Numerical Algorithms (2023) 93:1509–1525
https://doi.org/10.1007/s11075-022-01476-8

ORIGINAL PAPER

Lattice Boltzmann Method Analysis Tool (LBMAT)

Radek Fučík[1] · Pavel Eichler[1] · Jakub Klinkovský[1] · Robert Straka[2] · Tomáš Oberhuber[1]

Received: 25 April 2022 / Accepted: 30 November 2022 / Published online: 16 December 2022

Abstract

A general computational tool for the derivation of equivalent partial differential equations (EPDEs) is presented for the lattice Boltzmann method (LBM) with general collision operators that include single relaxation time (SRT-LBM), multiple relaxation time (MRT-LBM), central LBM (CLBM), or cumulant LBM (CuLBM). The method can be used to recover the advection–diffusion equations (ADEs), Navier–Stokes equations (NSEs), and other problems that could be solved by LBM in all dimensions. The derivation of EPDEs starts with the discrete (lattice) Boltzmann equation for raw moments and uses spatio-temporal Taylor expansion of these moments to obtain a system of partial differential equations. Then, to recover the desired ADEs or NSEs with additional partial differential terms up to a given order, a computationally feasible algorithm is proposed to eliminate higher order moments. The algorithm for the derivation of EPDEs, under the name of LBMAT (Lattice Boltzmann Method Analysis Tool), is implemented in C++ using the GiNaC library for symbolic algebraic computations. In order to optimize memory demands for higher dimension LBM models such as D3Q27, a custom-tailored data structure for storing the terms of partial differential expressions is proposed. The implementation of LBMAT is available to the community as open-source software under the terms and conditions of the GNU general public license (GPL).

Keywords Lattice Boltzmann method · Equivalent partial differential equation · MRT-LBM · Central LBM · Cumulant LBM

1 Introduction

The lattice Boltzmann method (LBM) [1–5] is based on the solution of the Boltzmann transport equation for density distribution functions in a discretized form and

✉ Radek Fučík
fucik@fjfi.cvut.cz

Extended author information available on the last page of the article.

serves as an efficient computational method for solving evolution equations such as the Navier–Stokes equations (NSEs), advection–diffusion equations (ADEs), and others [6, 7]. However, in contrast to traditional numerical methods like the finite difference, finite elements, or finite volume methods, these macroscopic equations are not the forerunners of the LBM numerical scheme. They need to be recovered using various techniques that include the asymptotic analysis [8, 9] or the derivation of equivalent finite difference equations (EFDEs) and subsequent use of the Taylor expansion [10] to obtain the desired equivalent partial differential equations (EPDEs) or just direct Taylor expansion of the lattice Boltzmann equation [11, 12]. Alternatively, recent article [13] describes other techniques that could be used to reproduce macroscopic equations. The objective of the recovery procedure is to produce a system of equivalent partial differential equations (EPDEs) up to a given order of derivatives and, furthermore, to transform EPDEs into a system of spatial EPDEs (SEPDEs) by eliminating higher-order temporal derivatives. SEPDEs are the evolution partial differential equations for macroscopic quantities which contain spatial derivatives only, except for the first-order temporal derivatives of the quantities [10]. Consequently, these SEPDEs represent the recovered macroscopic equations (NSEs, ADEs, or others) but with additional higher-order derivatives. However, especially in 3D, the derivation of EPDEs may become extremely difficult due to the immense complexity of the equations.

We explore the approach based on EFDEs and Taylor expansion which simplifies, generalizes, and expands the method proposed in [10]. In order to derive EPDEs and SEPDEs for a general LBM, a new method and a computational tool under the name of LBMAT (Lattice Boltzmann Method Analysis Tool) is proposed and implemented. LBMAT starts with EFDEs for macroscopic quantities (raw moments), employs Taylor expansion of these quantities to obtain EPDEs, and uses sophisticated algorithms to obtain the desired SEPDEs. LBMAT is implemented as an open-source software in C++ using the GiNaC library [14] (www.ginac.de) for symbolic algebraic computations. Its applicability is demonstrated on a series of selected popular LBM variants such as the single relaxation time (SRT-LBM), multiple relaxation time (MRT-LBM), central LBM (CLBM) [15], and the cumulant LBM (CuLBM) [16, 17] for both ADE and NSE. Furthermore, by its design, LBMAT can be potentially used to improve or propose new collision operators, or even to explore possibilities of recovering SEPDEs other than ADEs or NSEs.

The paper is organized as follows. Section 2 contains the description of LBM together with the definition of raw and central moments, derivation of EFDEs, and the assessment of all assumptions required for LBMAT to work. In Section 3, detailed derivations of EPDEs and SEPDEs are presented and two algorithms, essential for the derivation, are given. In Section 4, key points of the implementation are discussed and the computational performance of LBMAT is demonstrated using computations on a common personal computer. Then, in Section 5, SEPDEs in 3D are given and briefly discussed. The resulting SEPDEs for all aforementioned LBM variants (including their mutual comparison) are given in the Supplementary Materials for popular models D1Q3, D2Q5, D2Q9, D3Q7, and D3Q27.

2 Definitions and assumptions

2.1 Lattice Boltzmann equation

The lattice Boltzmann method solves the evolution equation for the discrete density distribution functions f_i, $i = 1, 2, \ldots, q$, on a lattice covering a computational domain in $\mathbb{R}^d$, where d denotes the dimension and q is the number of discrete velocities discretizing the velocity space [5]. For such a so-called DdQq model, the lattice Boltzmann equation is given by

$$f_i(t+\delta_t, \boldsymbol{x}+\delta_t \boldsymbol{c}_i) = f_i(t, \boldsymbol{x}) + C_i(f_1(t, \boldsymbol{x}), f_2(t, \boldsymbol{x}), \ldots, f_q(t, \boldsymbol{x})), \tag{1}$$

where $i = 1, 2, \ldots, q$, $t \in \mathbb{R}_0^+$ and $\boldsymbol{x} \in \mathbb{R}^d$ are the temporal and spatial coordinates, respectively, δ_t is the time step, $\boldsymbol{c}_i$ is the discrete velocity associated with f_i, and C_i denotes a discrete collision operator such as, for instance, one of the aforementioned SRT-LBM, MRT-LBM, CLBM, or CuLBM. The distance between neighboring lattice sites along each axis is considered the same and denoted by δ_l. Note that for those components of $\boldsymbol{c}_i$ (denoted by $[\boldsymbol{c}_i]_\alpha$) that are nonzero, $\delta_l = |[\boldsymbol{c}_i]_\alpha|\delta_t$, $i = 1, 2, \ldots, q$ and $\alpha = 1, \ldots, d$.

In (1), for the sake of simplicity, all quantities are considered dimensionless [4, 5]. Although $\delta_t = 1$ and $\delta_l = 1$ are employed in practice, symbols δ_t and δ_l are used throughout the paper to underline the temporal and spatial stepping, especially in the Taylor expansion later in Section 3.

In a vector form, (1) can be considered as

$$\sum_{i=1}^{q} \mathbf{E}_i \boldsymbol{f}(t+\delta_t, \boldsymbol{x}+\delta_t \boldsymbol{c}_i) = \boldsymbol{f}(t, \boldsymbol{x}) + \boldsymbol{C}(\boldsymbol{f}(t, \boldsymbol{x})), \tag{2}$$

where $\boldsymbol{f} := (f_1, f_2, \ldots, f_q)^T$, $\boldsymbol{C} := (C_1, C_2, \ldots, C_q)^T$, and $\mathbf{E}_i$ denotes the i-th row-selector $q \times q$ matrix, for which $[\mathbf{E}_i]_{i,i} = 1$ is the only nonzero element, $i = 1, 2, \ldots, q$.

Raw moments $m_{\boldsymbol{\alpha}}$, which represent the macroscopic physical quantities [5], are defined by

$$m_{\boldsymbol{\alpha}} := \sum_{i=1}^{q} f_i \boldsymbol{c}_i^{\boldsymbol{\alpha}}, \tag{3}$$

where $\boldsymbol{\alpha} = (\alpha_1, \alpha_2, \ldots, \alpha_d) \in \mathbb{N}_0^d$ denotes a multi-index (as a row vector) and $\boldsymbol{c}_i^{\boldsymbol{\alpha}} := \prod_{j=1}^{d} [\boldsymbol{c}_i]_j^{\alpha_j}$.

Definition 1 (Conserved raw moments) A raw moment $m_{\boldsymbol{\alpha}}$ is said to be conserved during the collision, if it is a collision invariant [18], i.e., for all $\boldsymbol{f}$,

$$\sum_{i=1}^{q} C_i(\boldsymbol{f}) \boldsymbol{c}_i^{\boldsymbol{\alpha}} = \mathbf{0}. \tag{4}$$

A particular choice of a linear combination of raw moments represented by $\boldsymbol{\mu} = (\mu_1, \mu_2, \ldots, \mu_q)^T$ allows to transform (2) into EFDEs for these macroscopic moments. The transformation is realized using

$$\boldsymbol{\mu} = \mathbf{M}\boldsymbol{f}, \tag{5}$$

where $\mathbf{M}$ is a constant, custom-selected nonsingular transformation matrix between the discrete density distribution functions and raw moments.

2.2 Conserved and nonconserved quantities

Although $\mathbf{M}$ in (5) can be selected arbitrarily, additional assumptions need to be placed upon the structure (ordering) of $\mathbf{M}$ in order to enable deriving the desired SEPDEs.

Assumption 1 *Let $\mathbf{M}$ be selected such that the components of $\boldsymbol{\mu} = \mathbf{M}\boldsymbol{f}$ are ordered as*

$$\boldsymbol{\mu} = \begin{pmatrix} \boldsymbol{\gamma} \\ \boldsymbol{\nu} \end{pmatrix} = (\gamma_1, \gamma_2, \ldots, \gamma_c, \nu_1, \nu_2, \ldots, \nu_n)^T, \tag{6}$$

where c and n denote the number of conserved (denoted by $\{\gamma_i := \mu_i\}_{i=1}^{c}$) and nonconserved (denoted by $\{\nu_i := \mu_{c+i}\}_{i=1}^{n}$) quantities, respectively, with $c + n = q$, $\boldsymbol{\gamma} := (\gamma_1, \gamma_2, \ldots, \gamma_c)^T$, and $\boldsymbol{\nu} := (\nu_1, \nu_2, \ldots, \nu_n)^T$.

In Assumption 1, the conserved quantities (moments) $\gamma_i, i = 1, 2, \ldots, c$, represent the physical macroscopic quantities that are collision invariants [18] and for which we aim to derive SEPDEs; e.g., $c = 1$ and $c = d + 1$ for ADEs and NSEs, respectively. Similar to [10], γ_1 is the zeroth-order raw moment ($\gamma_1 = \mu_1 = m_{\mathbf{0}}$) that corresponds to scalar quantities such as the density ρ (NSEs) or concentration (ADEs) and, in the case of NSEs, $\gamma_2, \ldots, \gamma_{d+1}$ correspond to the hydrodynamic momentum components ($\gamma_2 = \mu_2 = m_{(1,0,0)} = \rho v_1$, $\gamma_3 = \mu_3 = m_{(0,1,0)} = \rho v_2$, $\gamma_4 = \mu_4 = m_{(0,0,1)} = \rho v_3$, where $(v_1, v_2, v_3)^T$ denotes the macroscopic velocity vector in 3D). Note that the definition of the conserved quantities γ can be extended to represent a general, custom-selected linear combination of conserved raw moments.

2.3 Equivalent finite difference equations

From (2), the EFDEs are obtained by substitution of $\boldsymbol{f} = \mathbf{M}^{-1}\boldsymbol{\mu}$ (i.e., the inverse of (5)) and by multiplying (2) by $\mathbf{M}$ from the left. The resulting EFDEs read

$$\left[\sum_{i=1}^{q} \mathbf{M}\mathbf{E}_i\mathbf{M}^{-1}\boldsymbol{\mu}(t + \delta_t, \boldsymbol{x} + \delta_t\boldsymbol{c}_i)\right] - \boldsymbol{\mu}(t, \boldsymbol{x}) = \mathbf{M}\boldsymbol{C}(\mathbf{M}^{-1}\boldsymbol{\mu}(t, \boldsymbol{x})). \tag{7}$$

In (7), the first c equations can be regarded as those defining the conserved quantities $\{\gamma_i\}_{i=1}^{c}$ and the rest of them as those associated with the nonconserved quantities $\{\nu_i\}_{i=1}^{n}$. Let the right-hand sides of (7) corresponding to c conserved and n nonconserved equations be represented by vectors $\boldsymbol{R_c}$ and $\boldsymbol{R_n}$ with components

$$\left[\boldsymbol{R_c}(\boldsymbol{\gamma}, \boldsymbol{\nu})\right]_i := \left[\mathbf{M}\boldsymbol{C}(\mathbf{M}^{-1}\boldsymbol{\mu})\right]_i, \quad i = 1, 2, \ldots, c, \tag{8a}$$

and

$$\left[\boldsymbol{R_n}(\boldsymbol{\gamma}, \boldsymbol{\nu})\right]_i := \left[\mathbf{M}\boldsymbol{C}(\mathbf{M}^{-1}\boldsymbol{\mu}))\right]_{c+i}, \quad i = 1, 2, \ldots, n, \tag{8b}$$

respectively. In Assumption 1, the conserved quantities $\{\gamma_i\}_{i=1}^{c}$ defined by the first c rows of $\mathbf{M}$ are collision invariants and as follows from Definition 1, the left multiplication by $\mathbf{M}$ in (7) implies $\boldsymbol{R_c} = \mathbf{0}$. Assumption 2 summarizes the properties of $\boldsymbol{R_n}$ which are required for the derivation of EPDEs.

Assumption 2 *Let $\boldsymbol{R_n}$ be expressed using the first-order Taylor polynomial with respect to $\boldsymbol{\nu} \in \mathbb{R}^n$ at $\boldsymbol{\nu} = \vec{0}$ for all $\boldsymbol{\gamma} \in \mathbb{R}^c$ as*

$$\boldsymbol{R_n}(\boldsymbol{\gamma}, \boldsymbol{\nu}) = \boldsymbol{Q}(\boldsymbol{\gamma}) + \mathbf{L}(\boldsymbol{\gamma})\boldsymbol{\nu} + \boldsymbol{N}(\boldsymbol{\gamma}, \boldsymbol{\nu}), \tag{9a}$$

where $\boldsymbol{Q}(\boldsymbol{\gamma}) := \boldsymbol{R_n}(\boldsymbol{\gamma}, \mathbf{0})$, $\mathbf{L}(\boldsymbol{\gamma}) := (\mathrm{D}_{\boldsymbol{\nu}}\boldsymbol{R_n})(\boldsymbol{\gamma}, \mathbf{0})$ is a $n \times n$ matrix (with $\mathrm{D}_{\boldsymbol{\nu}}$ denoting the differential operator with respect to $\boldsymbol{\nu}$), and $\boldsymbol{N}(\boldsymbol{\gamma}, \boldsymbol{\nu})$ denotes the remainder. It is assumed that for all $\boldsymbol{\gamma} \in \mathbb{R}^c$, $\mathbf{L}(\boldsymbol{\gamma})$ is nonsingular and $\boldsymbol{N}(\boldsymbol{\gamma}, \boldsymbol{\nu})$ is polynomial in $\boldsymbol{\nu}$.

Finally, under the notation given by Assumption 2, matrix $\mathbf{M}$ needs to satisfy the following property given by Assumption 3.

Assumption 3 *Let $\mathbf{M}$ be selected such that for all $\boldsymbol{\gamma} \in \mathbb{R}^c$ and $k = 1, 2, \ldots, n$, $[\mathbf{L}(\boldsymbol{\gamma})^{-1}\boldsymbol{N}(\boldsymbol{\gamma}, \boldsymbol{\nu})]_k$ does not depend on $\nu_k, \nu_{k+1}, \ldots \nu_n$.*

For any matrix-LBM (such as SRT-LBM, MRT-LBM, CLBM), Assumption 3 is always satisfied since $\boldsymbol{N}(\boldsymbol{\gamma}, \boldsymbol{\nu}) = \vec{0}$ for all $\boldsymbol{\gamma} \in \mathbb{R}^c$ and $\boldsymbol{\nu} \in \mathbb{R}^n$, however, for nonlinear collision operators with $\boldsymbol{N}(\boldsymbol{\gamma}, \boldsymbol{\nu}) \neq \vec{0}$, $\mathbf{M}$ needs to be chosen carefully. In the case of CuLBM, it was found that $\mathbf{M}$ can be chosen as the matrix defining the raw moments for the MRT-LBM. The definition of $\mathbf{M}$ for each case considered in this paper is given in Section 1.3 of each Supplementary material.

3 Equivalent partial differential equations

The derivation of equivalent partial differential equations described in this section is possible under the assumption of sufficient smoothness of μ_j, $j = 1, 2, \ldots, q$ that allows to consider commutation of partial derivatives involved. From now on, it is assumed that μ_j are continuously differentiable functions of order D in time and space.

3.1 Taylor expansion

In order to transform (7) into a system of EPDEs, a Taylor expansion of μ_j, $j = 1, 2, \ldots, q$, in time and space centered around $(t, \boldsymbol{x})$ up to a given degree D is considered in the form

$$\mu_j(t+\delta_t, \boldsymbol{x}+\delta_l \boldsymbol{h}) = \mu_j(t, \boldsymbol{x}) + \sum_{1\leq|\boldsymbol{\sigma}|\leq D} \binom{|\boldsymbol{\sigma}|}{\boldsymbol{\sigma}} \delta_t^{\sigma_1} \delta_l^{|\boldsymbol{\sigma}|-\sigma_1} \Big(\prod_{m=1}^{d} [\boldsymbol{h}]_m^{\sigma_{m+1}} \Big) \mathrm{D}_{\boldsymbol{\sigma}} \mu_j(t, \boldsymbol{x}), \tag{10}$$

where $\boldsymbol{\sigma} \in \mathbb{N}_0^{d+1}$ is a multi-index, $|\boldsymbol{\sigma}| := \sum_{i=1}^{d+1} \sigma_i$, $\boldsymbol{h} \in \mathbb{Z}^d$, and $\mathrm{D}_{\boldsymbol{\sigma}}$ denotes the differential operator

$$\mathrm{D}_{\boldsymbol{\sigma}} := \frac{\partial^{|\boldsymbol{\sigma}|}}{\partial t^{\sigma_1} \partial x_1^{\sigma_2} \dots \partial x_d^{\sigma_{d+1}}}. \tag{11}$$

In (10), $\boldsymbol{h}$ represents $\frac{\delta_t}{\delta_l} \boldsymbol{c}_i$ from (7) and, to ease the notation in the following, $\boldsymbol{h}_i := \frac{\delta_t}{\delta_l} \boldsymbol{c}_i$ is used for all $i = 1, 2, \dots, q$.

After combining Taylor-expanded moments given by (10) with (7), the resulting *raw* EPDEs read

$$\begin{aligned} &\sum_{i=1}^{q} \mathbf{M}\mathbf{E}_i \mathbf{M}^{-1} \Big[\sum_{1\leq|\boldsymbol{\sigma}|\leq D} \binom{|\boldsymbol{\sigma}|}{\boldsymbol{\sigma}} \delta_t^{\sigma_1} \delta_l^{|\boldsymbol{\sigma}|-\sigma_1} \Big(\prod_{m=1}^{d} [\boldsymbol{h}_i]_m^{\sigma_{m+1}} \Big) \mathrm{D}_{\boldsymbol{\sigma}} \boldsymbol{\mu}(t, \boldsymbol{x}) \Big] = \\ &\mathbf{M}\boldsymbol{C}(\mathbf{M}^{-1} \boldsymbol{\mu}(t, \boldsymbol{x})), \end{aligned} \tag{12}$$

where the fact that $\sum_{i=1}^{q} \mathbf{E}_i$ gives the identity matrix is used to cancel out the zeroth-order derivatives (ZOD) of $\boldsymbol{\mu}$ from the left-hand side of the equation.

Employing the notation introduced in (8), (12) can be written as

$$\sum_{1\leq|\boldsymbol{\sigma}|\leq D} \begin{pmatrix} \mathbf{P}_{\boldsymbol{c},\boldsymbol{c}}^{(\boldsymbol{\sigma})} & \mathbf{P}_{\boldsymbol{c},\boldsymbol{n}}^{(\boldsymbol{\sigma})} \\ \mathbf{P}_{\boldsymbol{n},\boldsymbol{c}}^{(\boldsymbol{\sigma})} & \mathbf{P}_{\boldsymbol{n},\boldsymbol{n}}^{(\boldsymbol{\sigma})} \end{pmatrix} \begin{pmatrix} \mathrm{D}_{\boldsymbol{\sigma}} \boldsymbol{\gamma}(t, \boldsymbol{x}) \\ \mathrm{D}_{\boldsymbol{\sigma}} \boldsymbol{\nu}(t, \boldsymbol{x}) \end{pmatrix} = \begin{pmatrix} \boldsymbol{0} \\ \boldsymbol{R}_{\boldsymbol{n}}(t, \boldsymbol{x}), \end{pmatrix} \tag{13}$$

where $\{\mathbf{P}_{\boldsymbol{\alpha},\boldsymbol{\beta}}^{(\boldsymbol{\sigma})}\}_{\boldsymbol{\alpha},\boldsymbol{\beta}\in\{\boldsymbol{c},\boldsymbol{n}\}}$ denote constant matrices defined by

$$\begin{pmatrix} \mathbf{P}_{\boldsymbol{c},\boldsymbol{c}}^{(\boldsymbol{\sigma})} & \mathbf{P}_{\boldsymbol{c},\boldsymbol{n}}^{(\boldsymbol{\sigma})} \\ \mathbf{P}_{\boldsymbol{n},\boldsymbol{c}}^{(\boldsymbol{\sigma})} & \mathbf{P}_{\boldsymbol{n},\boldsymbol{n}}^{(\boldsymbol{\sigma})} \end{pmatrix} := \sum_{i=1}^{q} \binom{|\boldsymbol{\sigma}|}{\boldsymbol{\sigma}} \delta_t^{\sigma_1} \delta_l^{|\boldsymbol{\sigma}|-\sigma_1} \Big(\prod_{m=1}^{d} [\boldsymbol{h}_i]_m^{\sigma_{m+1}} \Big) \mathbf{M}\mathbf{E}_i \mathbf{M}^{-1}. \tag{14}$$

3.2 Elimination of nonconserved quantities

Assumption 2 together with (13) allows to express $\boldsymbol{\nu}$ from EPDEs for the nonconserved quantities as

$$\begin{aligned} \boldsymbol{\nu} = &\sum_{1\leq|\boldsymbol{\sigma}|\leq D} \Big[\mathbf{L}(\boldsymbol{\gamma})^{-1} \mathbf{P}_{\boldsymbol{n},\boldsymbol{c}}^{(\boldsymbol{\sigma})} \mathrm{D}_{\boldsymbol{\sigma}} \boldsymbol{\gamma} + \mathbf{L}(\boldsymbol{\gamma})^{-1} \mathbf{P}_{\boldsymbol{n},\boldsymbol{n}}^{(\boldsymbol{\sigma})} \mathrm{D}_{\boldsymbol{\sigma}} \boldsymbol{\nu} \Big] \\ &- \mathbf{L}(\boldsymbol{\gamma})^{-1} \boldsymbol{N}(\boldsymbol{\gamma}, \boldsymbol{\nu}) - \mathbf{L}(\boldsymbol{\gamma})^{-1} \boldsymbol{Q}(\boldsymbol{\gamma}). \end{aligned} \tag{15}$$

For all $k = 1, 2, \dots, n$, the ZODs of nonconservative quantities $\{\nu_i\}_{i=1}^{n}$ in the right-hand side of (15) are present in the nonlinear term $[\mathbf{L}(\boldsymbol{\gamma})^{-1} \boldsymbol{N}(\boldsymbol{\gamma}, \boldsymbol{\nu})]_k$ only. Since (15) can be regarded as a recursive system of equations for $\nu_1, \nu_2, \dots, \nu_n$, the assumption of $\boldsymbol{N}$ being polynomial with respect to $\boldsymbol{\nu}$ (Assumption 2) together with the Assumption 3 allows using a Gaussian-type recursive elimination procedure with truncation of higher order derivatives to completely eliminate all nonconservative quantities $\boldsymbol{\nu}$ from the right-hand side of (15). In Definition 2, the truncation strategy used here is described and referred to as the *second degree* truncation since the resulting partial differential expression is assumed in a form of the second-degree polynomial in terms of derivatives of $(\boldsymbol{\gamma}, \boldsymbol{\nu})$.

Note that in Definition 2, polynomials with higher degrees can also be used to represent the truncated expressions, however, it may not be feasible from the computational point of view because the complexity of the resulting expressions and subsequent memory demands for the derivation of EPDEs and SEPDEs will increase considerably. On the other hand, the second-degree polynomials are found to sufficiently represent the desired EPDEs and SEPDEs, therefore, only the second-degree polynomials are considered and implemented through Definition 2 in `LBMAT`.

The Gaussian-type elimination procedure applied to (15) is described by Algorithm 1. Note that based on Assumption 3, ZOD of ν_k is not present in the right-hand side of k-th (A) for all $k = 1, 2, \ldots, n$.

Definition 2 (A second-degree truncation strategy of order D for a polynomial partial differential expression) Let $\mathcal{F}^{(A,B,C)}(y_1(t,\boldsymbol{x}), y_2(t,\boldsymbol{x}), \ldots, y_q(t,\boldsymbol{x}))$ denote a partial differential expression where $\mathcal{F}^{(A,B,C)}$ is a polynomial of degree A with respect to derivatives of $\{y_i\}_{i=1}^q$ (of at least the first order) with general (nonlinear) coefficients, B denotes the highest order of derivatives of $\{y_i\}_{i=1}^q$ present in the expression, and C is the highest sum of orders of derivatives present in products, i.e.,

$$\mathcal{F}^{(A,B,C)}(y_1, y_2, \ldots, y_q) = a_0(y_1, y_2, \ldots, y_q) + \tag{16}$$

$$\sum_{i_1=1}^{q} \sum_{1\le|\boldsymbol{\sigma}_1|\le B} a_{i_1}^{(\boldsymbol{\sigma}_1)}(y_1, y_2, \ldots, y_q)\mathrm{D}_{\boldsymbol{\sigma}_1} y_{i_1} + \tag{17}$$

$$\sum_{i_1=1}^{q} \sum_{i_2=i_1}^{q} \sum_{\substack{1\le|\boldsymbol{\sigma}_1|\le B\\ 1\le|\boldsymbol{\sigma}_2|\le B\\ |\boldsymbol{\sigma}_1|+|\boldsymbol{\sigma}_2|\le C}} a_{i_1,i_2}^{(\boldsymbol{\sigma}_1,\boldsymbol{\sigma}_2)}(y_1, y_2, \ldots, y_q)\mathrm{D}_{\boldsymbol{\sigma}_1} y_{i_1}\mathrm{D}_{\boldsymbol{\sigma}_2} y_{i_2} + \cdots + \tag{18}$$

$$\sum_{i_1=1}^{q} \sum_{i_2=i_1}^{q} \cdots \sum_{i_A=i_{A-1}}^{q} \sum_{\substack{1\le|\boldsymbol{\sigma}_1|\le B\\ 1\le|\boldsymbol{\sigma}_2|\le B\\ \cdots\\ 1\le|\boldsymbol{\sigma}_A|\le B\\ \sum_{k=1}^{A}|\boldsymbol{\sigma}_k|\le C}} a_{i_1,i_2,\ldots,i_A}^{(\boldsymbol{\sigma}_1,\boldsymbol{\sigma}_2,\ldots,\boldsymbol{\sigma}_A)}(y_1, y_2, \ldots, y_q) \prod_{\ell=1}^{A} \mathrm{D}_{\boldsymbol{\sigma}_\ell} y_{i_\ell}, \tag{19}$$

where the polynomial's coefficients are represented by the symbol a. Then, the second-degree truncated partial differential expression of order D for $\mathcal{F}^{(A,B,C)}$ is given by $\mathcal{F}^{(2,D,D)}$.

After the Gaussian elimination with the second-degree truncation strategy of order D is done in (15), the nonconserved quantities can be expressed as functions of $\boldsymbol{\gamma}$

```
(15) is considered as

$$\nu_k = \mathcal{F}_k^{(2,D,D)}(\gamma_1, \gamma_2, \ldots, \gamma_c, \nu_1, \nu_2, \ldots, \nu_n), \quad k = 1, 2, \ldots, n, \qquad (A)$$

where $\{\mathcal{F}_k^{(2,D,D)}\}_{k=1}^n$ are functions given by Definition 2. In each step, whenever the right-hand-sides of (A) are modified, they are truncated using Definition 2 and re-used in the next step of the algorithm.

Part 1: **Diagonal and below-diagonal forward elimination**
• For $k = 1, 2, \ldots, n$:
    1. While $\exists \boldsymbol{\sigma} \in \mathbb{N}_0^{d+1}$ such that $\mathrm{D}_{\boldsymbol{\sigma}} \nu_k$ is present in the right-hand-side of $k$-th (A), repeat:
       For all $\boldsymbol{\sigma} \in \mathbb{N}_0^{d+1}$, $|\boldsymbol{\sigma}| = 1, 2, \ldots, D$:
        (a) Compute $\mathrm{D}_{\boldsymbol{\sigma}} \nu_k$ by differentiating $k$-th (A).
        (b) Substitute $\mathrm{D}_{\boldsymbol{\sigma}} \nu_k$ into the right-hand-side of $k$-th (A).
    2. For $\ell = k+1, k+2, \ldots, n$:
        (a) Substitute ZOD of $\nu_k$ from $k$-th (A) into the right-hand-side of $\ell$-th (A).
        (b) For all $\boldsymbol{\sigma} \in \mathbb{N}_0^{d+1}$, $|\boldsymbol{\sigma}| = 1, 2, \ldots, D$:
            (i) Compute $\mathrm{D}_{\boldsymbol{\sigma}} \nu_k$ by differentiating $k$-th (A).
            (ii) Substitute $\mathrm{D}_{\boldsymbol{\sigma}} \nu_k$ into the right-hand-side of $\ell$-th (A).

Part 2: **Above-diagonal backward elimination**
• For $k = n, n-1, \ldots, 1$:
    1. For all $\boldsymbol{\sigma} \in \mathbb{N}_0^{d+1}$, $|\boldsymbol{\sigma}| = 1, 2, \ldots, D$:
       For $\ell = k, k-1, \ldots, 1$:
        (a) Compute $\mathrm{D}_{\boldsymbol{\sigma}} \nu_k$ by differentiating $k$-th (A).
        (b) Substitute $\mathrm{D}_{\boldsymbol{\sigma}} \nu_k$ into the right-hand-side of $\ell$-th (A).
```

Algorithm 1 Gaussian-type recursive elimination of $\boldsymbol{\nu}$ in (15).

only:

$$\nu_k = b_k(\boldsymbol{\gamma}) + \sum_{j=1}^{c} \sum_{1 \le |\boldsymbol{\sigma}| \le D} b_{k,j}^{(\boldsymbol{\sigma})}(\boldsymbol{\gamma}) \mathrm{D}_{\boldsymbol{\sigma}} \gamma_j + \sum_{j_1=1}^{c} \sum_{j_2=j_1}^{c} \sum_{\substack{1 \le |\boldsymbol{\sigma}_1| \le D \\ 1 \le |\boldsymbol{\sigma}_2| \le D \\ |\boldsymbol{\sigma}_1| + |\boldsymbol{\sigma}_2| \le D}} b_{k,j_1,j_2}^{(\boldsymbol{\sigma}_1,\boldsymbol{\sigma}_2)}(\boldsymbol{\gamma}) \mathrm{D}_{\boldsymbol{\sigma}_1} \gamma_{j_1} \mathrm{D}_{\boldsymbol{\sigma}_2} \gamma_{j_2}, \qquad (20)$$

$k = 1, 2, \ldots, n$, where all coefficients denoted by the symbol b are produced by Algorithm 1.

In the vector form, (20) can be represented by

$$\boldsymbol{\nu} = \boldsymbol{b}(\boldsymbol{\gamma}) + \sum_{1 \le |\boldsymbol{\sigma}| \le D} \mathbf{B}_{(2)}^{(\boldsymbol{\sigma})}(\boldsymbol{\gamma}) \mathrm{D}_{\boldsymbol{\sigma}} \boldsymbol{\gamma} + \sum_{\substack{1 \le |\boldsymbol{\sigma}_1| \le D \\ 1 \le |\boldsymbol{\sigma}_2| \le D \\ |\boldsymbol{\sigma}_1| + |\boldsymbol{\sigma}_2| \le D}} \mathbf{B}_{(3)}^{(\boldsymbol{\sigma}_1,\boldsymbol{\sigma}_2)}(\boldsymbol{\gamma}) \mathrm{D}_{\boldsymbol{\sigma}_1} \boldsymbol{\gamma} \, \mathrm{D}_{\boldsymbol{\sigma}_2} \boldsymbol{\gamma}, \qquad (21)$$

where $\boldsymbol{b} = (b_1, b_2, \ldots, b_n)^T$, and $\mathbf{B}_{(2)}$ and $\mathbf{B}_{(3)}$ denote the second- and third-order tensors $(b_{k,j})_{k,j=1}^{n,c}$ and $(b_{k,j_1,j_2})_{k,j_1,j_2=1}^{n,c,c}$, respectively.

3.3 Spatial EPDEs for conserved quantities

Combining (21) with the equations corresponding to the conserved quantities in (13), the following system of c partial differential equations for $\{\gamma_i\}_{i=1}^{c}$ is obtained:

$$\sum_{1\le|\boldsymbol{\sigma}_0|\le D}\left[\mathbf{P}_{\boldsymbol{c},\boldsymbol{c}}^{(\boldsymbol{\sigma}_0)}\mathrm{D}_{\boldsymbol{\sigma}_0}\boldsymbol{\gamma}+\mathbf{P}_{\boldsymbol{c},\boldsymbol{n}}^{(\boldsymbol{\sigma}_0)}\mathrm{D}_{\boldsymbol{\sigma}_0}\boldsymbol{b}(\boldsymbol{\gamma})\right]+\sum_{\substack{1\le|\boldsymbol{\sigma}_0|\le D\\ 1\le|\boldsymbol{\sigma}_1|\le D}}\mathbf{P}_{\boldsymbol{c},\boldsymbol{n}}^{(\boldsymbol{\sigma}_0)}\mathrm{D}_{\boldsymbol{\sigma}_0}\left(\mathbf{B}_{(2)}^{(\boldsymbol{\sigma}_1)}(\boldsymbol{\gamma})\mathrm{D}_{\boldsymbol{\sigma}_1}\boldsymbol{\gamma}\right)$$
$$+\sum_{\substack{1\le|\boldsymbol{\sigma}_0|\le D\\ 1\le|\boldsymbol{\sigma}_1|\le D\\ 1\le|\boldsymbol{\sigma}_2|\le D\\ |\boldsymbol{\sigma}_1|+|\boldsymbol{\sigma}_2|\le D}}\mathbf{P}_{\boldsymbol{c},\boldsymbol{n}}^{(\boldsymbol{\sigma}_0)}\mathrm{D}_{\boldsymbol{\sigma}_0}\left(\mathbf{B}_{(3)}^{(\boldsymbol{\sigma}_1,\boldsymbol{\sigma}_2)}(\boldsymbol{\gamma})\mathrm{D}_{\boldsymbol{\sigma}_1}\boldsymbol{\gamma}\mathrm{D}_{\boldsymbol{\sigma}_2}\boldsymbol{\gamma}\right)=\mathbf{0}. \qquad (22)$$

Expanding all derivatives together with the truncation strategy from Definition 2, (22) is transformed into

$$\sum_{1\le|\boldsymbol{\sigma}_1|\le D}\mathbf{A}_{(2)}^{(\boldsymbol{\sigma}_1)}(\boldsymbol{\gamma})\mathrm{D}_{\boldsymbol{\sigma}_1}\boldsymbol{\gamma}+\sum_{\substack{1\le|\boldsymbol{\sigma}_1|\le D\\ 1\le|\boldsymbol{\sigma}_2|\le D\\ |\boldsymbol{\sigma}_1|+|\boldsymbol{\sigma}_2|\le D}}\mathbf{A}_{(3)}^{(\boldsymbol{\sigma}_1,\boldsymbol{\sigma}_2)}(\boldsymbol{\gamma})\mathrm{D}_{\boldsymbol{\sigma}_1}\boldsymbol{\gamma}\mathrm{D}_{\boldsymbol{\sigma}_2}\boldsymbol{\gamma}=\mathbf{0}, \qquad (23)$$

where $\mathbf{A}_{(2)}$ and $\mathbf{A}_{(3)}$ denote the resulting second- and third-order tensors, respectively.

Equation (23) is a system of partial differential equations for the conserved quantities $\boldsymbol{\gamma}$ with coefficients independent of $\boldsymbol{\nu}$. As such, (23) represents the sought EPDEs of (1). However, as in [10], (23) can be further processed in order to produce SEPDEs by eliminating all temporal derivatives of $\boldsymbol{\gamma}$ except for $\mathrm{D}_t\boldsymbol{\gamma}$ ($\mathrm{D}_t := \frac{\partial}{\partial t}$). Since (23) describes the evolution of conserved quantities, it is expected that $\mathbf{A}_{(2)}^{(\boldsymbol{\tau})}(\boldsymbol{\gamma})$, with $\boldsymbol{\tau} := (1, \mathbf{0})^T \in \mathbb{N}_0^{d+1}$, is nonsingular for all $\boldsymbol{\gamma}$ and $\mathrm{D}_t\boldsymbol{\gamma}$ can be expressed from (23) as

$$\mathrm{D}_t\boldsymbol{\gamma} = -\sum_{\substack{1\le|\boldsymbol{\sigma}_1|\le D\\ \boldsymbol{\sigma}_1\ne\boldsymbol{\tau}}}\mathbf{A}_{(2)}^{(\boldsymbol{\tau})}(\boldsymbol{\gamma})^{-1}\mathbf{A}_{(2)}^{(\boldsymbol{\sigma}_1)}(\boldsymbol{\gamma})\mathrm{D}_{\boldsymbol{\sigma}_1}\boldsymbol{\gamma}+$$
$$\sum_{\substack{1\le|\boldsymbol{\sigma}_1|\le D\\ 1\le|\boldsymbol{\sigma}_2|\le D\\ |\boldsymbol{\sigma}_1|+|\boldsymbol{\sigma}_2|\le D}}\mathbf{A}_{(2)}^{(\boldsymbol{\tau})}(\boldsymbol{\gamma})^{-1}\mathbf{A}_{(3)}^{(\boldsymbol{\sigma}_1,\boldsymbol{\sigma}_2)}(\boldsymbol{\gamma})\mathrm{D}_{\boldsymbol{\sigma}_1}\boldsymbol{\gamma}\mathrm{D}_{\boldsymbol{\sigma}_2}\boldsymbol{\gamma}. \qquad (24)$$

Equation (24) forms a recursive equation for $\mathrm{D}_t\boldsymbol{\gamma}$ that, as in [10], allows to eliminate all temporal derivatives of $\boldsymbol{\gamma}$ from its right-hand side (again with the truncation strategy described in Definition 2). This procedure is described by Algorithm 2.

The following series of equations is considered

$$\mathrm{D}_t^r \gamma_k = \mathcal{G}_{k,r}^{(2,D,D)}(\gamma_1, \gamma_2, \ldots, \gamma_c), \quad k = 1, 2, \ldots, c, \quad r = 1, 2, \ldots, D, \tag{B}$$

where $\mathcal{G}_{k,r}^{(2,D,D)}$ are functions defined by Definition 2 and $\mathrm{D}_t^r := \frac{\partial^r}{\partial t^r}$. For $r = 1$, (B) represents (24) and for all $r \geq 1$, $\mathcal{G}_{k,r+1}^{(2,D,D)} := \mathrm{D}_t \mathcal{G}_{k,r}^{(2,D,D)}$ with the truncation strategy based on Definition 2 applied. In each step, whenever the right-hand-sides of (B) are modified, they are truncated using Definition 2, and re-used in the next step of the algorithm.

```
• For $r = 1, 2, \ldots, D$:
  For $k = 1, 2, \ldots, c$:
    1. If $r \geq 2$:
       For $s = 1, 2, \ldots, r - 1$:
       For $\ell = 1, 2, \ldots, c$:
         (a) Substitute $\mathrm{D}_t^s \gamma_k$ from $(k, s)$-th (B) into the right-hand-side of $(\ell, r)$-th (B).
         (b) If a spatial derivative of $D_t^s \gamma_k$ of order $m$ (denoted by $\mathrm{D}_{\boldsymbol{x}}^{(m)} \mathrm{D}_t^s \gamma_k$) is present in the
             right-hand-side of $(\ell, r)$-th (B):
               (i) Spatially differentiate $(k, s)$-th (B) up to the spatial derivative of order
                   $m$.
               (ii) Use the obtained expression to replace $\mathrm{D}_{\boldsymbol{x}}^{(m)} \mathrm{D}_t^s \gamma_k$ in the right-hand-side
                    of $(\ell, r)$-th (B).
    2. Substitute $\mathrm{D}_t^r \gamma_k$ from $(k, r)$-th (B) recursively into the right-hand-side of $(k, r)$-th (B).
    3. For $m = 1, 2, \ldots, D$:
       While a spatial derivative of $\mathrm{D}_t^r \gamma_k$ of order $m$ (denoted by $\mathrm{D}_{\boldsymbol{x}}^{(m)} \mathrm{D}_t^r \gamma_k$) is present in the
       right-hand-side of $(k, r)$-th (B), repeat:
         (a) Spatially differentiate the right-hand-side of $(k, r)$-th (B) up to the spatial
             derivative of order $m$.
         (b) Use the obtained expression to replace $\mathrm{D}_{\boldsymbol{x}}^{(m)} \mathrm{D}_t^r \gamma_k$ in the right-hand-side of
             $(k, r)$-th (B).
    4. For $\ell = 1, 2, \ldots, c$ and $\ell \neq k$:
       For $m = 1, 2, \ldots, D$:
       If a spatial derivative of $D_t^r \gamma_k$ of order $m$ (denoted by $\mathrm{D}_{\boldsymbol{x}}^{(m)} \mathrm{D}_t^r \gamma_k$) is present in the
       right-hand-side of $(\ell, r)$-th (B):
         (a) Spatially differentiate $(k, r)$-th (B) up to the spatial derivative of order $m$.
         (b) Use the obtained expression to replace $\mathrm{D}_{\boldsymbol{x}}^{(m)} \mathrm{D}_t^r \gamma_k$ in the right-hand-side of
             $(\ell, r)$-th (B).
```

Algorithm 2 Elimination of temporal derivatives from (24).

3.4 Summary

In essence, the derivation of SEPDEs from the lattice Boltzmann equation given by (1), for which the collision operator satisfies Assumption 2, consists of the following steps:

1. Select matrix $\mathbf{M}$ that satisfies Assumptions 1 and 3.
2. Assemble EPDE in (13).
3. Use Algorithm 1 to eliminate nonconserved quantities and produce (23).

4. Use Algorithm 2 to eliminate temporal derivatives of $\boldsymbol{\gamma}$ in the right-hand side of (24) and produce SEPDEs.

4 Implementation

The computational algorithm summarized in Section 3.4 is implemented in `C++` using the `GiNaC` library for symbolic algebraic computation [14]. The code is made available to the LBM community as open-source software `LBMAT` (Lattice Boltzmann Method Analysis Tool) under the terms and conditions of the GNU general public license (GPL):

https://mmg-gitlab.fjfi.cvut.cz/gitlab/lbm/lbmat

Due to the extreme complexity of recurrently substituted partial differential expressions in Algorithms 1 and 2, the computational code is required to be carefully designed with respect to the computational efficiency and memory demands. In the following section, the most important implementation challenges are discussed.

4.1 Representation of truncated partial differential expressions

In the code, the truncated partial differential expressions given by Definition 2 are represented by structure `TPDE` which also provides all required operations on `TPDE` such as addition, multiplication by another `TPDE`, differentiation of a general order, substitution of a coefficient or a derivative for another `TPDE`, and simplification of the expression. By design, the second-degree truncation strategy given by Definition 2 is implemented automatically inside `TPDE` by storing the desired coefficients and discarding all that correspond to higher order derivatives or higher degree polynomial than D with respect to these derivatives.

Since all the aforementioned operations can become computationally expensive and substantially memory demanding, it is crucial to use an efficient storage for the coefficients. For this task, the structure `std::unordered_map` from the `C++` Standard Template Library was found to be the best choice. Unordered map is a container in which elements are stored in buckets addressed by a hash of keys [19]. In `LBMAT`, these keys correspond to multi-indices $\boldsymbol{\alpha}$ that define the derivatives of the quantities.

4.2 Symbolic substitutions

During the recurrent substitutions of partial differential expressions in Algorithms 1 and 2, the process of simplification of algebraic expressions within `GiNaC` is the most computationally expensive and memory demanding operation. If an algebraic expression involved in the recurrent substitutions consists of more than one summand, the number of summands in the resulting expression may increase exponentially in each iteration of the algorithms, especially for large q. Therefore, all expressions with more than one summand are replaced by a symbolic variable and the substitution is listed for later re-use.

To obtain the final EPDEs after Algorithms 1 and 2 completed, the deferred symbolic variables need to be replaced recursively by the substitutions established during the execution of the algorithms. This replacement is implemented as a post-processing step: first, the substitutions are offloaded into files stored on a disk to reduce RAM usage and outputs the EPDEs with symbolic variables, then, a separate Python script (using `GiNaC` internally to simplify algebraic expressions) is executed to perform recursive replacements. The dependencies between symbolic variables in the listed substitutions are analyzed and only those variables that are needed in the final EPDEs are actually substituted.

Using this approach, computational times and the amount of memory needed for both algorithms (including the post-processing step) decrease considerably and even for the most complex LBM models in 3D (CLBM or CuLBM in D3Q27), EPDEs and SEPDEs can be computed on a personal computer, see Table 1 for illustrative computational times. The execution times of the Python script are marginal with respect to the execution of `LBMAT` and they are not included in the computational times listed in Table 1.

5 Illustrative SEPDEs in 3D

In order to demonstrate the applicability of `LBMAT`, SEPDEs for ADEs and NSEs in 3D using D3Q7 and D3Q27 models, respectively, are presented in this section. In the Supplementary Materials, the definitions of LBM models SRT, MRT1, MRT2, CLBM1, CLBM2, CuLBM1, and CuLBM2 are given together with the list of all SEPDEs coefficients (marked in green) that mutually differ among these models.

5.1 ADE D3Q7

In this section, for the sake of brevity, the velocity vector $\boldsymbol{v}$ is considered constant in time and space and the SEPDE for ADE is shown up to the fourth-order derivative (truncated as $\mathcal{F}^{(2,4,4)}$ based on Definition 2). For the non-constant velocity case, the coefficients of SEPDE for ADE are listed in the Supplementary materials (ADE, model D3Q7).

$$\frac{\partial \rho}{\partial t} + \frac{\delta_l}{\delta_t}\boldsymbol{v}\cdot\nabla\rho - \nabla\cdot\left(\frac{\delta_l^2}{\delta_t}\mathbf{D}\nabla\rho\right) + \sum_{\substack{\boldsymbol{\alpha}\in\mathbb{N}_0^3\\ 3\le|\boldsymbol{\alpha}|\le 4}} C^{(0),\mathrm{ADE}}_{\mathrm{D}_{x_1}^{\alpha_1}\mathrm{D}_{x_2}^{\alpha_2}\mathrm{D}_{x_3}^{\alpha_3}\rho}\,\frac{\delta_l^{|\boldsymbol{\alpha}|}}{\delta_t}\,\frac{\partial^{|\boldsymbol{\alpha}|}\rho}{\partial x_1^{\alpha_1}\partial x_2^{\alpha_2}\partial x_3^{\alpha_3}} = 0, \quad (25)$$

where the diffusion tensor is given by

$$\mathbf{D} = \left(\frac{1}{\omega}-\frac{1}{2}\right)\begin{pmatrix} c_s^2 & -v_1v_2 & -v_1v_3 \\ -v_2v_1 & c_s^2 & -v_2v_3 \\ -v_3v_1 & -v_3v_2 & c_s^2 \end{pmatrix} \quad (26a)$$

Table 1 Illustrative computational times and peak memory usage for all cases computed on a personal computer equipped with a single Intel Core i9-9900KF CPU (at 3.6 GHz) and 64 GB RAM

			Computations without symbolic substitutions		Computations with symbolic substitutions	
		Model	Computational time [s]	Peak memory [MB]	Computational time [s]	Peak memory [MB]
ADE	D1Q3[a)]	SRT	3.6	29	3.3	30
		MRT	3.5	29	3.0	30
		CLBM	6.1	30	3.6	30
	D2Q5[b)]	SRT	17.1	93	19.4	95
		MRT1	19.5	95	19.4	95
		MRT2	19.5	95	19.5	95
		CLBM1	30.1	95	24.0	96
		CLBM2	30.9	95	24.5	96
	D3Q7[c)]	SRT	71.9	222	96.4	233
		MRT1	90.3	229	95.5	233
		MRT2	89.5	229	95.8	232
		CLBM1	115.9	231	108	234
		CLBM2	115.5	231	107	233
NSE	D1Q3[d)]	SRT	2.8	27	3.4	27
		MRT	3.5	27	3.2	27
		CLBM	3.1	27	3.0	27
	D2Q9[e)]	SRT	234	297	311	347
		MRT1	346	326	311	347
		MRT2	344	326	311	347
		CLBM1	3925	558	483	353
		CLBM2	3953	558	484	354
		CuLBM1	3904	582	476	359
		CuLBM2	4155	578	674	403
	D3Q27[f)]	SRT	18725	5452	18488	5884
		MRT1	32477	6236	18460	5884
		MRT2	32501	6237	18047	5883
		CLBM1	> 1000000[⋆)]	> 64000[⋆)]	60854	6014
		CLBM2	> 1000000[⋆)]	> 64000[⋆)]	66119	6013
		CuLBM1	> 1000000[⋆)]	> 64000[⋆)]	57820	6795
		CuLBM2	> 1000000[⋆)]	> 64000[⋆)]	141897	7743

Definitions of models used in the table are given in the following Supplementary materials: [a)]supp_d1q3_ade.pdf, [b)]supp_d2q5_ade.pdf, [c)]supp_d3q7_ade.pdf, [d)]supp_d1q3_nse.pdf, [e)]supp_d2q9_nse.pdf, [f)]supp_d3q27_nse.pdf

[⋆]The computation failed because the computer ran out of memory for SRT, and by

$$\mathbf{D} = \begin{pmatrix} \left(\frac{1}{\omega_2} - \frac{1}{2}\right) c_s^2 & \left(\frac{1}{2} - \frac{1}{2\omega_2} - \frac{1}{2\omega_3}\right) v_1 v_2 & \left(\frac{1}{2} - \frac{1}{2\omega_2} - \frac{1}{2\omega_4}\right) v_1 v_3 \\ \left(\frac{1}{2} - \frac{1}{2\omega_2} - \frac{1}{2\omega_3}\right) v_2 v_1 & \left(\frac{1}{\omega_3} - \frac{1}{2}\right) c_s^2 & \left(\frac{1}{2} - \frac{1}{2\omega_3} - \frac{1}{2\omega_4}\right) v_2 v_3 \\ \left(\frac{1}{2} - \frac{1}{2\omega_2} - \frac{1}{2\omega_4}\right) v_3 v_1 & \left(\frac{1}{2} - \frac{1}{2\omega_3} - \frac{1}{2\omega_4} -\right) v_3 v_2 & \left(\frac{1}{\omega_4} - \frac{1}{2}\right) c_s^2 \end{pmatrix} \tag{26b}$$

for MRT1, MRT2, CLBM1, and CLBM2. The diffusion tensor contains extra terms, which is in accordance with [20].

5.2 NSE D3Q27

The conservation of mass is recovered as

$$\begin{aligned} &\frac{\partial \rho}{\partial t} + \frac{\delta_l}{\delta_t} \nabla \cdot (\rho \boldsymbol{v}) + \frac{\delta_l^3}{12\delta_t} \sum_{k=1}^{3} \left[(3c_s^2 - 1 + v_k^2) v_k \frac{\partial^3 \rho}{\partial x_k^3} + (c_s^2 - 1 + 3v_k^2) \rho \frac{\partial^3 v_k}{\partial x_k^3} \right] \\ &\quad - \frac{\rho c_s^2}{6} \frac{\delta_l^3}{\delta_t} \left[\frac{\partial^3 v_1}{\partial x_1 \partial x_2^2} + \frac{\partial^3 v_1}{\partial x_1 \partial x_3^2} + \frac{\partial^3 v_2}{\partial x_1^2 \partial x_2} + \frac{\partial^3 v_2}{\partial x_2 \partial x_3^2} + \frac{\partial^3 v_3}{\partial x_1^2 \partial x_3} + \frac{\partial^3 v_3}{\partial x_2^2 \partial x_3} \right] \\ &\quad + \sum_{\beta \in \{\rho, v_1, v_2, v_3\}} \sum_{\substack{\boldsymbol{\alpha} \in \mathbb{N}_0^3 \\ |\boldsymbol{\alpha}| = 4}} C^{(0),\mathrm{NSE}}_{\mathrm{D}_x^{\alpha_1} \mathrm{D}_y^{\alpha_2} \mathrm{D}_z^{\alpha_3} \beta} \frac{\delta_l^4}{\delta_t} \frac{\partial^{|\boldsymbol{\alpha}|} \beta}{\partial x_1^{\alpha_1} \partial x_2^{\alpha_2} \partial x_3^{\alpha_3}} = 0, \end{aligned} \tag{27}$$

and for all $i = 1, 2, 3$, the conservation of momentum ρv_i is given by:

$$\begin{aligned} &\frac{\partial (\rho v_i)}{\partial t} + \frac{\delta_l}{\delta_t} \nabla \cdot (\rho v_i \boldsymbol{v}) + c_s^2 \frac{\delta_l}{\delta_t} \frac{\partial \rho}{\partial x_i} + \frac{\delta_l^2}{\delta_t} \frac{\partial \rho}{\partial x_i} \left[\sum_{j=1}^{3} C^{(i)\mathrm{NSE}}_{\mathrm{D}_{x_i} \rho, \mathrm{D}_{x_j} v_j} \frac{\partial v_j}{\partial x_j} \right] \\ &+ \frac{\delta_l^2}{\delta_t} \left[\sum_{\substack{j=1 \\ j \neq i}}^{3} \frac{\partial \rho}{\partial x_j} \left(C^{(i)\mathrm{NSE}}_{\mathrm{D}_{x_j} \rho, \mathrm{D}_{x_j} v_i} \frac{\partial v_i}{\partial x_j} + C^{(i)\mathrm{NSE}}_{\mathrm{D}_{x_j} \rho, \mathrm{D}_{x_i} v_j} \frac{\partial v_j}{\partial x_i} \right) + C^{(i)\mathrm{NSE}}_{\mathrm{D}_{x_i} \mathrm{D}_{x_j} v_j} \frac{\partial^2 v_j}{\partial x_i \partial x_j} \right] \\ &+ \frac{\delta_l^2}{\delta_t} \left[\sum_{j=1}^{3} C^{(i)\mathrm{NSE}}_{\mathrm{D}_{x_i} \mathrm{D}_{x_j} \rho} \frac{\partial^2 \rho}{\partial x_i \partial x_j} + C^{(i)\mathrm{NSE}}_{\mathrm{D}_{x_i} v_j, \mathrm{D}_{x_j} v_j} \frac{\partial v_j}{\partial x_i} \frac{\partial v_j}{\partial x_j} + C^{(i)\mathrm{NSE}}_{\mathrm{D}^2_{x_j} v_i} \frac{\partial^2 v_i}{\partial x_j^2} \right] \\ &+ \sum_{\beta \in \{\rho, v_1, v_2, v_3\}} \sum_{\substack{\boldsymbol{\alpha} \in \mathbb{N}_0^3 \\ 3 \leq |\boldsymbol{\alpha}| \leq 4}} C^{(i),\mathrm{NSE}}_{\mathrm{D}_{x_1}^{\alpha_1} \mathrm{D}_{x_2}^{\alpha_2} \mathrm{D}_{x_3}^{\alpha_3} \beta} \frac{\delta_l^{|\boldsymbol{\alpha}|}}{\delta_t} \frac{\partial^{|\boldsymbol{\alpha}|} \beta}{\partial x_1^{\alpha_1} \partial x_2^{\alpha_2} \partial x_3^{\alpha_3}} = 0. \end{aligned} \tag{28}$$

The resulting mass and momentum conservation equations (27), (28) are in accordance with macroscopic equations given in [5]. In order to match the coefficients in (28) and in the Supplementary Materials, the coefficients' denotations in (28) need to be arranged lexicographically due to commutation of products or partial derivatives, i.e., for instance, $C^{(i)}_{\mathrm{D}_{x_a} v_b, \mathrm{D}_{x_c} v_d} = C^{(i)}_{\mathrm{D}_{x_c} v_d, \mathrm{D}_{x_a} v_b}$ or $C^{(i)}_{\mathrm{D}_{x_a} \mathrm{D}_{x_b} v_c} = C^{(i)}_{\mathrm{D}_{x_b} \mathrm{D}_{x_a} v_c}$, respectively.

6 Conclusion

A computational analysis tool LBMAT has been proposed and implemented for the derivation of equivalent partial differential equations of the lattice Boltzmann method. The applicability of LBMAT has been demonstrated using the popular velocity models DdQq and collision operators (SRT, MRT, CLBM, and CuLBM) for both advection–diffusion and Navier–Stokes equations. The LBMAT computer code is available to the LBM community under the GPL license and can be used to analyze existing variants of LBM. By its general design, it can be employed in the development of new variants of LBM in the future.

Supplementary Information The online version contains supplementary material available at https://doi.org/10.1007/s11075-022-01476-8.

Funding The work was supported by the Ministry of Education, Youth and Sports of the Czech Republic under OP RDE grants no. CZ.02.1.01/0.0/0.0/16_019/0000765 and no. CZ.02.1.01/0.0/0.0/16_019/0000753, by the Ministry of Health of the Czech Republic project No. NV19-08-00071, by the Czech Science Foundation project no. 21-09093S, and by the National Science Center, Poland grant number UMO2018/31/B/ST8/00622.

Data availability All data generated or analyzed during this study are included in this published article and its supplementary information files.

Code availability Implementations of the algorithms used can be found at https://mmg-gitlab.fjfi.cvut.cz/gitlab/lbm/lbmat.

Declarations

Conflict of interest The authors declare no competing interests.

References

1. Wolf-Gladrow, D.A.: Lattice-gas Cellular Automata and Lattice Boltzmann Models: An Introduction, vol. 1725. Springer, Berlin (2000). https://doi.org/10.1007/b72010
2. Succi, S.: The Lattice Boltzmann Equation: for Fluid Dynamics and Beyond. Oxford University Press, Oxford (2001)
3. Sukop, M.C. Jr., D.T.T.: Lattice Boltzmann Modeling: An Introduction for Geoscientists and Engineers. Springer, Berlin (2006). https://doi.org/10.1007/3-540-27982-2
4. Guo, Z., Shu, C.: Lattice Boltzmann Method and Its Applications in Engineering, vol. 3. World Scientific, Singapore (2013). https://doi.org/10.1142/8806
5. Krüger, T., Kusumaatmaja, H., Kuzmin, A., Shardt, O., Silva, G., Viggen, E.M.: The lattice boltzmann method springer. https://doi.org/10.1007/978-3-319-44649-3 (2017)
6. Sharma, K.V., Straka, R., Tavares, F.W.: Current status of lattice boltzmann methods applied to aerodynamic, aeroacoustic, and thermal flows. Prog. Aerosp. Sci. **100616**, 115 (2020). https://doi.org/10.1016/j.paerosci.2020.100616
7. Geier, M., Fakhari, A., Lee, T.: Conservative phase-field lattice boltzmann model for interface tracking equation. Phys. Rev. E **91**(6), 063309 (2015). https://doi.org/10.1103/PhysRevE.91.063309
8. Chen, S., Doolen, G.: Lattice Boltzmann Method for fluid flows. Ann. Rev. Fluid Mech. **30**(1), 329–364 (1998). https://doi.org/10.1146/annurev.fluid.30.1.329

9. Hosseini, S.A., Darabiha, N., Thévenin, D.: Theoretical and numerical analysis of the lattice kinetic scheme for complex-flow simulations. Phys. Rev. E **99**(2), 023305 (2019). https://doi.org/10.1103/PhysRevE.99.023305
10. Fučík, R., Straka, R.: Equivalent finite difference and partial differential equations for the lattice Boltzmann method. Comput. Math. Appl. **90**(1), 96–103 (2021). https://doi.org/10.1016/j.camwa.2021.03.014
11. Farag, G., Zhao, S., Chiavassa, G., Boivin, P.: Consistency study of lattice-boltzmann schemes macroscopic limit. Phys. Fluids **33**, 037101 (2021). https://doi.org/10.1063/5.0039490
12. Dubois, F., Lallemand, P.: On single distribution lattice boltzmann schemes for the approximation of navier stokes equations. arXiv:2206.13261. https://doi.org/10.48550/arXiv.2206.13261 (2022)
13. Chai, Z., Shi, B.: Multiple-relaxation-time lattice boltzmann method for the navier-stokes and nonlinear convection-diffusion equations: modeling, analysis, and elements. Phys. Rev. E **102**, 023306 (2020). https://doi.org/10.1103/PhysRevE.102.023306
14. Bauer, C., Frink, A., Kreckel, R.: Introduction to the GiNaC framework for symbolic computation within the C++ programming language. J. Symb. Comput. **33**(1), 1–12 (2002). https://doi.org/10.1006/jsco.2001.0494
15. Geier, M., Greiner, A., Korvink, J.G.: Cascaded digital lattice Boltzmann automata for high Reynolds number flow. Phys. Rev. E **73**(6), 066705 (2006). https://doi.org/10.1103/PhysRevE.73.066705
16. Geier, M., Schönherr, M., Pasquali, A., Krafczyk, M.: The cumulant lattice Boltzmann equation in three dimensions: Theory and validation. Computers & Mathematics with Applications **70**(4), 507–547 (2015). https://doi.org/10.1016/j.camwa.2015.05.001
17. Geier, M., Pasquali, A., Schönherr, M.: Parametrization of the cumulant lattice Boltzmann method for fourth order accurate diffusion Part II: Application to flow around a sphere at drag crisis. J. Comput. Phys. **348**, 889–898 (2017). https://doi.org/10.1016/j.jcp.2017.05.040
18. Kremer, G.M.: An Introduction to the Boltzmann Equation and Transport Processes in Gases. Springer, Berlin (2010). https://doi.org/10.1007/978-3-642-11696-4
19. Drozdek, A.: Data structures and algorithms in c++ cengage learning (2012)
20. Chopard, B., Falcone, J.L., Latt, J.: The lattice boltzmann advection-diffusion model revisited. The European Physical Journal Special Topics **171**(1), 245–249 (2009). https://doi.org/10.1140/epjst/e2009-01035-5

Affiliations

Radek Fučík[1] · Pavel Eichler[1] · Jakub Klinkovský[1] · Robert Straka[2] · Tomáš Oberhuber[1]

Pavel Eichler
eichlpa1@fjfi.cvut.cz

Jakub Klinkovský
jakub.klinkovsky@fjfi.cvut.cz

Robert Straka
straka@metal.agh.edu.pl

Tomáš Oberhuber
tomas.oberhuber@fjfi.cvut.cz

1 Department of mathematics, Faculty of Nuclear Sciences and Physical Engineering, Czech Technical University in Prague, Trojanova 13, Prague, 12000, Czech Republic

2 AGH University of Science and Technology, al. Mickiewicza 30, Krakow, 30-059, Poland