Journal of Computational Physics 327 (2016) 503–518

Contents lists available at ScienceDirect

Journal of Computational Physics

www.elsevier.com/locate/jcp

An implicit block LU-SGS finite-volume lattice-Boltzmann scheme for steady flows on arbitrary unstructured meshes

Weidong Li [a,b], Li-Shi Luo [b,a,*]

[a] Department of Mathematics & Statistics, Old Dominion University, Norfolk, VA 23529, USA
[b] Computational Science Research Center, Hai-Dian District, Beijing 100193, China

ARTICLE INFO

Article history:
Received 25 March 2016
Received in revised form 4 September 2016
Accepted 16 September 2016
Available online 22 September 2016

Keywords:
Lattice Boltzmann equation
Finite-volume method
Fully implicit scheme
Block LU-SGS
Low-Mach number flows
Viscous and inviscid steady flows

ABSTRACT

This work proposes a fully implicit lattice Boltzmann (LB) scheme based on finite-volume (FV) discretization on arbitrary unstructured meshes. The linear system derived from the finite-volume lattice Boltzmann equation (LBE) is solved by the block lower-upper (BLU) symmetric-Gauss-Seidel (SGS) algorithm. The proposed implicit FV-LB scheme is efficient and robust, and has a low-storage requirement. The effectiveness and efficiency of the proposed implicit FV-LB scheme are validated and verified by the simulations of three test cases in two dimensions: (a) the laminar Blasius flow over a flat plate with $\mathrm{Re} = 10^5$; (b) the steady viscous flow past a circular cylinder with $\mathrm{Re} = 10$, 20, and 40; and (c) the inviscid flow past a circular cylinder. The proposed implicit FV-LB scheme is shown to be not only effective and efficient for simulations of steady viscous flows, but also robust and efficient for simulations of inviscid flows in particular.

© 2016 Elsevier Inc. All rights reserved.

1. Introduction

Since its inception in 1989 [1,2], the lattice Boltzmann method (LBM) has been established as an effective and efficient explicit scheme for nearly incompressible flows with a second-order spatial-temporal accuracy on *uniform* Cartesian meshes in both two and three dimensional space [3–5]. To overcome the restriction of uniform Cartesian meshes, the lattice Boltzmann (LB) schemes with local grid refinement have been developed [6,7]. However, a much less developed area is the fully implicit LB schemes for stead flows, which is the focus of the present work.

There have been some existing work on the implicit LB scheme. Tölke *et al.* [7] developed the first fully implicit LB scheme based on a finite-difference (FD) discretization of the Boltzmann equation with the discrete velocity set and the equilibrium distribution based on the lattice Boltzmann equation (LBE); the linear system derived the FD-LBE is solved by the point-Jacobi and Newton-Raphson methods. It is demonstrated that the implicit LB scheme can significantly enhance computational efficiency, especially for high-Reynolds-number flows [7]. Lee and Lin [8] developed a fully implicit LB scheme based on the implicit Taylor-Galerkin finite-element (FE) discretization. The linear system derived from the FE-LBE is solved by bi-conjugate gradient stabilized scheme. This approach does significantly increase the time-step size thus relaxes the restriction due to Courant-Friedrichs-Lewy (CFL) condition. However, the accuracy of this approach seems to depend on the CFL number (cf. the results given in Table 1 in [8]). It should also be noted that both aforementioned approaches [7,8] have not been thoroughly evaluated by comparison with their explicit counterparts in terms of accuracy and efficiency.

* Corresponding author at: Department of Mathematics & Statistics, Old Dominion University, Norfolk, VA 23529, USA.
E-mail addresses: lwd_1982.4.8@163.com (W. Li), lluo@odu.edu, lluo@csrc.ac.cn (L.-S. Luo).

http://dx.doi.org/10.1016/j.jcp.2016.09.038
0021-9991/© 2016 Elsevier Inc. All rights reserved.

Most recently, Huang, Yang and Cai [9] proposed a fully implicit LB scheme based on a finite-difference (FD) discretization. The resulting large sparse nonlinear system is solved by the Newton-Krylov-RAS (restricted additive Schwarz) algorithm. While this approach has high parallelization efficiency, the algorithm is rather complicated and the implicit Jacobian matrix has to be computed and stored on the fly, which may become a major impediment for three dimensional LB models.

In this work we propose and develop a fully implicit LB scheme based on the finite-volume (FV) LB scheme we have developed previously [10]. The proposed implicit FV-LB scheme addresses two specific issues. The first issue is storage requirement. The fully implicit block lower-upper (BLU) symmetric-Gauss-Seidel (SGS) [11] is used to solve the linear system derived from the FV-LBE. In this approach, both the collision and convection terms are implicit and linearized, but it does not need to store the Jacobian matrix, as opposed to the aforementioned approaches [7–9] which require storage of Jacobian matrices. The storage requirement for Jacobian matrices may become a significant issue for models with a large number of discrete velocities in three dimensional space. The second, and more important, issue is the computational efficiency in a wide range of Reynolds number. As shown later, the speed-up of the proposed implicit FV-LB scheme increases when compared with its explicit counterpart as the Reynolds number increases. We will demonstrate that the proposed implicit FV-LB scheme has a low-storage requirement and is computationally efficient when compared to its explicit counterpart; it is also rather robust — it can be used to simulate inviscid flows as well.

The remainder of this paper is organized as follows. Section 2 discusses in detail the formulation of the implicit FV-LB scheme. Section 3 provides the numerical results obtained with the proposed implicit FV-LB scheme for the following three cases in two-dimensions: (a) the laminar Blasius flow over a flat plate with $\mathrm{Re} = 10^5$; (b) the steady viscous flow past a circular cylinder with $\mathrm{Re} = 10$, 20, and 40; and (c) the inviscid flow past a circular cylinder. Finally, Section 4 concludes the paper.

2. Formulation and numerics of the implicit FV-LBM

In this section, we discuss the formulation of the implicit finite-volume lattice Boltzmann (FV-LB) scheme proposed in this work. We first discuss briefly in Sec. 2.1 the spatial discretization of the lattice Boltzmann equation (LBE) on arbitrary unstructured meshes, which forms the basis of the FV-LB scheme [10]. We next discuss in Sec. 2.3 the ghost-cell method for boundary conditions used in the FV-LB scheme. Section 2.4 is central part of the formulation of the proposed implicit FV-LB scheme, which provides the details of the proposed block lower-upper symmetric Gauss-Seidel (BLU-SGS) scheme implemented with the FV-LBE.

2.1. Spatial discretization of the LBE on unstructured meshes

The Boltzmann equation with a set of q discrete velocities, $\{\boldsymbol{\xi}_i | i = 0, 1, \ldots, (q-1)\}$, in d dimensional space is written concisely as the following:

$$\frac{\partial \mathbf{f}}{\partial t} + \nabla \cdot \mathbf{J} = \boldsymbol{\Omega}, \tag{1}$$

with $\mathbf{f} \in \mathbb{R}^q$ is the vector of the distribution functions corresponding the discrete velocities, $\{f_i | i = 0, 1, \ldots, (q-1)\}$, $\mathbf{J} \in \mathbb{R}^{q \times d}$ the momentum flux tensor, and $\boldsymbol{\Omega} \in \mathbb{R}^q$ is vector of collision terms; and they can be written as the following:

$$\mathbf{f} = \begin{pmatrix} f_0 \\ f_1 \\ \vdots \\ f_{q-1} \end{pmatrix}, \quad \mathbf{J} = \begin{pmatrix} \boldsymbol{\xi}_0 f_0 \\ \boldsymbol{\xi}_1 f_1 \\ \vdots \\ \boldsymbol{\xi}_{q-1} f_{q-1} \end{pmatrix}, \quad \boldsymbol{\Omega} = \begin{pmatrix} \Omega_0 \\ \Omega_1 \\ \vdots \\ \Omega_{q-1} \end{pmatrix}. \tag{2}$$

For the nearly incompressible flows to be studied in this work, the only relevant conserved variables are the flow mass density ρ and momentum $\rho \boldsymbol{u} := \rho(u_1, \ldots, u_d)$, where $\boldsymbol{u}$ is the flow velocity. In what follows we will restrict to the two-dimensional space, *i.e.*, $d = 2$.

We denote the vector of conserved quantities as $\mathbf{Q} \in \mathbb{R}^{d+1}$, which can be obtained as the following:

$$\mathbf{Q} = \begin{pmatrix} \rho \\ \rho u_x \\ \rho u_y \end{pmatrix} = \mathbf{C}\mathbf{f}, \quad \mathbf{C}\boldsymbol{\Omega} = \mathbf{0}, \tag{3}$$

where $\mathbf{C} \in \mathbb{R}^{(d+1) \times q}$ projects the distribution functions to the conserved velocity moments ρ and $\rho \boldsymbol{u}$, and it is the tensor of collisional invariants in the discrete velocity space, *i.e.*,

$$\mathbf{C} = \begin{pmatrix} 1 & 1 & \cdots & 1 \\ \xi_{0x} & \xi_{1x} & \cdots & \xi_{(q-1)x} \\ \xi_{0y} & \xi_{1y} & \cdots & \xi_{(q-1)y} \end{pmatrix}. \tag{4}$$

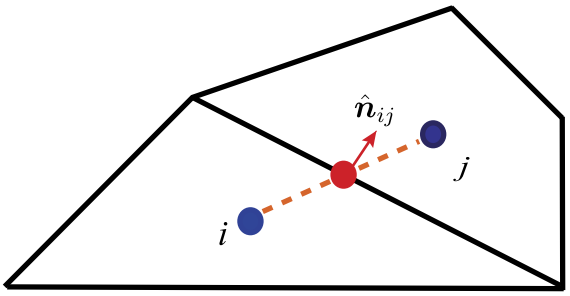

Fig. 1. Illustration of discretized space. The cell i and j share an interface ij of the area S_{ij}. The unit vector $\hat{\boldsymbol{n}}_{ij}$ is out-normal to the interface area S_{ij}.

Clearly, the first row of $\mathbf{C}$ is $|\boldsymbol{\xi}_i|^0 \equiv 1$, and the second and third rows are ξ_{ix} and ξ_{iy}, respectively.

We discretize the space $\boldsymbol{x} \in \mathbb{R}^d$ into cells. Integrating Eq. (1) over a cell i with the cell volume V_i and the boundary ∂V_i leads to:

$$\partial_t \int_{V_i} \mathbf{f}\,\mathrm{d}\boldsymbol{x} + \oint_{\partial V_i} \mathbf{F}\,\mathrm{d}S = \int_{V_i} \boldsymbol{\Omega}\,\mathrm{d}\boldsymbol{x}, \tag{5}$$

where $\mathbf{F} := \mathbf{J} \cdot \hat{\boldsymbol{n}} \in \mathbb{R}^q$ is the convection flux along the direction of $\hat{\boldsymbol{n}}$, and $\hat{\boldsymbol{n}}$ is the unit vector out normal to the surface element $\mathrm{d}S$. If both $\mathbf{f}$ and $\boldsymbol{\Omega}$ are approximated by their values at the cell center, $\mathbf{f}_i$ and $\boldsymbol{\Omega}_i$, respectively, and the surface integration by a simple discrete quadrature, then Eq. (5) becomes:

$$V_i \partial_t \mathbf{f}_i + \sum_{j \in \mathbb{N}_i} \mathbf{F}_{ij} S_{ij} = \boldsymbol{\Omega}_i V_i, \tag{6}$$

where $\mathbf{f}_i \in \mathbb{R}^q$ denotes the values of $\mathbf{f}$ at the centroid $\boldsymbol{r}_i$ of the cell i, *i.e.*, $\mathbf{f}_i := \mathbf{f}(\boldsymbol{r}_i)$, similarly $\boldsymbol{\Omega}_i := \boldsymbol{\Omega}(\boldsymbol{r}_i)$, $\mathbf{F}_{ij} := \mathbf{F}_{ij}(\mathbf{f}_i, \mathbf{f}_j)$ is the flux at the interface ij depending on both $\mathbf{f}_i$ and $\mathbf{f}_j$, and the set $\mathbb{N}_i(j)$ is defined as:

$$\mathbb{N}_i(j) := \{j | \partial V_j \cap \partial V_i = S_{ij},\ S_{ij} \neq \emptyset\},$$

that is, $\mathbb{N}_i(j)$ includes all neighboring cells j which share interfaces ij of non-zero area S_{ij} with the cell i, as illustrated in Fig. 1.

The fluxes $\mathbf{F}_{ij}$ in Eq. (6) are obtained with the low-diffusion Roe scheme [12]:

$$\mathbf{F}_{ij} = \frac{1}{2}\left[\mathbf{F}(\mathbf{f}_\mathrm{L}) + \mathbf{F}(\mathbf{f}_\mathrm{R}) - U_\mathrm{c}\,(\mathbf{f}_\mathrm{R} - \mathbf{f}_\mathrm{L}) \max_{0 \le k \le (q-1)} \left(\left|\boldsymbol{\xi}_k \cdot \hat{\boldsymbol{n}}_{ij}\right|\right)\right], \tag{7}$$

where $\mathbf{F} := \mathbf{J} \cdot \hat{\boldsymbol{n}}$, $\mathbf{f}_\mathrm{L} := \mathbf{f}_i$ and $\mathbf{f}_\mathrm{R} := \mathbf{f}_j$ are the values of $\mathbf{f}$ at the left and right side of interface ij (cf. Fig. 1), and the *local* characteristic velocity U_c is defined as:

$$U_\mathrm{c} = \max\left[\min\left(\lambda|\bar{\boldsymbol{u}}_{ij}|,\ 1.0\right),\ \nu/\Delta x,\ \epsilon\right], \tag{8}$$

where $\bar{\boldsymbol{u}}_{ij} := (\boldsymbol{u}_i + \boldsymbol{u}_j)/2 := [\boldsymbol{u}(\boldsymbol{r}_i) + \boldsymbol{u}(\boldsymbol{r}_j)]/2$, ν is the viscosity, $\Delta x := |\boldsymbol{r}_i - \boldsymbol{r}_j|/2$, $\boldsymbol{r}_i$ and $\boldsymbol{r}_j$ are the centroid positions of the cell i and j, respectively; the cut-off parameter ϵ is set to 10^{-5} and the adjustable parameter λ is set to 1.0 unless otherwise stated. The values of $\mathbf{f}_\mathrm{L}$ and $\mathbf{f}_\mathrm{R}$ in Eq. (7) at both sides of the interface S_{ij} need to be reconstructed from the cell values $\mathbf{f}_i$ and $\mathbf{f}_j$. In this work, we use a linear reconstruction scheme with the least-square method to compute the gradients of the distributions $\mathbf{f}_i$ at the cell centers [10]. It should be pointed out that, there is no need for a limiter in the reconstruction because the flow variables of nearly incompressible flows are continuous.

2.2. The collision model

The collision models used in the LBE are derived from the linearized collision operator of the Boltzmann equation. In particular, the collision model with multiple relaxation times (MRT) due to d'Humières [13] is used in this work. The MRT model can be written as

$$\boldsymbol{\Omega} = -\mathbf{M}^{-1}\mathbf{S}\left[\mathbf{m} - \mathbf{m}^{(0)}\right], \tag{9}$$

where $\mathbf{m}$ and $\mathbf{m}^{(0)}$ denote the vectors of the moments and their equilibria, respectively, $\mathbf{M}$ is the $q \times q$ transformation matrix mapping the distributions $\mathbf{f}$ to the corresponding moments $\mathbf{m}$, *i.e.*,

$$\mathbf{m} = \mathbf{M}\mathbf{f}, \quad \mathbf{m} = \mathbf{M}^{-1}\mathbf{m}, \tag{10}$$

and $\mathbf{S}$ is the $q \times q$ diagonal matrix of relaxation rates $\{s_i\}$:

$$\mathbf{S} := \mathrm{diag}(s_0, s_1, \ldots, s_{q-1}). \tag{11}$$

We will restrict ourselves to two dimensional space, and the specific model to be used is the so-called D2Q9 model with nine discrete velocities. The D2Q9 model therefore has nine moments:

$$\mathbf{m} = (\rho, e, \varepsilon, j_x, q_x, j_y, q_y, p_{xx}, p_{xy})^\dagger, \tag{12}$$

where † denotes transpose; $\rho := m_0$ is mass density of the flow and the zeroth-order velocity moment, $e := m_1$ is the second-order moment related to energy, which is *not* a conserved quantity in the model, $\varepsilon := m_2$ is fourth-order moment, $j_x := \rho u_x := m_3$ and $j_y := \rho u_y := m_5$ are the first-order moments corresponding to the x and y components of the flow momentum, respectively, $q_x := m_4$ and $q_y := m_6$ are the third-order moments related to the x and y components of the energy fluxes, respectively, and $p_{xx} := m_7$ and $p_{xy} := m_8$ are the second-order moments related to the components of the stress tensor. The corresponding equilibrium moments are [13,14]:

$$m_1^{(0)} := e^{(0)} = -\rho(2 - 3u^2), \tag{13a}$$

$$m_2^{(0)} := \varepsilon^{(0)} = \rho(1 - 3u^2), \tag{13b}$$

$$m_{4,6}^{(0)} := q_{x,y}^{(0)} = -\rho u_{x,y}, \tag{13c}$$

$$m_7^{(0)} := p_{xx}^{(0)} = \rho(u_x^2 - u_y^2), \tag{13d}$$

$$m_8^{(0)} := p_{xy}^{(0)} = \rho u_x u_y. \tag{13e}$$

It should be noted that the equilibria of the conserved moments, *i.e.*, ρ and $\boldsymbol{j} := (j_x, j_y) := \rho\boldsymbol{u} := \rho(u_x, u_y)$, are themselves. The LB model used here is athermal (instead of isothermal), because the thermal energy, which is related to the moment $m_1 := e$, is not a conserved quantity.

Based on the (arbitrary) ordering of moments stipulated in Eq. (12), the transformation matrix $\mathbf{M}$ is [13,14]:

$$\mathbf{M} = \begin{pmatrix} 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 \\ -4 & -1 & -1 & -1 & -1 & 2 & 2 & 2 & 2 \\ 4 & -2 & -2 & -2 & -2 & 1 & 1 & 1 & 1 \\ 0 & 1 & 0 & -1 & 0 & 1 & -1 & -1 & 1 \\ 0 & -2 & 0 & 2 & 0 & 1 & -1 & -1 & 1 \\ 0 & 0 & 1 & 0 & -1 & 1 & 1 & -1 & -1 \\ 0 & 0 & -2 & 0 & 2 & 1 & 1 & -1 & -1 \\ 0 & 1 & -1 & 1 & -1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & -1 & 1 & -1 \end{pmatrix}. \tag{14}$$

The first, fourth, and sixth row of $\mathbf{M}$ are the first, second, and third row of $\mathbf{C}$ defined by Eq. (4), respectively. The fourth and sixth rows of $\mathbf{M}$ uniquely define the labeling (or the ordering) of the discrete velocities $\{\boldsymbol{\xi}_i\}$. It can be seen that $\mathbf{M}^{-1} \propto \mathbf{M}^\dagger$, because $\mathbf{M}$ is orthogonal. Hence $\mathbf{M}^{-1}$ can be obtained straightforwardly. The diagonal matrix of relaxation rates, $\mathbf{S}$, is

$$\mathbf{S} = \mathrm{diag}(0, s_e, s_\varepsilon, 0, s_q, 0, s_q, s_\nu, s_\nu), \tag{15}$$

where the zeros are the relaxation rates s_i $(i = 0, 3, 5)$ for the conserved modes ρ and $\rho\boldsymbol{u}$. Of course, s_i is in the unit of $1/\delta t$.

The equilibria given by Eqs. (13) lead to the speed of sound in the system:

$$c_s = \frac{1}{\sqrt{3}}c, \tag{16}$$

and the shear viscosity ν and the bulk viscosity ζ

$$\nu = \frac{1}{3s_\nu}c^2, \quad \zeta = \frac{1}{3s_e}c^2. \tag{17}$$

It should be emphasized that, because both space and time are continuous in Eq. (1), therefore the dissipation related to a non-conserved moment m_i (*i.e.*, $i \neq 0, 3, 5$) is proportional to $1/s_i$, and not to $(1/s_i - 1/2)$ as in the fully discrete LB models [13,14]. The equation of state in the system is that for ideal gases, *i.e.*, the pressure is given by $p = \rho {c_s}^2$.

In our previous work, we observe that the stability of the FV-LB scheme is not so sensitive to the values of the relaxation rates s_i for the higher-order moments ϵ and (q_x, q_y). In addition, the "magic relationship" between s_q and s_ν in the traditional LB model [13,14] no longer exists in the FV-LB model — the relaxation rate s_q for the third-order moments $(q_x, q_y) := \boldsymbol{q}$ has little effect on the boundary conditions. Based on our experience, we set $s_e = s_\epsilon = s_\nu$ and $s_q = 1.3 s_\nu$ to maintain the numerical stability of the FV-LB scheme [10]. Thus, the collision model used in this work has, in effect, two relaxation rates.

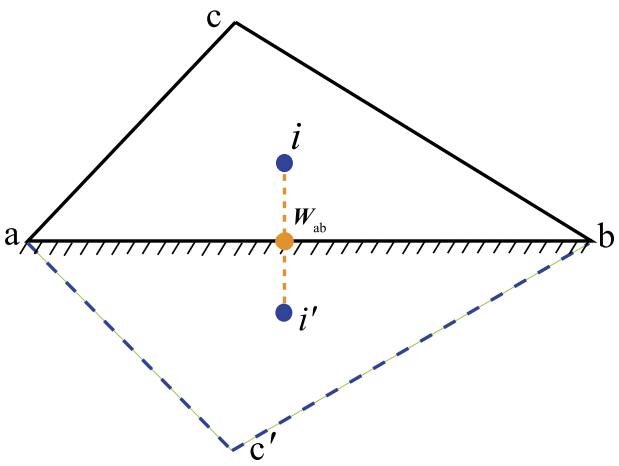

Fig. 2. Illustration of the ghost cell method. A boundary cell i with vertices "a", "b", and "c", and its ghost cell i' with vertices "a", "b", and "d".

2.3. Ghost-cell method for boundary conditions

The Dirichlet and the Neumann boundary conditions for the flow variables will be realized by the ghost cell method [15], which is illustrated in Fig. 2. The cell i with vertices "a", "b", and "c" and its centroid located at point $\boldsymbol{r}_i$ is a cell adjacent to the boundary — its edge "ab" is a part of the boundary. The cell i' with vertices "a", "b", and "c′" and its centroid at point $\boldsymbol{r}_{i'}$ is the reflective image of the cell i about the boundary segment "ab". The point W_{ab} is the intersection between the line i–i' connecting the two centroids and the boundary segment "ab".

At any time t_n, the flow variables at the boundary are given by the boundary conditions, however, the distribution functions $\mathbf{f}$ are not, because the flow variables only determine the equilibrium part of $\mathbf{f}$, but not the nonequilibrium part. To obtain $\mathbf{f}$ at the boundary, we use the scheme based on the ghost cell method to be described in what follows.

Because the point $\boldsymbol{r}_i$ and the point $\boldsymbol{r}_{i'}$ are symmetric about the boundary point W_{ab}, and the boundary is impenetrable, so the mass flux is zero at the boundary, hence $\hat{\boldsymbol{n}} \cdot \nabla \rho = 0$, where $\hat{\boldsymbol{n}}$ is the unit vector normal to the wall. This means ρ attains a local extremum at the point W_{ab} along the wall normal direction, and consequently $\rho(\boldsymbol{r}_i) = \rho(\boldsymbol{r}_{i'})$. Similarly, $\boldsymbol{u}(\boldsymbol{r}_i) = -\boldsymbol{u}(\boldsymbol{r}_{i'})$ due to the no-slip boundary conditions. The gradients $\nabla\rho$ and $\nabla\boldsymbol{u}$ at the cell centroid $\boldsymbol{r}_i$ are obtained by the least-square method, and their values at $\boldsymbol{r}_{i'}$ are known from their values at $\boldsymbol{r}_i$ based on the symmetry argument. Thus the value of $\mathbf{f}_{i'}$ at $\boldsymbol{r}_{i'}$ can be obtained by the second-order Chapman–Enskog approximation from ρ, $\boldsymbol{u}$ and their gradients. The equilibrium part of $\mathbf{f}$, $\mathbf{f}^{(0)}$, is known at W_{ab} because of the boundary conditions, and the nonequilibrium part of $\mathbf{f}$ at W_{ab} can be computed from their values at $\boldsymbol{r}_i$ and $\boldsymbol{r}_{i'}$ by second-order interpolations. In principle, this approach is better than using extrapolations [16,17] in terms of accuracy and stability. Also, in this approach, there is no distinction between the boundary and interior cells when it comes to compute the gradients — they are done in an identical manner, this thus simplified the algorithm considerably.

2.4. Implicit block LU-SGS scheme for the FV-LB scheme

As will be shown later, because of the collision term $\boldsymbol{\Omega}$ in the LBE, the traditional LU-SGS implicit scheme for fully compressible flows [18–20] cannot be directly applied to the FV-LB system. In what follows, we will introduce a block LU-SGS (BLU-SGS) algorithm for the FV-LB scheme.

With the first order approximation of the time term, an implicit form of Eq. (6) can be derived as the following:

$$\frac{\delta \mathbf{f}_i^n}{\Delta t} V_i + \mathbf{R}_i^{n+1} = \boldsymbol{\Omega}_i^{n+1} V_i, \tag{18}$$

where $\delta \mathbf{f}_i^n := \mathbf{f}_i^{n+1} - \mathbf{f}_i^n$ and $\mathbf{R}_i$ is the vector of the convective fluxes

$$\mathbf{R}_i := \sum_{j \in \mathbb{N}_i} \mathbf{F}_{ij} S_{ij}. \tag{19}$$

For a steady state, the time derivative term in the right-hand side of Eq. (18) vanished, thus Eq. (18) becomes

$$\mathbf{R}_i^{n+1} = \boldsymbol{\Omega}_i^{n+1} V_i, \tag{20}$$

in which the implicit convection fluxes on the left-hand side and the implicit collision term on the right-hand side can be linearized as the following:

$$\mathbf{R}_i^{n+1} \approx \mathbf{R}_i^n + \sum_{j\in\mathbb{N}_i}\left(\frac{\partial \mathbf{F}_{ij}}{\partial \mathbf{f}_i}\delta\mathbf{f}_i^n + \frac{\partial \mathbf{F}_{ij}}{\partial \mathbf{f}_j}\delta\mathbf{f}_j^n\right) S_{ij}, \tag{21a}$$

$$\mathbf{\Omega}_i^{n+1} \approx \mathbf{\Omega}_i^n + \frac{\partial \mathbf{\Omega}_i}{\partial \mathbf{f}_i}\delta\mathbf{f}_i^n. \tag{21b}$$

Substituting Eqs. (21) into Eq. (20) yields the following implicit scheme:

$$\sum_{j\in\mathbb{N}_i}\frac{\partial \mathbf{F}_{ij}}{\partial \mathbf{f}_i}\delta\mathbf{f}_i^n S_{ij} + \sum_{j\in\mathbb{N}_i}\frac{\partial \mathbf{F}_{ij}}{\partial \mathbf{f}_j}\delta\mathbf{f}_j^n S_{ij} - \frac{\partial \mathbf{\Omega}_i}{\partial \mathbf{f}_i}\delta\mathbf{f}_i^n V_i = \mathbf{\Omega}_i^n V_i - \mathbf{R}_i^n. \tag{22}$$

In the above equation, except the second term in the left-hand side, *i.e.*, the term of $\partial\mathbf{F}_{ij}/\partial\mathbf{f}_j$, all other terms are *local* in the sense that they are fully determined by $\mathbf{f}$ within a given cell i, while the term of $\partial\mathbf{F}_{ij}/\partial\mathbf{f}_j$ involves the values of $\mathbf{f}$ in the neighboring cells, *i.e.*, $\{\mathbf{f}_j | j \in \mathbb{N}_i\}$.

Suppose the computational domain is discretized with N_c number of cells and $m := q \times N_c$ is the degrees of freedom of the system (22), then Eq. (22) can be written in the following matrix form:

$$\mathbf{A}\delta\mathbf{f}^n = \mathbf{\Psi}^n - \mathbf{R}^n, \tag{23}$$

where $\delta\mathbf{f}^n$, $\mathbf{\Psi}^n$, and $\mathbf{R}^n \in \mathbb{R}^m$ are, respectively, the vectors of the variation of $\mathbf{f}$, the collision term given by $\mathbf{\Psi}_i := \mathbf{\Omega}_i^n V_i$, and the convective fluxes; and the coefficient matrix $\mathbf{A} \in \mathbb{R}^{m\times m}$ can be decomposed into three parts: the upper, lower, and "diagonal" part, denoted by $\mathbf{L}$, $\mathbf{U}$, and $\mathbf{D}$, respectively:

$$\mathbf{A} = \mathbf{L} + \mathbf{D} + \mathbf{U}, \tag{24}$$

therefore the system (22) is rewritten as:

$$(\mathbf{L} + \mathbf{D} + \mathbf{U})\,\delta\mathbf{f}^n = \mathbf{\Psi}^n - \mathbf{R}^n. \tag{25}$$

Generally, directly calculation the Jacobian matrices in Eq. (22) for high-order fluxes are rather expensive. Also, the directly computed Jacobians are poorly conditioned comparing with those of the first-order flux functions, so the Jacobians are often approximated by their counterparts of the first-order flux functions in the implicit system, *i.e.*,

$$\frac{\partial \mathbf{F}_{ij}}{\partial \mathbf{f}_i} := \frac{\partial \mathbf{F}(\mathbf{f}_i, \mathbf{f}_j)}{\partial \mathbf{f}_i} \approx \frac{1}{2}\left[\frac{\partial \mathbf{F}(\mathbf{f}_i)}{\partial \mathbf{f}_i} + (\Lambda_c)_{ij}\mathbf{I}\right], \tag{26a}$$

$$\frac{\partial \mathbf{F}_{ij}}{\partial \mathbf{f}_j} := \frac{\partial \mathbf{F}(\mathbf{f}_i, \mathbf{f}_j)}{\partial \mathbf{f}_j} \approx \frac{1}{2}\left[\frac{\partial \mathbf{F}(\mathbf{f}_j)}{\partial \mathbf{f}_j} - (\Lambda_c)_{ij}\mathbf{I}\right], \tag{26b}$$

where $\mathbf{F}_{ij} := \mathbf{F}(\mathbf{f}_i, \mathbf{f}_j)$ is given by Eq. (7), $\mathbf{F}(\mathbf{f}_i) = +\mathbf{J}\cdot\hat{\boldsymbol{n}}_{ij}$, $\mathbf{F}(\mathbf{f}_j) = -\mathbf{J}\cdot\hat{\boldsymbol{n}}_{ij}$ (cf. Eq. (2) for $\mathbf{J}$), and $(\Lambda_c)_{ij}$ is the spectral radius of the convective term defined as:

$$(\Lambda_c)_{ij} = \max_{0\le k\le(q-1)}\{|\boldsymbol{\xi}_k \cdot \hat{\boldsymbol{n}}_{ij}|\}.$$

Consequently, the $q \times q$ matrix $\mathbf{D}_i$ at *each* cell i, which forms the block diagonal of the matrix $\mathbf{D}$ in the linear system (25), is given by:

$$\mathbf{D}_i = -\frac{\partial \mathbf{\Omega}_i}{\partial \mathbf{f}_i}V_i + \frac{1}{2}\mathbf{I}\sum_{j\in\mathbb{N}_i}(\Lambda_c)_{ij}S_{ij}, \tag{27}$$

where $\mathbf{I}$ is the $q\times q$ identity matrix and $\partial\mathbf{\Omega}_i/\partial\mathbf{f}_i$ is the $q\times q$ Jacobian matrix of the collision term evaluated at the centroid $\boldsymbol{r}_i$.

The traditional lower-upper symmetric-Gauss-Seidel (LU-SGS) algorithm is designed for the case in which the matrix $\mathbf{D}$ is strictly diagonal. It is obvious that the Jacobian $\partial\mathbf{\Omega}_i/\partial\mathbf{f}_i$ is a full $q\times q$ matrix, hence $\mathbf{D}$ in Eq. (25) is not a diagonal, but a block-diagonal matrix. Therefore, the traditional LU-SGS algorithm cannot be directly applied to solve the linear system (25). Nevertheless, the LU-SGS algorithm can be modified to solve the system (25). Specifically, the system (25) can be solved by using symmetric Gauss–Seidel iterative method with an inner iteration process to accelerate convergence, and within each Gauss-Seidel iteration step, the inversion of a small $q\times q$ matrix $\mathbf{D}_i$ at each cell i can be achieved by the LU decomposition method. The proposed scheme is based on a block LU-SGS (BLU-SGS) algorithm.

Let k be the index of the inner iteration, Eq. (25) can be solved by the BLU-SGS algorithm with the following two steps in each inner iteration loop:

1. Forward sweep from $i = 1, 2, \ldots, N_c$:

$$\mathbf{D}_i\delta\mathbf{f}_i^{(*)} = \mathbf{\Psi}_i^n - \mathbf{R}_i^n - \frac{1}{2}\sum_{j\in\mathbb{L}_i(j)}\left[\Delta\mathbf{F}_j^{(*)} - (\Lambda_c)_{ij}\delta\mathbf{f}_j^{(*)}\right]S_j - \frac{1}{2}\sum_{j\in\mathbb{U}_i(j)}\left[\Delta\mathbf{F}_j^{(k-1)} - (\Lambda_c)_{ij}\delta\mathbf{f}_j^{(k-1)}\right]S_j, \tag{28}$$

where $\mathbb{L}_i(j) := \{j < i | j \in \mathbb{N}_i(j)\}$ and $\mathbb{U}_i(j) := \{j > i | j \in \mathbb{N}_i(j)\}$;

2. Backward sweep from $i = N_c, N_c - 1, \ldots, 1$:

$$\mathbf{D}_i \delta\mathbf{f}_i^{(k)} = \mathbf{\Psi}_i^n - \mathbf{R}_i^n - \frac{1}{2}\sum_{j\in\mathbb{L}_i(j)}\left[\Delta\mathbf{F}_j^{(*)} - (\Lambda_c)_{ij}\delta\mathbf{f}_j^{(*)}\right]S_j - \frac{1}{2}\sum_{j\in\mathbb{U}_i(j)}\left[\Delta\mathbf{F}_j^{(k)} - (\Lambda_c)_{ij}\delta\mathbf{f}_j^{(k)}\right]S_j, \tag{29}$$

where

$$\Delta\mathbf{F}_j^{\square} = \frac{\partial\mathbf{F}(\mathbf{f}_j)}{\partial\mathbf{f}_j}\delta\mathbf{f}_j^{\square} \approx \mathbf{F}(\mathbf{f}_j^n + \delta\mathbf{f}_j^{\square}) - \mathbf{F}(\mathbf{f}_j^n), \tag{30}$$

and $\square$ denotes the superscript (k), $(k-1)$, or $(*)$, and $\mathbf{f}^{(*)}$ is the intermediate result which is obtained and used in the following way:

$$\text{Forward sweep:} \qquad \mathbf{D}\delta\mathbf{f}^{(*)} := \mathbf{\Psi}^n - \mathbf{R}^n - \mathbf{L}\delta\mathbf{f}^{(*)} - \mathbf{U}\delta\mathbf{f}^{n,k-1}, \tag{31a}$$

$$\text{Backward sweep:} \qquad \mathbf{D}\delta\mathbf{f}^{n,k} := \mathbf{\Psi}^n - \mathbf{R}^n - \mathbf{L}\delta\mathbf{f}^{(*)} - \mathbf{U}\delta\mathbf{f}^{n,k}, \tag{31b}$$

that is, in the forward sweep, the missing unknowns in the upper diagonal part are replaced by their values obtained in the previous inner iteration step, and in the backward sweep, the missing unknowns in the lower diagonal part are replaced by their values obtained in the forward sweep.

In summary, the implicit procedure for the inner iteration at time t_n reads as follows:

1. Initialize $\delta\mathbf{f}_i^{n,0} = \mathbf{0}$;
2. Carry out the following iterations K times:
   Do $k = 1,\ K$
   Solving system (25) with the BLU-SGS algorithm prescribed by Eqs. (28) and (29);
   Enddo
3. Set $\delta\mathbf{f}_i^n = \alpha \cdot \delta\mathbf{f}_i^{n,K}$ with the relaxation factor $0 < \alpha \leq 1$.

Note that α is a global parameter in this work. The relaxation process in step 3 is necessary for the sake of stability, because of the approximation of the flux Jacobian in Eq. (30) and omission of the time derivative term. If the flux Jacobian is not approximated and the linear system (25) is to be solved *exactly*, then the present method becomes the Newton iteration method, which is unstable without good initial value.

In the proposed scheme described above, the inversion of the small $q \times q$ matrix $\mathbf{D}_i$ needs to be computed and stored at each cell i and at each time step before each BLU-SGS sweeping, which does increase the memory requirement a little. However, the inversion of $\mathbf{D}_i$ can be efficiently done by the exact LU decomposition method. Therefore, the proposed implicit scheme is both efficient in CPU time and low-cost in memory requirement.

3. Numerical results and discussions

In this section, we demonstrate the effectiveness and efficiency of the proposed implicit FV-LB scheme based on the BLU-SGS algorithm by using the numerical results of several steady flows in 2D. Specifically, three cases are chosen to be studied: the Blasius boundary-layer flow past a semi-infinite flat plate in 2D, and steady viscous and inviscid flows past a cylinder in 2D. We compare the CPU time T_{CPU} and the number of iterations required to attain the steady state by the first-order explicit Euler time marching scheme and the proposed implicit FV-LB scheme. For a computed quantity u, we measure the L_2-normed error at a given number of iteration n, $L_2^{(n)}(u)$, and the relative residual $\|\delta u\|_2$ defined as the following:

$$L_2^{(n)}(u) := \|u^n - u^{n-1}\|_2 := \sqrt{\sum_{i=1}^{N_c}\left(u_i^n - u_i^{n-1}\right)^2}, \tag{32a}$$

$$\|\delta u\|_2 = \frac{L_2^{(n)}(u)}{L_2^{(1)}(u)}, \tag{32b}$$

where u is a computed quantity, $u_i^n := u(\mathbf{r}_i, t_n)$ is the value of u at the cell i and n-th iteration, and N_c is the total number of cells in a simulation. In what follows, we use the residue of the streamwise velocity u_x as the measure of convergence in all calculations.

3.1. Laminar boundary-layer flow past a semi-infinite flat plate in 2D

Fig. 3 illustrates the flow configuration, a typical mesh, and boundary conditions for the Blasius boundary-layer flow past a semi-infinite flat plate in 2D. The domain size in the simulations is $L_x \times L_y = 122.5 \times 50$, $(x,\ y) \in [-22.5,\ 100] \times [0,\ 50]$, and the origin of the coordinate system sits at the tip of the flat plate, and the length of the plate is $L = 100$. The parameters

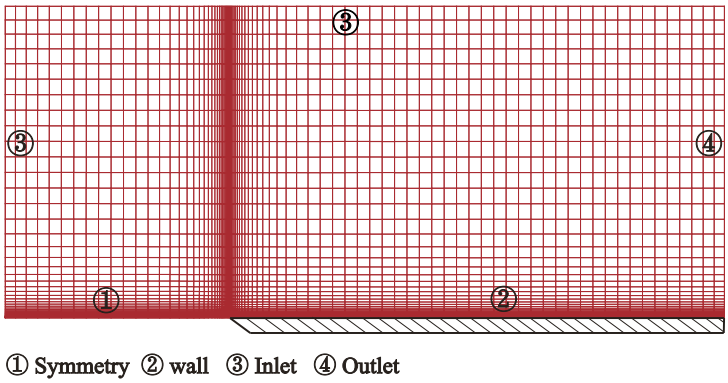

Fig. 3. The flow configuration, mesh, and boundary conditions for the Blasius flow past a semi-infinite flat plate.

for the nonuniform meshes covering the flow domain are the height of the first grid $h_0 = 10^{-4}L$ and the grid spacing stretching factor $r = 1.05$ along the normal direction of the plate. The total number of cells is $N_c = 12\,576$. The relaxation factor in the implicit inner-iteration procedure is set to $\alpha = 0.25$ in this case.

The boundary conditions for the flow are specified as the following. On the left boundary of the domain, the velocity field is specified by the inflow boundary conditions of $\boldsymbol{U} = (U_\infty, 0)$, and the pressure boundary conditions are $\partial p/\partial \hat{\boldsymbol{n}} = \boldsymbol{0}$, where $\hat{\boldsymbol{n}}$ is the unit vector out-normal to the left boundary. On the right boundary, both the velocity and the pressure are imposed with the zero-gradient outflow conditions. On the top boundary, the boundary conditions are identical to that of the inflow conditions at the left boundary. At the bottom boundary, the symmetric boundary conditions and the no-slip boundary conditions are applied to the forefront of the flat plate, *i.e.*, $-22.5 \le x \le 0$, and along the plate, *i.e.*, $0 \le x \le 100$, respectively. The boundary conditions are realized by the ghost cell method when appropriate. The Reynolds number of the flow is

$$\mathrm{Re} = \frac{U_\infty L}{\nu}.$$

Unless otherwise stated, $U_\infty = 0.1$ and $\mathrm{Re} = 10^5$ are used in this case.

The Blasius flow admits the following self-similar solutions for the flow field:

$$\bar{u} := \frac{u_x}{U_\infty} = f(\eta), \qquad \bar{v} := \frac{u_y}{V_\infty} = g(\eta), \tag{33a}$$

$$\eta := \frac{y}{x}\sqrt{\mathrm{Re}_x}, \quad \mathrm{Re}_x := \frac{U_\infty x}{\nu}, \quad V_\infty := \frac{U_\infty}{\sqrt{\mathrm{Re}_x}}, \tag{33b}$$

where both $f(\eta)$ and $g(\eta)$ are solutions of ordinary differential equations [21].

The computed dimensionless velocity profiles, $\bar{u}$ and $\bar{v}$, are shown in Fig. 4. The velocity profiles obtained by the implicit FV-LB scheme are compared with the results obtained with the explicit FV-LB scheme and the analytic solutions. The accuracy of the explicit FV-LB scheme for this flow has been investigated previously and the convergence of the FV-LB scheme have been investigated [10]. The effectiveness of the implicit FV-LB scheme is to be ensured by its agreement to the explicit FV-LB scheme. With the velocity residue of 10^{-5}, the results obtained with both the explicit and implicit FV-LB schemes are almost identical to each other, as expected. Fig. 4 shows the analytic solutions of $\bar{u}(\eta)$ and $\bar{v}(\eta)$ and compares them with the numerical results obtain by using the explicit and implicit FV-LB schemes with the velocity residue of 10^{-5} and 10^{-12}, respectively. Clearly the results obtained by using the implicit FV-LB scheme agree well with the analytic solutions.

To demonstrate the efficiency of the proposed implicit FV-LB scheme, we show in Fig. 5 the convergence histories of the residual defined by Eq. (32b) for both the implicit and explicit LB schemes in terms of both the CPU time, T_{CPU}, and the number of iterations, n. It is clear that the explicit LB scheme converges very slowly when it converges, due to a large Reynolds number $\mathrm{Re} = 10^5$. In fact, we simply cannot afford the time to use the explicit FV-LB scheme with a convergence criterion of $\|\delta \boldsymbol{u}\|_2 \le O(10^{-12})$. Thus we can only compare the CPU time (T_{CPU}) to attain the steady state with a larger residue of $\|\delta \boldsymbol{u}\|_2 \le O(10^{-5})$, and the results are given in Table 1. The proposed implicit FV-LB scheme is about 45 times faster than its explicit counterpart (cf. the case of $K = 3$ *vs.* the explicit case in Table 1). More importantly, the implicit scheme can drive the residue as small as $O(10^{-12})$, as evidently shown in Fig. 5, while it is very difficult for the explicit FV-LB scheme to do so, if possible at all.

The speed of convergence of the implicit FV-LB scheme depends on the number of inner iterations K, as evident in the data shown in both Table 1 and Fig. 5. While the total number of iterations, n, to attain the convergence monotonically decreases as K increases, however, the actual computational time, *i.e.*, the elapsed CPU time, taken to achieve the convergence, T_{CPU}, does not depend on K monotonically. As shown in both Fig. 5 and Table 1, the elapsed CPU time T_{CPU} is optimal when

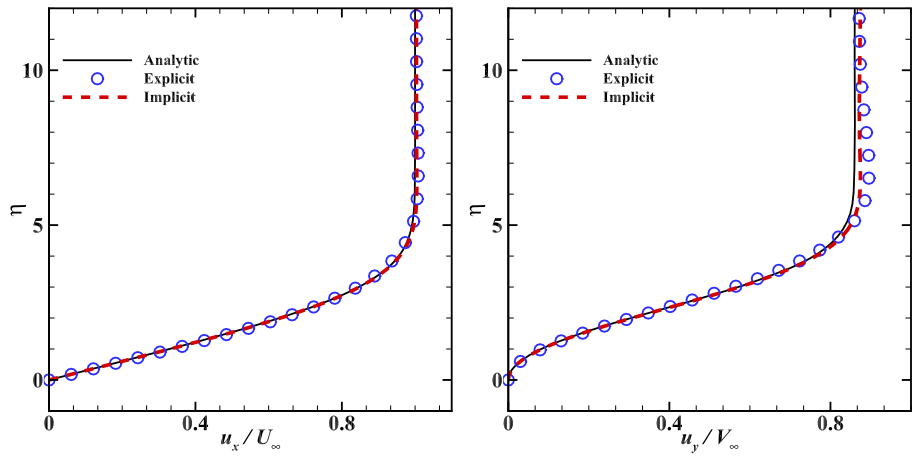

Fig. 4. The Blasius boundary-layer flow past a semi-infinite flat plate at $\mathrm{Re} = 10^5$ and $x = 40$. The dimensionless horizontal velocity $\bar{u} := u_x/U_\infty$ (left) and vertical velocity $\bar{v} := u_y/V_\infty$ (right) obtained numerically by explicit LB (symbol ∘) and implicit LB scheme (dashed lines), compared with the analytic solutions (solid lines).

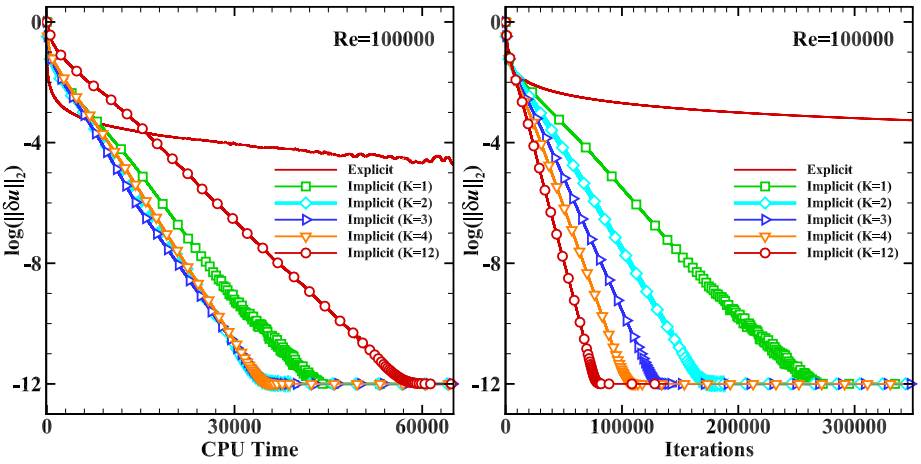

Fig. 5. The Blasius flow at $\mathrm{Re} = 10^5$. The convergence histories of the elapsed CPU time T_{CPU} (left) and the total number of iterations n (right) for the relative residual $\|\delta u\|_2$ obtained with explicit and implicit LB schemes.

Table 1
The Blasius flow in 2D. The CPU time (in seconds) required to obtain a steady state with a residue of $O(10^{-5})$ by using the explicit FV-LB scheme and the implicit FV-LB scheme with the number of inner iterations $K = 1$, 2, 3, 4, and 12. The number in italic indicates the minimum of T_{CPU}.

Method	Explicit	$K = 1$	$K = 2$	$K = 3$	$K = 4$	$K = 12$
T_{CPU} (s)	550 000	16 000	13 100	*12 200*	14 000	24 000

$K = 3$, while the case of $K = 12$ takes the longest CPU time, and the ratio between the longest ($K = 12$) and the shortest ($K = 3$) CPU times is about 2.

3.2. Steady viscous flow past a circular cylinder in 2D

The second test case is the steady flow past a cylinder in 2D at the Reynolds number $\mathrm{Re} = 10$, 20, and 40. The flow domain in this case has curved boundary, as opposed to the linear boundary in the previous case. The configuration of the computational domain and the relevant boundary conditions are illustrated in Fig. 6. Because of the flow symmetry about the horizontal axis along the streamwise direction, the computational domain uses one half of the entire flow domain as shown in Fig. 6. The size of the computational domain is $L \times H = 120R \times 40R$, where R is the cylinder radius. The relaxation factor is set to $\alpha = 0.25$ in this case.

The boundary conditions are the following. On the left and top boundaries of the domain, a constant velocity $\boldsymbol{U} = (U_\infty, 0)$ are imposed through the equilibrium distribution functions, and the boundary conditions for the pressure are that

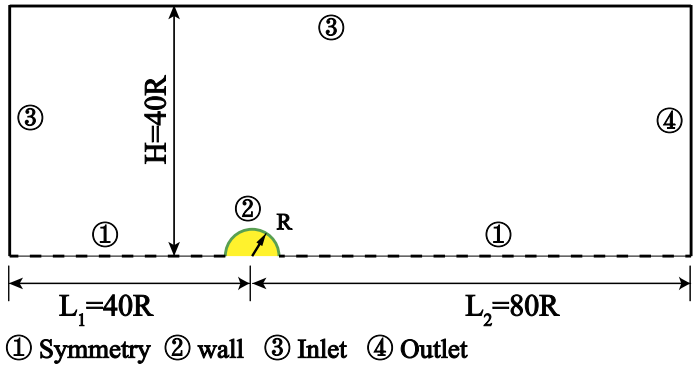

Fig. 6. Schematics of the computational domain for the flow past a 2D cylinder.

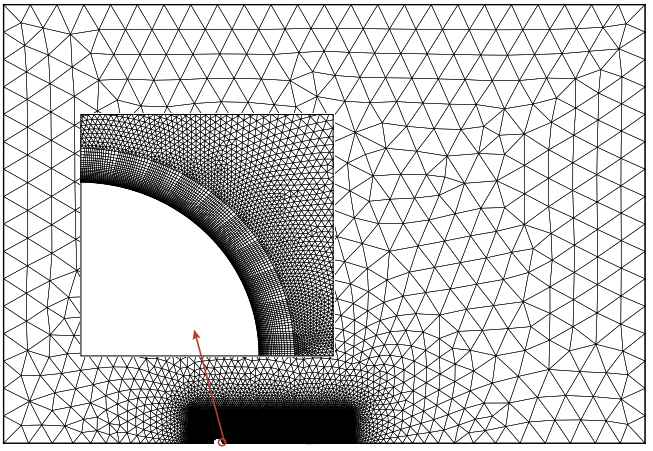

Fig. 7. The configuration of the hybrid unstructured mesh for the flow past a 2D cylinder.

$\partial p/\partial \hat{\boldsymbol{n}} = \mathbf{0}$, where $\hat{\boldsymbol{n}}$ is the unit vector out-normal to the left and top boundaries. At the bottom boundary, the no-slip boundary conditions are imposed on the cylinder surface, and the symmetric boundary conditions are imposed on the two straight segments. All the boundary conditions are realized by the ghost cell method discussed in Sec. 2.3. The Reynolds number of the flow is based on the cylinder diameter $D = 2R$:

$$\mathrm{Re} = \frac{U_\infty D}{\nu},$$

where $D = 0.01$ and $U_\infty = 0.1$. We compute the flows with $\mathrm{Re} = 10$, 20, and 40 by varying the viscosity ν.

The results shown in what follows are obtained with a mesh consisting of 54 174 cells, and the first-layer cell height is $h_0 = 10^{-3}D$ and the stretching factor along the radial direction is $r = 1.1$. The angular resolution is $\delta\theta = 0.6°$. To accurately capture the details of the flow separation region behind the cylinder, we use hybrid meshes with quadrilateral cells about the cylinder and nearly isotropic triangular cells away from the cylinder, as shown in Fig. 7. The convergence study for this particular flow using the explicit FV-LB scheme has been investigated in our previous study [10]. For the explicit FV-LB scheme [10], we use $\mathrm{CFL} = 0.1$, and for the implicit FV-LB scheme, we use $\alpha = 0.1$ for the relaxation in the inner iteration.

We compute the following quantities for the flow: the friction coefficient $C_\mathrm{f}(\theta)$ along the cylinder boundary, the drag coefficient C_D, the pressure coefficient at both the rear stagnation point ($\theta = 0$), $C_\mathrm{P}(0)$, and the front stagnation point ($\theta = \pi = 180°$), $C_\mathrm{P}(\pi)$, the separation angle θ_s, and the dimensionless separation length L_s/R. The separation length L_s is determined by the location where $u_x = 0$ along the horizontal symmetric line of the flow. The separation angle θ_s is determined at the cylinder wall where the friction force vanishes, *i.e.*, the friction coefficient $C_\mathrm{f} = 0$. To verify and validate the proposed implicit FV-LB scheme, we have tested in a few cases to ensure that all the results obtained by using both the implicit and explicit schemes are indeed nearly identical when the residual errors reach 10^{-12}.

The results of C_D, $C_\mathrm{P}(0)$ and $C_\mathrm{P}(\pi)$, θ_s, and L_s/R are tabulated in Table 2, along with some existing results obtained by finite difference (FD) [22,23],[1] the interpolation-supplemented LB (IS-LB) [24], and finite-difference LB (FD-LB) [25] methods.

[1] The values of θ_s in [23] are digitized from Fig. 16 in [23].

Table 2
The steady flow past a cylinder in 2D with Re = 10, 20, and 40. C_D, $C_P(\theta=0)$, $C_P(\theta=\pi)$, L_s/R, and θ_s, obtained by the FD [22,23], the IS-LB [24], the FD-LB [25], and the present method.

Re	Method	C_D	$-C_P(0)$	$C_P(\pi)$	L_s/R	θ_s
10	FD [22]	2.846	0.742	1.489	0.53	29.6°
	IS-LB [24]	3.170	0.687	1.393	0.474	26.0°
	FD-LB [25]	2.801	0.6733	1.4867	0.4891	29.83°
	Present	2.964	0.6584	1.5240	0.6477	33.88°
20	FD [22]	2.045	0.589	1.269	1.88	43.7°
	FD [23]	2.0001	0.54	1.28	1.82	45.3°
	IS-LB [24]	2.152	0.567	1.233	1.842	42.96°
	FD-LB [25]	2.021	0.5465	1.2659	1.8480	43.58°
	Present	2.092	0.5386	1.3315	1.9934	45.79°
40	FD [22]	1.522	0.509	1.144	4.69	53.8°
	FD [23]	1.4980	0.46	1.14	4.48	54.89°
	IS-LB [24]	1.499	0.487	1.113	4.490	52.84°
	FD-LB [25]	1.515	0.4808	1.154	4.454	51.86°
	Present	1.546	0.4771	1.207	4.625	54.99°

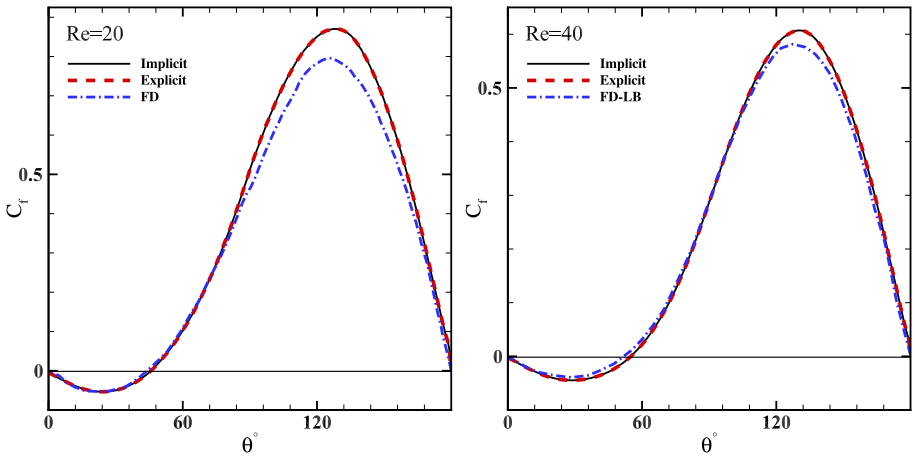

Fig. 8. The steady viscous flow past a circular cylinder in 2D. The surface friction coefficient $C_f(\theta)$ along the cylinder surface at Re = 20 (left) and 40 (right). The results obtained with both the implicit and explicit FD-LB schemes are compared with that obtained by the finite-difference (FD) method [23] for Re = 20 and finite-difference LB (FD-LB) scheme [25] for Re = 40.

While our values of C_D and $C_P(0)$ fall within the range of existing data [22–25], our values of $C_P(\pi)$, θ_s, and L_s/R are consistently larger than the existing data [22–25], as clearly shown in Table 2. The discrepancy can be attributed to several factors, all related to the details in the numerics. The first reason may be due to the mesh resolution. As we have noted previously [10], we use a mesh which is finer than all those used in previous work [22–25], especially in the radial direction — the finest angular and radial resolutions are $\delta\theta = 0.6°$ and $h_0 = 10^{-3}D$, respectively. Our mesh ensures an adequate resolution of the boundary layer with a given Reynolds number [10], which is not the case in the previous studies [22–25]. Also, we use a fine mesh to cover the entire recirculation zone; and the domain size is different from the ones used in the previous studies. Second, the computational efficiency of the proposed implicit LB scheme allows us to reduce the residual error in the velocity field to $O(10^{-12})$, which demands considerable longer CPU time for the its explicit counterpart [10], if it converges at all. We note that the values of both the separation length L_s and the separation angle θ_s depend on the size of the residual error. Third and the last is the post processing. In our previous study [10], we relied on a visualization software to extract L_s and θ_s, which may introduce errors. To make this point, we show in Fig. 8 the friction coefficient along the cylinder surface, $C_f(\theta)$.

To demonstrate the efficiency of the implicit FV-LB scheme, which is the main thrust of this work, we first tabulate the CPU time T_{CPU} to achieve the steady state with a residual velocity field $\|\delta \boldsymbol{u}\|_2 \leq 10^{-7}$ by using the explicit FV-LB scheme and its implicit counterpart with the number of the inner iterations $K = 1$, 2, 3, 4, and 12. We observe that the optimal value of the number of inner iterations K increases as the Reynolds number Re increases — for Re = 10, 20, and 40, the optimal number of the inner sweepings K is 2, 3, and 12, respectively, It is clear that the proposed implicit FV-LB scheme significantly reduces the CPU time to obtain the steady state.

Fig. 9 shows the histories of residual velocity field defined in Eq. (32b) as a function of the CPU time T_{CPU} and the number of iterations n, for the explicit FV-LB scheme and the implicit FV-LB scheme with different number of the inner iterations K. As clearly shown in the figure, both the CPU time T_{CPU} and the iteration number n for the explicit FV-LB

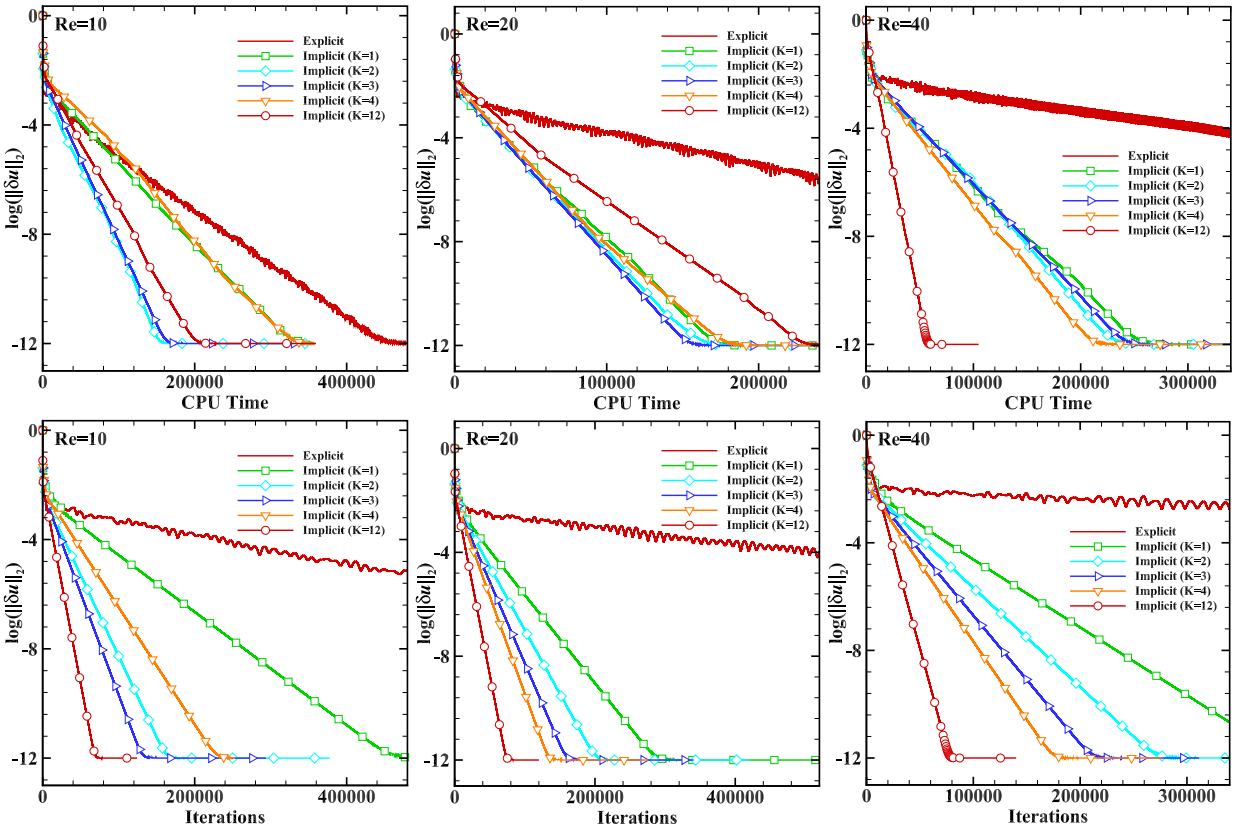

Fig. 9. Steady viscous flow past a cylinder in 2D at Re = 10 (left), 20 (middle), and 40 (right). Convergence histories of the residual velocity field as a function of the CPU time T_{CPU} (top) and the total number of iterations n (bottom) for the explicit LBE and implicit LB scheme with different number of inner iterations K.

Table 3
The viscous flow past a cylinder in 2D. The CPU time (in seconds) required to obtain a steady state with the residue $\|\delta \boldsymbol{u}\|_2 \leq 10^{-7}$ by using the explicit FV-LB scheme and the implicit FV-LB scheme with the number of inner iterations $K = 1$, 2, 3, 4, and 12. The numbers in italic are the minimal CPU times with a fixed Reynolds number Re.

Method	Explicit	$K = 1$	$K = 2$	$K = 3$	$K = 4$	$K = 12$
Re = 10	200 000	160 000	*76 500*	79 000	167 000	105 000
Re = 20	350 000	84 200	78 500	*71 500*	80 500	113 000
Re = 40	4 400 000	200 000	140 000	110 000	85 000	*45 000*

scheme to achieve convergence grows rapidly as the Reynolds number Re increases. In contrast, the Reynolds number has a weak effect on the CPU time T_{CPU} or the number of iterations n for the implicit FV-LB scheme. For the case of Re = 10, the implicit FV-LB scheme is at least three times faster than its explicit counterpart. However, for the cases of Re = 20 and 40, we cannot afford to carry out the calculations using the explicit FV-LB scheme with the convergence criterion $\|\delta \boldsymbol{u}\|_2 \leq 10^{-12}$, for they will take too long if they converge at all. The data in Table 3 indicates that the implicit FV-LB scheme is about 100 times faster than its explicit counterpart for the case Re = 40.

3.3. The inviscid flow past a cylinder in 2D

To demonstrate the robustness of the proposed implicit FV-LB scheme, it is used to simulate the *inviscid* flow past a cylinder in 2D. In this case, the boundary conditions on the cylinder wall are that for the inviscid slip wall. While we cannot set the viscosity ν equal to zero, we use a very large Reynolds number to approximate the inviscid flow — we set $\text{Re} = U_\infty D/\nu = 1.0 \times 10^{14}$ in this flow, where $U_\infty = 0.1$ and $D = 1.0$, thus $\nu = 1.0 \times 10^{-15}$. The relaxation factor is set to $\alpha = 0.15$ in this case. It should be noted that no traditional LB scheme with a *linear* collision operator is stable with such a small viscosity. The explicit FV-LB scheme [10] is also unstable with $\nu = 1.0 \times 10^{-15}$.

The computational domain we used is an annulus with the outer radius of $39.5R$ and the inner radius of R, where R is the radius of the cylinder, and set $R = D/2 = 0.5$. The mesh is a structured one with quadrilateral cells, although the data structure in the code is written for unstructured meshes. The angular resolution is uniform and $\delta\theta = 360°/400 = 0.9°$.

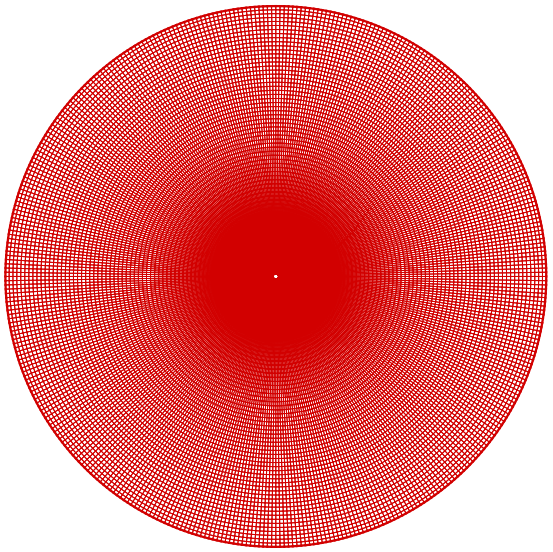

Fig. 10. The mesh for the inviscid flow past a cylinder in 2D.

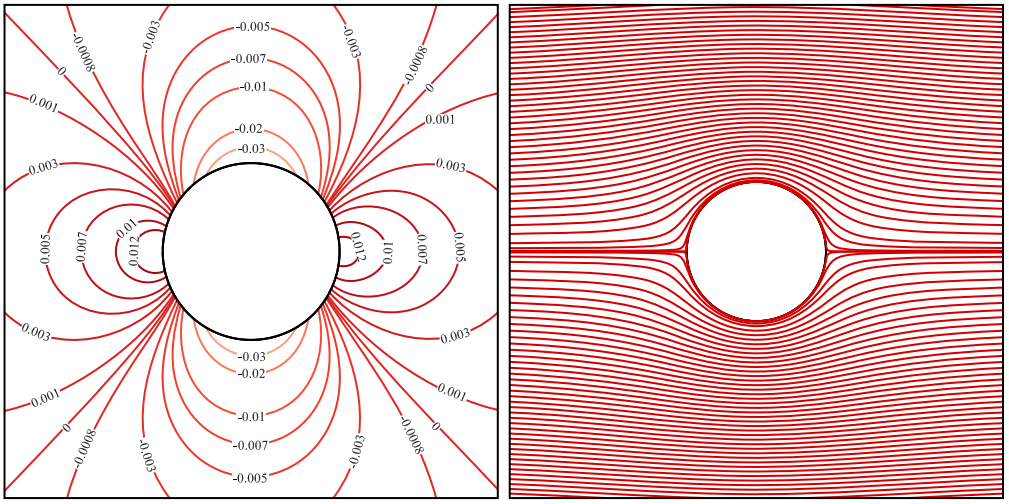

Fig. 11. The inviscid flow past a cylinder in 2D: the contours of the normalized pressure fluctuation $\delta\bar{p} := p/p_0 - 1$, $p_0 = 1/3$ (left) and the steamlines (right) about the cylinder.

The height of the first grid at the cylinder surface is $h_0 = 0.01D$. The domain is divided into two concentric annuli, *i.e.*, $0.5 \le r \le 4.5$ and $4.5 \le r \le 39.5$. The grid stretching factor along the radial direction is 1.05 and 1.2, resulting in 93 and 67 grid points along the radial direction r, for the inner and outer annulus, respectively. The total number of cells is 64 000. The mesh used for the inviscid flow is shown in Fig. 10.

The contours of the normalized pressure fluctuation $\delta\bar{p} := p/p_0 - 1$, $p_0 = 1/3$ (in the lattice units), and the streamlines about the cylinder are shown in Fig. 11. The figures show that the both the pressure and the stream function are symmetric about the x and y axes, as they are expected for the inviscid flow in this case. The further quantify the efficacy for the proposed implicit FV-LB scheme to simulate inviscid flows, we compute both the drag coefficient C_D and the lift coefficient C_L for the flow past a cylinder in 2D, $C_\mathrm{D} \approx 1.2535 \times 10^{-3}$ and $C_\mathrm{L} \approx -6.2471 \times 10^{-6}$. For an ideal inviscid flow, C_D should be zero. However, we are using a viscous solution with very small viscosity ($\nu = 10^{-15}$) to approximate its inviscid counterpart, and an error is expected, as measured by the value of C_D. The value of C_L can be used as a measure for the symmetry of the flow about x axis. The value of C_L is indeed consistent with the residual error ($O(10^{-8})$) in this case.

We compute the pressure coefficient distribution $C_\mathrm{P}(\theta)$ along the cylinder surface:

$$C_\mathrm{P}(\theta) = \frac{p(\theta) - p_\infty}{\frac{1}{2}\rho_\infty U_\infty^2}, \tag{34}$$

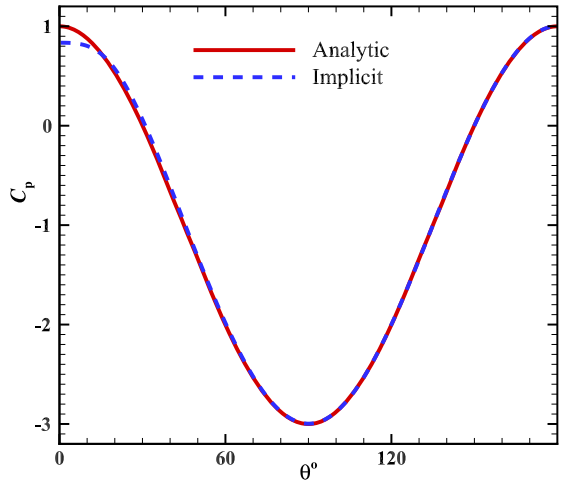

Fig. 12. The inviscid flow past cylinder in 2D. The pressure distribution on the cylinder surface.

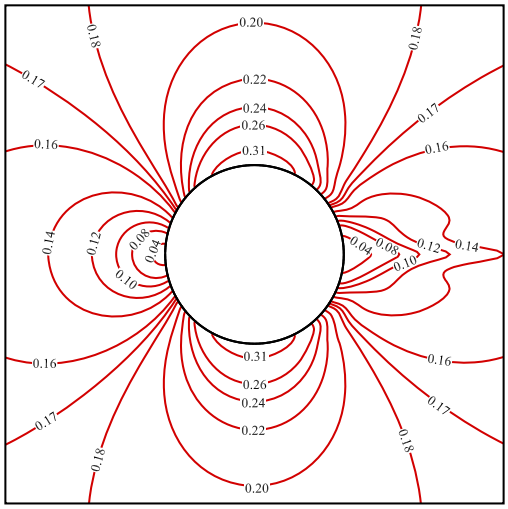

Fig. 13. The inviscid flow past a cylinder in 2D. The contours of local Mach number $|\boldsymbol{u}(\boldsymbol{x})|/c_s$.

where $\rho_\infty = 1$ is the density at the far-field boundary. The computed and the analytic pressure coefficient distributions are shown in Fig. 12. The simulated pressure coefficient distribution agrees with the analytic solution very well except for the small region near the rear stagnation point. The cause of this discrepancy between the numerical and analytic solutions is the viscous effect in the scheme, regardless however small that is — the viscous effect due to numerical viscosity and an extreme small viscosity ($\nu = 10^{-15}$) in the system induces a very small, invisible wake at the rear stagnation point ($\theta = 0$), and its effect is clearly shown in $C_P(\theta)$ near $\theta = 0$. To make this point, we show the contours of local Mach number $|\boldsymbol{u}(\boldsymbol{x})|/c_s$ in Fig. 13. Clearly, the velocity is asymmetric about the y axis, and has an oscillation along the x axis behind the cylinder. This is a manifestation of viscous effect, mainly a numerical one here. This type of effect may be removed by using, for instance, low-Mach-number preconditioning technique [20,26].

The convergence histories of the residual velocity field $\|\delta\boldsymbol{u}\|_2$ as a function of the CPU time T_{CPU} and the number of iterations n are shown in Fig. 14. It can be observed that the convergence behavior for the inviscid case differs from that for the viscous flows, as shown in Figs. 14 and 9, respectively. First of all, the residual errors at coarser grids decay first at a rate faster than that at finer grids, and this process at coarser grids takes much longer in the inviscid case than the viscous case. For the viscous case shown in Fig. 9, this process completes so fast that it is not clearly seen. And secondly, the residual error $\|\delta\boldsymbol{u}\|_2$ in the inviscid case, especially with $K \leq 3$, oscillates far more severely than in the viscous cases (cf. Fig. 9). It is evident that the implicit FV-LB scheme is especially robust and efficient to solve the flows with extremely small viscosity, which cannot be handled by its explicit counterpart.

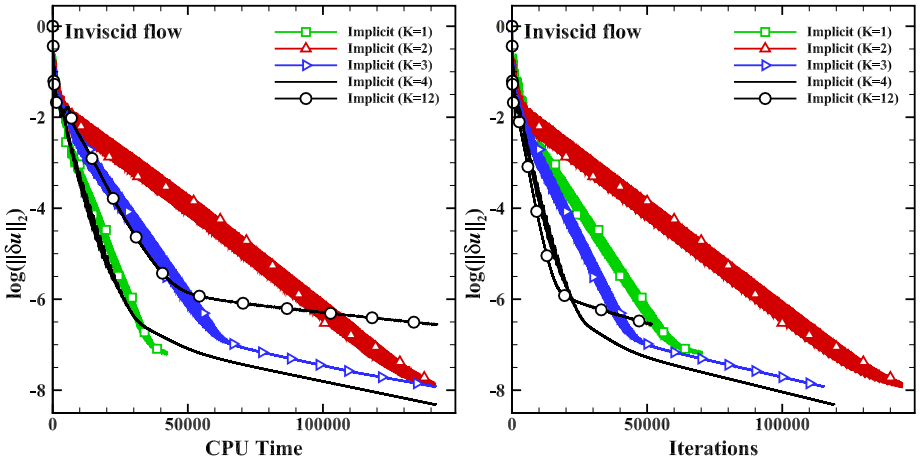

Fig. 14. The inviscid flow past a cylinder in 2D. The convergence histories of the residual error of the velocity field as a function of the CPU time T_{CPU} (left) and the number of iterations n (right). Note that $\|\delta \boldsymbol{u}\|_2$ is highly oscillatory, especially for the cases of $K \leq 3$.

4. Conclusion

In this work we propose and develop a fully implicit finite-volume lattice-Boltzmann scheme on arbitrary unstructured meshes in 2D. The proposed implicit FV-LB scheme is based on the block LU-SGS relaxation technique, which is used to solve the implicit system arising from the linearization of collision and advection terms in the lattice Boltzmann equation. The implicit FV-LB scheme is particularly efficient for steady flow simulations because it is free of the CFL-number restriction in its explicit FV-LB counterpart. The proposed implicit FV-LB scheme is mesh-type independent and allows local mesh refinement thus it has a great flexibility to handle complex geometries.

The efficacy and efficiency of the proposed implicit FV-LB scheme compared to its explicit counterpart are demonstrated through three test cases in 2D: the laminar Blasius flow past a flat plate at the Reynolds number $\text{Re} = 10^5$, the steady viscous flow past a circular cylinder at $\text{Re} = 10$, 20, and 40, and the inviscid flow past a circular cylinder. The results clearly show that the implicit FV-LB scheme is as accurate as its explicit counterpart [10], and converges much faster. In terms of CPU time, for the Blasius flow and the steady viscous flow past a cylinder, the proposed implicit FV-LB scheme is at least 45 and 100 times, respectively, faster than its explicit counterpart. For the case of the inviscid flow past a cylinder which cannot be simulated with the explicit scheme, the proposed implicit scheme is shown to be particularly robust and efficient.

Clearly, the implicit FV-LB scheme proposed in this work is not as computationally efficient as a good implicit scheme based on a direct FV discretization of the incompressible Navier–Stokes equation. The reason is mainly due to the difference in these two sets of equations to be solved. The Boltzmann equation with a fixed set of discrete velocities, Eq. (1), is a first-order PDE — the advection term is *linear*, and its nonlinearity resides in the *local* collision term, which is also responsible for the viscous effect, whereas the Navier–Stokes equation is a second-order nonlinear PDE — the advection term is *nonlinear*, and the viscous term is linear but of second-order. In the advection process in Eq. (1), the distribution functions are completely decoupled; and their coupling is only through the local collision term. The Jacobian of the advection term in Eq. (1) is a *constant diagonal* matrix, and the resulting linear system is block-diagonal, as oppose to a band-structured system from the Navier-Stokes equation. Consequently, the linear system from the LBE is *point* implicit, as opposed to *line* implicit or a fully coupled system from the Navier-Stokes equation. Hence, the implicit FV-LB scheme converges slower than an implicit scheme based on a direct FV discretization of the Navier-Stokes equation.

Finally, we note that while the BLU-SGS technique developed in the present work is limited to the lattice Boltzmann equation, it can also be used to solve other discrete kinetic equations, such as the discrete velocity models [27–30] and the discrete-ordinates method [31,32]. Although the proposed implicit FV-LB scheme is second-order accurate in space, its accuracy can be enhanced by employing advanced techniques such as spectral difference, which will be a subject of our future studies.

Acknowledgements

The authors are grateful to Professor Manfred Krafczyk at Institut für rechnergestützte Modellierung im Bauingenieurwesen (iRMB), Technischen Universität Braunschweig, Germany, for providing computing resources with which part of this research is carried out. The authors would like to acknowledge the generous support from the Richard F. Barry Jr. Endowment at Old Dominion University.

References

[1] G.R. McNamara, G. Zanetti, Use of the lattice Boltzmann to simulate lattice-gas automata, Phys. Rev. Lett. 61 (1988) 2332–2335.

[2] F.J. Higuera, J. Jiménez, Boltzmann approach to lattice gas simulations, Europhys. Lett. 9 (7) (1989) 663–668.
[3] D. Yu, R. Mei, L.-S. Luo, W. Shyy, Viscous flow computations with the method of lattice Boltzmann equation, Prog. Aerosp. Sci. 39 (5) (2003) 329–367.
[4] L.-S. Luo, M. Krafczyk, W. Shyy, Lattice Boltzmann method for computational fluid dynamics, in: R. Blockley, W. Shyy (Eds.), Encyclopedia of Aerospace Engineering, Wiley, New York, 2010, pp. 651–660, Ch. 56.
[5] P.J. Dellar, L.-S. Luo, Lattice Boltzmann methods, in: B. Engquist (Ed.), Encyclopedia of Applied and Computational Mathematics, Springer, Berlin, 2015, pp. 774–778.
[6] O. Filippova, D. Hänel, Grid refinement for lattice-BGK models, J. Comput. Phys. 147 (1) (1998) 219–228, http://dx.doi.org/10.1006/jcph.1998.6089.
[7] J. Tölke, M. Krafczyk, M. Schulz, E. Rank, R. Berrios, Implicit discretization and nonuniform mesh refinement approaches for FD discretizations of LBGK models, Int. J. Mod. Phys. C 9 (1998) 1143–1157.
[8] T. Lee, C.-L. Lin, An Eulerian description of the streaming process in the lattice Boltzmann equation, J. Comput. Phys. 185 (2) (2003) 445–471, http://dx.doi.org/10.1016/S0021-9991(02)00065-7.
[9] J. Huang, C. Yang, X.-C. Cai, A fully implicit method for lattice Boltzmann equations, SIAM J. Sci. Comput. 37 (5) (2015) S291–S313, http://dx.doi.org/10.1137/140975346.
[10] W. Li, L.-S. Luo, Finite-volume lattice Boltzmann method for nearly incompressible flows on arbitrary unstructured meshes, Commun. Comput. Phys. 20 (2) (2016) 301–324, http://dx.doi.org/10.4208/cicp.211015.040316a.
[11] A. Jameson, S. Yoon, Lower–upper implicit schemes with multiple grids for the Euler equations, AIAA J. 25 (7) (1987) 929–935.
[12] P. Roe, Approximate Riemann solvers, parameter vectors, and difference schemes, J. Comput. Phys. 43 (2) (1981) 357–372.
[13] D. d'Humières, Generalized lattice-Boltzmann equations, in: B.D. Shizgal, D.P. Weave (Eds.), Rarefied Gas Dynamics: Theory and Simulations, in: Prog. Astronaut. Aeronaut., vol. 159, AIAA, Washington, D.C., 1992, pp. 450–458.
[14] P. Lallemand, L.-S. Luo, Theory of the lattice Boltzmann method: dispersion, dissipation, isotropy, galilean invariance, and stability, Phys. Rev. E 61 (6) (2000) 6546–6562.
[15] J. Blazek, Computational Fluid Dynamics: Principles and Applications, 2nd edition, Elsevier, New York, 2005.
[16] Z. Guo, C. Zheng, B. Shi, An extrapolation method for boundary conditions in lattice Boltzmann method, Phys. Fluids 14 (6) (2002) 2007–2010, http://dx.doi.org/10.1063/1.1471914.
[17] Z. Guo, C. Zheng, B. Shi, Non-equilibrium extrapolation method for velocity and pressure boundary conditions in the lattice Boltzmann method, Chin. Phys. 11 (4) (2002) 366–374.
[18] D. Sharov, K. Nakahashi, Reordering of 3-D hybrid unstructured grids for vectorized LU-SGS Navier–Stokes computations, AIAA-97-2102, in a collection of technical papers in: 13th AIAA Computational Fluid Dynamics Conference, June 29–July 2, 1997, Snowmass Village, CO, 2009.
[19] H. Luo, J. Baum, R. Löhner, A fast, matrix-free implicit method for computing low Mach number flows on unstructured grids, Int. J. Comput. Fluid Dyn. 14 (2) (2001) 133.
[20] W. Li, M. Kaneda, K. Suga, A new matrix-free, high efficient preconditioned LU-SGS method for low Mach axisymmetric flows on unstructured mesh, in: Proceedings of the 4th Asian Symposium on Computational Heat Transfer and Fluid Flow, June 3–6, 2013, Hong Kong, China, 2013.
[21] H. Schlichting, K. Gersten, Boundary Layer Theory, 8th edition, Springer, Berlin, 2000.
[22] S.C.R. Dennis, C.-Z. Chang, Numerical solutions for steady flow past a circular cylinder at Reynolds numbers up to 100, J. Fluid Mech. 42 (1970) 471–489, http://dx.doi.org/10.1017/S0022112070001428.
[23] B. Fornberg, A numerical study of steady viscous flow past a circular cylinder, J. Fluid Mech. 98 (4) (1980) 819–855, http://dx.doi.org/10.1017/S0022112080000419.
[24] X. He, G.D. Doolen, Lattice Boltzmann method on curvilinear coordinates systems: flow around a circular cylinder, J. Comput. Phys. 134 (1997) 306–315, http://dx.doi.org/10.1006/jcph.1997.5709.
[25] K. Hejranfar, E. Ezzatneshan, Implementation of a high-order compact finite-difference lattice Boltzmann method in generalized curvilinear coordinates, J. Comput. Phys. 267 (2014) 28–49.
[26] E. Turkel, Preconditioned methods for solving the incompressible and low-speed compressible equations, J. Comput. Phys. 72 (2) (1987) 277–298, http://dx.doi.org/10.1016/0021-9991(87)90084-2.
[27] K. Aoki, K. Kanba, S. Takata, Numerical analysis of a supersonic rarefied gas flow past a flat plate, Phys. Fluids 9 (4) (1997) 1144–1161, http://dx.doi.org/10.1063/1.869204.
[28] H. Babovsky, Discretization and numerical schemes for steady kinetic model equations, Comput. Math. Appl. 35 (1–2) (1998) 29–40, http://dx.doi.org/10.1016/S0898-1221(97)00256-3.
[29] L. Mieussens, Discrete-velocity models and numerical schemes for the Boltzmann-BGK equation in plane and axisymmetric geometries, J. Comput. Phys. 162 (2) (2000) 429–466, http://dx.doi.org/10.1006/jcph.2000.6548.
[30] H. Cabannes, R. Gatignol, L.-S. Luo, The Discrete Boltzmann Equation: Theory and Applications, Lecture Notes given by Cabannes at University of California, Berkeley, 1980 2003.
[31] B. Shizgal, A Gaussian quadrature procedure for use in the solution of the Boltzmann-equation and related problems, J. Comput. Phys. 41 (2) (1981) 309–328, http://dx.doi.org/10.1016/0021-9991(81)90099-1.
[32] J.-Y. Yang, J.C. Huang, L. Tsuei, Numerical solutions of the nonlinear model Boltzmann equations, Proc. R. Soc. Lond. Ser. A 448 (1932) (1995) 55–80, http://dx.doi.org/10.1098/rspa.1995.0003.