymbol{x} + \boldsymbol{c}_i \triangle t, t + \triangle t) = f_i^{\diamond}(\boldsymbol{x}, t)\,. \tag{8}$$

In the absence of body forces and boundaries, the scheme described by (6) is second order accurate in space and time [11].

Preserving second order accuracy for space- and time-dependent $\boldsymbol{F} \neq \boldsymbol{0}$, Guo et al. [18] defined a forcing scheme based on a shift in invariants of $\boldsymbol{\Omega}$ and a specific choice of $\hat{\boldsymbol{F}}$. Earlier, Shan and Chen [19] proposed a forcing scheme which is applicable for coupled TEQs describing for example multiphase flows. A summary of various methods for force inclusion to the LBE which have been derived in the past, is given in [11].

Macroscopic boundary conditions ascribed to the TEQ have to be treated with boundary methods on the mesoscopic level. Update rules on the populations f_i near or at the boundary need to ensure the consistent behavior for the macroscopic flow quantities. Typical issues arise when comparing the accuracy order of the scheme within the bulk flow with the one obtained at the boundary. Solution approaches to match the orders of bulk and boundary methods are summarized in [11] and references therein.

Concerning the definitions of $\boldsymbol{\Omega}$, the application-selective enhancement of LBM, has caused a variety in models for the collision process. Starting from the Bhatnagar–Gross–Krook (BGK) collision model [20,21] with a single relaxation time, further models were proposed, extending the former to two relaxation times (TRT) [22] or multiple relaxation times (MRT) [23], regularizing it with modifications in the collision step (RLB) [24,25], or including viscosity regulating schemes via entropy corrections (ELB) [26]. Sub-branches and composites of these collision models have been recently developed and successfully applied to numerical benchmark tests and applications. For an extensively detailed comparison of several models the work of Coreixas et al. [27] should be considered.

Available implementations of boundary methods, forcing schemes and collision models within OpenLB are summarized in Section 3.

2.2. Target equations

For the sake of simplicity, the BGK collision scheme is used for the description of the necessary modifications for reaching the following TEQs. Further, mathematically rigorous convergence proofs under consideration of initial and boundary conditions are omitted in the discussion. General results for such topics can be found for example in [28,29].

Please cite this article as: M.J. Krause, A. Kummerländer, S.J. Avis et al., OpenLB—Open source lattice Boltzmann code, Computers and Mathematics with Applications (2020), https://doi.org/10.1016/j.camwa.2020.04.033.

The BGK version of (6) reads

$$f_i\left(\boldsymbol{x}+\boldsymbol{c}_i\triangle t, t+\triangle t\right) = f_i\left(\boldsymbol{x}, t\right) + \frac{\triangle t}{\tau}\left[f_i^{\mathrm{eq}}\left(\boldsymbol{x}, t\right) - f_i\left(\boldsymbol{x}, t\right)\right] + \triangle t\hat{F}_i, \tag{9}$$

where $\tau > 0.5$ is the relaxation time and f_i^{eq} denote the equilibrium populations. In a scaling limit, Eq. (9) can be directly associated to macroscopic TEQs via asymptotic expansion approaches proposed for example by Chapman–Enskog [30] or Grad [31]. A derivation of the requirements on f_i^{eq} is typically done via truncating a Hermite expansion series of the continuous Maxwell–Boltzmann equilibrium distribution function [11]. Equilibrium populations to obtain two specific types of TEQs are given below. Other classical equilibrium functions from kinetic theory are for example the Bose–Einstein or the Fermi–Dirac equilibrium functions.

2.2.1. Incompressible Navier–Stokes equations

The force-free incompressible Navier–Stokes equations (NSE) read

$$\begin{cases} \partial_t \boldsymbol{u} + \left(\boldsymbol{u}\cdot\nabla\right)\boldsymbol{u} & = -\frac{1}{\rho}\nabla p + \nu\triangle\boldsymbol{u} \\ \nabla\cdot\boldsymbol{u} & = 0 \end{cases}, \tag{10}$$

where $\boldsymbol{u} : \mathcal{X}\times\mathcal{I}\to\mathbb{R}^d$ is the macroscopic flow velocity, $p : \mathcal{X}\times\mathcal{I}\to\mathbb{R}$ denotes the pressure, $\nu > 0$ is a given kinematic viscosity and $\rho > 0$ is a given density. Pertaining to (10), the equilibrium populations for the force-free version of (9) are defined as

$$f_i^{\mathrm{eq}}\left(\rho, \boldsymbol{u}\right) = \rho w_i\left[1 + \frac{c_{i\alpha}u_\alpha}{c_s^2} + \frac{u_{\alpha\beta}\left(c_{i\alpha}c_{i\beta} - c_s^2\delta_{\alpha\beta}\right)}{2c_s^4}\right], \tag{11}$$

with w_i as lattice weights and c_s denoting the lattice speed of sound. For the sake of simplicity, a distinction between macroscopic flow variables appearing in (10) and their mesoscopically obtained counterpart is omitted. It is notable that, due to the linkage described above, the macroscopic flow velocity and density are approximately recovered through moments (3) induced by $\mathcal{G}^{\mathrm{inv}}$, where the simulated pressure obeys $p = c_s^2\rho$ and ν is coupled with τ via [11]

$$\nu = c_s^2\left(\tau - \frac{\triangle t}{2}\right). \tag{12}$$

In OpenLB the velocity sets *D2Q9*, *D3Q13*, *D3Q15*, *D3Q19* and *D3Q27* are implemented to approximate the solution of (10).

2.2.2. Advection–diffusion equation

The advection–diffusion equation (ADE)

$$\partial_t\chi + \nabla\left(\boldsymbol{v}\chi\right) - \nabla\cdot\left(B\nabla\chi\right) = 0 \tag{13}$$

governs the transport of a scalar-valued macroscopic quantity $\chi : \mathcal{X}\times\mathcal{I}\to\mathbb{R}$ through the velocity field $\boldsymbol{v} : \mathcal{X}\times\mathcal{I}\to\mathbb{R}^d$ under a diffusion coefficient $B > 0$. Typical advected quantities are density, energy or temperature. To recover (13), the equilibrium populations are defined as

$$k_i^{\mathrm{eq}}\left(\chi\right) = \chi w_i\left[1 + \frac{c_{i\alpha}v_\alpha}{c_s^2}\right]. \tag{14}$$

Thus, the macroscopic quantity χ in (13) is approximated through the 0th order moment of the populations k_i complying a force-free version of (9) [32]. By Chapman–Enskog analysis [11], it can be shown that the diffusion coefficient exhibits a similar coupling to the relaxation time as for the NSE equilibrium, namely

$$B = c_s^2\left(\tau - \frac{\triangle t}{2}\right). \tag{15}$$

OpenLB offers the velocity sets *D2Q5* and *D3Q7* for the ADE. In case a reaction term such as $R\chi$, where $R\in\mathbb{R}$ denotes the reaction coefficient, is added to the left-hand side of (13), the resulting advection–diffusion–reaction equation (ADRE) can be approximated via modifying the equilibrium in (14) with a corresponding additional term [33]. For couplings of the ADRE with the NSE through $\boldsymbol{v}\equiv\boldsymbol{u}$, OpenLB utilizes two lattices with respective populations, where the reaction part is realized via forcing schemes. Applications employing such couplings between NSE and equations akin to ADE are listed for example in Sections 4, 5 and 7.

3. Software development

3.1. Scope and overview

OpenLB is an object-oriented implementation of LBM. It is the first implementation of a generic platform for LBM programming which is sustainably maintained and shared with the open source community (GPLv2). Since the first release

Please cite this article as: M.J. Krause, A. Kummerländer, S.J. Avis et al., OpenLB—Open source lattice Boltzmann code, Computers and Mathematics with Applications (2020), https://doi.org/10.1016/j.camwa.2020.04.033.

in 2007 [9], the code has been continuously improved and extended across nine releases. The latest release is versioned as 1.3 [34] and dates from 20 May 2019. The OpenLB code is written in C++ and is used by application programmers as well as developers. Due to its ability to implement custom models, OpenLB supports complex data structures that allow simulations in complex geometries and parallel execution using MPI and OpenMP on high-performance computers. The source code uses the concepts of interfaces and templates, so that efficient, direct and intuitive implementations of the LBM become possible. The efficiency and scalability has been checked and proved by code reviews. Both, a user guide [35] and source code documentation, are available on the project website. An overview of the features offered by release 1.3 [34] as well as the application areas (physical characterization) are given in the following.

OpenLB supports a wide variety of lattice types for 2D and 3D simulations such as $D2Q9$, $D3Q15$, $D3Q19$ and $D3Q27$ in addition to various LBM collision models including BGK, TRT, MRT, ELB and RLB. Parallel grids are automatically generated and may contain both local and non-local boundary conditions as per Inamuro [36], He and Zou [37], Latt (regularized LB) [24,25] and Bouzidi et al. [38]. OpenLB's core feature set is completed with checkpointing for robust and resumable program executions, parallelism for shared and distributed memory platforms, various visualization options and straightforward coupling to external tools for pre- and post-processing.

This flexible toolbox enables application in a broad set of areas (physical characterizations) such as incompressible Newtonian and non-Newtonian fluid flows, multiphase and multicomponent flows, light and thermal radiation as well as thermal flows, particulate flows using both Euler–Euler and Euler–Lagrange (with resolved and sub-grid scale models), turbulent flow models (large eddy models (LES) and wall function approaches) and porous media models. Table 1 provides a collection of currently implemented LB features. Note that the summary herein focuses on the most regularly used features. The complete list, including dependency graphs can be accessed via the DoxyGen documentation [4]. Further, each release comprises a collection of application examples. An overview table of the currently included set of examples is available online [35].

3.2. Implementation

OpenLB is written using an object-oriented programming paradigm where data is abstracted into objects that in turn provide methods, both for exposing and regulating data access as well as for performing data manipulations. Inheritance and virtual method call resolution is utilized to provide generic interfaces. Class templates are used to write floating-point-type agnostic code. Furthermore, template meta-programming is used to select optimized code paths for certain LB models in addition to enabling expressive compile time data layout descriptions.

The OpenLB library provides modular classes to be used in the development of applications that solve transport problems using LBM. A typical application consists of the following steps:

1. Initialize the simulation by defining a converter between physical and lattice units as well as the underlying LB model.
2. Prepare the geometry by discretizing the analytical problem as a regular lattice (parallel meshing) and assigning material numbers to individual cells.
3. Prepare the lattice by binding *dynamics* (cf. Section 3.2.1) to material numbers according to the specific boundary and bulk behavior required to model the problem.
4. Start the main simulation loop controlled by a timer.

Parallel meshing is implemented by using an octree data structure to decompose the cuboidal bounding volume of the simulation geometry into a hierarchy of blocks [39]. Based on this structure blocks are either discarded if they contain no geometry or shrunk where possible to further reduce memory consumption. For each block the number of fluid and boundary voxels are computed but not stored. The distribution of the resulting sparse set of blocks between the available computation units is handled in a separate load balancing step (see Section 3.2.4). Then, the actual meshing is done and the mesh with the voxels is generated and stored in parallel. The mesh generation process is summarized in Fig. 1.

The main loop may execute additional steps besides the necessary call to collide (7) and stream (8) operations such as updating boundary conditions or logging performance statistics. Further possibilities are convergence checks and further processing of simulations results via *functors* (cf. Section 3.2.2) to e.g. calculate and plot error norms or VTK output for later visualization. All of these steps are supported by a rich set of utility classes.

Please cite this article as: M.J. Krause, A. Kummerländer, S.J. Avis et al., OpenLB—Open source lattice Boltzmann code, Computers and Mathematics with Applications (2020), https://doi.org/10.1016/j.camwa.2020.04.033.

Table 1
Currently supported, essential LB model features in OpenLB (release 1.3 [34]) under physical categorization with name and reference. Further information on implementations and feature linkage is available online [4] via respective keyword search within the DoxyGen documentation.

Category		Name	Reference	Keyword
Velocity sets		*D2Q5*, *D2Q9* *D3Q7*, *D3Q13*, *D3Q15*, *D3Q19*, *D3Q27*		`latticeDescriptors`
Collision dynamics	NSE	Bhatnagar–Gross–Krook (BGK)	[20,21]	`BGK`
		Regularized lattice Boltzmann (RLB)	[24]	`RLB`
		Two-relaxation-times (TRT)	[22]	`TRT`
		Multiple-relaxation-times (MRT)	[23]	`MRT`
		Entropic lattice Boltzmann (ELB)	[26]	`entropic`
	ADE	Bhatnagar–Gross–Krook (BGK)	[40]	`advectionDiffusionBGK`
		Regularized lattice Boltzmann (RLB)	[41]	`advectionDiffusionRLB`
		Two-relaxation-times (TRT)	[42]	`advectionDiffusionTRT`
		Multiple-relaxation-times (MRT)	[43,44]	`advectionDiffusionMRT`
		Source term	[45]	`sourcedAdvectionDiffusion`
Force models		Shan–Chen	[46]	`forcedShanChen`
		Guo	[47]	`forced`
		Exact difference method	[48]	`forcedKupershtokh`
Multiphysics couplings		Shan–Chen (multicomponent/multiphase flow)	[19]	`shanChen`
		Free energy model (multicomponent flow)	[49]	`freeEnergy`
		Boussinesq approximation (thermal fluid flow, laminar)	[50]	`navierStokesAdvectionDiffusion`
		Boussinesq approximation and Smagorinsky LES (thermal fluid flow)	[51]	`smagorinskyBoussinesq`
		Momentum exchange (particulate flow)	[52]	`ladd`, `particle`
		Discrete element method (DEM) (particulate flow, one-way/two-way coupling)	[53,54]	`particle`
Turbulence models		Approximate deconvolution model (ADM)	[55,56]	`ADM`
		Wall-adapting local eddy-viscosity (WALE) model	[57,58]	`WALE`
		Consistent Smagorinsky model	[59]	`conSmagorinsky`
		Consistent strain Smagorinsky model	[59]	`conStrainSmagorinsky`
		Dynamic Smagorinsky model	[60]	`dynSmagorinsky`
		Shear-improved Smagorinsky model	[61]	`shearSmagorinsky`
		Smagorinsky model	[62]	`smagorinsky`
		Smagorinsky model with van Driest damping	[63]	`externalTauEffLES`, `computeVanDriest`
		Shear-improved Smagorinsky model with Kalman filter	[61,64]	`kalmanShearSmagorinsky`
Other transport models		Porous media	[65,66]	`porous`, `guoZhao`
		Power law	[67,68]	`powerLaw`
		Homogenized lattice Boltzmann method (HLBM)	[69]	`hlbm`
		Radiative transport lattice Boltzmann method (RTLBM)	[70,71]	`rtlbm`, `radiative`
Boundary methods	Wet-node	Regularized	[25]	`createLocalBoundary`
		Finite difference (FD) velocity gradients	[72,73]	`createInterpBoundary`
		Inamuro	[36]	`inamuro`
		Zou–He	[74]	`zouHe`
		Non-linear FD velocity gradients	[72,73]	`createExtendedFdBoundary`
		LES wall function	[63,75]	`wallFunction`
	Link-wise	Fullway bounce-back	[76]	`bounceBack`
		Bouzidi (first order)	[38]	`bouzidi`
		Slip, symmetry	[77]	`slip`
		Periodic	[78]	`setPeriodicity`
	Advection–diffusion	Temperature boundary	[25]	`advectionDiffusionBoundary`
		Convection boundary	[79]	`advectionDiffusionBoundary`

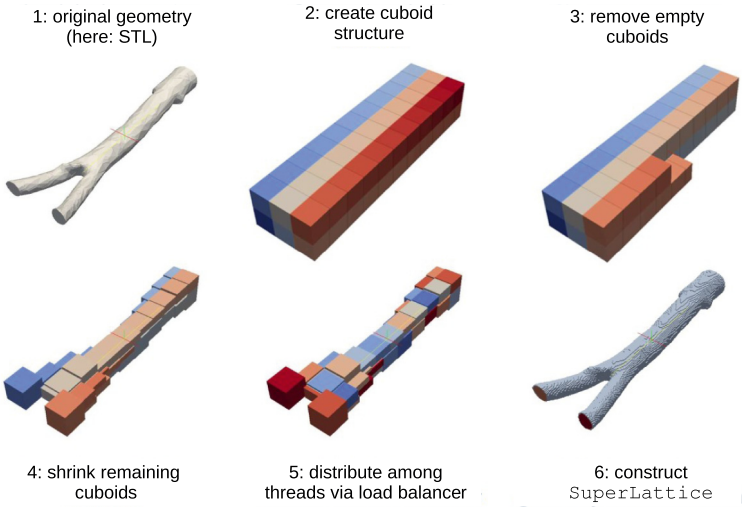


Fig. 1. Sketch of the parallel meshing in OpenLB, containing six steps for generating the final mesh geometry from a given STL file.

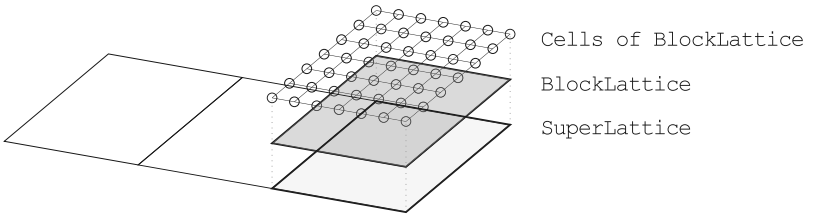


Fig. 2. Data structure in OpenLB: `BlockLattice(2,3)D` instances build a `SuperLattice(2,3)D` to model higher level software constructs like multi-block and parallelized lattices.

3.2.1. Data structure design

At the highest level, OpenLB's class hierarchy is separated into classes for problems in two and three spatial dimensions, respectively. Within each set of 2D or 3D classes it is distinguished between super- and block-level structures (see Fig. 2). The composition of multiple block-level structures into a super-level class, referred to as `SuperLattice(2,3)D` where `(2,3)` denotes that the block is either two or three dimensional, fulfills multiple purposes [16], i.e. decomposing complex simulation domains enables sparse memory consumption and allows for a simple and composable implementation of the LB algorithm on regular blocks as well as efficient data parallelism for distributed computation.

The core of this block structure is an efficient array-like construct called `BlockLattice(2,3)D`. This *block lattice* class represents a cuboidal subset of the underlying regular and homogeneous lattice $\mathcal{X}_h$ with direction-independent grid spacing $h \in \mathbb{R}^+$. As such, this class provides the scaffolding for implementing an LB algorithm in a very traditional sense, i.e. splitting the LBE into collision (7) and streaming (8) operations as described in Section 2. As the subgrid modeled by a single block lattice is very simple, the implementation of these steps is straight forward. Note that the distinction between super- and block-level structures is tied directly to the implementation of OpenLB's hybrid parallelization scheme detailed in Section 3.2.4. `BlockLattice(2,3)D` maintains a `Cell` instance for each node of its assigned subgrid block.

The `Cell` structures are the central interface for accessing the data tied to a given lattice coordinate. The specific combination of populations and additional fields exposed via this interface is described using so called descriptors (see Section 3.2.3). Within the simulation loop all these cells are processed iteratively to perform a local collision step followed by a non-local streaming step. The streaming step is independent of the choice of LB dynamics and remains invariant. On the other hand, the collision step determines the physics of the model and can be configured by assigning a `Dynamics` object to each `Cell`. In this manner, it is easy to implement inhomogeneous fluids which use a different type of physics from one cell to another. For each time step, the collision and streaming step can be executed as two separate loops over all cells or fused into a single iteration. For many applications the fused method is preferable as its dedicated swapping technique preserves the locality of the stored data and avoids costly reloading of data into the cache in a second loop [16].

3.2.2. Functor concept

While `BlockLattice(2,3)D` and its surroundings implement both data storage and manipulation, all further processing of simulation results is encapsulated by so called read-only *functor* classes. OpenLB provides a rich set of such objects to e.g. calculate moments, aerodynamic coefficients and error norms.

Please cite this article as: M.J. Krause, A. Kummerländer, S.J. Avis et al., OpenLB—Open source lattice Boltzmann code, Computers and Mathematics with Applications (2020), https://doi.org/10.1016/j.camwa.2020.04.033.

Roughly speaking, a functor is a class that behaves like a function. Objects of a functor class perform computations by overloading the `operator()`. The ultimate purely virtual base class of all functors is called `GenericF` which enables generic usage of all functors. At a high level functors may be grouped into lattice functors that are defined for each cell, analytical functors that are defined on the full analytical simulation domain and indicator functors used to e.g. model subsets of both lattices and geometries.

As functor classes provide overloads for C++'s arithmetic operators, all existing functors may be easily composed into more complex evaluations via *functor arithmetic*. In fact, functors such as `SuperRelativeErrorLpNorm(2,3)D` are implemented as the composition of lower-level L_p-norms and arithmetic functors.

3.2.3. Descriptor concept

Every LB-based simulation can be characterized by a set of constants such as the modeled spatial dimension D, the number of neighbors Q in the underlying regular grid, the weights w_i used to compute equilibrium distributions or the lattice speed of sound c_s. Due to OpenLB's goal of offering a wide variety of LB models to address many different kinds of transport problems, the constants are not hard-coded throughout the code base but rather maintained in compile-time data structures. Any usage of these constants can then refer to the characterizing *descriptor* data structure.

As an example, Listing 1 provides the full definition of the *D2Q9* descriptor including all constants. These constants are exposed via an adaptable set of free functions templated on the descriptor type as can be seen in Listing 2. This also opens up the possibility of *tagging* descriptors with empty fields to e.g. mark MRT descriptors requiring special treatment.

Listing 1: Definition of the `D2Q9` descriptor

```
template <typename... FIELDS>
struct D2Q9 : public DESCRIPTOR_BASE<2,9,POPULATION,FIELDS...> {
  D2Q9() = delete;
};

namespace data {

template <>
constexpr int vicinity<2,9> = 1;                                    // distance of neighbors

template <>
constexpr int c<2,9>[9][2] = {                                      // discrete velocities
  { 0, 0}, {-1, 1}, {-1, 0}, {-1,-1}, { 0,-1},
  { 1,-1}, { 1, 0}, { 1, 1}, { 0, 1}
};

template <>
constexpr int opposite<2,9>[9] = { 0, 5, 6, 7, 8, 1, 2, 3, 4 };

template <>
constexpr Fraction t<2,9>[9] = {                                    // equilibrium weights
  {4, 9}, {1, 36}, {1, 9}, {1, 36}, {1, 9},
  {1, 36}, {1, 9}, {1, 36}, {1, 9}
};

template <>
constexpr Fraction cs2<2,9> = {1, 3};                               // lattice speed of sound

}
```

All descriptors inherit the `DESCRIPTOR_BASE` class template instantiated using the desired lattice constants in addition to a list of fields. As many parts of the code do not depend on which specific descriptor is used, most classes and functions are templates that accept any user-defined descriptor type. This allows selection of descriptor specific optimizations via plain template specializations. The main area in OpenLB's code base where this possibility is used is the automatic selection of optimized collision step implementations. These optimized implementations are mainly obtained by common subexpression elimination (CSE).

Listing 2: Free functions for accessing descriptor constants

```
using T = double;
using DESCRIPTOR = descriptors::D2Q9<>;

// number of discrete velocities
const int q = descriptors::q<DESCRIPTOR>();                         // == 9

// second discrete velocity vector
const Vector<int,2> c1 = descriptors::c<DESCRIPTOR>(1);             // == {-1,1}

// weight of the first discrete velocity
const T w = descriptors::t<T,DESCRIPTOR>(0);                        // == 4./9.
```

Please cite this article as: M.J. Krause, A. Kummerländer, S.J. Avis et al., OpenLB—Open source lattice Boltzmann code, Computers and Mathematics with Applications (2020), https://doi.org/10.1016/j.camwa.2020.04.033.

The descriptor concept is tightly coupled to the definition of the cells that make up the block lattice. The `Cell` class locally maintains the data for all fields described by a descriptor (discrete populations in addition to e.g. an external force) and as such implements a collision-optimized *array of structures* memory layout. In order to determine the locations of individual fields inside a cell's memory one has to calculate offsets by accumulating the preceding field sizes. OpenLB's descriptor field concept allows for doing this statically and safely during compile time via template meta-programming.

Descriptor fields are defined as the parametrization of a multilinear function on the foundational constants D and Q of each descriptor, i.e. each field describes its size as a function

$$f : \mathbb{N}_0^3 \to \mathbb{N}_0, (a, b, c) \mapsto a + bD + cQ. \tag{16}$$

Parameters of such a field functions are expressed as non-type template parameters of the `DESCRIPTOR_FIELD_BASE` template. As per Listing 3, this allows for one-line definitions of field types.

Listing 3: Definition of various descriptor fields

```
// (Field size parametrized by: Cs + Ds*D + Qs*Q)      Cs Ds Qs
struct POPULATION       : public DESCRIPTOR_FIELD_BASE<0, 0, 1> { };
struct FORCE            : public DESCRIPTOR_FIELD_BASE<0, 1, 0> { };
struct VELOCITY         : public DESCRIPTOR_FIELD_BASE<0, 1, 0> { };
struct OMEGA            : public DESCRIPTOR_FIELD_BASE<1, 0, 0> { };
```

Lists of such types can then be passed around via variadic template arguments. This allows for compile-time handling of fields in a manner that is both flexible and consistent across all descriptors.

Listing 4: Accessing automatically determined field locations

```
// D2Q9 descriptor extended by three additional fields
using DESCRIPTOR = D2Q9<FORCE,VELOCITY,OMEGA>;

// Check whether DESCRIPTOR contains the field VELOCITY
DESCRIPTOR::provides<VELOCITY>();                                    // == true

// Get cell—local memory location of the VELOCITY field
const int offset = DESCRIPTOR::index<VELOCITY>();                    // == 11

// Get size of the descriptor's VELOCITY field
const int size = DESCRIPTOR::size<VELOCITY>();                       // == 1
```

OpenLB provides a memory-safe interface for interacting with field data on top of the index-based functions illustrated by Listings 4 and 5.

Listing 5: Higher level interface for accessing a cell's field data

```
// Get a pointer to the memory location of a cell's force vector
double* force = cell.getFieldPointer<FORCE>();

// Read a cell's force vector as an OpenLB vector value
Vector<double,2> force = cell.getField<FORCE>();

// Set a cell's force vector to zero
cell.setField<FORCE>(Vector<double,2>(0.0, 0.0));
```

In general, this usage of expressive compile-time data structures for providing generic access to lattice constants and memory locations has proven itself to be a useful tool for writing code that is readable and extensible while providing optimal performance. This is achieved due to the possibility of using C++ templates for writing abstract code that is fully resolved during compile-time.

As another benefit, most library and application code can be written without strict dependencies on a specific memory layout, which opens up the possibility of transparently implementing different layouts and streaming schemes. To expand, while OpenLB's latest release [34] uses a *array of structures* layout with the SWAP propagation pattern [80], this can be changed to a *structure of arrays* layout with a SSS pattern [81] without adapting anything but the `Cell` and field storage implementation details.

Please cite this article as: M.J. Krause, A. Kummerländer, S.J. Avis et al., OpenLB—Open source lattice Boltzmann code, Computers and Mathematics with Applications (2020), https://doi.org/10.1016/j.camwa.2020.04.033.

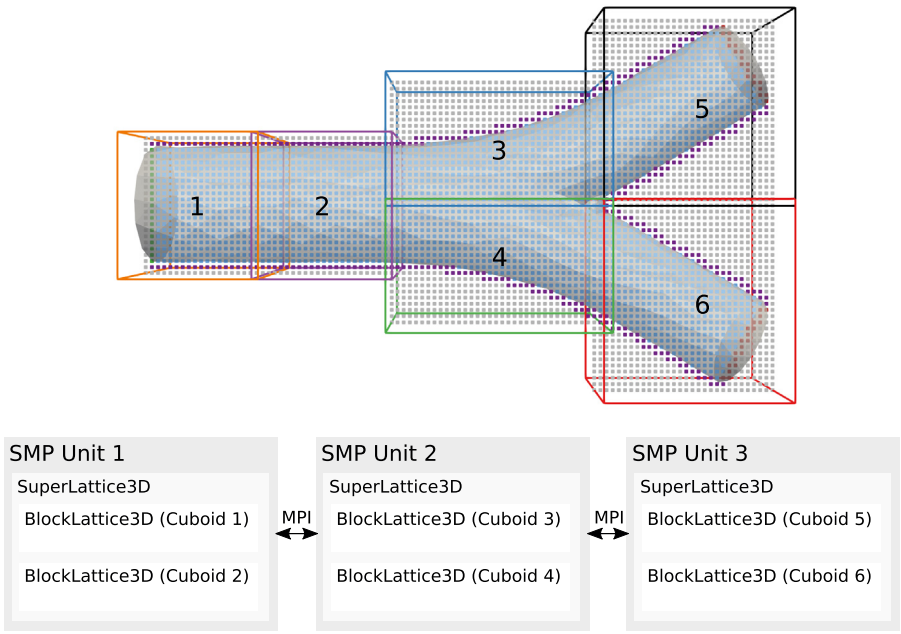


Fig. 3. Architecture of OpenLB's parallel data structure. Exemplary illustration of the distribution process of `BlockLattice(2,3)D` instances over the available symmetric multiprocessing (SMP) units.

3.2.4. Hybrid parallelization

Efficient parallelization of the LBM algorithm on distributed computing platforms that are made up of individual shared memory systems is non-trivial, as such a distributed system combines the requirements of two distinct computation architectures.

OpenLB's hybrid parallelization concept [82,83] follows the classical approach of partitioning the data used to model the simulation domain according to its geometrical origin. The resulting elements of the spatial block decomposition in the form of `BlockLattice(2,3)D` instances are then distributed over the available symmetric multiprocessing (SMP) units as is illustrated in Fig. 3. As a consequence, the set of block lattices maintained by the `SuperLattice(2,3)D` instance specific to each shared memory unit is disjoint from all other nodes of the distributed system—with the exception of overlap areas required for synchronization. For this exchange of data between the SMP units a communication-based paradigm is employed.

The most time demanding steps in LB simulations are the collision and the streaming. Since the collision step is purely local and the streaming step only requires data of the neighboring nodes, parallelizing by spatial domain partitioning in this fashion leads to low communication costs and is therefore efficient. Assignment of block lattices to individual SMP unit is handled by a `LoadBalancer` for which different implementations are available. In addition to a straight forward heuristic implementation that balances the total number of cells per computation node, sophisticated graph partitioning algorithms can be applied to minimize the communication between nodes [84].

The actual implementation of this hybrid concept in OpenLB utilizes MPI for communications between SMP units while processing of individual block lattices is parallelized via OpenMP—i.e. the overlap between any two instances of `BlockLattice(2,3)D` is synchronized by explicitly communicating it via MPI primitives. Collision and streaming loops within a single block are automatically parallelized across all cells by appropriate OpenMP pragmas.

3.2.5. Performance results

As LBM simulations are bound by the available memory bandwidth it is useful to compare how close the actual performance gets to this theoretical limit. While an ideal implementation would transfer exactly $B_{\text{cell}} := 2QB_{\text{fpt}}$ bytes of memory per cell (where `Q` is the number of discrete velocities and B_{fpt} is the number of bytes per floating point number), in practice one requires an additional memory transfer to distinguish between e.g. boundary and bulk cells. Due to this additional transfer a simplified model for the theoretical performance limit in MLUPs is given by $\overline{\text{MLUPs}} := \overline{B}/(1e6B_{\text{cell}})$ where $\overline{B}$ is the maximum bandwidth of the targeted platform in bytes per second and $B_{\text{cell}} := 2QB_{\text{fpt}} + 4$ is the number of bytes that is transferred per cell. Table 2 compares the lid driven cavity performance of OpenLB 1.3.1 to both, this theoretical model and an implementation of the SSS pattern [81] for the upcoming release 1.4 as well as a hand-optimized reference implementation.

Further performance evaluations of OpenLB's hybrid parallelization concept are available in [82] and [83], while [84] discusses the impact of graph-based partitioning.

Please cite this article as: M.J. Krause, A. Kummerländer, S.J. Avis et al., OpenLB—Open source lattice Boltzmann code, Computers and Mathematics with Applications (2020), https://doi.org/10.1016/j.camwa.2020.04.033.

Table 2
Selected double precision performance results on a Intel Core i7-4790K (4 cores, theoretical limit w.r.t. STREAM bandwidth measurements, $\sim$ 16.7 GiB/s). Comparison of the latest release [34] (OpenLB 1.3.1) and the upcoming release (OpenLB with SSS) which features the SSS pattern [81].

Example	OpenLB 1.3.1	OpenLB with SSS	Experimental	Theoretical
LDC, D2Q9, 1000^2	112 MLUPs	115 MLUPs	120 MLUPs	121 MLUPs
LDC, D3Q19, 200^3	26 MLUPs	46 MLUPs	52 MLUPs	58 MLUPs

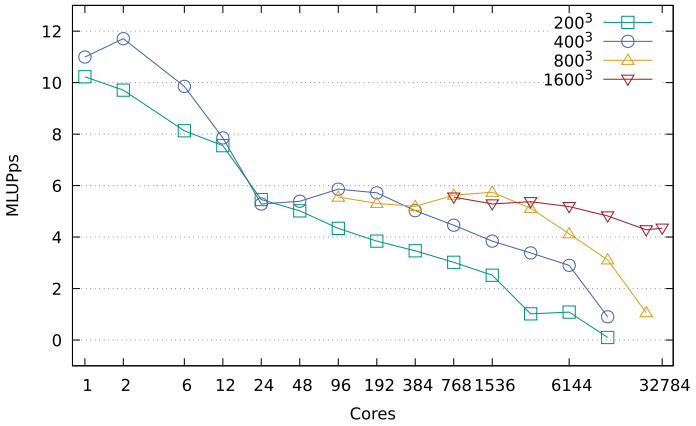


Fig. 4. Magnus performance results. MLUPps (mega lattice updates per processing unit and second) as a function of cores for several resolutions.

Recently in March 2019, the performance of OpenLB was evaluated for realistic scenarios at the Pawsey Magnus super computer (TOP500 at 358, November 2018 [85]). Up to 32,784 of the available 35,712 cores were deployed, obtaining 142,479 MLUPs, i.e. about 142 billion fluid cells were updated in one second. Fig. 4 illustrates the performance of the common lid-driven cavity example at various resolutions and visualizes the computational efficiency and scalability of OpenLB. Hence, the conducted tests confirm its suitability for applications in solving large-scale fluid flow problems.

3.3. Open source community

Documentation. Besides the user guides and technical reports [4], further details concerning the computational concepts realized in OpenLB can be found in [10,16]. An overall code documentation generated by DoxyGen is also available. Moreover, scientific articles published since the initial launch of OpenLB, directly utilizing the numerical framework or using it as a reference, both, with and without the participation of active developers, are summarized online [4]. Additionally, a complete list of the current core programmers as well as former contributors is provided and regularly updated on the webpage [4].

Awards. OpenLB has played a major role in three successful award proposals. For the Itanium Innovation Awards, OpenLB was recognized twice for presenting solution strategies for numerical simulations of human respiratory flows, and was awarded in the category of Humanitarian Impact as Finalist in 2007 and as Honorable Mention Finalist in 2009. The contest is organized by the Itanium Solutions Alliance, a global consortium of notable hardware, operating system and application vendors. A detailed presentation of the subject of the two proposals as well as a press release for the 2009 award can be found in [16]. In 2011 OpenLB's pre-processing approach for innovative patient-specific intranasal flow simulations, which was later published by Krause et al. [39], won the Mimics Innovation Award.

Spring school software lab. Grown out of active discussions of fundamentally theoretical as well as technical nature within the OpenLB forum, the first spring school software lab on OpenLB was organized in 2017 in Hammamet (Tunesia). The intention of creating an international platform with courses for beginners in LBM, as well as researchers and developers from academia and industry was fulfilled with a response of 49 participants from 14 countries. Continuations, 2018 in Karlsruhe (Germany) and 2019 in Mannheim (Germany), followed and a subsequent fourth spring school for 2020 in Berlin is planned. The success of the event is based on the interlaced educational concept of comprehensive and personal courses offered by the core developer team, the local organizing group as well as professional guest lecturers within the field of LBM.

4. Multicomponent flows

One of the most important and popular areas of applications for LBM is on multiphase and multicomponent flows due to their wide-ranging relevance in natural phenomena and engineering processes. Examples in nature include

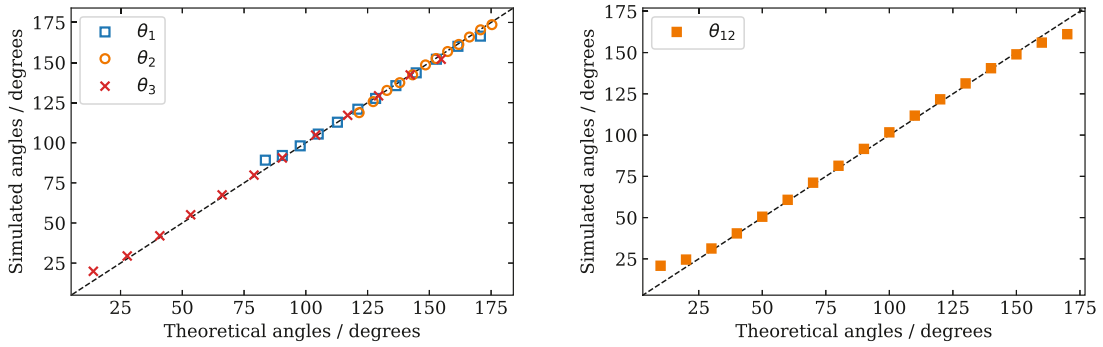


Fig. 5. Benchmark tests comparing simulated Neumann angles θ_i between three fluids (left) and contact angles θ_{ij} (right) to the theoretical values calculated from the κ_i and h_i parameters. Only the contact angles between fluids 1 and 2 are shown. This is because the same results are obtained regardless of which fluid components are used, due to the symmetry between the three fluids.

the formation of clouds, rain and snow, and the flow of the components of blood through blood vessels. Meanwhile, engineering applications include the combustion of gas or oil, and enhanced oil recovery using air and water.

The keywords multiphase and multicomponent are often used interchangeably, but in principle they describe two rather different concepts. Multicomponent flows consist of different substances that cannot interchange, such as oil and water, whereas multiphase flows consist of multiple states of the same substance, such as water and vapor [11]. The distinction in practice becomes less well-defined because many flows of interest are in fact a mixture between the two, where the liquid and gas phases can separately comprise of several components.

Several different approaches have been proposed for simulating multiphase and multicomponent fluids with the LBM. These include the color-gradient [86], Shan–Chen (or pseudopotential) [19], free energy [87], and phase-field [88] methods. A critical comparison of the strengths and weaknesses of these approaches will not be given here. Instead, the reader is referred to a number of articles, reviews and books that have covered this issue [89–92]. In the long term, both the Shan–Chen and free energy models are planned to be supported in OpenLB, which are perhaps the most popular approaches in the community. This article will consider the free energy approach for multicomponent flows, as this has been the focus of recent development.

4.1. Free energy model

In the free energy approach, an expression is first selected to describe the free energy of the system. The free energy model which has been implemented in OpenLB is due to Semprebon et al. [49] where the free energy of the system reads

$$F = \int_V \sum_i \left[\frac{\kappa_i}{2} C_i^2 (1 - C_i)^2 + \frac{\alpha^2 \kappa_i}{2} (\nabla C_i)^2 \right] \mathrm{d}V - \int_{\partial V} \sum_i h_i C_i \, \mathrm{d}S. \tag{17}$$

Here V denotes the domain considered, κ_i, h_i and α are model parameters, and C_i refers to the local concentration of the fluid component indexed with $i = 1, 2, 3$, respectively. The first term in (17) is needed to realize three different bulk fluids. The second term describes the interfaces between the three fluids in the system. Finally, the third term corresponds to interactions between the fluids and the solid surface. Hence, the above construction of F enables independently varying each of the appearing surface tensions, contact angles and the interfacial width by adjusting κ_i, h_i and α, respectively. These surface tensions and contact angles are benchmarked by comparing the simulated Neumann angles and contact angles to the theoretical values. The corresponding results are shown in Fig. 5. Additionally, while this approach is designed for three fluid components, it can also be used for two-component systems simply by assigning the third fluid concentration to be zero.

It is notable that the present method has a drawback in the sense that the different fluids are all assigned equal densities. Therefore this version of the model is limited to situations where the fluid densities are similar, or where the Reynolds number is sufficiently low. Consequently, its application to liquid–gas mixtures is limited. There are, however, approaches that could be undertaken to enable high density ratios between the fluids [93–95].

The present OpenLB implementation employs three LBE, with respective populations and equilibria. The first is used to recover the density $\rho = C_1 + C_2 + C_3$, capturing the NSE (10), with an additional force term, in the continuum limit. To distinguish the three fluid components and track their interfaces, at least two order parameters are needed, chosen as $\phi = C_1 - C_2$ and $\psi = C_3$. Describing the evolution of the order parameters over time, the second and third LBE are used to reproduce the governing Cahn–Hilliard equations,

$$\partial_t \phi + \nabla \cdot (\phi \boldsymbol{u}) = M_\phi \Delta \mu_\phi, \tag{18}$$

Please cite this article as: M.J. Krause, A. Kummerländer, S.J. Avis et al., OpenLB—Open source lattice Boltzmann code, Computers and Mathematics with Applications (2020), https://doi.org/10.1016/j.camwa.2020.04.033.

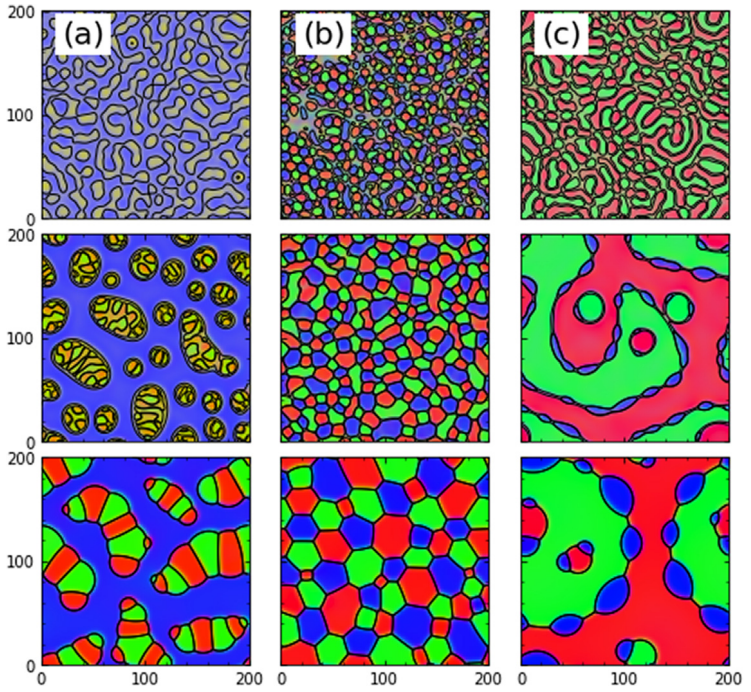


Fig. 6. Phase separation for initial concentrations: (a) $C_1 = C_2 = 0.25$, $C_3 = 0.5$; (b) $C_1 = C_2 = C_3 = 1/3$; (c) $C_1 = C_2 = 0.4$, $C_3 = 0.2$.
Source: Reproduced from Ref. [49].

$$\partial_t \psi + \nabla \cdot (\psi \boldsymbol{u}) = M_\psi \Delta \mu_\psi , \tag{19}$$

where M_ϕ and M_ψ are the respective mobility parameters. The free energy described in Eq. (17) enters the equations of motion of the fluid through the force term present in the NSE as well as the chemical potential terms μ_ϕ and μ_ψ in (18) and (19), respectively.

Several boundary conditions have also been implemented in OpenLB that can be used in conjunction with the free energy model. As stated previously, the contact angles are controllable at solid surfaces. This is done by adjusting the relative affinity for each fluid to the surface. In addition, inflow and outflow boundaries with constant pressure or velocity can be prescribed by using a multicomponent extension to the method of Zou and He [74].

To illustrate the capabilities of the model, several applications that have been simulated using this model are described below.

4.2. Phase separation

The herein described free energy model has been used in the introductory paper by Semprebon et al. [49] to study the phase separation dynamics of ternary fluid systems. By beginning with a mixed fluid and allowing it to separate, the manner by which the separation occurred was shown to depend strongly upon the initial fluid concentrations (Fig. 6). These observations were found to be consistent with previous results obtained by using other methods.

4.3. Flow-focusing microchannels

Double emulsions consist of droplets of one fluid contained within droplets of another. These have various applications including encapsulation of drugs for targeted delivery. Microfluidic channels can be used to produce these droplets with a controlled size. One such channel, consisting of two flow-focusing junctions, has been recently investigated by Wang et al. [96]. The resulting droplet morphology produced is highly dependent on the input flow rates and the channel geometry. A case where a double emulsion is created has been reproduced using OpenLB, as shown in Fig. 7.

5. Particle-laden flows

The simulation of particles submerged in a fluid has a wide range of applications, as aerosols and suspensions have to be considered in processes like protein purification [97], exhaust aftertreatment systems or even in the respiratory system [53]. Given the extend of relevant parameter ranges, one has to choose an appropriate modeling approach.

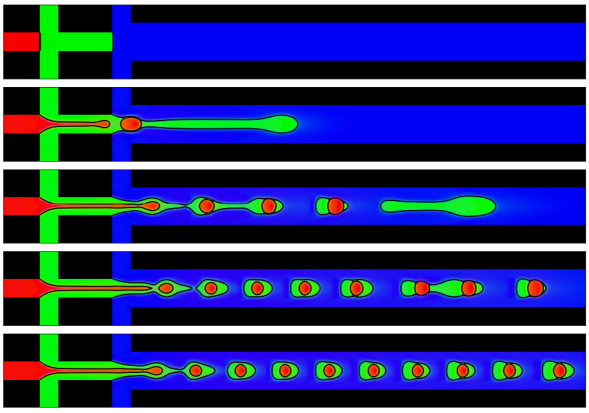


Fig. 7. Flow-focusing channel simulation for the first $4 \cdot 10^5$ time-steps. The specific flow rates prescribed at the inflow boundaries effectuate the production of a double emulsion.

Therefore, the acting forces, the operating length scales, the necessary degree of detail, the coupling between phases and the solid volume fraction have to be considered.

This eventually leads to two commonly distinguished classes of approaches for particle-laden flow simulations. A categorization by the flow model mainly distinguishes between an Eulerian and a Lagrangian description of the particle phase. The Eulerian approach describes the solid component based on its particle concentration, continuously modeled by an ADE. For the Lagrangian approach, on the other hand the trajectory of each discrete particle is computed according to Newton's law of motion. While the carrier phase is usually modeled as an incompressible Newtonian fluid by the NSE, the respective simulation approaches are denoted as Euler–Euler and Euler–Lagrange method.

A further categorization by the interaction of the two components is based on the distinction between one-way, two-way and four-way coupling. While the first one exclusively models the fluid's influence on the particles, a two-way coupled system additionally accounts for the particle's feedback force on the fluid. A four-way coupling in turn also includes the modeling of inter-particle and particle–wall interactions. The necessary coupling method usually depends on the considered solid volume fraction.

As the simulation of particle-laden flows is currently a significant part of the ongoing research within the framework of OpenLB, multiple related studies have been performed and several papers have been published. These efforts in each of the previously described categories are presented in the following.

5.1. Euler–Lagrange approach

Dilute solid component. One of the first efforts to incorporate particulate flows into OpenLB was done by Henn et al. [53] employing a Lagrangian formalism. Having an efficiently parallelizable fluid flow framework based on the fixed domain decomposition approach (Section 3.2.4) already at hand, the main focus of this initial work was on an adequate parallelization strategy for the particle phase. Next to convergence studies, performance tests were conducted using a simplified geometry of the human lungs. The implemented strategy was then applied to the simulation of time-dependent particulate flows of micro-particles in a patient-specific geometry of a human nasal cavity, including paranasal sinuses. After comparing the two parallelization strategies *load optimal* and *communication optimal* for the particle trajectory computation, the latter was found to be the most efficient under the assumption of a homogeneous particle distribution. With the application to the nasal cavity, specific deposition locations could be related to particle size, Stokes number, and the injection method. Eventually it was concluded that transient simulations are necessary when simulating particle-laden flows in the nasal cavities. Besides the application in [53], the Euler–Lagrange approach realized in OpenLB is suitable for particle-laden flows appearing in traditional filtration processes, where trajectory computation is necessary and the number of particles as well as the Reynolds number *Re* are typically large. As an example, the trajectory computation can be used for qualitative evaluation of dynamic cross-flow filtration. Fig. 8 visualizes a rotor-induced particle-laden flow, which is tangential to a filter membrane. The subsequent prevention of particle blocking to ensure an efficient and continuous process, can be assessed by efficiently computing particle trajectories.

Dense solid component. For high solid volume fractions, the impact of submerged particles on the fluid has to be taken into account, leading to a two-way or four-way coupled system. This is especially the case for comparably large particle depositions or regions of varying density within a liquid suspension. Therefore in Maier et al. [54], the Lagrangian framework introduced into OpenLB by Henn et al. [53] was extended by a mesoscopic coupling method. This was achieved by relating the corresponding drag forces to the occurring momentum exchange. The sedimentation of a single sphere for various radii was used to validate both the preexisting one-way coupling and the newly introduced two-way coupled

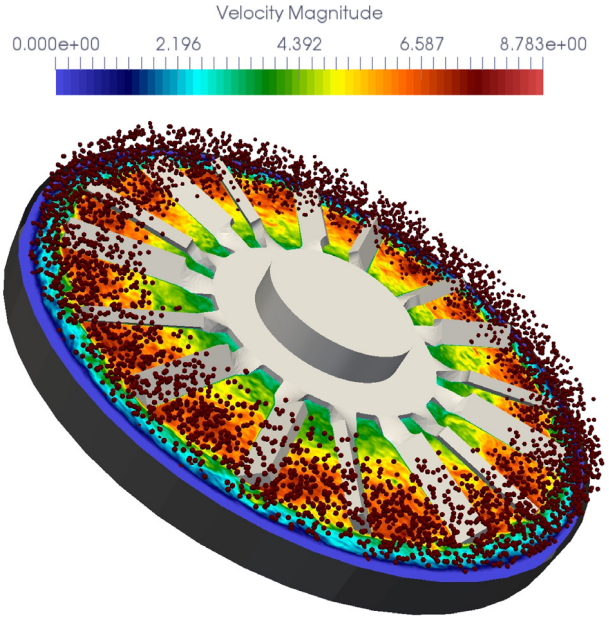


Fig. 8. Dynamic cross-flow filtration. OpenLB simulation of particle-laden flow with the Euler–Lagrange approach at $Re \approx 477000$ for 10000 particles.

system. Additionally, the sedimentation of 1800 particles in a cubical cavity was performed to qualitatively investigate the temporal evolution of the settling particle block. In order to investigate inter-particle collisions for the dense suspension, a contact force model was introduced and tested by colliding two spherical particles. Results were then compared to data from Kruggel-Emden et al. [98]. It was found that the relative errors in the dynamic and stationary part of the particle velocity in the one-way coupling decline for decreasing time step sizes with a fixed relaxation time. The experimental order of convergence of the particle phase was confirmed to be equal to one for the explicit Euler scheme applied for the calculation of the particle's velocity. Furthermore it was deduced that the contact forces are in very good agreement with the reference data.

Direct numerical representation. In the previously described schemes particles are considered to be perfect spheres as simplification, valid for a wide range of applications. In order to investigate arbitrarily shaped particles submerged in a viscous fluid, Krause et al. [69] proposed a *homogenized* LBM (HLBM) for particle-laden flows. The standard LBM was extended by a moving porous media approach, depicting the actual shape of the particle on the computational grid, by utilizing an adaptation of the velocity components in the equilibrium distribution function. Different scenarios, including the flow around a cylinder by Schäfer and Turek [99], the sedimentation of a single sphere, the drafting–kissing–tumbling (DKT) benchmark and the sedimentation of 24 particles with different shapes were simulated to test the method and to show its capabilities. The first two cases were used as benchmarks and delivered results agreeing well with literature data while exhibiting at least linear experimental order of convergence. The general behavior of DKT could be recovered. Ultimately it was argued that pressure fluctuation addressed in Ladd's approach [100] could be significantly reduced by means of the proposed approach. An extension of HLBM to 3D simulations along with a strategy for the treatment of arbitrary particle shapes e.g. taken from computer tomography (CT) scans was finally proposed by Trunk et al. [101]. Results of numerical experiments, validating the construction of a discrete particle representation, the particle dynamics and the corresponding forces were presented. The generation of physical parameters, like the moment of inertia and the volume from a surface representation of the particle, was validated by comparison with analytical solutions. Analogously to Krause et al. [69], the flow around a cylinder and a settling sphere were used as benchmark cases to verify the dynamics and computations of hydrodynamic forces. Furthermore, the 3D extension was applied to the simulation of 15 limestone particles based on geometry data generated from CT scans (Fig. 9). While a grid independence study yielded a quadratic convergence for the drag coefficient, the results of both benchmark cases were reported as agreeing well with existing literature. From differences in the average settling velocity of the limestone bodies, it could be concluded that the individual particle shape influences the macroscopic flow behavior of a suspension.

5.2. Euler–Euler approach

Dilute solid component. A detailed investigation of deposition patterns however leads to the necessity of much higher particle concentrations. Even with an appropriate parallelization strategy the approach of Henn et al. [53] already revealed

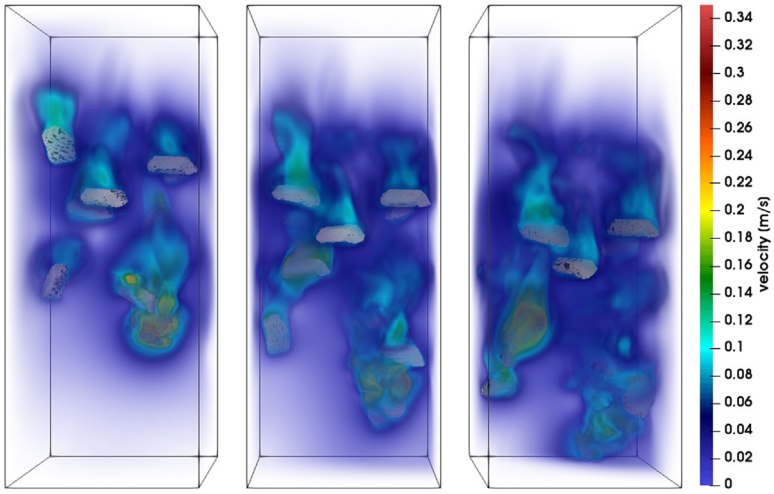


Fig. 9. Sedimenting limestone particles. Particles of arbitrary shape simulated with HLBM. *Source:* Reprint from Trunk et al. [101].

a great computational effort as it scales with the number of particles (or the square of the number of particles for four-way coupled systems). Since the computational costs of an Euler–Euler scheme solely scale with the resolution of the computational domain and not with the amount of particles, Trunk et al. [102] introduced the respective scheme into OpenLB. In this context, a stabilized extension to existing Euler–Euler formulations was proposed and the two-component system was applied to a simplified geometry of a human lungs bifurcation. Validation of the resulting scheme was done by comparing the numerical results to those determined by Lagrangian particle formulations, focusing on the deposition and its dependency on the Stokes number. It was concluded that the scheme leads to viable results regarding the deposition in the human lungs bifurcation. Depending on the Stokes number, the fraction of deposited particles was determined and the individual solutions were found to fit the respective physical model.

Dense solid component. An alternative approach to simulate dense particle-laden flows was proposed by Höcker et al. [103] by introducing an LBM for the volume-averaged Navier–Stokes (VANS) equation. Relying on an adaptation of the streaming step, which scales part of the density distribution function for the fluid phase, depending on the local fluid volume fraction, the approach refrains from requiring additional forcing terms. Four test cases with analytical solutions, two of which contain spatially or temporally fluctuating particle concentrations, were used to investigate the LBM's convergence to the VANS equations. The method was combined with an advection–diffusion LBM to obtain a simple multicomponent model, which was validated using a Rayleigh–Taylor instability test case. Finally, the VANS LBM's performance for particulate flows and the behavior in the hydrodynamic limit was investigated. The momentum equation of the VANS equations could be retrieved in the hydrodynamic limit by assuming small pressure fluctuations. A second order convergence was shown for the VANS LBM in the presence of a static particle-phase for both an explicit and an implicit forcing scheme. For the latter however, the temporal development of the concentration was reported as closely resembling other simulations of Rayleigh–Taylor instabilities. For a non-static case, the results were interpreted as agreeing well with an analytical solution. Eventually, it was concluded that the Euler–Euler VANS LBM accurately incorporates the desired coupling.

6. Turbulent fluid flows

Turbulent flows are characterized by a three-dimensional, unsteady and chaotic behavior. The increased vorticity, turbulent diffusion and dissipation are related to the turbulent energy transport. Furthermore, turbulence is a multiscale phenomenon, i.e. turbulence appears on a broad range of length and time scales [104]. Turbulent flows are present in many engineering applications, e.g. the flow around vehicles, mixing processes or flows in pipelines. The study of turbulence is therefore indispensable to explore both, negative and positive effects of turbulence on process variables. The complex multiscale nature of turbulence is challenging in numerical simulation and requires a fundamental knowledge of distinct physical model concepts. Three common approaches to model turbulent flows are direct numerical simulation (DNS), large-eddy simulation (LES) and Reynolds averaged Navier–Stokes equations (RANS).

DNS solves the Navier–Stokes equations for turbulent flows. This approach does not require turbulence modeling, but all time and length scales must be resolved. Although DNS is essential to studies of turbulence phenomena and model development, the increased computational effort renders the method not feasible for many engineering applications. In contrast, LES models the small scales and solely simulates the large energy containing scales. This reduces the numerical effort and changes the target equation. Based on an explicit or implicit filter, the filtered Navier–Stokes equations, which

Please cite this article as: M.J. Krause, A. Kummerländer, S.J. Avis et al., OpenLB—Open source lattice Boltzmann code, Computers and Mathematics with Applications (2020), https://doi.org/10.1016/j.camwa.2020.04.033.

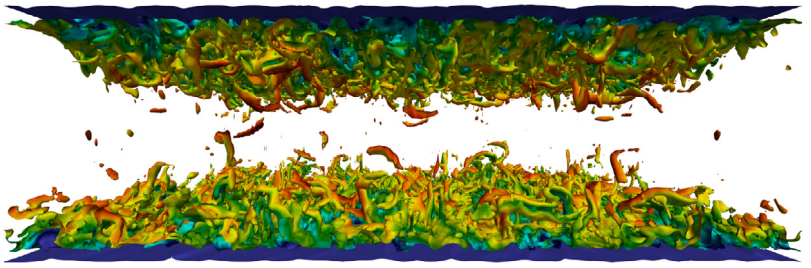


Fig. 10. Turbulent channel flow. Iso volume representation of the instantaneous vorticity field at friction Reynolds number $Re_\tau = 1000$.

are closed by sub-grid scale (SGS) model, are solved. The third group of models are using a statistical approach to access the turbulence. The RANS equations based on the Reynolds decomposition calculate mean quantities of the flow field. Statistical stationary flows can be solved quickly for high Reynolds numbers, but the high degree of modeling can significantly affect the accuracy.

The OpenLB community dealt with various aspects of turbulence simulations in recent years. Two frequently considered benchmark cases are illustrated in Figs. 10 and 11. The next sections summarize the results of turbulence research that have benefited from the features of OpenLB.

6.1. Study of wake flows

The first attempt to validate the turbulence capabilities of OpenLB was made by Nathen et al. [105] in 2013. They focused on the simulation of turbulent flows around moving boundaries of arbitrarily shaped obstacles. The validation of the MRT collision operator was performed in a two-dimensional double shear layer and the three-dimensional Taylor–Green vortex. Afterwards an explicit SGS model coupled to the MRT scheme was investigated for the test case of a flow around a square cylinder at Reynolds number $Re = 21{,}400$. Compared to experimental data, good agreement was reached for the flow field in the wake. The simulation of rotating cylinders showed that the MRT-LES scheme using an interpolated bounce back boundary was able to predict the lift reasonably well. However, the drag coefficient was overestimated compared to experimental data and reference simulations. An extension to three-dimensional flows showed a reasonable numerical stability. Furthermore, characteristic flow structures caused by rotating geometries could be identified in the wake flow.

Another study that deals with turbulent external flows was performed by Nadim et al. [106]. The flow around a NACA0012 airfoil was simulated and analyzed for different angles of attack. A BGK-LES with a standard Smagorinsky SGS model was used to obtain flow field details with acceptable calculation costs. The used half-way bounce back boundary scheme was sufficient to capture aerodynamic coefficients.

6.2. Turbulence modeling for large eddy simulations

Also turbulence modeling and implementation was performed with the aid of OpenLB. Nathen et al. [107] extended the approximate deconvolution method (ADM) for LBM by a viscosity dependent filter technique. The filter uses a time dependent coupling between resolved scales and a phase averaged strain-rate. The advantage of this filter is that the common adjustment parameters that often depend on the Reynolds number, the flow type and the grid resolution are not needed. The transfer function of the filter is able to dynamically introduce artificial viscosity in underresolved regions to correct the energy transfer. Firstly, they tested the selective ADM approach for different Reynolds numbers, resolutions and filter stencils in the context of decaying homogeneous isotropic turbulence (DHIT).

The results for filter stencils of second and third order matched the DNS reference data well, assuming that the energy containing scales are resolved. An underresolved LES simulation at higher resolutions showed that the third order stencil introduces more dissipation, which coincides with previous studies. A further analysis of the numerical viscosity in the spectral space validated the capabilities of the model to dynamically adapt for different grid resolutions and Reynolds numbers. The selective ADM was also benchmarked for wall-bounded turbulence. Therefor, the test case of the turbulent channel flow was studied for distinct Reynolds numbers and grid resolutions. The results showed good agreement to the DNS reference data. These promising findings approved the suitability of the selective ADM for simulating several types of turbulent flows.

Another important topic of turbulence modeling is wall-modeled LES for large-scale industrial applications. To study wall-bounded turbulent flows in highly underresolved grids, Haussmann et al. [75] proposed a BGK-LES model coupled to a wall function. The evaluation of different algorithm schemes were performed in the bi-periodic turbulent channel flow. Using an extrapolation scheme as velocity boundary condition combined with a wall function based on either the Musker equation or a three layer equation, yielded accurate results at $Re_\tau = 1000$. The simulation results of higher Reynolds friction numbers corresponded to the DNS reference data for first and second order turbulence statistics obtained by

Please cite this article as: M.J. Krause, A. Kummerländer, S.J. Avis et al., OpenLB—Open source lattice Boltzmann code, Computers and Mathematics with Applications (2020), https://doi.org/10.1016/j.camwa.2020.04.033.

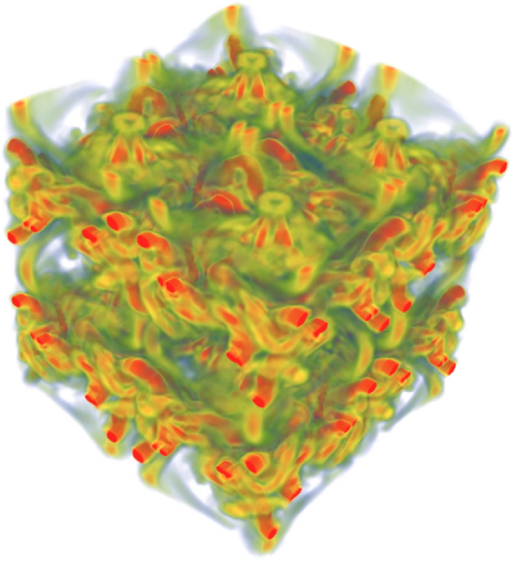


Fig. 11. Taylor–Green vortex flow. Volume rendering representation of the instantaneous vorticity field at Reynolds number $Re = 1600$. A recent implementation can be found in the example section of OpenLB 1.3 [34].

higher grid resolutions. This wall function approach was then applied to estimate the pressure drop in the geometry of a Coriolis mass flowmeter. The use of the wall model reduces the error by two orders of magnitude in comparison to a common no-slip boundary condition.

6.3. Collision operator comparison for direct numerical simulations

The suitability of different collision operators to simulate turbulent flows is of topical interest. These comparisons of different collision operators particularly benefit from the large open source framework OpenLB. Nathen et al. [108] investigated the accuracy and stability of three different collision schemes, namely BGK, MRT and RLB, for turbulent flows. The suitability to simulate DHIT was estimated with the Taylor–Green vortex flow. In underresolved settings the BGK collision operator was found to suffer from instabilities. The reasons are found in the amplification of nonphysical moments. However, mesh convergence and accurate results were achieved in a resolved setup. Applying MRT enhanced the stability in underresolved simulations. In contrast, higher resolutions at low Mach numbers lead to unstable and non-converging results. The RLB scheme was able to stabilize the simulation at each tested configuration but suffers from large numerical dissipation that reduces the accuracy. The effects for wall-bounded turbulence showed similar results. Although MRT was stable, velocity oscillations in the bulk affected the accuracy and mesh convergence at higher Reynolds numbers. The use of RLB led to a relaminarization of the flow at low Reynolds numbers as a result of the increased artificial viscosity contribution.

Based on these findings, Haussmann et al. [109] extended the study with two additional collision operators, namely TRT and ELB. The influence of the Mach number was investigated by applying acoustic scaling (AS) and diffusive scaling (DS) on DHIT in the TGV. For DS, the lattice Mach number Ma_L decreases with increasing resolution, whereas it is kept constant in AS leading to a non-vanishing compressibility error of order $\mathcal{O}\left(Ma_L^2\right)$ for weakly compressible simulations. The results for BGK agree with the previous study and emphasize the suitability for accurate DNS computations in the incompressible limit. BGK and TRT showed similar results for DS and AS at high Reynolds numbers, whereas at small Reynolds numbers, the constant truncation error influenced the accuracy. Employing the RLB with AS, the convergence speed was increased at high Reynolds numbers. MRT showed stable and accurate results using AS; the instabilities described by Nathen et al. [107] when using DS are only found at low Mach numbers. A comparison of the BGK and ELB results, exhibited that for the TGV neither instabilities for low resolutions are diminished, nor the accuracy is increased by the entropy-corrective extension. The authors concluded that the higher computational costs of DS and the diminishing influence of the truncation error at larger Reynolds numbers, vindicate a lattice velocity of $u_L = 0.1$ for high Reynolds number flows.

7. Thermal fluid flows

Spatial and temporal temperature distributions are of particular relevance in many areas of research and development, from mechanical engineering and process engineering to weather forecasting. Most of the problems that arise cannot be solved analytically, especially when complex domains or coupled thermal transport by turbulent convection are involved.

Please cite this article as: M.J. Krause, A. Kummerländer, S.J. Avis et al., OpenLB—Open source lattice Boltzmann code, Computers and Mathematics with Applications (2020), https://doi.org/10.1016/j.camwa.2020.04.033.

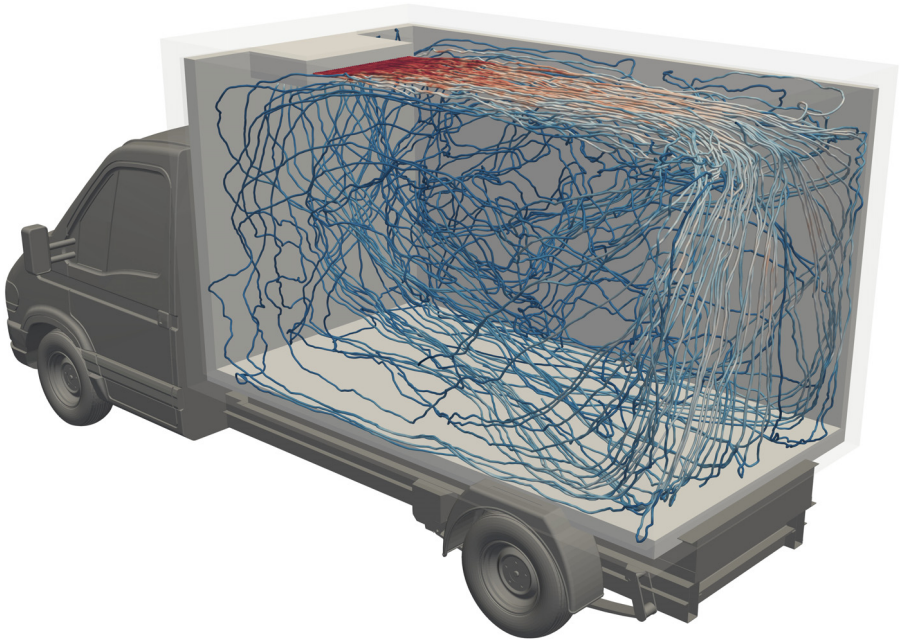


Fig. 12. Streamline representation of the simulated air flow inside the refrigerated vehicle.
Source: Reprint from Gaedtke et al. [51].

Thus, a number of numerical methods have been proposed to solve these problems by iterative algorithms. Besides the FEM and the FVM, much effort has also been put into solving various thermal transport problems with the LBM, see for example Krüger et al. [11] for an overview.

OpenLB follows the approach of multiple distributions functions to enable the simultaneous solution for the incompressible NSE and ADE for the temperature by the means of two corresponding distribution functions as described in Sections 2.2.1 and 2.2.2, respectively. In order to take natural convection into account in the simulations, a buoyancy force can be included in the coupling procedure. Usually this is done by the Boussinesq approximation. Gaedtke et al. [51] presented a rigorous validation of the model and its implementation in the open source code OpenLB. This validation study includes a grid convergence study showing that the scheme is second order in space by comparing it with the analytical solution of the porous plate problem. This is followed by a comparison with the benchmark of natural convection in a square cavity for different Rayleigh numbers between 10^3 and 10^{10} of laminar to turbulent natural convection. For the latter, the double distribution function algorithm is extended by a Smagorinsky subgrid model with a constant turbulent Prandtl number to take the subgrid contribution of thermal diffusion into account.

7.1. Application to refrigerated trucks

Subsequently, the developed and validated model is used for the investigation of more sustainable refrigerated trucks under consideration of the reduction of fuel consumption and emissions related to deep frozen food transports in Gaedtke et al. [51]. A comparison with measurements for the internal heat transport and flow inside of a refrigerated vehicle at $Re \approx 53,000$ (see Fig. 12) is also presented. To provide a deeper understanding of the refrigerated vehicle's insulation processes, the vehicles insulation walls are included in the simulations by conjugate heat transfer. This setup allows for precise simulations of velocity and temperature distributions within the cooled loading area and heat flux through the insulation walls. The authors find good agreement with the respective reference values measured during an open door test at four characteristic velocity and 13 temperature positions in the truck.

In a later article [110], simulations with OpenLB are used to investigate two concepts for optimized refrigerated vehicles. The authors consider the inclusion of vacuum insulation panels (VIP) in the refrigerated body's walls and the introduction of a latent heat storage (LHS) to replace fuel-driven air conditioning (AC). The proposed concept of VIP inclusion is shown capable of halving the cooling energy required as well as reducing the temperature fluctuations of the chilled goods during cooling operation. Replacing the AC system with an LHS mounted near the top of the refrigerator body and supported by a small ventilation system of lower overall capacity results in comparable temperature homogeneity of the deep frozen food products in the simulations. Finally, the maximum downtime of the AC system is increased from 8 min in case of combined PUR and VIP insulation to a total of approximately 11 min for the case with LHS.

Please cite this article as: M.J. Krause, A. Kummerländer, S.J. Avis et al., OpenLB—Open source lattice Boltzmann code, Computers and Mathematics with Applications (2020), https://doi.org/10.1016/j.camwa.2020.04.033.

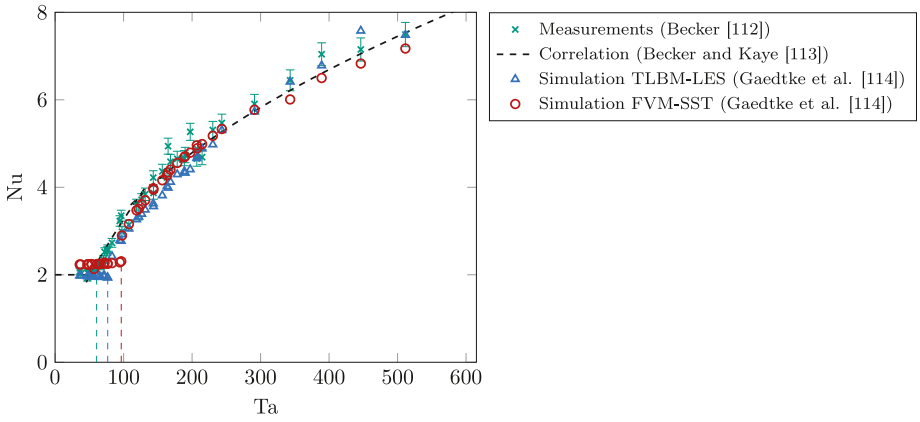


Fig. 13. Nusselt number over Taylor number: Comparison of measured data from Becker [114] and the correlation from Becker and Kaye [116] against simulation results by TLBM-LES and FVM-SST from Gaedtke et al. [113].

7.2. Application to vacuum insulation panels

VIPs offer superior thermal insulation performance. With their reduced pressure and filling with a powder of precipitated silicic acid, they exhibit reduced thermal conductivity. In order to gain a deep understanding of the underlying heat transfer mechanisms, detailed microscale simulations were performed by Ross-Jones et al. [111]. A periodic compressed packing of precipitated silicic acid particles is generated with the open source discrete element method software YadeDEM [112]. This aggregate packing is subsequently imported into OpenLB and used to simulate the effects on heat transfer, while resolving the geometry at the pore scale. These simulations include heat transfer by conduction through the solid material, through the residual gas as well as thermal radiation. The authors justify this new holistic approach by the fact that it offers a decisive advantage by enabling direct control and adjustment of particle packing characteristics.

7.3. Taylor–Couette flow

In [113], Gaedtke et al. applied the thermal large eddy LBM (TLBM-LES) to Taylor–Couette flow simulations, allowing detailed analysis of local heat transport over a wide range of Taylor numbers. The influence of the rotational speed of an inner cylinder with Taylor numbers from 36 to 511 is numerically investigated. The simulations are validated on basis of the global Nusselt number, with good agreement with a measurement series by Becker [114], a correlation and FVM simulations using the Shear Stress Transport (SST) turbulence model [115], see Fig. 13. The measured critical Taylor number is found to be reproduced almost exactly by DNS with TLBM. TLBM-LES show a slightly higher critical Taylor number, while simulations using the SST model overestimate the first occurrence of Taylor vortices even further.

8. Optimal control

Novel methods and algorithms in the field of optimal control and parameter identification for fluid flow problems with LBM, have been proposed and investigated by Krause et al. [117]. Therein, a distributed control problem was solved with an adjoint LBM approach with parallelization in 3D for up to 1.6 million control variables. A second approach was implemented in [118], based on automatic differentiation (AD) which enables solving parameter identification problems in a highly generic manner. Allowing the computation of sensitivities these approaches prove beneficial for example when identifying unknown boundary conditions for an optimal fit to given measurements. Other applications of optimization problems solved in OpenLB are the parameter estimation of ion current formulations [119] and gait and design parameters for bipedal robots [120].

Another application for optimal control of fluid flow is the combination of magnetic resonance imaging (MRI) and CFD, called CFD–MRI. With the CFD–MRI method, a flow measurement is first performed using MRI. The generally noisy measurement results represent values averaged over time and space. At the same time they are a solution to a flow problem characterized by a mathematical model with boundary conditions and corresponding geometry, which can be described by the NSE. The CFD–MRI method utilizes the knowledge of the model to reduce the noise on the one hand and to deduce fine structures of the geometry based on averaging on the other hand.

The coupling is done by formulating the problem as optimal control problem, where the following *goal function*

$$J(f,\alpha) = \frac{1}{2}\|\boldsymbol{u}_f(\alpha) - \boldsymbol{u}^*\|^2_{L^2(V)} \tag{20}$$

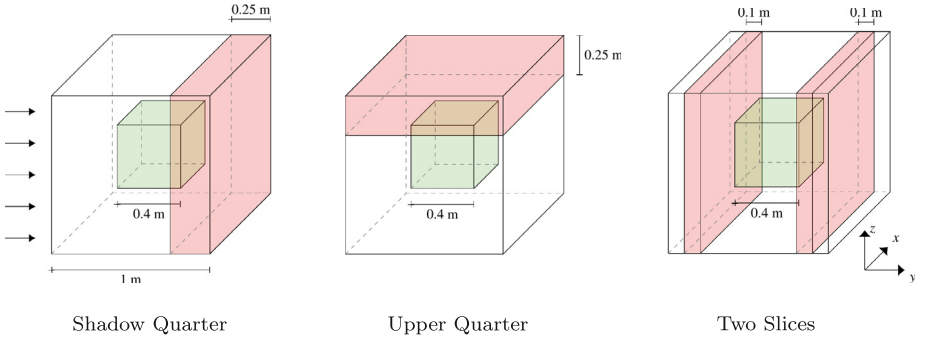


Fig. 14. Cube identification from partial data. Within the design area (green) lies the object to be identified. The target area (red) is reduced to different parts of the total area, such that only partial information of the flow data is used for the optimization algorithm. (For interpretation of the references to color in this figure legend, the reader is referred to the web version of this article.)
Source: Reprint from [121].

is to be minimized. Here $\boldsymbol{u}^*$ is the flow velocity of the measurement, imaged in some domain V. The simulated velocity is denoted as $\boldsymbol{u}_f$, which is controlled by the parameter α, therefore also called *control*. The subscript f indicates that the velocity is considered as moment of the distribution functions of the BE (1), which needs to hold. Therefore, only those solutions are considered, which satisfy physical equations of fluid flow. The sensitivities required to solve the optimization problem are calculated using adjoint methods, which are characterized by high efficiency with a large number of variables. Thus, to realize the CFD–MRI method, the adjoint LBE [117] was extended by the porous media model proposed by Spaid and Phelan [65]. In this model a lattice dependent artificial quantity, here called porosity, is used to insert the physical permeability into the model. In order to use the porosity as control for the optimization problem, a projection method is utilized, which can preserve the physical properties of the flows in porous media and couples the control variable with the permeability. A specific projection was proposed, rendering the method more robust, grid-independent, and consistent with the underlying porous media model. In addition, the exponential nature of the projection effectuates higher convergence rates, allowing the problem to be solved in fewer optimization steps [121].

To validate the flow domain identification of the method, a generic test case is constructed, where the geometry information to be identified is available a priori [121]. First, the fluid flow around a solid geometry is simulated and the resulting flow field is saved. Subsequently, the optimization algorithm is applied. The test case here is the identification of a cube in the middle of a wind tunnel. The available data is then reduced to analyze the accuracy of the method with only partially available data, see Fig. 14.

The results of the three partial data test cases and the full data test case are shown in Fig. 15. The relative error from simulation to given data reaches very low values for all three partial data test cases, indicating that the method works very well. After only 10 steps, the error virtually reaches its minimum in all cases. For all test cases the error is minimized, whereby the case of all given information works best, as expected, and the error approaches 0.01%. Fig. 16 visualizes the object identification of the cube in the case of full data. Observably, the cube was recovered with high porosity after 10 steps, decreasing further for the next steps until the solid cube is identified.

The method was applied to the coupling of real MRI data and simulation in [122]. This setting challenges the CFD–MRI method in terms of the MRI data being solely available two-dimensionally in space and one-dimensionally in velocity. Nonetheless, the measurement noise is significantly reduced, as shown in Fig. 17, and the imaged object inside the flow is found, cf. Fig. 18. Recovering the outline of the object with high permeability required 5 steps. After 8 steps the method detects a solid shell, and a total of 18 steps results in both the inner and outer layers being almost completely impermeable. The CFD–MRI method was able to locate the object and accurately determine the fluid flow, although only 2*D* spatially resolved MRI data were used. Furthermore, the measurement noise was significantly reduced.

9. Other transport problems

Beyond the five categories particulate, turbulent, thermal, multicomponent fluid flow simulation and optimal control of fluid flows, LBM could prove to be well-suited to solve other transport problems. LBM is especially successful in areas where its algorithmic properties and mesoscopic modeling origin make an impact—for example, the transport of mass, momentum and/or energy in resolved porous media in rocks or soil in geology [123,124] or filters in process engineering [125]. Other applications are also found in medicine, where LBM enables e.g. transient blood flow simulations in vessels, which has been extracted from CT from individual patients. In all these cases, the LB algorithm efficiently uses the parallelism of CPUs, clusters and accelerator cards like GPUs, where simple voxel meshes are generated and distributed among the available processing units. The mesoscopic nature of LBM allows an intuitive modeling of many transport problems, such as radiative transport in volume, microfluids or chemical reactions of anisotropic nature. Here, LBM serve as an efficient alternative to particulate approaches such as molecular dynamics or to stochastic Monte Carlo simulations, but also to finite volume methods.

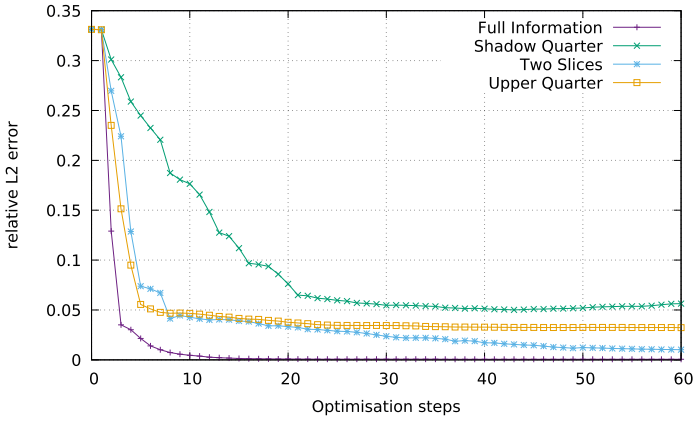
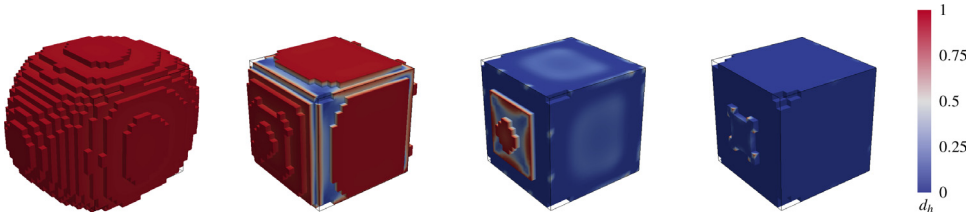


Fig. 15. Relative error of flow characterization for various given data.
Source: Reprint from [121].



(a) 5 steps (b) 10 steps (c) 20 steps (d) 35 steps

Fig. 16. Result of the object identification with full available data.
Source: Reprint from [121].

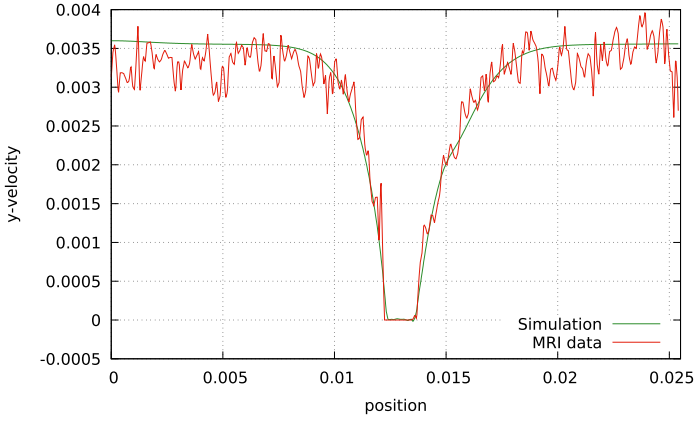


Fig. 17. Comparison of the velocity profile. The experimental data are displayed in red and the simulation results in green. A clear reduction of the measurement noise is observed. (For interpretation of the references to color in this figure legend, the reader is referred to the web version of this article.)
Source: Reprint from [122].

9.1. Automated pre-processing for patient-specific flow simulations

When simulating flow in medical applications, for instance in human respiratory tract or in human blood vessels, one often faces challenges dealing with complex geometries. It begins with segmenting raw measurement data, e.g. from CT, continues with the meshing and partitioning for a parallel executions of the flow domain and ends with the handling of huge data for visualizing and analyzing the results. Consequently, an automated pre-processing and parallel simulation setup becomes essential. In this regard, customized methods and techniques were proposed in the framework of OpenLB

Please cite this article as: M.J. Krause, A. Kummerländer, S.J. Avis et al., OpenLB—Open source lattice Boltzmann code, Computers and Mathematics with Applications (2020), https://doi.org/10.1016/j.camwa.2020.04.033.

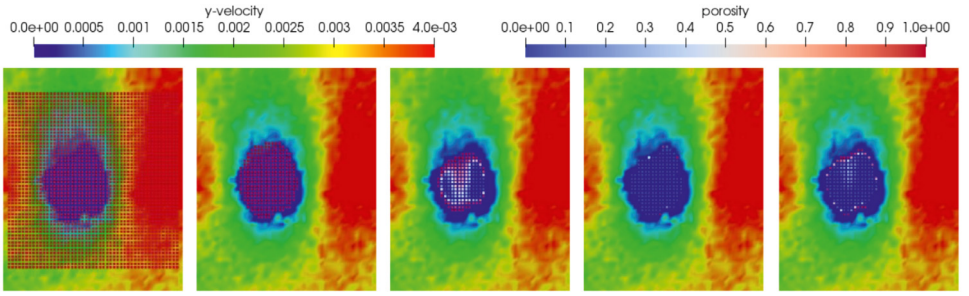


Fig. 18. Results for object identification for different optimization steps. The MRI data are displayed in the background and the permeability of the simulation as points in the foreground. Red indicates high and blue low permeability values and show a clear identification of the imaged object by the method. (For interpretation of the references to color in this figure legend, the reader is referred to the web version of this article.)
Source: Reproduced from [122].

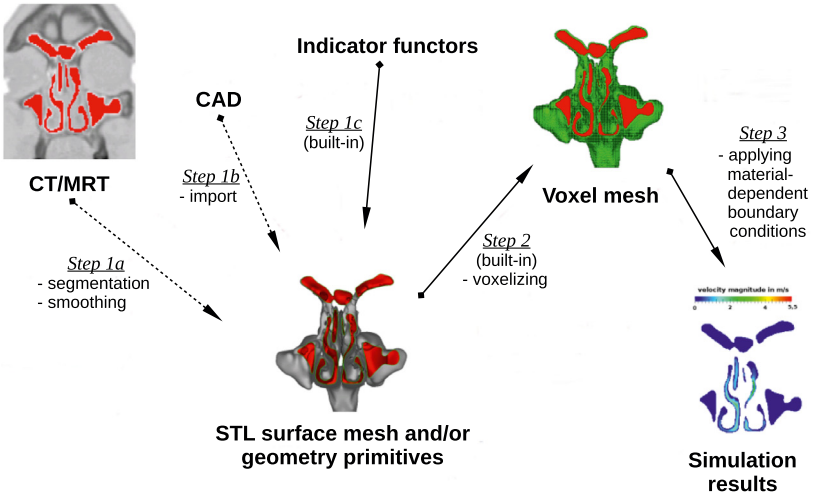


Fig. 19. OpenLB's automated pre-processing for complex geometries is illustrated for the processing of the flow domain resulting from CT data of the human nasal cavity.
Source: Reproduced from [39].

including a hybrid parallelization strategy for LBM [83,84,126] and an automated pre-processing approach (cf. Fig. 19) described in [127]. Both techniques were applied to several test cases in the human lungs [128,129] and nasal cavity [129].

9.2. Validated blood simulations in the human aorta

A patient-specific blood flow through a transverse aortic arch, with a moderate thoracic aortic coarctation, was simulated in the framework of a benchmark competition [130], showing a valid realization and grid convergence for the complex setup. Particular attention was paid to the blood pressure gradient through the coarctation. The challenge, in this context, is the complex geometry containing a stenosis, which results in complex flow patterns. The discretization and simulation were realized using the OpenLB library. A realistic transient flow profile, of the cardiac output for a human at rest, was used to specify the inflow boundary condition at the aortic root, whereas the outflow at the descending aorta was modeled by a pressure boundary condition. A detailed convergence study revealed a linear convergence of the pressure drop over the stenosis ($\pi_1 - \pi_2$ in Fig. 20). The results were also found to be in good agreement with those of other groups which contributed to the challenge. Due to the high scalability of the parallel code, fine temporal and spatial resolved flow fields are computed in a computational time, competitive to those of traditional CFD schemes, which in turn renders the proposed scheme suitable for patient-specific simulations. The automated fast mesh generation and simulation-setup was recognized advantageous compared to standard approaches.

9.3. Computational hemodynamics—a clinical validation study

By means of a clinical study [131], LBM are introduced as a novel technique in hemodynamics to noninvasively measure pressure gradients in patients with a coarctation of the aorta (CoA). In order to provide evidence on the accuracy of

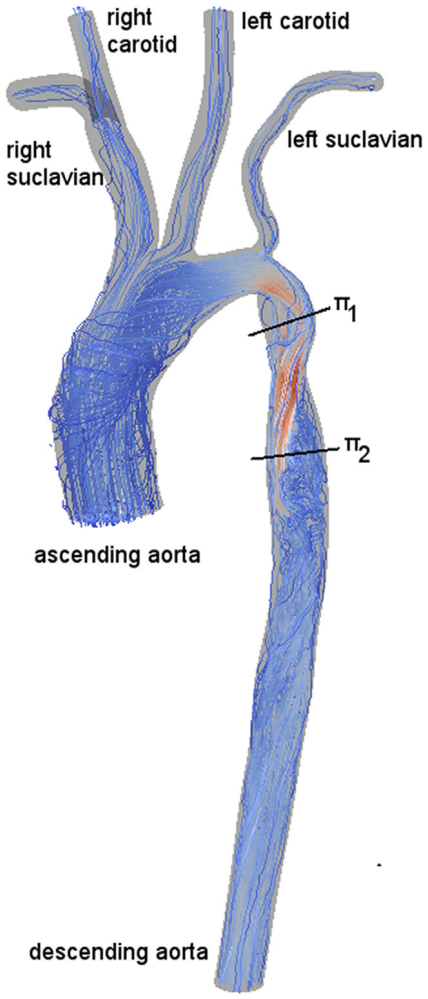


Fig. 20. Validated simulation of the aortic arc with up to 37 million fluid voxels. Flow visualizations at the time point of highest flow rate. The color indicates the magnitude of the velocity.
Source: Reprint from [130].

the proposed scheme, the computed pressure drop values are compared against those obtained using the reference standard method of catheterization. Then, the approach has been applied for the prediction of pressure gradients for 12 patients with CoA for the time point of peak systole, using the open source library OpenLB. Four-dimensional flow-sensitive phasecontrast MRI at 1.5 Tesla was used to acquire flow and to setup the simulation. The vascular geometry was reconstructed using 3D whole-heart MRI. Patients underwent pre- and post-interventional pressure catheterization as a reference standard. The results indicate a reasonable agreement between the simulation results and the catheter measurements. Due to its easy setup procedure and low computational costs, LBM-based computational hemodynamics are concluded to be an attractive alternative to traditional CFD schemes for noninvasive pressure calculations and to be considered to assist in diagnosis and therapy planning.

9.4. Modeling unconfined gas mixing in anaerobic digestion

Anaerobic digestion allows for transforming sludge, the by-product of waste water industry, into a more stable compound and hence, enables the recovery of energy through methane-rich bio gas harvesting. Climate change, increasing need of water, food and energy [135] and increasing energy consumption from waste water treatment [136] mean that it is necessary to reduce energy-intensive mixing for digestion operation while keeping constant, if not increasing, bio gas production. CFD has been proven to be a valid alternative to hazardous and expensive experimental measurements [134].

Dapelo et al. proposed the first LB model for gas mixing in anaerobic digestion [134] and validated it against laboratory data [132,133]. Qualitative results reproduced the flow pattern of a cylindrical tank filled with transparent, non-Newtonian sludge substitute and a central rising bubble column (Fig. 21), where the position of the main vortex was tracked with a

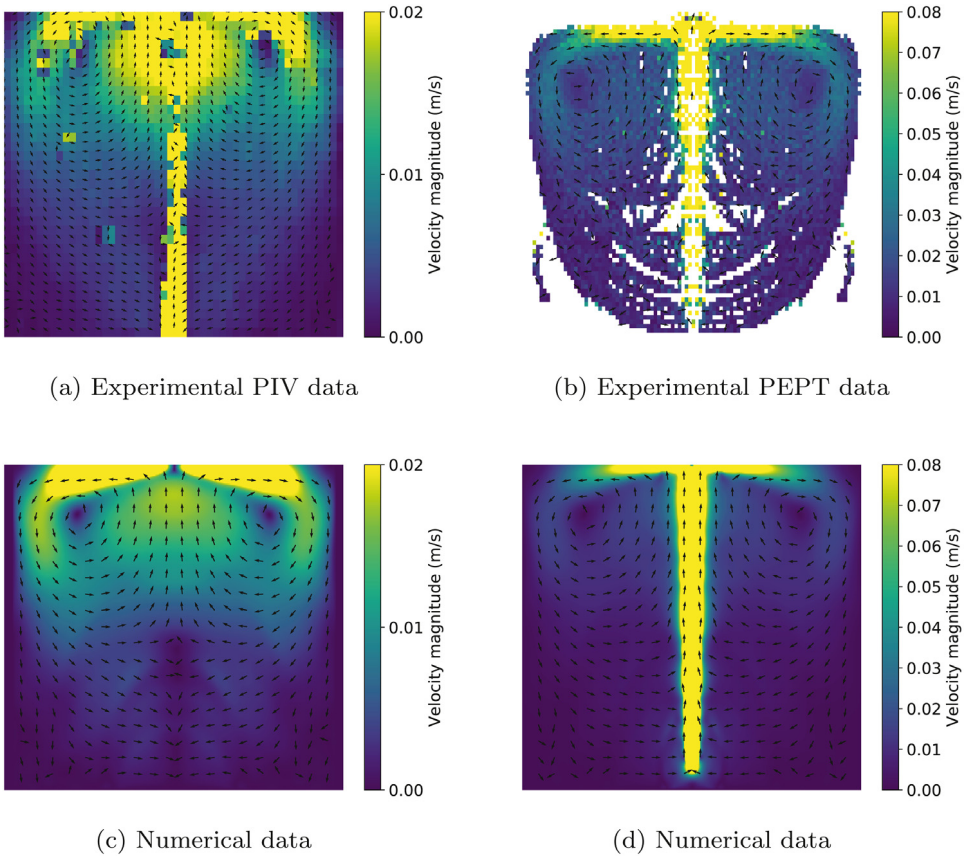


Fig. 21. Averaged flow patterns. (a) Particle Image Velocimetry (PIV) experimental data, [132,133]. The high-velocity column is an optical artifact due to the average effect of laser refraction across the bubble–liquid phase interface. (b) Positron Emission Particle Tracking (PEPT) experimental data, [133]. (c) Numerical data, to be compared to PIV data (a). (d) Numerical data, to be compared to PEPT data (b).
Source: Reproduced from [134].

precision of 0.26%. Convergence tests showed stable behavior for $N = 400$ cells per diameter. The standard BGK LBM with a $D3Q27$ velocity set was used. The bubbly phase was modeled through HLBM [69,101], due to its freedom from high-density-ratio-related pressure fluctuations and parasitic currents occurring in Shan–Chen [19] and free energy [87,137] models, and because it does not require mesh adaptation procedures (which are necessary to immersed boundary methods [138,139]). Furthermore, the OpenLB implementation of HLBM makes use of only one `SuperLattice3D` object, thus allows a significant reduction of memory and computational time. Power law viscosity and turbulence in the liquid phase were treated together through combined application of the Boyd [67] and Smagorinsky [140] models. A partial-slip boundary condition [141,142] was defined at the top boundary as a linear combination of half-way bounce-back and free-slip. The reason for introducing this boundary condition was to reproduce the energy dissipation occurring at the surface due to a liquid bulge above the bubble column and ripples departing from it, which could not be simulated otherwise as the liquid–atmosphere interface was not tracked for the sake of simplicity. The work described in [134] demonstrates the practical applicability of HLBM in OpenLB to a relevant problem of water engineering.

9.5. Light and flow simulations in photobioreactors

A 3D radiative transport LBM (RTLBM) for light simulation in participating media was proposed by Mink et al. [70,71]. The light transport in highly scattering media, can be approximated by a macroscopic diffusion equation with additional sink term. This approach known as P1-Approximation or Diffusion approximation is reported to be increasingly accurate the higher the portion of scattering [143,144]. The proposed RTLBM is realized with the OpenLB framework and evaluated with regard to relative error against analytical solutions [70] and against Monte-Carlo data [71]. For steady diffusion processes the convergence speed is found to be of order two. Further it is shown, that for absent absorption the existing mesoscopic RTLBM coincide with both, the collapsed modeling equation and the approximated macroscopic target equation. The comprehensive study confirms the suitability of LBM to solve radiative transport problems in heavily scattering media such as micro algae suspension in photobioreactors (PBR) and provides a fundamental approach to solve

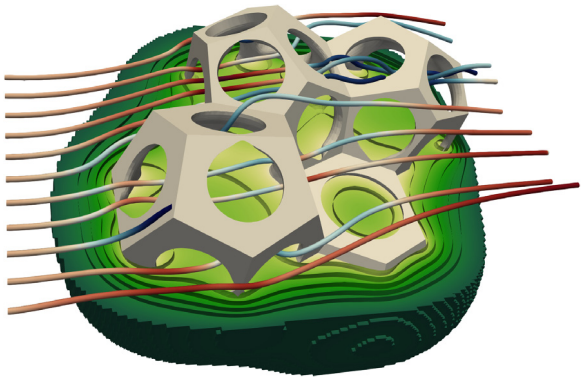


Fig. 22. Simulation of the light distribution (green colors) and fluid flow (streamlines) in a spongy PBR. (For interpretation of the references to color in this figure legend, the reader is referred to the web version of this article.)

anisotropic transport and inhomogeneous media with LBM. The numerical approach has been applied to numerous PBR designs and helped to improve its efficiency. Recently, it has been applied to spongy PBR geometries, where the LBM approach and OpenLB realization could illustrate their strengths for simulations in complex geometries (cf. Fig. 22).

9.6. Modeling and simulation of adsorption processes

Maier et al. proposed an adsorption model that consists of an LB-based Euler–Euler–Lagrange approach [97]. It considers adsorption processes of a solute (adsorptive) on moving particles suspended in water. The water flow was modeled by the NSE, the adsorptive concentration by the ADRE and the adsorption itself by a sink term in the ADRE. The Henry adsorption isotherm was used as a simple linear adsorption model. All particles were assumed to be point masses and described with a sub-grid scale discrete particle model (DPM). The models and the model couplings were implemented within the OpenLB framework. A static mixer was chosen in order to validate the approach by comparisons with several conducted experiments. The simulation results and MRI measurements of the velocity field show good accordance in the magnitude of the velocity (in the coronal cross-section) and its components (in the axial cross-section). The results match very well, qualitatively and quantitatively (maximum relative error is about 9%). The experiments conducted to charge validate the fluid mixing were done with water and an ink solution. Compared with the simulation, both show a similar concentration profile in the mixer. In the cross section of the simulation result, the convective and the diffusive mixing of the fluid components is visible. Finally, the feasibility of the integral approach was confirmed by a qualitative analysis of the dynamical adsorption process measuring the adsorbed quantities on the moving particles.

10. Summary

The present work introduces OpenLB as a general solver for PDEs occurring in transport problems for various applications, where basic implementations within the underlying concept and parallel data structure design are focused. Further, the paper summarizes the findings, applications and validations from previous publications utilizing OpenLB. The framework's modular structure and its open source character facilitate modifications of existing code and likewise enable the direct and intuitive implementation of custom LBM models. Apart from the object-oriented implementation, the open source framework utilizes complex data structures, hybrid parallelization as well as conceptual interface and template meta-programming, to offer overall increased performance, efficiency and scalability. These key features evidently effectuated the herein listed achievements in various fields of applications for LBM, which are highlighted below.

The OpenLB implementations of several particle-laden flow models are fitted to specific purposes within the field and cover a wide spectrum of applications—for example cost-efficient and diffusion-recovering Euler–Euler approaches for similar particles or HLBM for surface force measurements on arbitrarily shaped particles. Topics in the field of turbulent flows are accessible through OpenLB simulations with a broad range of turbulence models incorporated in LBM, such as ADM coupling or wall-modeled LES. Further, insights to stability in turbulence simulations for commonly used collision schemes as a result of under-resolved settings or computations along the incompressible limit, are possible through readily available implementations of commonly used LBM collision schemes. The modular OpenLB blocks for thermal fluid flow simulations allow the treatment of issues such as the numerical analysis of local heat transport in specific test cases for a wide range of Taylor numbers or investigations of heat transfer mechanisms in vacuum insulation panels. Within the field of optimal control for fluid flow problems, the coupling of MRI and CFD via topology optimization was successfully executed for the first time with adjoint LBM via its realization for fluid flow domain identification problems in OpenLB. Moreover notable are, the application of RTLBM for light simulations in photobioreactors, the modeling of unconfined gas mixing, or patient-specific flow simulations with automated pre-processing.

Please cite this article as: M.J. Krause, A. Kummerländer, S.J. Avis et al., OpenLB—Open source lattice Boltzmann code, Computers and Mathematics with Applications (2020), https://doi.org/10.1016/j.camwa.2020.04.033.

In summary, it can be stated that after a decade of software development, LBM research, and its application to various transport problems, the achievements reached with OpenLB prove that through its open source universal access to prior implementations, the successful reproduction of published results is facilitated and promoted. Reproducibility and software sustainability are ensured by automatically testing all code developments using extensive unit tests in a Continuous Integration (CI) setup. Testing of examples against known reference results ensures reproducibility across individual software releases and large-scale code restructuring. Consequently, it not only improves the visibility of LBM outside the community due to its public availability, but further has an impact on the applicability of LBM in complex scenarios in science, industry, and medicine. Hence OpenLB represents a successful transfer from the prototypical research implementations to open access solution possibilities for various engineering problems, not only in CFD. Moreover, as a LBM library comprising multiple collision schemes and boundary methods, OpenLB enables comparisons of distinct LBM, and hence supports extended numerical analysis for qualitative research on the theoretical background of LBM.

Meanwhile OpenLB accounts for a well-established open access platform for developers and applicants and thus traces the initial aims of the project. Nevertheless, sustainable software development demands both, continuous open access maintenance and adjustments to state-of-the-art theoretical achievements, which further on constitute the mainstays of the project. For the upcoming release, further significant performance improvements will be realized by introducing the SSS pattern [81]. Limitations will be removed and new LB features, such as a phase change model [145] including a benchmark showcase, will be made available. Due to the open source realization of the project, future improvements depend on the voluntary participation of open-minded researchers by means of sharing ideas, source codes and documentations. The authors hereby invite such researchers to contribute to OpenLB, thus enriching the LBM community and the scope of LBM applications.

CRediT authorship contribution statement

Mathias J. Krause: Conceptualization, Software, Investigation, Writing - original draft, Visualization, Supervision, Project administration, Funding acquisition. **Adrian Kummerländer:** Software, Investigation, Writing - original draft, Visualization. **Samuel J. Avis:** Software, Investigation, Writing - original draft, Visualization. **Halim Kusumaatmaja:** Software, Investigation, Writing - original draft, Visualization, Supervision. **Davide Dapelo:** Software, Investigation, Writing - original draft, Visualization. **Fabian Klemens:** Software, Investigation, Writing - original draft, Visualization. **Maximilian Gaedtke:** Software, Investigation, Writing - original draft, Visualization. **Nicolas Hafen:** Software, Investigation, Writing - original draft, Visualization. **Albert Mink:** Software, Investigation, Writing - original draft, Visualization. **Robin Trunk:** Software, Investigation, Writing - original draft, Visualization. **Jan E. Marquardt:** Software, Investigation, Writing - original draft, Visualization. **Marie-Luise Maier:** Software, Investigation, Writing - original draft, Visualization. **Marc Haussmann:** Conceptualization, Software, Investigation, Writing - original draft, Visualization. **Stephan Simonis:** Conceptualization, Software, Investigation, Writing - original draft, Writing - review & editing, Visualization.

References

[1] U. Frisch, B. Hasslacher, Y. Pomeau, Lattice-gas automata for the Navier–Stokes equation, Phys. Rev. Lett. 56 (14) (1986) 1505–1508, http://dx.doi.org/10.1103/PhysRevLett.56.1505.
[2] G. Wellein, T. Zeiser, S. Donath, G. Hager, On the single processor performance of simple lattice Boltzmann kernels, Comput. & Fluids 35 (8–9) (2006) 910–919, http://dx.doi.org/10.1016/j.compfluid.2005.02.008.
[3] T. Zeiser, G. Wellein, A. Nitsure, K. Iglberger, U. Rude, G. Hager, Introducing a parallel cache oblivious blocking approach for the lattice Boltzmann method, Prog. Comput. Fluid Dyn. 8 (1–4) (2008) 179–188, http://dx.doi.org/10.1504/PCFD.2008.018088.
[4] M.J. Krause, OpenLB – Open Source Lattice Boltzmann Code, online. URL https://www.openlb.net.
[5] FlowKit Ltd., Palabos – CFD, Complex Physics, online. URL http://www.palabos.org.
[6] U. Rüde, waLBerla – widely applicable Lattice Boltzmann from Erlangen, online. URL https://www.walberla.net/.
[7] Exa Corporation, a Dassault Systèmes company, SIMULIA PowerFLOW, online. URL https://exa.com/en/product/simulation-tools/powerflow-cfd-simulation.
[8] NUMECA International, OMNIS/LB, online. URL https://www.numeca.com/product/omnislb.
[9] J. Latt, M. Krause, OpenLB Release 0.3: Open Source Lattice Boltzmann Code, Zenodo, 2007, http://dx.doi.org/10.5281/zenodo.3625765.
[10] V. Heuveline, J. Latt, The OpenLB project: an open source and object oriented implementation of lattice Boltzmann methods, Internat. J. Modern Phys. C 18 (2007) 627–634, http://dx.doi.org/10.1142/S0129183107010875.
[11] T. Krüger, H. Kusumaatmaja, A. Kuzmin, O. Shardt, G. Silva, E.M. Viggen, The Lattice Boltzmann Method: Principles and Practice, Springer, Berlin, Germany, 2017, http://dx.doi.org/10.1007/978-3-319-44649-3.
[12] S. Succi, The Lattice Boltzmann Equation for Fluid Dynamics and Beyond, Clarendon Press, Oxford, 2001.
[13] D.A. Wolf-Gladrow, Lattice-Gas, Cellular Automata and Lattice Boltzmann Models, An Introduction, in: Lecture Notes in Mathematics, Springer, Heidelberg, Berlin, 2000.
[14] P.J. Dellar, Bulk and shear viscosities in lattice Boltzmann equations, Phys. Rev. E 64 (2001) 031203, http://dx.doi.org/10.1103/PhysRevE.64.031203, https://link.aps.org/doi/10.1103/PhysRevE.64.031203.
[15] P.J. Dellar, An interpretation and derivation of the lattice Boltzmann method using strang splitting, Comput. Math. Appl. (2011) http://dx.doi.org/10.1016/j.camwa.2011.08.047, URL http://www.sciencedirect.com/science/article/pii/S0898122111007206.
[16] M. Krause, Fluid Flow Simulation and Optimisation with Lattice Boltzmann Methods on High Performance Computers: Application to the Human Respiratory System (Ph.D. thesis), Karlsruhe Institute of Technology (KIT), Karlsruhe, Germany, 2010, URL http://digbib.ubka.uni-karlsruhe.de/volltexte/1000019768.
[17] S. Ubertini, P. Asinari, S. Succi, Three ways to lattice Boltzmann: A unified time-marching picture, Phys. Rev. E 81 (2010) 016311, http://dx.doi.org/10.1103/PhysRevE.81.016311, URL https://link.aps.org/doi/10.1103/PhysRevE.81.016311.

Please cite this article as: M.J. Krause, A. Kummerländer, S.J. Avis et al., OpenLB—Open source lattice Boltzmann code, Computers and Mathematics with Applications (2020), https://doi.org/10.1016/j.camwa.2020.04.033.

[18] Z. Guo, C. Zheng, B. Shi, Discrete lattice effects on the forcing term in the lattice Boltzmann method, Phys. Rev. E 65 (2002) 046308, http://dx.doi.org/10.1103/PhysRevE.65.046308.
[19] X. Shan, H. Chen, Lattice Boltzmann model for simulating flows with multi phases and components, Phys. Rev. E 47 (3) (1993) 1815–1819, http://dx.doi.org/10.1103/PhysRevE.47.1815.
[20] P.L. Bhatnagar, E.P. Gross, M. Krook, A model for collision processes in gases. I. Small amplitude processes in charged and neutral one-component systems, Phys. Rev. 94 (3) (1954) 511–525, http://dx.doi.org/10.1103/PhysRev.94.511.
[21] S. Chen, G. Doolen, Lattice Boltzmann method for fluid flows, Annu. Rev. Fluid Mech. 30 (1998) 329–364, http://dx.doi.org/10.1146/annurev.fluid.30.1.329.
[22] I. Ginzburg, F. Verhaeghe, D. d'Humieres, Two-relaxation-time lattice Boltzmann scheme: About parametrization, velocity, pressure and mixed boundary conditions, Commun. Comput. Phys. 3 (2) (2008) 427–478.
[23] D. d'Humieres, I. Ginzburg, M. Krafczyk, P. Lallemand, L.-S. Luo, Multiple-relaxation-time lattice Boltzmann models in three dimensions, Philos. Trans.: Math. Phys. Eng. Sci. 360 (1792) (2002) 437–451, http://dx.doi.org/10.1098/rsta.2001.0955.
[24] J. Latt, B. Chopard, Lattice Boltzmann method with regularized non-equilibrium distribution functions, Math. Comput. Simulation 72 (2006) 165–168, http://dx.doi.org/10.1063/1.868766.
[25] J. Latt, Hydrodynamic Limit of Lattice Boltzmann Equations (Ph.D. thesis), University of Geneva, Geneva, Switzerland, 2007, http://dx.doi.org/10.13097/archive-ouverte/unige:464.
[26] S. Ansumali, I.V. Karlin, Single relaxation time model for entropic lattice Boltzmann methods, Phys. Rev. E 65 (5) (2002) 056312, http://dx.doi.org/10.1103/PhysRevE.65.056312.
[27] C. Coreixas, B. Chopard, J. Latt, Comprehensive comparison of collision models in the lattice Boltzmann framework: Theoretical investigations, Phys. Rev. E 100 (2019) 033305, http://dx.doi.org/10.1103/PhysRevE.100.033305, https://link.aps.org/doi/10.1103/PhysRevE.100.033305.
[28] M. Junk, Z. Yang, Convergence of lattice Boltzmann methods for Navier–Stokes flows in periodic and bounded domains, Numer. Math. 112 (1) (2009) 65–87, http://dx.doi.org/10.1007/s00211-008-0196-0.
[29] M. Junk, Z. Yang, L2 convergence of the lattice Boltzmann method for one dimensional convection-diffusion-reaction equations, Commun. Comput. Phys. 17 (5) (2015) 1225–1245, http://dx.doi.org/10.4208/cicp.2014.m369.
[30] S. Chapman, T.G. Cowling, D. Burnett, The Mathematical Theory of Non-Uniform Gases: An Account of the Kinetic Theory of Viscosity, Thermal Conduction and Diffusion in Gases, Cambridge University Press, 1990.
[31] H. Grad, On the kinetic theory of rarefied gases, Comm. Pure Appl. Math. 2 (4) (1949) 331–407, http://dx.doi.org/10.1002/cpa.3160020403.
[32] S. Simonis, M. Frank, M.J. Krause, On relaxation systems and their relation to discrete velocity Boltzmann models for scalar advection–diffusion equations, Phil. Trans. R. Soc. A (2019) 20190400, http://dx.doi.org/10.1098/rsta.2019.0400.
[33] M.K. Rheinländer, Analysis of Lattice-Boltzmann Methods: Asymptotic and Numeric Investigation of a Singularly Perturbed System (Ph.D. thesis), Universität Konstanz, Konstanz, Germany, 2007, URL http://kops.ub.uni-konstanz.de/volltexte/2007/3635.
[34] M. Krause, S. Avis, D. Dapalo, N. Hafen, M. Haußmann, M. Gaedtke, F. Klemens, A. Kummerländer, M.-L. Maier, A. Mink, J. Ross-Jones, S. Simonis, R. Trunk, OpenLB Release 1.3: Open Source Lattice Boltzmann Code, Zenodo, 2019, http://dx.doi.org/10.5281/zenodo.3625967.
[35] OpenLB user guide, associated to release 1.3 of the code, 2019, online. URL https://www.openlb.net/user-guide.
[36] T. Inamuro, M. Yoshina, F. Ogino, A non-slip boundary condition for lattice Boltzmann simulations, Phys. Fluids 7 (1995) 2928–2930, http://dx.doi.org/10.1063/1.868766.
[37] X. He, Q. Zou, L.-S. Luo, M. Dembo, Analytic solutions of simple flows and analysis of nonslip boundary conditions for the lattice Boltzmann BGK model, J. Stat. Phys. 87 (1) (1997) 115–136, http://dx.doi.org/10.1007/BF02181482.
[38] M. Bouzidi, M. Firdaouss, P. Lallemand, Momentum transfer of a Boltzmann-lattice fluid with boundaries, Phys. Fluids 13 (11) (2001) 3452–3459, http://dx.doi.org/10.1063/1.1399290.
[39] M.J. Krause, T. Gengenbach, R. Mayer, S. Zimney, V. Heuveline, How to breathe life into CT-data, Comput. Aided Med. Eng. 2 (2011) 29–33.
[40] D. Wolf-Gladrow, A lattice Boltzmann equation for diffusion, J. Statist. Phys. 79 (5–6) (1995) 1023–1032.
[41] J. Latt, B. Chopard, Lattice Boltzmann method with regularized non-equilibrium distribution functions, 2005, arXiv preprint physics/0506157.
[42] I. Ginzburg, D. d'Humières, A. Kuzmin, Optimal stability of advection-diffusion lattice Boltzmann models with two relaxation times for positive/negative equilibrium, J. Stat. Phys. 139 (6) (2010) 1090–1143, http://dx.doi.org/10.1007/s10955-010-9969-9.
[43] H. Wu, J. Wang, Z. Tao, Passive heat transfer in a turbulent channel flow simulation using large eddy simulation based on the lattice Boltzmann method framework, Int. J. Heat Fluid Flow 32 (6) (2011) 1111–1119, http://dx.doi.org/10.1016/j.ijheatfluidflow.2011.09.001.
[44] Q. Liu, Y.-L. He, Double multiple-relaxation-time lattice Boltzmann model for solid–liquid phase change with natural convection in porous media, Physica A 438 (2015) 94–106, http://dx.doi.org/10.1016/j.physa.2015.06.018.
[45] T. Seta, Implicit temperature-correction-based immersed-boundary thermal lattice Boltzmann method for the simulation of natural convection, Phys. Rev. E 87 (6) (2013) 063304, http://dx.doi.org/10.1103/PhysRevE.87.063304.
[46] X. Shan, H. Chen, Lattice Boltzmann model for simulating flows with multiple phases and components, Phys. Rev. E 47 (3) (1993) 1815.
[47] Z. Guo, C. Zheng, B. Shi, Discrete lattice effects on the forcing term in the lattice Boltzmann method, Phys. Rev. E 65 (4) (2002) 046308, http://dx.doi.org/10.1103/PhysRevE.65.046308.
[48] A. Kupershtokh, D. Medvedev, D. Karpov, On equations of state in a lattice Boltzmann method, Comput. Math. Appl. 58 (5) (2009) 965–974, http://dx.doi.org/10.1016/j.camwa.2009.02.024.
[49] C. Semprebon, T. Krüger, H. Kusumaatmaja, Ternary free-energy lattice Boltzmann model with tunable surface tensions and contact angles, Phys. Rev. E 93 (3) (2016) 033305, http://dx.doi.org/10.1103/PhysRevE.93.033305.
[50] Z. Guo, B. Shi, C. Zheng, A coupled lattice BGK model for the Boussinesq equations, Internat. J. Numer. Methods Fluids 39 (4) (2002) 325–342, http://dx.doi.org/10.1002/fld.337.
[51] M. Gaedtke, S. Wachter, M. Rädle, H. Nirschl, M. Krause, Application of a lattice Boltzmann method combined with a Smagorinsky turbulence model to spatially resolved heat flux inside a refrigerated vehicle, Comput. Math. Appl. 76 (10) (2018) 2315–2329, http://dx.doi.org/10.1016/j.camwa.2018.08.018.
[52] A.J. Ladd, Numerical simulations of particulate suspensions via a discretized Boltzmann equation. Part 1. Theoretical foundation, J. Fluid Mech. 271 (1994) 285–309, http://dx.doi.org/10.1017/S0022112094001771.
[53] T. Henn, G. Thäter, W. Dörfler, H. Nirschl, M. Krause, Parallel dilute particulate flow simulations in the human nasal cavity, Comput. & Fluids 124 (2016) 197–207, http://dx.doi.org/10.1016/j.compfluid.2015.08.002.
[54] M.-L. Maier, T. Henn, G. Thaeter, H. Nirschl, M.J. Krause, Towards validated multiscale simulation with a two-way coupled LBM and DEM, Chem. Eng. Technol. 40 (9) (2017) 1591–1598, http://dx.doi.org/10.1002/ceat.201600547.
[55] O. Malaspinas, P. Sagaut, Advanced large-eddy simulation for lattice Boltzmann methods: The approximate deconvolution model, Phys. Fluids 23 (10) (2011) 105103, http://dx.doi.org/10.1063/1.3650422.
[56] P. Nathen, M. Haussmann, M.J. Krause, N.A. Adams, Adaptive filtering for the simulation of turbulent flows with lattice Boltzmann methods, Comput. & Fluids 172 (2018) 510–523, http://dx.doi.org/10.1016/j.compfluid.2018.03.042.
[57] F. Nicoud, F. Ducros, Subgrid-scale stress modelling based on the square of the velocity gradient tensor, Flow Turbul. Combust. 62 (3) (1999) 183–200, http://dx.doi.org/10.1023/A:1009995426001.

Please cite this article as: M.J. Krause, A. Kummerländer, S.J. Avis et al., OpenLB—Open source lattice Boltzmann code, Computers and Mathematics with Applications (2020), https://doi.org/10.1016/j.camwa.2020.04.033.

[58] M. Weickert, G. Teike, O. Schmidt, M. Sommerfeld, Investigation of the LES WALE turbulence model within the lattice Boltzmann framework, Comput. Math. Appl. 59 (7) (2010) 2200–2214, http://dx.doi.org/10.1016/j.camwa.2009.08.060.
[59] O. Malaspinas, P. Sagaut, Consistent subgrid scale modelling for lattice Boltzmann methods, J. Fluid Mech. 700 (2012) 514–542, http://dx.doi.org/10.1017/jfm.2012.155.
[60] K.N. Premnath, M.J. Pattison, S. Banerjee, Dynamic subgrid scale modeling of turbulent flows using lattice-Boltzmann method, Physica A 388 (13) (2009) 2640–2658, http://dx.doi.org/10.1016/j.physa.2009.02.041.
[61] E. Lévêque, F. Toschi, L. Shao, J.-P. Bertoglio, Shear-improved Smagorinsky model for large-eddy simulation of wall-bounded turbulent flows, J. Fluid Mech. 570 (2007) 491–502, http://dx.doi.org/10.1017/S0022112006003429.
[62] S. Hou, J. Sterling, S. Chen, G. Doolen, A lattice Boltzmann subgrid model for high Reynolds number flows, 1994, arXiv preprint comp-gas/9401004.
[63] O. Malaspinas, P. Sagaut, Wall model for large-eddy simulation based on the lattice Boltzmann method, J. Comput. Phys. 275 (2014) 25–40, http://dx.doi.org/10.1016/j.jcp.2014.06.020.
[64] J. Boudet, E. Lévêque, P. Borgnat, A. Cahuzac, M.C. Jacob, A Kalman filter adapted to the estimation of mean gradients in the large-eddy simulation of unsteady turbulent flows, Comput. & Fluids 127 (2016) 65–77, http://dx.doi.org/10.1016/j.compfluid.2015.12.006.
[65] M.A.A. Spaid, F.R. Phelan Jr., Lattice Boltzmann methods for modeling microscale flow in fibrous porous media, Phys. Fluids 9 (9) (1997) 2468–2474, http://dx.doi.org/10.1063/1.869392.
[66] Z. Guo, T. Zhao, Lattice Boltzmann model for incompressible flows through porous media, Phys. Rev. E 66 (3) (2002) 036304, http://dx.doi.org/10.1103/PhysRevE.66.036304.
[67] J. Boyd, J. Buick, S. Green, A second-order accurate lattice Boltzmann non-Newtonian flow model, J. Phys. A: Math. Gen. 39 (46) (2006) 14241–14247, http://dx.doi.org/10.1088/0305-4470/39/46/001.
[68] H. Yu, S. Girimaji, L.-S. Luo, DNS and LES of decaying isotropic turbulence with and without frame rotation using lattice Boltzmann method, J. Comput. Phys. 209 (2) (2005) 599–616, http://dx.doi.org/10.1016/j.jcp.2005.03.022.
[69] M. Krause, F. Klemens, T. Henn, R. Trunk, R. Nirschl, Particle flow simulations with homogenised lattice Boltzmann methods, Particuology 34 (2017) 1–13, http://dx.doi.org/10.1016/j.partic.2016.11.001.
[70] A. Mink, G. Thäter, H. Nirschl, M. Krause, A 3D lattice Boltzmann method for light simulation in participating media, J. Comput. Sci. 17 (Part 2) (2016) 431–437, http://dx.doi.org/10.1016/j.jocs.2016.03.014.
[71] A. Mink, C. McHardy, L. Bressel, C. Rauh, M.J. Krause, Radiative transfer lattice Boltzmann methods: 3D models and their performance in different regimes of radiative transfer, J. Quant. Spectrosc. Radiat. Transfer 243 (2020) 106810, http://dx.doi.org/10.1016/j.jqsrt.2019.106810.
[72] P. Skordos, Initial and boundary conditions for the lattice Boltzmann method, Phys. Rev. E 48 (6) (1993) 4823–4842, http://dx.doi.org/10.1103/PhysRevE.48.4823.
[73] J. Latt, B. Chopard, O. Malaspinas, M. Deville, A. Michler, Straight velocity boundaries in the lattice Boltzmann method, Phys. Rev. E 77 (2008) 056703, http://dx.doi.org/10.1103/PhysRevE.77.056703.
[74] Q. Zou, X. He, On pressure and velocity boundary conditions for the lattice Boltzmann BGK model, Phys. Fluids 9 (1997) 1591–1598, http://dx.doi.org/10.1063/1.869307.
[75] M. Haussmann, A.C. Barreto, G.L. Kouyi, N. Rivière, H. Nirschl, M.J. Krause, Large-eddy simulation coupled with wall models for turbulent channel flows at high reynolds numbers with a lattice Boltzmann method—Application to coriolis mass flowmeter, Comput. Math. Appl. (2019) http://dx.doi.org/10.1016/j.camwa.2019.04.033.
[76] M. Sukop, J. DT Thorne, Lattice Boltzmann Modeling Lattice Boltzmann Modeling, Springer, 2006.
[77] S. Succi, The Lattice Boltzmann Equation: For Fluid Dynamics and Beyond, Oxford University Press, 2001.
[78] R.S. Maier, R.S. Bernard, D.W. Grunau, Boundary conditions for the lattice Boltzmann method, Phys. Fluids 8 (7) (1996) 1788–1801, http://dx.doi.org/10.1063/1.868961.
[79] D. Yu, R. Mei, W. Shyy, Improved treatment of the open boundary in the method of lattice Boltzmann equation, Prog. Comput. Fluid Dyn. 5 (1–2) (2005) 3–12, http://dx.doi.org/10.1504/PCFD.2005.005812.
[80] K. Mattila, J. Hyväluoma, T. Rossi, M. Aspnäs, J. Westerholm, An efficient swap algorithm for the lattice Boltzmann method, Comput. Phys. Comm. 176 (2007) 200–210, http://dx.doi.org/10.1016/j.cpc.2006.09.005.
[81] M. Mohrhard, G. Thäter, J. Bludau, B. Horvat, M.J. Krause, Auto-vectorization friendly parallel lattice Boltzmann streaming scheme for direct addressing, Comput. & Fluids 181 (2019) 1–7, http://dx.doi.org/10.1016/j.compfluid.2019.01.001.
[82] V. Heuveline, M.J. Krause, OpenLB: Towards an efficient parallel open source library for lattice Boltzmann fluid flow simulations, in: International Workshop on State-of-the-Art in Scientific and Parallel Computing. PARA, Vol. 9, 2010.
[83] V. Heuveline, M. Krause, J. Latt, Towards a hybrid parallelization of lattice Boltzmann methods, Comput. Math. Appl. 58 (5) (2009) 1071–1080, http://dx.doi.org/10.1016/j.camwa.2009.04.001.
[84] J. Fietz, M.J. Krause, C. Schulz, P. Sanders, V. Heuveline, Optimized hybrid parallel lattice Boltzmann fluid flow simulations on complex geometries, in: C. Kaklamanis, T. Papatheodorou, P. Spirakis (Eds.), Euro-Par 2012 Parallel Processing, in: Lecture Notes in Computer Science, vol. 7484, Springer, Berlin Heidelberg, 2012, pp. 818–829, http://dx.doi.org/10.1007/978-3-642-32820-6_81.
[85] TOP 500.org, TOP500 – The list, online. URL https://www.top500.org/system/178449.
[86] A.K. Gunstensen, D.H. Rothman, S. Zaleski, G. Zanetti, Lattice Boltzmann model of immiscible fluids, Phys. Rev. A 43 (8) (1991) 4320–4327, http://dx.doi.org/10.1103/PhysRevA.43.4320.
[87] M.R. Swift, W.R. Osborn, J.M. Yeomans, Lattice Boltzmann simulation of nonideal fluids, Phys. Rev. Lett. 75 (5) (1995) 830–833, http://dx.doi.org/10.1103/PhysRevLett.75.830.
[88] X. He, S. Chen, R. Zhang, A lattice Boltzmann scheme for incompressible multiphase flow and its application in simulation of Rayleigh–Taylor instability, J. Comput. Phys. 152 (2) (1999) 642–663, http://dx.doi.org/10.1006/JCPH.1999.6257.
[89] H.-B. Huang, X.-Y. Lu, M. Sukop, Numerical study of lattice Boltzmann methods for a convection–diffusion equation coupled with Navier–Stokes equations, J. Phys. A 44 (5) (2011) 055001, http://dx.doi.org/10.1088/1751-8113/44/5/055001.
[90] J. Yang, E.S. Boek, A comparison study of multi-component lattice Boltzmann models for flow in porous media applications, Comput. Math. Appl. 65 (6) (2013) 882–890, http://dx.doi.org/10.1016/J.CAMWA.2012.11.022.
[91] Q. Li, K. Luo, Q. Kang, Y. He, Q. Chen, Q. Liu, Lattice Boltzmann methods for multiphase flow and phase-change heat transfer, Prog. Energy Combust. Sci. 52 (2016) 62–105, http://dx.doi.org/10.1016/J.PECS.2015.10.001.
[92] H. Huang, M.C. Sukop, X.-Y. Lu, Multiphase Lattice Boltzmann Methods : Theory and Application, John Wiley & Sons, Ltd, Chichester, 2015, http://dx.doi.org/10.1002/9781118971451.
[93] M. Wöhrwag, C. Semprebon, A. Mazloomi Moqaddam, I. Karlin, H. Kusumaatmaja, Ternary free-energy entropic lattice Boltzmann model with a high density ratio, Phys. Rev. Lett. 120 (23) (2018) 234501, http://dx.doi.org/10.1103/PhysRevLett.120.234501.
[94] R. Haghani Hassan Abadi, M.H. Rahimian, A. Fakhari, Conservative phase-field lattice-Boltzmann model for ternary fluids, J. Comput. Phys. 374 (2018) 668–691, http://dx.doi.org/10.1016/j.jcp.2018.07.045.
[95] Z. Chen, C. Shu, D. Tan, X.D. Niu, Q.Z. Li, Simplified multiphase lattice Boltzmann method for simulating multiphase flows with large density ratios and complex interfaces, Phys. Rev. E 98 (6) (2018) 1–18, http://dx.doi.org/10.1103/PhysRevE.98.063314.

Please cite this article as: M.J. Krause, A. Kummerländer, S.J. Avis et al., OpenLB—Open source lattice Boltzmann code, Computers and Mathematics with Applications (2020), https://doi.org/10.1016/j.camwa.2020.04.033.

[96] N. Wang, C. Semprebon, H. Liu, C. Zhang, H. Kusumaatmaja, Modelling double emulsion formation in planar flow-focusing microchannels, 2019, arXiv:1906.01034.
[97] M.-L. Maier, S. Milles, S. Schuhmann, G. Guthausen, H. Nirschl, M. Krause, Fluid flow simulations verified by measurements to investigate adsorption processes in a static mixer, Comput. Math. Appl. 76 (11) (2018) 2744–2757, http://dx.doi.org/10.1016/j.camwa.2018.08.066.
[98] H. Kruggel-Emden, S. Rickelt, S. Wirtz, V. Scherer, A study on the validity of the multi-sphere discrete element method, Powder Technol. 188 (2) (2008) 153–165, http://dx.doi.org/10.1016/j.powtec.2008.04.037.
[99] M. Schäfer, S. Turek, Benchmark computations of laminar flow around cylinder, in: Flow Simulation with High-Performance Computers II, in: Notes on Numerical Fluid Mechanics, vol. 52, Vieweg, Wiesbaden, 1996, pp. 547–566, http://dx.doi.org/10.1007/978-3-322-89849-4_39.
[100] A.J.C. Ladd, Numerical simulations of particulate suspensions via a discretized Boltzmann equation. Part 1. Theoretical foundation, J. Fluid Mech. 271 (1994) 285–309, http://dx.doi.org/10.1017/S0022112094001771.
[101] R. Trunk, J. Marquardt, G. Thäter, H. Nirschl, M. Krause, Towards the simulation of arbitrarily shaped 3D particles using a homogenised lattice Boltzmann method, Comput. & Fluids (172) (2018) 621–631, http://dx.doi.org/10.1016/j.compfluid.2018.02.027.
[102] R. Trunk, T. Henn, W. Dörfler, H. Nirschl, M. Krause, Inertial dilute particulate fluid flow simulations with an euler-euler lattice Boltzmann method, J. Comput. Sci. 17 (Part 2) (2016) 438–445, http://dx.doi.org/10.1016/j.jocs.2016.03.013.
[103] S. Höcker, R. Trunk, W. Dörfler, M. Krause, Towards the simulations of inertial dense particulate flows with a volume-averaged lattice Boltzmann method, Comput. & Fluids 166 (2018) 152–162, http://dx.doi.org/10.1016/j.compfluid.2018.02.011.
[104] S.B. Pope, Turbulent Flows, Cambridge University Press, 2000, http://dx.doi.org/10.1017/CBO9780511840531.
[105] P. Nathen, D. Gaudlitz, M.J. Krause, J. Kratzke, An extension of the lattice Boltzmann method for simulating turbulent flows around rotating geometries of arbitrary shape, in: 21st AIAA Computational Fluid Dynamics Conference, American Institute of Aeronautics and Astronautics, 2013, http://dx.doi.org/10.2514/6.2013-2573.
[106] N. Nadim, T.T. Chandratilleke, M.J. Krause, LBM-LES modelling of low Reynolds number turbulent flow over NACA0012 aerofoil, in: Fluid-Structure-Sound Interactions and Control, Springer, 2016, pp. 205–210, http://dx.doi.org/10.1007/978-3-662-48868-3_33.
[107] P. Nathen, M. Haussmann, M.J. Krause, N.A. Adams, Adaptive filtering for the simulation of turbulent flows with lattice Boltzmann methods, Comput. & Fluids 172 (2018) 510–523, http://dx.doi.org/10.1016/j.compfluid.2018.03.042.
[108] P. Nathen, D. Gaudlitz, M.J. Krause, N.A. Adams, On the stability and accuracy of the BGK, MRT and RLB Boltzmann schemes for the simulation of turbulent flows, Commun. Comput. Phys. 23 (3) (2018) 846–876, http://dx.doi.org/10.4208/cicp.OA-2016-0229.
[109] M. Haussmann, S. Simonis, H. Nirschl, M.J. Krause, Direct numerical simulation of decaying homogeneous isotropic turbulence-numerical experiments on stability, consistency and accuracy of distinct lattice Boltzmann methods, Internat. J. Modern Phys. C (2019) http://dx.doi.org/10.1142/S0129183119500748.
[110] M. Gaedtke, S. Wachter, S. Kunkel, S. Sonnick, M. Rädle, H. Nirschl, M.J. Krause, Numerical study on the application of vacuum insulation panels and a latent heat storage for refrigerated vehicles with a large eddy lattice Boltzmann method, Heat Mass Transf. (2019) 1–13, http://dx.doi.org/10.1007/s00231-019-02753-4.
[111] J. Ross-Jones, M. Gaedtke, S. Sonnick, M. Rädle, H. Nirschl, M. Krause, Conjugate heat transfer through nano scale porous media to optimize vacuum insulation panels with lattice Boltzmann methods, Comput. Math. Appl. 77 (2019) 209–221, http://dx.doi.org/10.1016/j.camwa.2018.09.023.
[112] J. Kozicki, F.V. Donze, YADE-OPEN DEM: An opensource software using a discrete element method to simulate granular material, Eng. Comput. (Swans. Wales) 26 (7) (2009) 786–805, http://dx.doi.org/10.1108/02644400910985170.
[113] M. Gaedtke, T. Hoffmann, V. Reinhardt, G. Thäter, H. Nirschl, M. Krause, Flow and heat transfer simulation with a thermal large eddy lattice Boltzmann method in an annular gap with an inner rotating cylinder, Internat. J. Modern Phys. C 30 (02n03) (2019) 1950013, http://dx.doi.org/10.1142/S012918311950013X.
[114] K.M. Becker, An Experimental and Theoretical Study of Heat Transfer in an Annulus with an Inner Rotating Cylinder (Ph.D. thesis), Massachusetts Institute of Technology, 77 Massachusetts Ave, Cambridge, MA 02139, USA, 1957, p. 192.
[115] F.R. Menter, Two-equation eddy-viscosity turbulence models for engineering applications, AIAA J. 32 (8) (1994) 1598–1605, http://dx.doi.org/10.2514/3.12149.
[116] K.M. Becker, J. Kaye, The influence of a radial temperature gradient on the instability of fluid flow in an annulus with an inner rotating cylinder, J. Heat Transfer 84 (2) (1962) 106–110, http://dx.doi.org/10.1115/1.3684306.
[117] M. Krause, G. Thäter, V. Heuveline, Adjoint-based fluid flow control and optimisation with lattice Boltzmann methods, Comput. Math. Appl. 65 (6) (2013) 945–960, http://dx.doi.org/10.1016/j.camwa.2012.08.007.
[118] M. Krause, V. Heuveline, Parallel fluid flow control and optimisation with lattice Boltzmann methods and automatic differentiation, Comput. & Fluids 80 (2013) 28–36, http://dx.doi.org/10.1016/j.compfluid.2012.07.026.
[119] A. Loewe, M. Wilhelms, J. Schmid, M. Krause, F. Fischer, D. Thomas, E. Scholz, O. Dössel, G. Seemann, Parameter estimation of ion current formulations requires hybrid optimization approach to be both accurate and reliable, Front. Bioeng. Biotechnol. 3 (209) (2016) http://dx.doi.org/10.3389/fbioe.2015.00209.
[120] U. Römer, C. Kuhs, M. Krause, A. Fidlin, Simultaneous optimization of gait and design parameters for bipedal robots, in: 2016 IEEE International Conference on Robotics and Automation (ICRA), 2016, pp. 1374–1381, http://dx.doi.org/10.1109/ICRA.2016.7487271.
[121] K. Klemens, B. Förster, M. Dorn, G. Thäter, M.J. Krause, Solving fluid flow domain identification problems with adjoint lattice Boltzmann methods, Comput. Math. Appl. (2018) http://dx.doi.org/10.1016/j.camwa.2018.07.010.
[122] F. Klemens, S. Schuhmann, G. Guthausen, G. Thäter, M. Krause, CFD-MRI: A coupled measurement and simulation approach for accurate fluid flow characterisation and domain identification, Comput. & Fluids 166 (2018) 218–224, http://dx.doi.org/10.1016/j.compfluid.2018.02.022.
[123] Q. Kang, P.C. Lichtner, D. Zhang, Lattice Boltzmann pore-scale model for multicomponent reactive transport in porous media, J. Geophys. Res.: Solid Earth 111 (B5) (2006) http://dx.doi.org/10.1029/2005JB003951.
[124] A.H. Kohanpur, Y. Chen, A.J. Valocchi, J. Tudek, D. Crandall, Comparison of pore-network and lattice Boltzmann models for pore-scale modeling of geological storage of CO2 in natural reservoir rocks, in: AGU Fall Meeting Abstracts, 2016, Provided by the SAO/NASA Astrophysics Data System. URL https://ui.adsabs.harvard.edu/abs/2016AGUFM.H51B1466K.
[125] O. Filippova, D. Hänel, Lattice-Boltzmann simulation of gas-particle flow in filters, Comput. & Fluids 26 (7) (1997) 697–712, http://dx.doi.org/10.1016/S0045-7930(97)00009-1.
[126] M.J. Krause, T. Gengenbach, V. Heuveline, Hybrid parallel simulations of fluid flows in complex geometries: application to the human lungs, in: M. Guarracino, F. Vivien, J. Traeff, M. Cannatoro, M. Danelutto, A. Hast, F. Perla, A. Knuepfer, B. Di Martino, M. Alexander (Eds.), Euro-Par 2010 Parallel Processing Workshops, in: Lecture Notes in Computer Science, vol. 6586, Springer, Berlin / Heidelberg, 2011, pp. 209–216, http://dx.doi.org/10.1007/978-3-642-21878-1_26.
[127] M.J. Krause, T. Gengenbach, R. Mayer, S. Zimney, V. Heuveline, A Preprocessing Approach for Innovative Patient-specific Intranasal Flow Simulations, in: EMCL Preprint Series, 2011, URL http://www.emcl.kit.edu/preprints/emcl-preprint-2011-07.pdf.
[128] T. Gengenbach, M.J. Krause, V. Heuveline, Numerical Simulation of the Human Lung: A Two–scale Approach, in: EMCL Preprint Series, 2011, URL http://www.emcl.kit.edu/preprints/emcl-preprint-2011-11.pdf.

Please cite this article as: M.J. Krause, A. Kummerländer, S.J. Avis et al., OpenLB—Open source lattice Boltzmann code, Computers and Mathematics with Applications (2020), https://doi.org/10.1016/j.camwa.2020.04.033.

[129] T. Henn, G. Thäter, W. Dörfler, H. Nirschl, M. Krause, Parallel dilute particulate flow simulations in the human nasal cavity, Comput. & Fluids (2015) http://dx.doi.org/10.1016/j.compfluid.2015.08.002.
[130] T. Henn, M.J. Krause, S. Ritterbusch, V. Heuveline, Lattice Boltzmann method meets aortic coarctation model, in: M.P. Oscar Camara, K. Rhode, M. Sermesant, A. Young (Eds.), Medical Image Computing and Computer-Assisted Intervention – MICCAI 2012, in: Lecture Notes in Computer Science, vol. 7746, Springer, Berlin Heidelberg, 2012, pp. 34–43.
[131] H. Mirzaee, T. Henn, M. Krause, L. Goubergrits, C. Schumann, M. Neugebauer, T. Kuehne, T. Preusser, A. Hennemuth, MRI-based computational hemodynamics in patients with aortic coarctation using the lattice Boltzmann methods: Clinical validation study, J. Magn. Reson. Imaging 45 (1) (2016) 139–146, http://dx.doi.org/10.1002/jmri.25366.
[132] D. Dapelo, F. Alberini, J. Bridgeman, Euler-Lagrange CFD modelling of unconfined gas mixing in anaerobic digestion, Water Res. 85 (2015) 497–511, http://dx.doi.org/10.1016/j.watres.2015.08.042.
[133] R.C. Sindall, D. Dapelo, T. Leadbeater, J. Bridgeman, Positron emission particle tracking (PEPT): A novel approach to flow visualisation in lab-scale anaerobic digesters, Flow Meas. Instrum. 54 (2017) 250–264, http://dx.doi.org/10.1016/j.flowmeasinst.2017.02.009.
[134] D. Dapelo, R. Trunk, M.J. Krause, J. Bridgeman, Towards lattice-Boltzmann modelling of unconfined gas mixing in anaerobic digestion, Comput. & Fluids 180 (2019) 11–21, http://dx.doi.org/10.1016/j.compfluid.2018.12.008.
[135] WWAP (World Water Assessment Programme), The united nations world water development report 4: Managing water under uncertainty and risk, in: UN Water Reports, Tech. rep., UNESCO, Paris, 2012, URL http://www.unesco.org/new/fileadmin/MULTIMEDIA/HQ/SC/pdf/WWDR4Volume1-ManagingWaterunderUncertaintyandRisk.pdf.
[136] European Environment Agency, Waterbase - UWWTD: Urban Waste Water Treatment Directive – Reported Data, Tech. rep., 2015, URL http://www.eea.europa.eu/data-and-maps/data/waterbase-uwwtd-urban-waste-water-treatment-directive-4.
[137] M. Swift, E. Orlandini, W. Osborn, J. Yeomans, Lattice Boltzmann simulations of liquid-gas and binary fluid systems, Phys. Rev. E 54 (5) (1996) 5041–5052, http://dx.doi.org/10.1103/PhysRevE.54.5041.
[138] C.S. Peskin, The immersed boundary method, Acta Numer. 11 (2002) 479–517, http://dx.doi.org/10.1017/S0962492902000077.
[139] M. Uhlmann, An immersed boundary method with direct forcing for the simulation of particulate flows, J. Comput. Phys. 209 (2) (2005) 448–476, http://dx.doi.org/10.1016/j.jcp.2005.03.017.
[140] S. Hou, J. Sterling, S. Chen, G. Doolen, A lattice Boltzmann subgrid model for high reynolds number flows, in: A. Lawniczak:, R. Kapral (Eds.), Pattern Formation and Lattice Gas Automata, Vol. 6, AMS - Fields Institute Communications, 1996, arXiv:9401004.
[141] M. Sbragaglia, S. Succi, Analytical calculation of slip flow in lattice Boltzmann models with kinetic boundary conditions, Phys. Fluids 17 (9) (2005) 093602, http://dx.doi.org/10.1063/1.2044829.
[142] R. Benzi, L. Biferale, M. Sbragaglia, S. Succi, F. Toschi, Mesoscopic modelling of heterogeneous boundary conditions for microchannel flows, J. Fluid Mech. 548 (2006) 257–280, http://dx.doi.org/10.1017/S0022112005007512.
[143] L. Wang, H. Wu, Biomedical Optics: Principles and Imaging, Wiley, 2007.
[144] S.T. Flock, M.S. Patterson, B.C. Wilson, D.R. Wyman, Monte Carlo modeling of light propagation in highly scattering tissues. I. Model predictions and comparison with diffusion theory, IEEE Trans. Biomed. Eng. 36 (12) (1989) 1162–1168, http://dx.doi.org/10.1109/TBME.1989.1173624.
[145] M. Gaedtke, S. Abishek, R. Mead-Hunter, A.J.C. King, B.J. Mullins, H. Nirschl, M.J. Krause, Total enthalpy-based lattice Boltzmann simulations of melting in paraffin/metal foam composite phase change materials, Int. J. Heat Mass Transfer 155 (2020) 119870, http://dx.doi.org/10.1016/j.ijheatmasstransfer.2020.119870.

Please cite this article as: M.J. Krause, A. Kummerländer, S.J. Avis et al., OpenLB—Open source lattice Boltzmann code, Computers and Mathematics with Applications (2020), https://doi.org/10.1016/j.camwa.2020.04.033.