A FULLY IMPLICIT METHOD FOR LATTICE BOLTZMANN EQUATIONS *

JIZU HUANG†, CHAO YANG‡, AND XIAO-CHAUN CAI§¶

Abstract. Existing approaches for solving the lattice Boltzmann equations with finite difference method are explicit and semi-implicit, both have certain stability constraints on the time step size. In this work, a fully implicit second-order finite difference scheme is developed. We focus on a parallel, highly scalable, Newton–Krylov–RAS algorithm for the solution of a large sparse nonlinear system of equations arising at each time step. Here, RAS is a restricted additive Schwarz preconditioner based on a first-order spatial discretization. We show numerically that by using the fully implicit method the time step size is no longer constrained by the CFL condition, and the Newton–Krylov–RAS algorithm is scalable on a supercomputer with more than ten thousand processors. Moreover, to calculate the steady state solution we investigate an adaptive time stepping strategy. The total compute time required by the implicit method with adaptive time stepping is much smaller than that of an explicit method for several test cases.

Key words. Lattice Boltzmann equations, fully implicit method, Newton–Krylov–RAS, domain decomposition, parallel scalability.

AMS subject classifications. 65Y05, 65M55, 76M20, 35F20

1. Introduction. In recent years, simulating complex fluid flows based on the lattice Boltzmann equations (LBEs) becomes increasingly popular [5, 17, 42]. Unlike the Navier–Stokes equations, the unknown in the LBEs is the density distribution function f_α of particles instead of the macroscopic variables and the macroscopic quantities (velocity and density) are obtained by taking the summation of the density distribution function f_α. The most popular approach for solving the LBEs is the lattice Boltzmann method (LBM) [14, 16, 35, 44]. Comparing to traditional CFD methods, the parallel implementation of LBM is much easier since the communication is purely local. A typical LBM implementation comprises a collision and a streaming step. In the collision step, the particle distribution function is updated at equally distributed lattice points, and in the streaming step, the particle distribution function is shifted between lattice points. The consistency requirement between the mesh points and lattice points makes LBM not convenient to apply in the case of nonuniform mesh or complex geometry. The Courant–Friedrichs–Lewy (CFL) number for LBM has to be 1, because the particle distribution function can only be shifted between neighboring lattice points. Whether an explicit, semi-implicit, or implicit LBM is applied, the time step size is severely limited by the CFL number. To obtain a steady state solution, a large number of time steps is usually required, especially for high resolution simulations.

To overcome the limitations of LBM, several approaches have been introduced to

*This work was supported in part by NSFC 61170075, 61120106005, and 973 program 2011CB309701. The research was supported in part by NSF CCF-1216314, 863 program 2015AA01A302, NSFC 91330111, and Shenzhen grant KQCX20130628112914303.

†Institute of Computational Mathematics and Scientific/Engineering Computing, Academy of Mathematics and Systems Science, Chinese Academy of Sciences, Beijing 100190, China (huangjz@lsec.cc.ac.cn).

‡Institute of Software, Chinese Academy of Sciences, Beijing 100190, China, and State Key Laboratory of Computer Science, Beijing 100190, China (yangchao@iscas.ac.cn).

§Department of Computer Science, University of Colorado Boulder, Boulder, CO 80309, USA (cai@cs.colorado.edu).

¶Corresponding author: Xiao-Chuan Cai.

directly discretize the temporal and spatial derivatives using finite difference method [30], finite volume method [48], or finite element method [40]. In reference [36], a fourth-order central scheme for space and an explicit fourth-order Runge–Kutta scheme for time are introduced on a uniform mesh. Cao *et al.* [13] provides an extension of this method to the case of non-uniform mesh. In reference [30] a semi-implicit finite difference method in curvilinear coordinates is studied using body-fitted non-uniform meshes. In the approach, the collision term is treated implicitly while other terms are discretized explicitly. Similar semi-implicit methods are studied in references [2, 33, 46, 47]. In reference [25] a special distribution function is used to remove the implicitness in the numerical scheme. By using the new distribution function, an explicit scheme is obtained even though the collision term is treated implicitly.

In the finite difference lattice Boltzmann method, the velocity-lattice is uncoupled from the spatial mesh which allows us to choose the discrete velocity model and the space-time discretization independently. In this paper, we employ a second-order spatial discretization and a fully implicit second-order temporal discretization. The fully implicit method treats both the collision term and the streaming term implicitly, and might be much more expensive than an explicit and semi-implicit method due to the need of solving nonlinear systems at each time step. Comparing with explicit and semi-implicit methods, the fully implicit method is usually more stable with much larger time step size, which depends only on the accuracy requirement. It is worth pointing out that similar fully implicit methods have recently been successfully applied to several classes of important applications [8, 22, 28, 32, 37, 39, 51], but we haven't seen any similar publications for the LBEs with a fully implicit finite difference method.

In the fully implicit method, the most expensive step is to solve a large sparse nonlinear algebraic system at every time step. We present a parallel, highly scalable, Newton–Krylov–RAS algorithm for solving such a system. In the Newton–Krylov algorithm, the nonlinear system is solved by an inexact Newton method whose Jacobian problem is solved with a preconditioned Krylov subspace method. In our approach, the Jacobian matrix is calculated analytically, and an overlapping RAS is used as the preconditioner in which the subdomain problem is solved with a point-block LU or ILU factorization. The point-block size is equal to the number of distribution functions per grid point. To reduce the cost and improve the scalability of the RAS preconditioner, a first-order discretization is developed just for the preconditioning step and the preconditioner is re-computed only once per time step.

In the implicit approach the time step size is selected based on the desired solution accuracy and certain features of the problem such as the oscillational rate of the solution about time. For steady state calculations, we propose an adaptive time stepping algorithm which changes the time step size as the system evolves. Several test cases are carefully investigated to show the performance of the proposed algorithm. For the purpose of comparison, we also implement an explicit algorithm and compare it with the fully implicit approach. The total compute time of the fully implicit method is much smaller than that of the explicit method. Scalability results on a supercomputer with more than ten thousand processors are reported.

The remainder of the paper is organized as follows. In section 2, we present the LBEs and a fully implicit second-order discretization scheme. The initial and boundary conditions are carefully discussed in the same section. The details of the parallel domain decomposition solver are described in section 3, in which an adaptive

time stepping method is also introduced. Several numerical results are reported in section 4 and concluding remarks are given in section 5.

2. A fully implicit second-order discretization. In this paper, we consider the LBEs with the D2Q9 model [34] without the external force,

$$\frac{\partial f_\alpha}{\partial t}(\mathbf{x},t)+\mathbf{e}_\alpha\cdot\nabla f_\alpha(\mathbf{x},t)=\Theta_\alpha,\quad \alpha=0,\,1,\cdots,8,\quad \mathbf{x}\in\Omega,\quad t\in(0,T), \tag{2.1}$$

where f_α is the particle distribution function, $\mathbf{e}_\alpha=(e_{\alpha1},e_{\alpha2})$ is the discrete particle velocity, Θ_α is the collision operator, $\Omega\in R^2$ is the computational domain, and $(0,T)$ is the time interval. The macroscopic density ρ and the macroscopic velocity $\mathbf{u}=(u_1,\,u_2)$ of the fluid are respectively induced from the particle distribution function by

$$\rho=\sum_\alpha f_\alpha,\qquad \mathbf{u}=\frac{1}{\rho}\sum_\alpha f_\alpha\mathbf{e}_\alpha. \tag{2.2}$$

Based on the kinetic theory, the collision operator is a six-fold integral in the phase space. After applying the single time relaxation approximation proposed in reference [6], the collision term is simplified to a function of the relaxation time τ, the density distribution function f_α and the local equilibrium distribution function (EDF) $f_\alpha^{(eq)}$. As suggested in reference [34], a suitable EDF can be chosen in the following form

$$f_\alpha^{(eq)}=w_\alpha\rho\Big[1+\frac{1}{c_s^2}\mathbf{e}_\alpha\cdot\mathbf{u}+\frac{1}{2c_s^4}(\mathbf{e}_\alpha\cdot\mathbf{u})^2-\frac{1}{2c_s^2}|\mathbf{u}|^2\Big], \tag{2.3}$$

where $|\mathbf{u}|=(u_1^2+u_2^2)^{1/2}$, and the discrete velocities are given by $\mathbf{e}_0=(0,0)$, and $\mathbf{e}_\alpha=\lambda_\alpha(\cos\theta_\alpha,\sin\theta_\alpha)$, with $\lambda_\alpha=1$, $\theta_\alpha=(\alpha-1)\pi/2$ for $\alpha=1,\,2,\,3,\,4$ and $\lambda_\alpha=\sqrt{2}$, $\theta_\alpha=(\alpha-5)\pi/2+\pi/4$ for $\alpha=5,\,6,\,7,\,8$, and $c_s=1/\sqrt{3}$ is the speed of sound [34]. The weighting factors are defined as $w_0=4/9$, $w_\alpha=1/9$ for $\alpha=1,\,2,\,3,\,4$ and $w_\alpha=1/36$ for $\alpha=5,\,6,\,7,\,8$. Using the EDF, the collision operator is defined as

$$\Theta_\alpha=-\frac{1}{\tau}(f_\alpha(\mathbf{x},t)-f_\alpha^{(eq)}(\mathbf{x},t)).$$

In the incompressible limit of low Mach number, the approximate Navier–Stokes equations can be derived from (2.1) through a Chapman–Enskog expansion procedure [17]

$$\begin{cases}\dfrac{\partial\rho}{\partial t}+\nabla\cdot(\rho\mathbf{u})=0,\\[2ex] \dfrac{\partial(\rho\mathbf{u})}{\partial t}+\nabla\cdot(\rho\mathbf{u}\cdot\mathbf{u})=-\nabla p+\nabla\cdot[\rho\nu(\nabla\cdot\mathbf{u}+\mathbf{u}\cdot\nabla)],\end{cases} \tag{2.4}$$

where $p=c_s^2\rho$ is the pressure and ν is the shear viscosity defined by

$$\nu=c_s^2\tau. \tag{2.5}$$

2.1. Spatial discretization. We suppose Ω is the unit square covered by a uniform $N\times N$ mesh with mesh size $h=1/(N-1)$. As shown in Fig. 2.1, let (x_1^i,x_2^j), $i,j=0,1\ldots,N-1$, be the mesh points. To obtain a second-order scheme

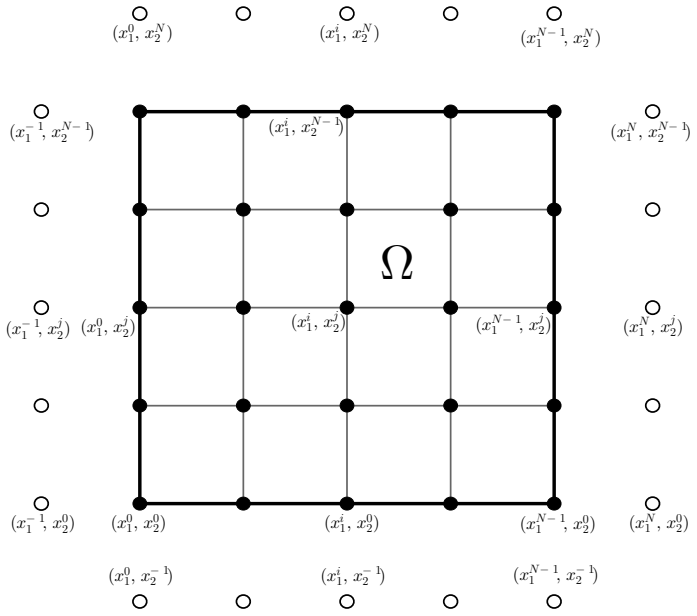

FIG. 2.1. *Computational domain with mesh points and ghost points. Here solid circles represent the mesh points and hollow circles represent the ghost points.*

in space, we introduce a layer of ghost points outside each boundaries of Ω denoted by (x_1^{-1}, x_2^j), (x_1^N, x_2^j), (x_1^i, x_2^{-1}), and (x_1^i, x_2^N), respectively. To discretize the spatial gradient operator ∇ in (2.1), the simplest method is a first-order upwinding scheme

$$\frac{\partial f_\alpha}{\partial x_k^i} = \begin{cases} \dfrac{1}{h}\big[f_\alpha(x_k^i,\cdot) - f_\alpha(x_k^{i-1},\cdot)\big] & \text{if} \quad e_{\alpha k} \geq 0,\ 1 \leq i \leq N-2, \\ -\dfrac{1}{h}\big[f_\alpha(x_k^i,\cdot) - f_\alpha(x_k^{i+1},\cdot)\big] & \text{if} \quad e_{\alpha k} < 0,\ 1 \leq i \leq N-2. \end{cases} \tag{2.6}$$

The scheme is easy to implement, however, our experiments show that the accuracy of the scheme is usually not acceptable. But, on the other hand, it can be used to construct an efficient preconditioner for a higher order scheme. To improve the accuracy, we study a family of fully implicit finite difference schemes originally proposed in reference [25, 30]. Let us define a scheme $\frac{\partial f_\alpha}{\partial x_k^i}\Big|_m$ in the family as

$$\frac{\partial f_\alpha}{\partial x_k^i}\Big|_m = \epsilon \frac{\partial f_\alpha}{\partial x_k^i}\Big|_u + (1-\epsilon)\frac{\partial f_\alpha}{\partial x_k^i}\Big|_c, \qquad k = 1, 2,\ i = 1, 2, \ldots, N-2, \tag{2.7}$$

where $\frac{\partial f_\alpha}{\partial x_k^i}\Big|_u$ is an upwinding scheme, $\frac{\partial f_\alpha}{\partial x_k^i}\Big|_c$ is a second-order central scheme, and $0 \leq \epsilon \leq 1$ is a control parameter to adjust the weights of the two components. Here $\frac{\partial f_\alpha}{\partial x_k^i}\big|_u$ and $\frac{\partial f_\alpha}{\partial x_k^i}\big|_c$ are respectively defined by

$$\frac{\partial f_\alpha}{\partial x_k^i}\Big|_u = \begin{cases} \dfrac{1}{2h}\big[3f_\alpha(x_k^i,\cdot) - 4f_\alpha(x_k^{i-1},\cdot) + f_\alpha(x_k^{i-2},\cdot)\big] & \text{if} \quad e_{\alpha k} \geq 0, \\ -\dfrac{1}{2h}\big[3f_\alpha(x_k^i,\cdot) - 4f_\alpha(x_k^{i+1},\cdot) + f_\alpha(x_k^{i+2},\cdot)\big] & \text{if} \quad e_{\alpha k} < 0, \end{cases}$$

and

$$\frac{\partial f_\alpha}{\partial x_k^i}\Big|_c = \frac{1}{2h}\big[f_\alpha(x_k^{i+1},\cdot) - f_\alpha(x_k^{i-1},\cdot)\big].$$

To make the schemes second-order accurate in space, we introduce a third-order approximation at the ghost points. For brevity, we only show the approximation approach for the ghost points below the bottom wall. For any given ghost point (x_1^i, x_2^{-1}), $i = 0, 1, ..., N-1$, a third-order approximation is given as

$$f_\alpha(x_1^i, x_2^{-1}) = 2f_\alpha(x_1^i, x_2^0) - 2f_\alpha(x_1^i, x_2^2) + f_\alpha(x_1^i, x_2^3).$$

2.2. Initial and boundary condtions. In traditional flow simulations, initial and boundary conditions are often available in terms of the macroscopic variables $\mathbf{u}$ and ρ. How to determine the initial and boundary values of the particle distribution function f_α in (2.1) is an important issue for Boltzmann based computations. In this paper, $f_\alpha(\mathbf{x}, 0)$ is set to be the EDF, i.e $f_\alpha(\mathbf{x}, 0) = f_\alpha^{(eq)}(\mathbf{x}, 0)$ by the equilibrium method.

Wall boundary conditions, originally taken from the lattice gas automata, are among the most broadly applied boundary conditions in Boltzmann models. In reference [29], a bounce-back scheme for the particle distribution function is used on the walls to obtain a no-slip velocity condition. The bounce-back scheme is easy to apply, but is only first-order accurate on the boundary [19]. To improve the numerical accuracy, other boundary treatments have been proposed. One is the extrapolation method [18] that views LBM as a special finite difference scheme. The method preserves the overall accuracy of LBM, and can be applied to a variety of boundary conditions. Based on it, Mei and Shyy [30] presented a method for their finite difference-based LBM. But the numerical stability of this extrapolation method is rather poor for high Reynolds number flows [52]. To overcome this difficulty, an alternative nonequilibrium extrapolation method (NEM) is introduced in reference [26].

In the NEM proposed in reference [26], the particle distribution function f_α at boundary points is decomposed into an equilibrium part $f_\alpha^{(eq)}$ and a nonequilibrium part $f_\alpha^{(neq)}$. The equilibrium part $f_\alpha^{(eq)}$ is calculated by (2.3) with the macroscopic boundary conditions, and the nonequilibrium part $f_\alpha^{(neq)}$ is approximately obtained by an extrapolation approximation. If we take a point (x_1^i, x_2^0) on the bottom wall as an example, the extrapolation approximation of the nonequilibrium part $f_\alpha^{(neq)}(x_1^i, x_2^0)$ is given by

$$f_\alpha^{(neq)}(x_1^i, x_2^0) \approx f_\alpha^{(neq)}(x_1^i, x_2^1) = f_\alpha(x_1^i, x_2^1) - f_\alpha^{(eq)}(x_1^i, x_2^1). \tag{2.8}$$

Since the extrapolation approximation given by (2.8) is only first-order accurate, we call it first-order NEM in the follows.

It shows later in the numerical tests that the accuracy of the first-order NEM is low, therefore we propose a second-order NEM here to improve the accuracy. In the second-order NEM, the equilibrium part $f_\alpha^{(eq)}$ is also calculated by (2.3), but a second-order extrapolation approximation is used to obtain the nonequilibrium part $f_\alpha^{(neq)}$. For the point (x_1^i, x_2^0) on the bottom wall, the second-order extrapolation approximation is defined as

$$f_\alpha^{(neq)}(x_1^i, x_2^0) \approx 2f_\alpha^{(neq)}(x_1^i, x_2^1) - f_\alpha^{(neq)}(x_1^i, x_2^2). \tag{2.9}$$

Some numerical results are given in Section 4 to compare the accuracy of the first-order NEM and the second-order NEM.

2.3. A fully implicit second-order temporal discretization. After spatially discretizing (2.1) with the second-order finite difference scheme, we obtain a semi-discrete system

$$\frac{\partial}{\partial t}\mathbf{X}(t) + \mathcal{G}(\mathbf{X}(t)) = 0. \tag{2.10}$$

Here $\mathcal{G}$ is a nonlinear function of $\mathbf{X}(t)$ which depends on the spatial discretization and the collision term. We organize the solution vector in a point-wise (field-coupling) order instead of a component-wise (field-splitting) order; i.e., the solution vector $\mathbf{X}(t)$ is defined as

$$\mathbf{X}(t) = (f_0^{00}(t), f_1^{00}(t), \cdots, f_8^{00}(t), f_0^{10}(t), f_1^{10}(t), \cdots, f_8^{10}(t), \cdots)^T.$$

Similarly, the point-wise order is used for $\mathcal{G}$. This order helps in improving not only the cache performance but also the parallel efficiency in load and communication balance; see [24] for example.

Assume $(0, T)$ is divided into time intervals $[t_n, t_{n+1}]$, where n is the time step index. At the $(n+1)^{th}$ time step, Let us define the time step size as $\Delta t_{n+1} = t_{n+1} - t_n$, and denote $\mathbf{X}^{n+1} \approx \mathbf{X}(t_{n+1})$ as the approximate solution of equations (2.1) at t_{n+1}. In order to obtain high numerical stability and accuracy in time, we employ a family of Explicit-first-step, Single-diagonal-coefficient Diagonally Implicit Runge–Kutta (ESDIRK) methods [7]

$$\begin{cases} \mathbf{X}^{(0)} = \mathbf{X}^n, \\ \dfrac{1}{\Delta t_{n+1}}\left(\mathbf{X}^{(p)} - \mathbf{X}^{(0)}\right) + \displaystyle\sum_{q=0}^{p} a_{pq}\mathcal{G}(\mathbf{X}^{(q)}) = 0, \quad p = 1, \ldots, s, \\ \mathbf{X}^{n+1} = X^{(s)}, \end{cases} \tag{2.11}$$

where $s \geq 1$ is the number of implicit stages. We denote an ESDIRK method with s implicit stages as ESDIRK(s). In this work, we focus on an ESDIRK(2) method with coefficients

$$a_{pp} = a_{10} = 1 - \sqrt{2}/2, \quad a_{20} = a_{21} = \sqrt{2}/4, \quad p = 1, 2,$$

which is both L-stable and second-order accurate [45].

For comparison purpose, we also implement an explicit second-order Strong Stability Preserving Runge–Kutta (SSP RK-2) method [4]

$$\begin{cases} \mathbf{X}^{(1)} = \mathbf{X}^n - \Delta t\mathcal{G}(\mathbf{X}^n), \\ \mathbf{X}^{n+1} = \dfrac{1}{2}\left(\mathbf{X}^n + \mathbf{X}^{(1)}\right) - \dfrac{\Delta t}{2}\mathcal{G}(\mathbf{X}^{(1)}), \end{cases} \tag{2.12}$$

in which we use a fixed time step size Δt determined by a given CFL number. Here the CFL number is calculated via

$$\mathrm{CFL} = \max_{\alpha \neq 0} \frac{(e_{\alpha 1}^2 + e_{\alpha 2}^2)^{1/2}\Delta t}{\min_{i,j}\left((x_1^i - x_1^{i+e_{\alpha 1}})^2 + (x_2^j - x_2^{j+e_{\alpha 2}})^2\right)^{1/2}}.$$

3. Parallel domain decomposition and adaptive time stepping. In the ESDIRK(2) method, two nonlinear algebraic systems are constructed and solved at each time step. In order to solve the nonlinear algebraic systems efficiently, we present a Newton–Krylov–Schwarz (NKS) [9, 10] type algorithm. In the algorithm, a Newton method is employed as the outer iteration, and a Krylov subspace method with Schwarz preconditioning is applied as inner iteration. Next let us discuss some details about the NKS method for a given nonlinear system $\mathcal{F}(\mathbf{X}) = 0$.

At the $(m+1)^{th}$ step of the Newton iteration, we obtain a new approximate solution $\mathbf{X}_{m+1}$ from the current approximation $\mathbf{X}_m$ through

$$\mathbf{X}_{m+1} = \mathbf{X}_m + \lambda_m \mathbf{S}_m, \qquad m = 0, 1, \cdots . \tag{3.1}$$

Here $\mathbf{X}_0$ is chosen as the solution at the previous time step, λ_m is the step-length determined by a line search procedure [20] and $\mathbf{S}_m$ is the search direction obtained by solving the following Jacobian system approximately using a Krylov subspace method with a preconditioner

$$J_m \mathbf{S}_m = -\mathcal{F}(\mathbf{X}_m), \tag{3.2}$$

where the Jacobian matrix $J_m = \mathcal{F}'(\mathbf{X}_m)$ is calculated at the current approximate solution $\mathbf{X}_m$. In the inexact Newton method, the linear system (3.2) does not need to be solved exactly. In our study, a restarted GMRES method [38] is applied to approximately solve the right-preconditioned system

$$J_m M_m^{-1} (M_m \mathbf{S}_m) = -\mathcal{F}(\mathbf{X}_m), \tag{3.3}$$

until the linear residual $\mathbf{r}_m = J_m \mathbf{S}_m + \mathcal{F}(\mathbf{X}_m)$ satisfies the stopping condition

$$\big\|\mathbf{r}_m\big\| \le \max\big\{\xi_r \|\mathcal{F}(\mathbf{X}_m)\|, \xi_a\big\},$$

where ξ_r, $\xi_a \ge 0$ are linear tolerances and M_m^{-1} is an additive Schwarz type preconditioner to be defined shortly. To achieve a more uniform distribution of residual errors of all time steps [49], the stoping condition for the Newton iteration (3.1) is adaptively determined by

$$\|\mathcal{F}(\mathbf{X}_{m+1})\| \le \min\Big\{\hat{\gamma}_a, \max\{\gamma_r \|\mathcal{F}(\mathbf{X}_0)\|, \gamma_{a,}\}\Big\}, \tag{3.4}$$

where the safeguard $\hat{\gamma}_a$ and the relative tolerance γ_r are both fixed for all time steps, and the absolute tolerance γ_a is chosen as $\gamma_{a,1} \in [0, \hat{\gamma}_a)$ at the first time step and then adaptively determined by

$$\gamma_a = \max_{i=1,2,\cdots,n-1} \big\{\|\mathcal{F}(\mathbf{X}^i)\|\big\},$$

at the n^{th} time step.

Newton methods can be implemented with or without the explicit formulation of the Jacobian matrix. In this study, we focus on the explicitly calculated Jacobian and compare different preconditioners. To define the restricted additive Schwarz preconditioner, we decompose the domain Ω into N_p non-overlapping subdomains Ω^i $(i = 1, \cdots, N_p)$, such that $\overline{\Omega} = \cup_{i=1}^{N_p} \overline{\Omega}^i$ and $\Omega^i \cap \Omega^{i'} = \emptyset$, $\forall i \ne i'$. Here N_p is the number of processors and also the number of subdomains. In order to obtain overlapping subdomains, we extend each subdomain Ω^i with δ layers to a larger subdomain $\Omega^{i,\delta} \subset \Omega$.

Let ϱ be the total number of mesh points in Ω and $\varrho^{i,\delta}$ the total number of mesh points in $\Omega^{i,\delta}$. Then the restriction operator $R_{i,\delta}$ is an $\varrho^{i,\delta} \times \varrho$ block matrix that is defined as follows: its 9×9 block element $(R_{i,\delta})_{p_1,p_2}$ is an identity block if the integer indices $1 \le p_1 \le \varrho^{i,\delta}$ and $1 \le p_2 \le \varrho$ correspond to a mesh point in $\Omega^{i,\delta}$, or a block of zeros otherwise. The $R_{i,\delta}$ serves as a restriction matrix because its multiplication by a block $\varrho \times 1$ vector results in a shorter $\varrho^{i,\delta}$ block vector by dropping the components corresponding to mesh points outside $\Omega^{i,\delta}$. Note that $R_{i,0}$ denotes the restriction to the nonoverlapping subdomain. For each of the overlapping subdomains, we define $B_i^m = R_{i,0} B_m (R_{i,\delta})^T$ as the restriction of a global matrix B_m to the overlapping subdomain $\Omega^{i,\delta}$. The one-level left restricted additive Schwarz (RAS) preconditioner is defined as [12]

$$M_m^{-1} = \sum_{i=1}^{N_p} (R_{i,0})^T (B_i^m)^{-1} R_{i,\delta}. \tag{3.5}$$

In this paper, the matrix-vector multiplication with $(B_i^m)^{-1}$ is calculated by a point-block LU or ILU factorization.

The global matrix B_m is computed at each Newton iteration by taking the derivative of a nonlinear function, which is constructed by a given spatial discretization scheme at $\mathbf{X}_m$ in the domain Ω. By using spatial discretization scheme (2.6) or (2.7) for the nonlinear function, we obtain a first-order discretization based preconditioner (FODBP) and a second-order discretization based preconditioner (SODBP), respectively. SODBP should be effective in terms of the number of GMRES iterations, but the compute time may be greater than that of FODBP due to the high bandwidth and a larger number of non-zeros in the sparse matrix. Some numerical results will be shown later to compare the performances of FODBP and SODBP. For the NKS algorithm with a standard RAS preconditioner, we need to update the RAS preconditioner at each Newton iteration. Since the cost of constructing the RAS preconditioner is high, we freeze the preconditioner for all Newton iterations within a time step.

For long-term simulation, especially when calculating the steady state solution, the evolution of the system usually admits various time scales and the calculation often lasts for a long time. Therefore, an adaptive time step control is necessary in the numerical simulation. The idea of the adaptive time stepping algorithm we use in the study is analogous to the switched evolution/relaxation method in reference [31] and [1]. When incorporating those schemes into the implicit method, divergence may occur if the time step size increases too fast. Some safeguard mechanisms are included in our method to avoid the situation. At the $(n+1)^{th}$ time step, $\|\mathcal{G}(\mathbf{X}^n)\|$ is the Euclidean norm of $\mathcal{G}(\mathbf{X}^n)$ which measures the departure from convergence. The numbers of nonlinear and linear iterations N_{its}^n, N_{lits}^n at the n^{th} time step can be viewed as a measure of the computational cost. We then use $\|\mathcal{G}(\mathbf{X}^n)\|$, $\|\mathcal{G}(\mathbf{X}^{n-1})\|$, N_{its}^n, and N_{lits}^n to determine the time step size at the $(n+1)^{th}$ time step. The algorithm goes as follows.

1. Start with a reasonably small time step size Δt_1 and choose the control parameters $\beta_1 = 1.5$, $\beta_2 = 0.75$. Set the maximum allowable number of nonlinear and linear iterations as N_{its}^{max} and N_{lits}^{max}, respectively.
2. At the $(n+1)^{th}$ time step, Δt_{n+1} is adjusted to

$$\Delta t_{n+1} = \min\Big\{\Delta t_{max}, \max\Big\{\frac{2}{3}, \min\Big\{\beta_1, \Big(\frac{\|\mathcal{G}(\mathbf{X}^{n-1})\|}{\|\mathcal{G}(\mathbf{X}^n)\|}\Big)^{\beta_2}\Big\}\Big\}\Delta t_n\Big\},$$

where β_1 and β_2 are defined as

$$\beta_1 = \begin{cases} 1, & \text{if} \quad N_{its}^n > N_{its}^{max} \quad \text{or} \quad N_{lits}^n > N_{lits}^{max}, \\ 1.5, & \text{otherwise}, \end{cases} \tag{3.6}$$

$$\beta_2 = \begin{cases} 0.75, & \text{if} \quad N_{its}^n > N_{its}^{max} \quad \text{or} \quad N_{lits}^n > N_{lits}^{max}, \\ 1.5, & \text{otherwise}. \end{cases} \tag{3.7}$$

Here Δt_{max} is a pre-determined safeguard time step size.

3. If the Newton iteration diverges with the current Δt_{n+1}, then a smaller time step size $0.5\Delta t_{n+1}$ is chosen to restart the Newton calculation. The maximum allowable number of nonlinear and linear iterations N_{its}^{max} and N_{lits}^{max} are adjusted to $N_{its}^{max} = 0.75 \times \min(N_{its}^{max}, N_{its}^n)$ and $N_{lits}^{max} = 0.75 \times \min(N_{lits}^{max}, N_{lits}^n)$, respectively.

The choice of Δt_1, N_{its}^{max}, N_{lits}^{max}, Δt_{max} is problem dependent and will be discussed in the next section. A similar but slightly simpler version of the algorithm is used in reference [50] for phase field problems. For the problems studied in this paper, the time step size should be relatively small at the beginning steps, otherwise divergence may occur in some later iterations.

REMARK 3.1. *We remark here that the class of Newton-Krylov-Schwarz algorithms has been applied successfully to solving Navier-Stokes equations [11], etc, but as far as we know this is the first time it is used for the LBEs. The subdomain solve we define in (3.5) implies that a zero Dirichlet boundary condition is applied to all 9 variables on the subdomain boundary that is in the interior of* Ω. *On the physical boundary, the original boundary is applied.*

REMARK 3.2. *For the steady state simulation, the stopping condition can be given as* $\|\frac{\mathbf{X}^n - \mathbf{X}^{n-1}}{\Delta t_n}\| \leq \varpi_a$ *or* $\|\mathcal{G}(\mathbf{X}^n)\| \leq \varpi_a$. *In the adaptive time stepping algorithm,* $\|\mathcal{G}(\mathbf{X}^n)\|$ *is already calculated, therefor we use* $\|\mathcal{G}(\mathbf{X}^n)\| \leq \varpi_a$ *to determine the convergence to the steady state.*

4. Numerical experiments. In this section, we investigate the numerical behavior and parallel performance of the newly proposed algorithm. We consider three test cases, namely a steady state Poiseuille flow, an unsteady Couette flow, and a steady state driven cavity flow. We focus on (1) the verification of the numerical accuracy of the fully implicit method, (2) a comparison of different preconditioners, (3) a comparison of the performance of an explicit method and the fully implicit method with adaptive time stepping, and (4) the parallel performance of the fully implicit method.

We implement the new algorithm described in the previous sections based on PETSc [3]. The numerical tests are carried out on the Tianhe-2 supercomputer, which tops the Top-500 list as of June, 2014. The computing nodes of Tianhe-2 are comprised of two 12-core Intel Ivy Bridge Xeon CPUs with 64GB local memory, and are interconnected via a proprietary high performance network. In the numerical experiments we use all 24 CPU cores in each node and assign one subdomain to each core. The stopping conditions for the nonlinear and linear iterations are as follows.

- The relative tolerance for nonlinear solver: $\gamma_r = 10^{-6}$;
- The absolute tolerance for nonlinear solver: $\gamma_{a,1} = 10^{-10}$;
- The safeguard tolerance for nonlinear solve: $\hat{\gamma} = 10^{-6}$;
- The relative tolerance for linear solver: $\xi_r = 10^{-3}$;
- The absolute tolerance for linear solver: $\xi_a = 10^{-11}$.

On each subdomain, we apply a sparse point-block LU or ILU factorization as the subdomain solver and set the overlapping size between subdomains to be $\delta = 0, h, 2h$, where h is the spatial mesh size. GMRES, restarts at every 30 iterations, is used to solve the Jacobian system at each Newton step. To test the accuracy of our scheme, we define the relative l_2 error as following

$$l_2 = \left(\frac{I\big(\sum_{k=0}^{8} (f_\alpha - \tilde{f}_\alpha)^2\big)}{I\big(\sum_{k=0}^{8} \tilde{f}_\alpha^2\big)} \right)^{\frac{1}{2}}, \qquad I(\square) = \sum_{i=0}^{N-1} \sum_{j=0}^{N-1} \square^{ij},$$

where f_α is the numerical solution, $\tilde{f}_\alpha$ is the analytical solution, and I is the discrete summation computed over all mesh points.

4.1. The Poiseuille flow. The Poiseuille flow is a channel flow driven by a constant force along the x_1 direction between two parallel plates. There is a steady state solution that can be expressed as a parabola centered around the axis of the channel

$$\tilde{u}_1(x_1, x_2) = 4U_0 \frac{x_2}{H}\Big(1 - \frac{x_2}{H}\Big), \quad \tilde{u}_2(x_1, x_2) = 0, \quad \text{for} \quad \forall x_1, 0 \le x_2 \le H, \tag{4.1}$$

where H is the channel height, $U_0 = FH^2/(8\rho_0\nu)$ is the peak velocity, F is the driven force, and ρ_0 is the initial density of the flow. Due to the existence of the external force, equations (2.1) should be changed to

$$\frac{\partial f_\alpha}{\partial t}(\mathbf{x}, t) + \mathbf{e}_\alpha \cdot \nabla f_\alpha(\mathbf{x}, t) = \Theta_\alpha + w_\alpha e_{\alpha 1} F/c_s^2, \quad \alpha = 0,\, 1, \cdots, 8. \tag{4.2}$$

The Reynolds number of the Poiseuille flow is a function of the peak velocity, the channel height H and the shear viscosity ν, $Re = HU_0/\nu$. Assuming Re is given, the driven force is set to be $F = 0.08\rho_0/(Re \times H)$ so that the peak velocity U_0 is equal to 0.1.

The density and velocity of the fluid are initialized to be $\rho_0 = 1.0$ and $(u_1, u_2) = (0, 0)$ for the whole domain. The equilibrium method is used to initialize the particle distribution function. The second-order NEM is applied to the top and bottom walls of the channel for no-slip boundary conditions, and periodic boundary conditions are applied to the inlet and exit of the channel. In the experiments, we set $Re =$ 10, 50, 100 and $L = H = 1.0$, where L is the length of the channel. A uniform mesh of size $N \times N$ is used. Since this is a steady state calculation, adaptive time stepping is used in the fully implicit algorithm.

Numerical results of both the explicit method ($\Delta t = 0.015$) and the fully implicit method with the central, upwinding, and mixed ($\epsilon = 0.1$, 0.5) schemes are shown in Fig. 4.1-(a), (b) together with the analytical solution, which verifies the consistency between the explicit and the fully implicit method. For the purpose of comparison, we also implement the first-order NEM to the top and bottom walls. The numerical results with the first-order NEM are presented in Fig. 4.1-(c), (d). In Fig. 4.1-(c) and (d), artificial wiggles appear in the central scheme, which shows that the central finite difference scheme with the first-order NEM is not stable for this test case. Table 4.1 shows the l_2 errors of u_1 for the implicit method and explicit method with different boundary conditions. From Table 4.1, we conclude that the second-order NEM is more accurate than the first-order NEM by orders of magnitude. Fig. 4.2 shows the history

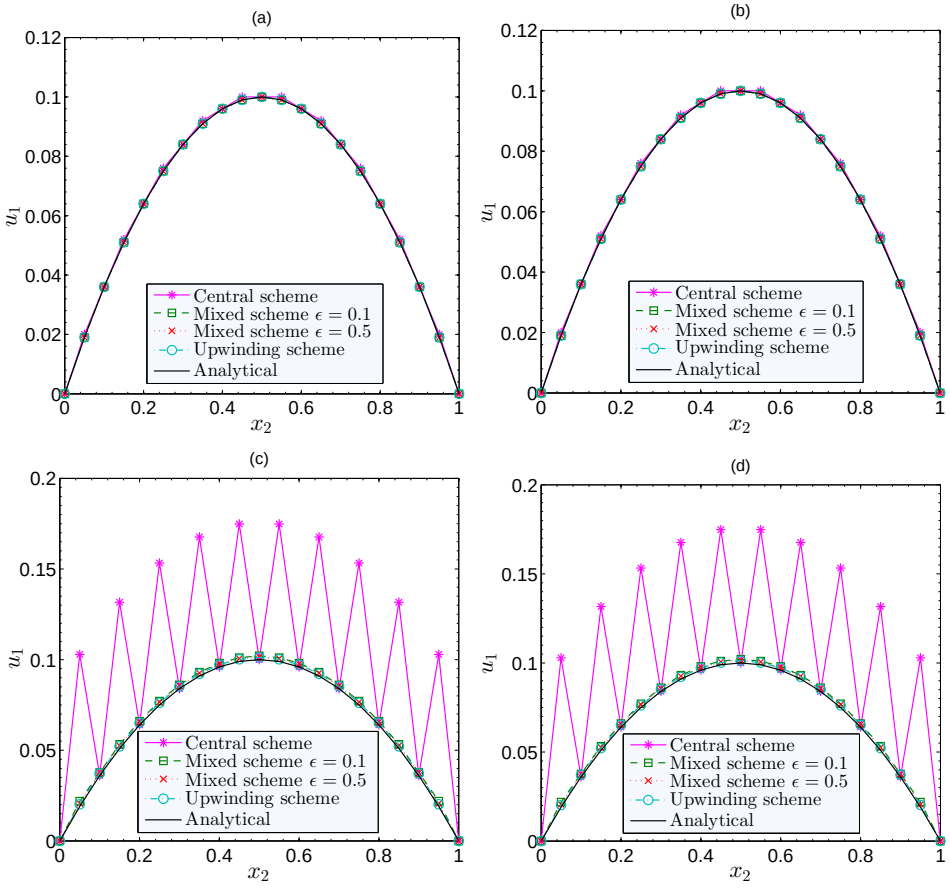

FIG. 4.1. *Velocity profiles of the Poiseuille flow, $Re = 10$, $N = 21$, $t = 100$, $\delta = h$. (a) The explicit method with the second-order NEM and $\Delta t = 0.015$. (b) The implicit method with the second-order NEM and adaptive time stepping. (c) The explicit method with the first-order NEM and $\Delta t = 0.015$. (d) The implicit method with the first-order NEM and adaptive time stepping.*

TABLE 4.1

The l_2 errors of u_1 for the implicit method and explicit method with different boundary conditions. The Poiseuille flow, $Re = 10$, $N = 21$, $t = 100$. The time step size for the explicit method is 0.015. In the implicit method, the adaptive time step algorithm is used to adaptively adjust time step size.

ϵ			0	0.1	0.5	1
l_2 error of u_1	Implicit	First-order NEM	0.766	0.0289	0.0210	0.0160
		Second-order NEM	0.0097	3.45e-5	3.14e-5	2.81e-5
	Explicit	First-order NEM	0.766	0.0288	0.0209	0.0160
		Second-order NEM	0.0096	5.44e-5	4.96e-5	4.47e-5

of the CFL numbers calculated from the adaptively selected time steps. Except the central finite difference scheme, the other three schemes increase the CFL smoothly. It is further observed that the adaptive time stepping algorithm works well even with a relatively large initial time step size, as shown in Fig. 4.2-(b). The performance of

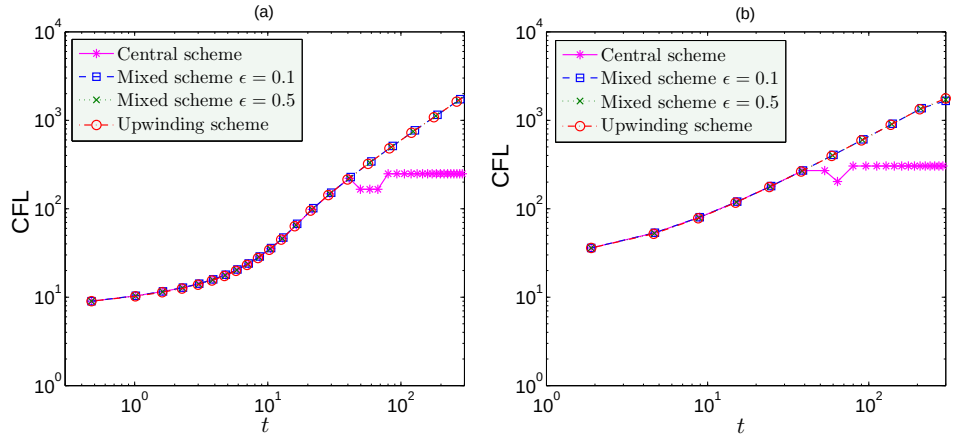

FIG. 4.2. *The Poiseuille flow. History of CFL number for the fully implicit method with adaptive time stepping,* $Re = 10$, $N = 21$, $N_{its}^{max} = 5$, $N_{lits}^{max} = 500$, *and* $\delta = 2h$: *(a)* $\Delta t_0 = 0.45$; *(b)* $\Delta t_1 = 1.8$.

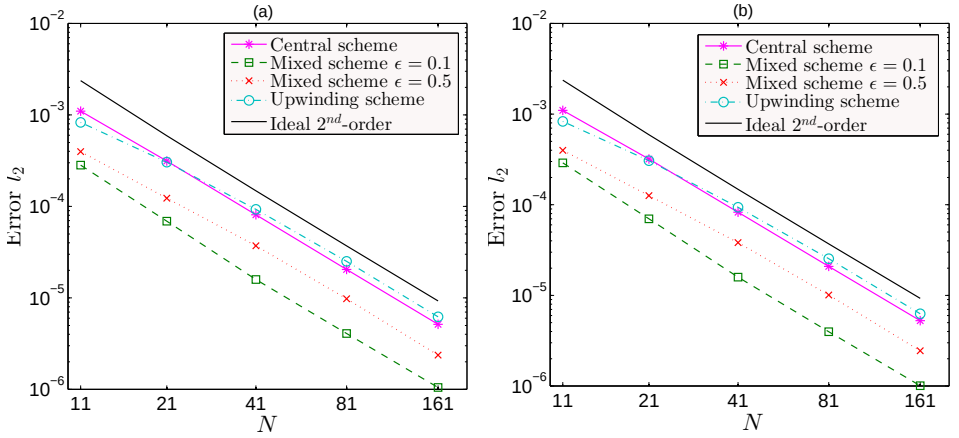

FIG. 4.3. l_2 *error of the Poiseuille flow,* $Re = 10$, $t = 4$, *(a) the explicit method with* $\Delta t = 0.1h$, *(b) the implicit method with adaptive time stepping,* $\Delta t_1 = 18h$.

the adaptive time stepping method is remarkable; the CFL number increases by over two orders of magnitude during the simulation.

To understand the accuracy of the spatial discretization scheme with the second-order NEM, we run the tests with $Re = 10$, 50 and 100 on increasingly refined meshes. The explicit method with a fixed time step size, and the fully implicit method with adaptive time stepping are considered. Since the particle distribution function f_α is unknown in the case, we take the numerical solution obtained by the explicit method with a very fine mesh 641×641 and small time step size $\Delta t = 1.56 \times 10^{-4}$ as the analytical solution $\tilde{f}_\alpha$. For $Re = 10$, the l_2 errors at $t = 4$ for the central, upwinding, and mixed ($\epsilon = 0.1$, 0.5) schemes are plotted in Fig. 4.3. The l_2 errors for $Re = 50$ and 100 are very similar to that of $Re = 10$. It is observed that the mixed ($\epsilon = 0.1$) scheme for both the explicit method and the fully implicit method produce smaller error than that of the other three schemes. Furthermore, all the four schemes show

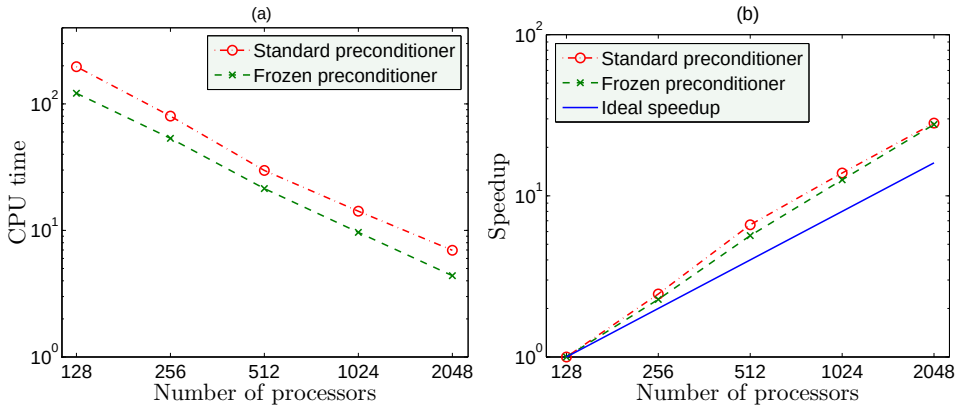

FIG. 4.4. *A comparison between the standard and the frozen preconditioner.* $1,024 \times 1,024$ *mesh (# of unknowns* $= 9,437,184$*), the time step size* $\Delta t = 0.02$*,* $Re = 10$*, 10 time steps,* $\delta = 1$*, and the point-block LU subdomain solver. Here, "standard preconditioner" denotes the standard FODBP; "frozen preconditioner" denotes the frozen FODBP.*

a second-order convergence rate in space. We then investigate the performance of the preconditioners introduced previously in the paper. We first look at FODBP and SODBP, and compare their performances in terms of the number of nonlinear and linear iterations, and the compute time. Table 4.2 shows that the change of preconditioners has no influence on the number of Newton iterations. It is important to note that the timing results obtained by using FODBP are much better than that with SODBP even though the number of GMRES iterations increases. We also observe that the fully implicit method with both preconditioners offer superlinear speedup with up to 2,048 processors. We believe that the reason for the superlinear speedup is due to the increasingly better cache performance in the sparse LU factorization as the subdomain problem becomes smaller [15, 21]. Since the RAS preconditioner does not change the accuracy of the Newton-Krylov solver, we further experiment by freezing the preconditioner at each time step. This technique saves some computational time by avoiding repeated factorizations of the subdomain matrix at each Newton step. Fig. 4.4 presents a comparison between the standard and the frozen preconditioner. The results indicate that the frozen technique can save nearly 50% of the compute time, while still maintain the superlinear speedup.

TABLE 4.2
A comparison of FODBP and SODBP. $1,024 \times 1,024$ *mesh (# of unknowns* $= 9,437,184$*), the time step size* $\Delta t = 0.02$*,* $Re = 10$*, 10 time steps,* $\delta = h$*, and the point-block LU subdomain solver.*

N_p	Newton/Time step		GMRES/Newton		Compute time (s)		Speedup	
	SODBP	FODBP	SODBP	FODBP	SODBP	FODBP	SODBP	FODBP
128	4.0	4.2	3.0	11.8	987.92	196.76	1	1
256	4.0	4.1	3.7	12.0	432.66	80.08	2.28	2.46
512	4.0	4.1	3.7	12.1	127.97	29.79	7.72	6.61
1024	4.0	4.0	3.7	11.8	61.65	14.20	16.03	13.86
2048	4.0	4.1	4.0	12.1	19.34	6.97	51.09	28.24

Now let we compare the fully implicit method with the explicit method. For the explicit method, we use a fixed time step size Δt determined by CFL$= 0.3$. For

the implicit method, we apply the adaptive time stepping algorithm, and RAS with FODBP. The mesh resolution increases from 16×16 to 256×256, and the number of processors increases from 2 to 512. From Table 4.3, it is clear that the total compute time of the fully implicit method is smaller than that of the explicit method.

TABLE 4.3
The Poiseuille flow. A comparison between the fully implicit method and the explicit method, mixed scheme with $\epsilon = 0.1$, $\delta = h$, $N_{its}^{max} = 5$, $N_{lits}^{max} = 1,000$, point-block LU subdomain solver, first-order discretization based preconditioner.

Mesh size N	16	32	64	128	256
Number of processors	2	8	32	128	512
Implicit time steps	8	9	10	16	28
Average CFL	281.3	516.7	945.0	1,190.6	1,366.1
Total Newton	59	54	55	76	142
Total GMRES	457	983	1,390	3,813	13,064
Total compute time	0.60	0.74	0.91	2.22	7.34
Explicit time steps	7,500	15,500	31,500	63,500	127,500
CFL	0.3	0.3	0.3	0.3	0.3
Total compute time	1.00	1.94	4.05	8.17	16.41

Table 4.3 also shows the weak scalability of the fully implicit method and the explicit method. Due to the stability restriction on the time step size, the total number of time steps as well as the total compute time for the explicit method doubles as the mesh is refined. For the fully implicit method, both the total number of implicit time steps and the total number of Newton iterations increase slowly as more processors are used, especially at low resolution; but the total compute time increases more rapidly due to the increased number of GMRES iterations. It is observed that neither the explicit method nor the implicit method reaches the ideal performance. We believe adding some coarse level corrections in the additive Schwarz preconditioner may improve the weak scaling of the fully implicit solver and plan to study this issue in the future.

4.2. The unsteady Couette flow. The unsteady Couette flow is a channel flow driven by the top plate which moves from left to right with a constant velocity $U_0 = 0.1$ along the x_1 direction. The Reynolds number is defined as $Re = HU_0/\nu$, where H is the channel height. If the initial velocity is zero, then the analytical solution is

$$\frac{\tilde{u}_1(x_2,t)}{U_0} = \frac{x_2}{H} + 2\sum_{m=1}^{\infty} \frac{(-1)^m}{\lambda_m H} \exp(-\nu\lambda_m^2 t)\sin(\lambda_m x_2), \quad \tilde{u}_2 = 0, \tag{4.3}$$

where $\lambda_m = m\pi/H$, $m = 1, 2, \cdots$. The density and velocity of the fluid are initialized to be $\rho_0 = 1.0$ and $(u_1, u_2) = (0, 0)$ and the equilibrium method is used to initialize the particle distribution function. The second-order NEM is applied on the top and bottom of the channel for no-slip boundary conditions, and a periodic boundary condition is applied to the inlet and exit. In the experiment, we set $Re = 10$, 50, 100 and $L = H = 1.0$, where L is the channel length. A uniform mesh of size $N \times N$ is used. A mixed scheme with $\epsilon = 0.1$ is considered and the fully implicit method with a fixed time step size $\Delta t = 0.25$ is applied. The time step size for the explicit method is fixed to be $\Delta t = 0.005$. Since the unsteady Couette flow slowly approaches the

steady-state solution $\tilde{u}_1(x_1, x_2) = U_0 x_2/H$, we also use the adaptive time stepping algorithm and compare the total compute time with the explicit method.

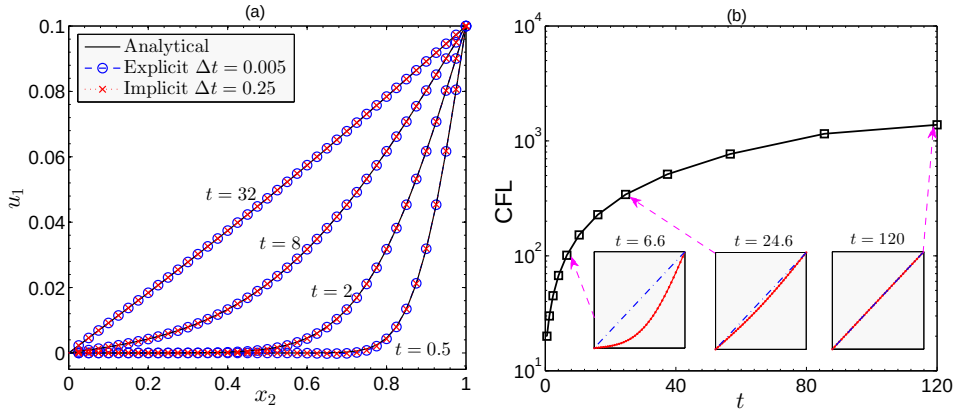

FIG. 4.5. *Results of the unsteady Couette flow on a* 41×41 *uniform mesh and* $Re = 10$. *(a) Velocity profiles at different times. (b) History of CFL number up to* $t = 120$ *with the adaptive time stepping algorithm,* $\Delta t_1 = 0.5$, $N_{its}^{max} = 5$, $N_{lits}^{max} = 500$, *and* $\delta = h$.

A series of velocity profiles at different time steps, together with the analytical solution, are shown in Fig. 4.5-(a). It is clear that the numerical solution of the explicit method and the fully implicit method both agree with the analytical solution. Numerical experiments with different Reynolds numbers are also carried out. All results match well with the analytical solution. The history of the CFL numbers up to $t = 120$ with the adaptive time stepping algorithm and some typical velocity profiles are presented in Fig. 4.5-(b). As seen in Fig. 4.5-(b), by using the adaptive time stepping algorithm, the CFL number increases from a relatively small number to about 200 times larger.

We then compare the fully implicit method and the explicit method in terms of the total compute time. The CFL number of the explicit method is fixed at 0.3. Details of the fully implicit method are shown in subsection 4.1. Table 4.4 shows a comparison of the computational costs of both method. It is observed that the total compute time of the fully implicit method is smaller than that of the explicit method for all tested meshes. Same to the Poiseuille flow, both methods have poor weak scalability.

For the purpose of comparison, we also compute the unsteady Couette flow by using the first-order NEM. Because of the rapid change of the velocity u_1 near the top wall, the first-order NEM on a uniform mesh can't depict a sufficiently accurate solution. To overcome this difficulty, the first-order NEM with a nonuniform mesh is used to solve the unsteady Couette flow [25]. It is worth pointing out that the numerical results with the second-order NEM is more accurate than that of the first-order NEM. So even on a uniform mesh, the second-order NEM also can produce a sufficiently accurate solution.

4.3. Lid-driven cavity flow. The third test case is a lid-driven cavity flow which has been used as a benchmark problem for many numerical methods due to its simple geometry and complicated flow behaviors. In the flow problem, the top plate moves from left to right along the x_1 direction with a constant velocity $U_0 = 0.1$, and the other three walls are fixed. The second-order NEM is applied as boundary

TABLE 4.4
The unsteady Couette flow. A comparison of the fully implicit method and the explicit method, mixed scheme with $\epsilon = 0.1$, $\delta = h$, $N_{its}^{max} = 5$, $N_{lits}^{max} = 1,000$, point-block LU subdomain solver, and first-order discretization based preconditioner.

Mesh size N	16	32	64	128	256
Number of processors	2	8	32	128	512
Implicit time steps	7	8	10	13	20
Average CFL	321.4	581.3	945	1,465.4	1,912.5
Total Newton	38	43	54	62	96
Total GMRES	321	904	1,826	5,093	19,705
Total compute time	0.43	0.71	1.14	2.74	10.34
Explicit time steps	7,500	15,500	31,500	63,500	127,500
Total compute time	0.97	1.95	4.09	8.13	16.35

conditions on all walls, and the equilibrium method is used to initialize the particle distribution function by setting $\rho = 1.0$ and $\mathbf{u} = (0, 0)$ in the cavity. The Reynolds number is defined as $Re = U_0 H/\nu$ with $H = 1.0$. In our experiments, Re is chosen to be 400, 1,000, 3,200, 5,000, 7,500, and 10,000.

Since the Navier–Stokes equations can be derived from the LBEs (2.1) through a Chapman–Enskog expansion in the incompressible limit of low Mach number, we compare the result of the LBEs solved by the fully implicit method with that of the Navier–Stokes equations given by Ghia *et al.* [23]. Vorticity contours for each case are plotted in Fig. 4.6. It is observed that the flow structures are visually identical to the benchmark results obtained in reference [23], which verifies the agreement between Navier–Stokes equations and the LBEs. The effect of the Reynolds number is clearly shown in these plots (more details in reference [23]).

To quantitatively compare the results, the two velocity components u_1 and u_2 along the vertical and horizontal lines through the cavity center are shown in Fig. 4.7, together with the benchmark solutions of Ghia *et al.* [23]. Agreements are obtained. We find that the profiles become nearly linear in the center of the cavity as Re increases. The computed velocities obtained by the fully implicit method are closer to the benchmark solutions than the velocities profile obtained by an explicit method given by [25], especially when the Reynolds number is high. The possible reason is that the explicit method stagnates at a pseudo-steady state and the fully implicit method with adaptive time stepping allows us to obtain the steady state solution.

For the lid-driven cavity flow, the vorticity contour and the streamline pattern are often used to describe the performance of the flow. It is worth pointing out that our result is the first report that shows the exact consistency between the Boltzmann model and the Navier–Stokes model. In reference [25], by solving the LBEs using an explicit finite difference method, streamline patterns for $Re = 400$, $1,000$, $3,200$ and $5,000$ are plotted. Vortex centers of streamline patterns for $Re = 3,200$ and 5,000 are different from that of the Navier–Stokes equations given by [23]. Reference vorticity contours and streamline patterns obtained by solving the LBEs with LBM can be found in reference [27]. It is clear that some discrepancies exist between two models.

4.3.1. Performance of the fully implicit method. In this subsection, we focus on the parallel performance of the method. Several issues are carefully discussed including the influence of different overlaps and subdomain solvers, the strong scalability study, and a comparison of the explicit method and the fully implicit method.

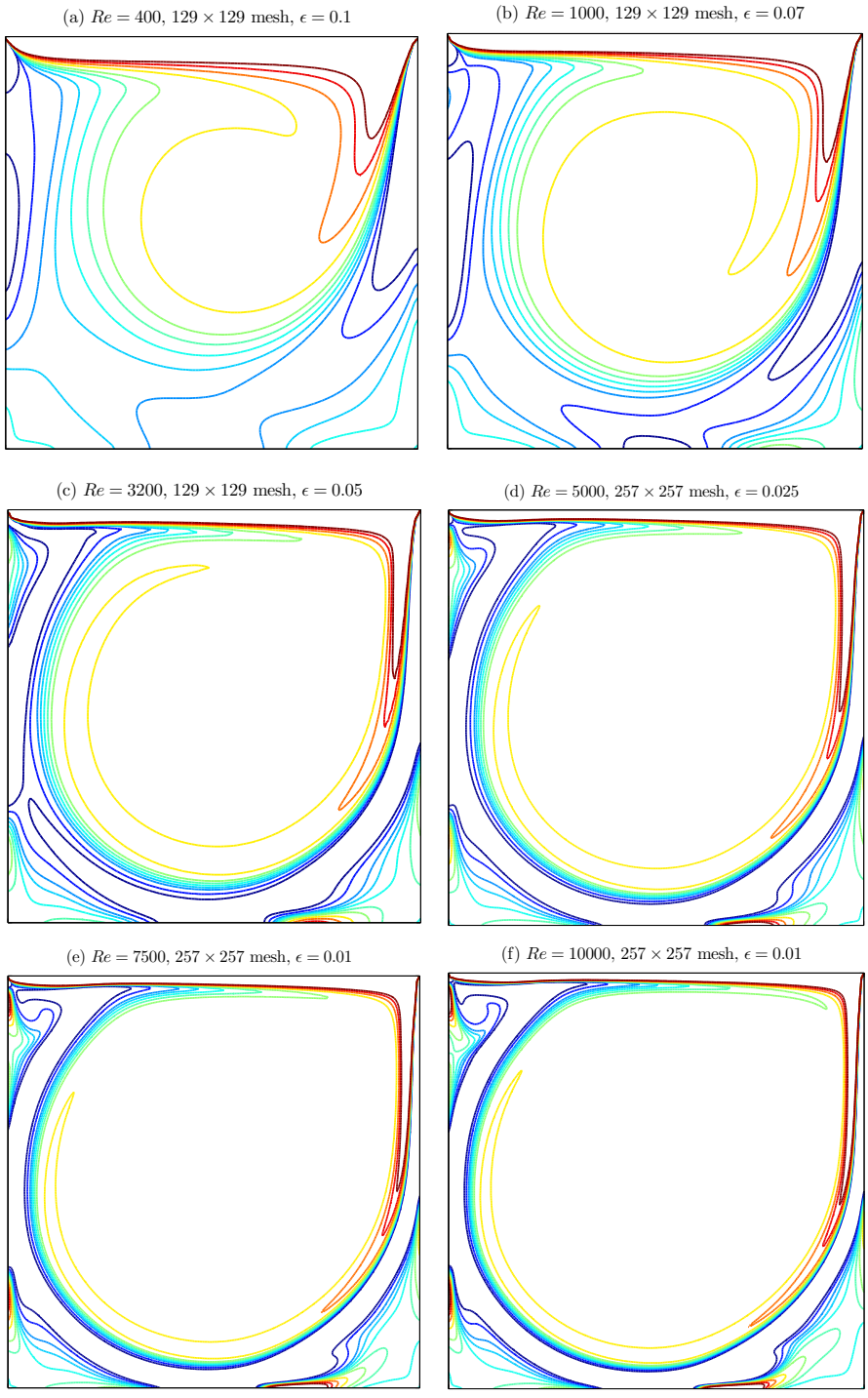

FIG. 4.6. *Vorticity contours for the driven cavity flow problem at different Reynolds numbers.*

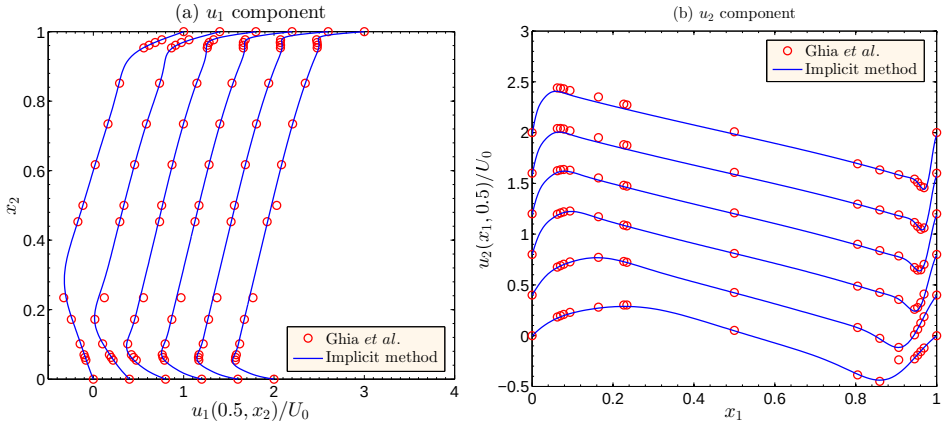

FIG. 4.7. *Velocity profiles of the cavity flow at different Reynolds numbers through the center of the computational domain. (a) u_1, from left to right: $Re = 400,\ 1,000,\ 2,000,\ 3,200,\ 5,000,\ 7,500,\ 10,000$; (b) u_2, from bottom to top: $Re = 400,\ 1,000,\ 2,000,\ 3,200,\ 5,000,\ 7,500,\ 10,000$. Note that the profiles are shifted for visual comparison.*

As is well known, the performance of the Schwarz preconditioner depends on the overlapping size and the subdomain solver. To investigate the influence, we focus on the lid-driven cavity flow with $Re = 1,000$ by running the tests with 256 processors on a $1,024 \times 1,024$ mesh with a fixed time step size $\Delta t = 0.098$ for the first ten time steps. The overlapping size is adjusted from 0 to $2h$ and different subdomain solvers including the point-block sparse LU factorization and the point-block sparse ILU factorization with different levels of fill-in are considered. We observe that the total number of Newton iterations is insensitive to the overlapping size and subdomain solver. The number of GMRES iterations and the compute time are shown in Table 4.5. It is clear that by increasing the overlapping size or by increasing the fill-in level, the number of GMRES iterations becomes smaller, but the compute time does not necessarily reduce because of the increased cost of communication or subdomain solver. In particular, when LU instead of ILU is used as the subdomain solver, GMRES converges with fewer iterations but the compute time is larger because LU is more expensive than ILU. In summary, we find that the optimal choice in terms of the total compute time is ILU(1) subdomain solver with $\delta = h$.

TABLE 4.5

Performance of NKS with different overlapping size and subdomain solvers. The lid-driven cavity flow with $Re = 1,000$, and $1,024 \times 1,024$ mesh with 256 processors. The time step size is fixed to $\Delta t = 0.098$ and the results are averaged over the first ten time steps.

δ	GMRES/Newton				Compute time (s)/Newton			
	LU	ILU(0)	ILU(1)	ILU(2)	LU	ILU(0)	ILU(1)	ILU(2)
0	20.1	27.3	23.8	22.8	1.578	0.637	0.620	0.644
h	19.3	25.7	21.8	20.8	1.610	0.623	0.596	0.622
$2h$	18.8	25.3	21.3	20.6	1.712	0.633	0.603	0.636

To study the parallel scalability, a $6,144 \times 6,144$ mesh is considered. We run test with $Re = 1,000$ and a fixed time step size $\Delta t = 0.0163$ for 10 time steps. In the experiment, we use both sparse ILU(1) and LU as the subdomain solver. The numbers of linear and nonlinear iterations and the compute time results are reported

in Tables 4.6. We can see that the number of nonlinear iterations per time step is almost independent of the number of processors and subdomain solvers. But the number of linear iterations grows slowly with the increase of the number of processors. A remarkable speedup is obtained from 1,024 to 16,384 processors. Figure 4.8 shows the total compute time and the speedup with respect to the number of processors. For the purpose of comparison, the performance results of the explicit SSP RK-2 method is also included. It is observed that both the explicit and the implicit methods with ILU(1) subdomain solver have nearly linear speedup, and the implicit method with LU subdomain solver has a better speedup performance. More importantly, the implicit method with ILU(1) subdomain solver outperforms the other methods by several times in terms of the total compute time.

REMARK 4.1. *For an elliptical problem with classical additive Schwarz preconditioner, the convergence theory indicates that the condition number of the preconditioned system satisfies* $\kappa \leq C(1 + H/\delta)/H^2$ *for the one-level method and* $\kappa \leq C(1 + H/\delta)$ *for the two-level method, where* H *is of order* $N_p^{-1/2}$ *for a 2D problem and* C *is a constant independent of* H, δ, *and the mesh size [43]. The condition number of the one-level method indicates an increase in the number of iterations with the increase of the number of processors. The two-level method is often preferred in order to remove the dependency of the number of iterations on the number of processors and also to obtain an ideal strong scalability. However, the optimal convergence theory of RAS is still not available, even for elliptic equations. The Jacobian systems in this paper are not elliptic and the above mentioned condition number estimates does not apply. However, our numerical results of the one-level RAS method suggest that the condition number increases more slowly than that in the elliptic case. Similar observation was made for another time dependent problem [49]. The time step size plays a role, but we don't theoretically understand how it impacts the convergence rate.*

TABLE 4.6

The lid-driven cavity flow problem. Test results using different subdomain solvers and number of processors, $6,144 \times 6,144$ *mesh (# of unknowns* $= 339,738,624$*),* $t_0 = 0$*, time step size* $\Delta t = 0.0163$*,* $CFL = 100$*,* $Re = 1,000$*, and 10 time steps.*

N_p	Newton(avg.)		GMRES/Newton		Compute time		Speedup	
	LU	ILU(1)	LU	ILU(1)	LU	ILU(1)	LU	ILU(1)
1024	6	6	15.7	16.7	1,231.9	318.9	1	1
2048	6	6	15.8	16.8	496.0	147.1	2.48	2.17
4096	6	6	16.3	17.2	265.4	89.4	4.64	3.57
8192	6	6	16.6	17.6	117.5	51.8	10.49	6.16
16384	6	6	17.0	17.9	62.6	29.0	19.67	11.01

The CFL histories for $Re = 5,000,\ 7,500,\ 10,000$ with the adaptive time stepping algorithm are given in Fig. 4.9 together with several plots of the vorticity contour at different times. It is clear that adaptive time stepping is able to select a large time step size while the solution changes slowly. This reduces tremendously the overall computational cost because large CFL numbers are used for almost 90% of the time, which reveals the advantage of the fully implicit method with adaptive time stepping for the steady state calculation. Table 4.7 gives a comparison of the compute time using the fully implicit method with adaptive time stepping and the explicit method. The fully implicit method is a clear winner, especially for high Reynolds number.

REMARK 4.2. *As shown in Table 4.7, the control parameter* ϵ *is adjusted according to the Reynolds number. The average CFLs for the implicit method with different*

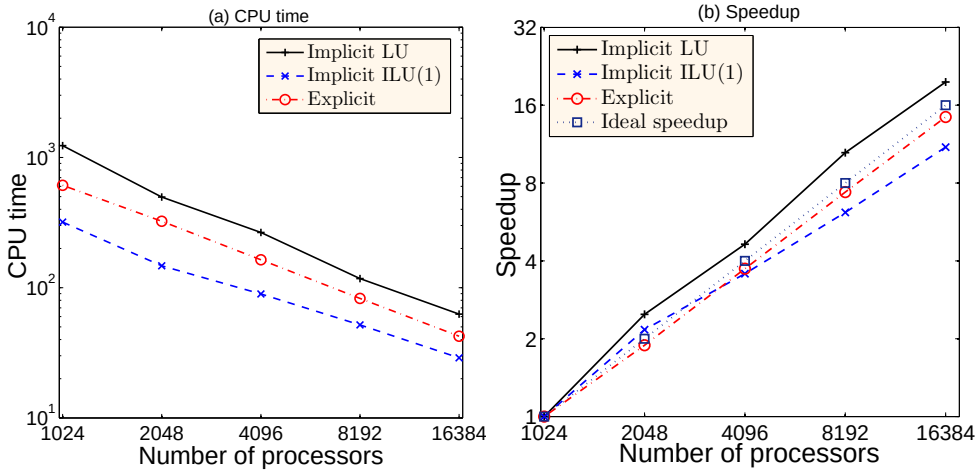

FIG. 4.8. *Strong scaling results for solving the driven cavity flow problem as in Table 4.6. Implicit method with different subdomain solvers and a fixed time step size* $\Delta t = 0.0163$*; the explicit method with a fixed time step size* $\Delta t = 9.8 \times 10^{-6}$.

TABLE 4.7

A comparison between the fully implicit method and the explicit method for the 2D lid-driven cavity flow, mixed scheme, $\delta = h$, $N_{its}^{max} = 5$, $N_{lits}^{max} = 1,000$, *point-block LU subdomain solver, and first-order discretization based preconditioner.*

Reynolds number	400	1,000	3,200	5,000	7,500	10,000
Mesh size N	129	129	129	257	257	257
Number of processors	128	128	128	512	512	512
ϵ	0.1	0.07	0.05	0.025	0.01	0.01
Implicit time steps	72	175	548	1,632	2,979	5,948
Average CFL	889	731	467	627	859	861
Newton	654	1,633	5,066	14,307	22,647	39,351
GMRES/Newton	113.97	87.50	71.98	139.73	149.72	131.6
Total compute time	46.5	91.6	238.0	769.4	1,649.6	3,964.8
Explicit time steps	4.0e+5	1.8e+6	1.3e+7	3.4e+7	1.3e+8	5.1e+8
CFL	0.16	0.07	0.02	0.03	0.02	0.01
Total compute time	52.4	245.2	1,702.5	4,437.5	16,717.5	67,074.7

Reynolds numbers are greater than 400*, which shows that the fully implicit method is unconditional stable and is quite robust with respect to the Reynolds number. However, the stability of the explicit method is strictly dependent on the control parameter* ϵ*. For this reason, the optimal CFL for the explicit method with different Reynolds number has to be picked by hand. For this test case, the CFL number for the explicit method is experimentally selected to be the largest allowable value for which the instability doesn't occur.*

5. Concluding remarks. In this paper, a fully implicit second-order finite difference method and a highly parallel solution algorithm were proposed and studied for solving the LBEs in 2D. The Newton–Krylov–RAS method converges well for several tests with various level of difficulties. For the steady state calculation, an adaptive

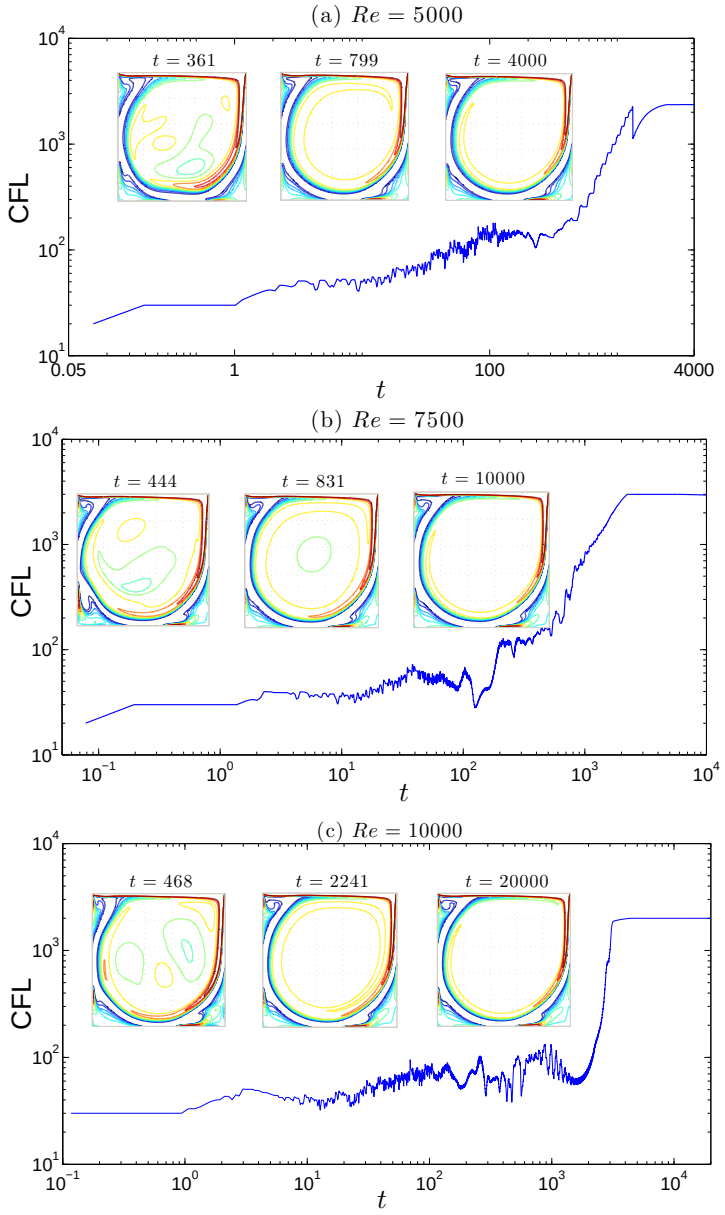

FIG. 4.9. *The histories of CFL number for the lid-driven cavity flow problem at different Reynolds numbers.*

time stepping strategy is successfully incorporated into the implicit time integration scheme so that large CFL numbers are allowed when the flow is close to steady state. Overlapping domain decomposition methods with both FODBP and SODBP were proposed, and we found that FODBP has a better performance in terms of the total compute time while the second-order discretization scheme (2.7) is used for the residual computation. A comparison between an explicit method with a fixed time step size and the fully implicit method with both fixed time step size and adaptive

time step sizes was presented. Accuracies of the explicit method and the fully implicit method are close to each other, but the fully implicit method with adaptive time stepping has lower computational cost in all steady state calculations. The fully implicit method exhibits a superlinear speedup for tests with up to 0.34 billion unknowns on a supercomputer with up to 16,384 processors. Excellent agreement between computed solutions of the LBEs and the Navier–Stokes equations for the 2D lid-driven cavity flow problem with high Reynolds numbers was achieved. As far as we know, such results were not available in any previously published reports.

Acknowledgments. The authors would like to thank the referees for many constructive suggestions leading to the improvement of the paper.

REFERENCES

[1] W. K. Anderson, W. D. Gropp, D. K. Kaushik, D. E. Keyes, and B. F. Smith, *Achieving high sustained performance in an unstructured mesh CFD application*, in Proceedings of the 1999 ACM/IEEE conference on Supercomputing, ACM, 1999, pp. 69: 1–13.

[2] P. Asinari, *Semi-implicit-linearized multiple-relaxation-time formulation of lattice Boltzmann schemes for mixture modeling*, Phys. Rev. E, 73 (2006), pp. 056705: 1–24.

[3] S. Balay, J. Brown, K. Buschelman, V. Eijkhout, W. Gropp, D. Kaushik, M. Knepley, L. C. McInnes, B. Smith, and H. Zhang, *PETSc Users Manual, Argonne National Laboratory*, 2013.

[4] G. D. Byrne and A. C. Hindmarsh, A polyalgorithm for the numerical solution of ordinary differential equations, ACM T. Math. Software, 1 (1975), pp. 71-96.

[5] R. Benzi, S. Succi, and M. Vergassola, *The lattice Boltzmann equation: theory and applications*, Phys. Rep., 222 (1992), pp. 145–197.

[6] P. L. Bhatnagar, E. P. Gross, and M. Krook, *A model for collision processes in gases. I. Small amplitude processes in charged and neutral one-component systems*, Phys. Rev., 94 (1954), pp. 511–525.

[7] H. Bijl, M. H. Carpenter, V. N. Vatsa, and C. A. Kennedy, *Implicit time integration schemes for the unsteady compressible Navier–Stokes equations: Laminar flow*, J. Comput. Phys., 179 (2002), pp. 313–329.

[8] P. N. Brown, D. E. Shumaker, and C. S. Woodward, *Fully implicit solution of large-scale non-equilibrium radiation diffusion with high order time integration*, J. Comput. Phys., 204 (2005), pp. 760–783.

[9] X.-C. Cai, W. D. Gropp, D. E. Keyes, R. G. Melvin, and D. P. Young, *Parallel Newton–Krylov–Schwarz algorithms for the transonic full potential equation*, SIAM J. Sci. Comput., 19 (1998), pp. 246–265.

[10] X.-C. Cai, W. D. Gropp, D. E. Keyes, and M. D. Tidriri, *Newton-Krylov-Schwarz methods in CFD*, Notes on Numer. Fluid Mech., 47 (1994), pp. 17–17.

[11] X.-C. Cai and X. F. Li, *Inexact Newton methods with restricted additive Schwarz based nonlinear elimination for problems with high local nonlinearity*, SIAM J. Sci. Comput., 33 (2011), pp. 746–762.

[12] X.-C. Cai and M. Sarkis, *A restricted additive Schwarz preconditioner for general sparse linear systems*, SIAM J. Sci. Comput., 21 (1999), pp. 792–797.

[13] N. Z. Cao, S. Y. Chen, S. Jin, and D. Martinez, *Physical symmetry and lattice symmetry in the lattice Boltzmann method*, Phys. Rev. E, 55 (1997), pp. R21–R24.

[14] H. D. Chen, S. Kandasamy, S. Orszag, R. Shock, S. Succi, and V. Yakhot, *Extended Boltzmann kinetic equation for turbulent flows*, Science, 301 (2003), pp. 633–636.

[15] R. Chen, Y. Wu, Z. Yan, Y. Zhao, and X.-C. Cai, *A parallel domain decomposition method for 3D unsteady incompressible flows at high Reynolds number*, J. Sci. Comput., 58 (2014), pp. 275–289.

[16] S. Chen, *A large-eddy-based lattice Boltzmann model for turbulent flow simulation*, Appl. Math. Comput., 215 (2009), pp. 591–598.

[17] S. Y. Chen and G. D. Doolen, *Lattice Boltzmann method for fluid flows*, Annu. Rev. Fluid Mech., 30 (1998), pp. 329–364.

[18] S. Y. Chen, D. Martinez, and R. W. Mei, *On boundary conditions in lattice Boltzmann methods*, Phys. Fluids, 8 (1996), pp. 2527–2536.

[19] R. Cornubert, D. d'Humières, and D. Levermore, *A Knudsen layer theory for lattice gases*, Physica D: Nonlinear Phenomena, 47 (1991), pp. 241–259.
[20] J. E. Dennis and R. B. Schnabel, *Numerical Methods for Unconstrained Optimization and Nonlinear Equations*, SIAM, 1983.
[21] L. C. Dutto, C. Y. Lepage, W. G. Habashi, *Effect of the storage format of sparse linear systems on parallel CFD computations*, Comput. Methods Appl. Mech. Engrg., 188 (2000), pp. 441-453.
[22] K. J. Evans, D. A. Knoll, and M. Pernice, *Enhanced algorithm efficiency for phase change convection using a multigrid preconditioner with a SIMPLE smoother*, J. Comput. Phys., 223 (2007), pp. 121–126.
[23] U. Ghia, K. N. Ghia, and C.T. Shin, *High-Re solutions for incompressible flow using the Navier-Stokes equations and a multigrid method*, J. Comput. Phys., 48 (1982), pp. 387–411.
[24] W. D. Gropp, D. K. Kaushik, D. E. Keyes, and B. F. Smith, *High-performance parallel implicit CFD*, Parallel Comput., 27 (2001), pp. 337–362.
[25] Z. L. Guo and T. S. Zhao, *Explicit finite-difference lattice Boltzmann method for curvilinear coordinates*, Phys. Rev. E, 67 (2003), pp. 066709: 1–12.
[26] Z. L. Guo, C. G. Zheng, and B. C. Shi, *Non-equilibrium extrapolation method for velocity and pressure boundary conditions in the lattice Boltzmann method*, Chinese Phys., 11 (2002), pp. 366–374.
[27] S. L. Hou, Q. S. Zou, S. Y. Chen, G. Doolen, and A. C. Cogley, *Simulation of cavity flow by the lattice Boltzmann method*, J. Comput. Phys., 118 (1995), pp. 329–347.
[28] D. A. Knoll, L. Chacon, L. G. Margolin, and V. A. Mousseau, *On balanced approximations for time integration of multiple time scale systems*, J. Comput. Phys., 185 (2003), pp. 583–611.
[29] P. Lavallee, J. P. Boon, and A. Noullez, *Boundaries in lattice gas flows*, Physica D: Nonlinear Phenomena, 47 (1991), pp. 233–240.
[30] R. W. Mei and W. Shyy, *On the finite difference-based lattice Boltzmann method in curvilinear coordinates*, J. Comput. Phys., 143 (1998), pp. 426–448.
[31] W. A. Mulder and B. V. Leer, *Experiments with implicit upwind methods for the Euler equations*, J. Comput. Phys., 59 (1985), pp. 232–246.
[32] S. Ovtchinnikov, F. Dobrian, X.-C. Cai, and D. E. Keyes, *Additive Schwarz-based fully coupled implicit methods for resistive Hall magnetohydrodynamic problems*, J. Comput. Phys., 225 (2007), pp. 1919–1936.
[33] S. Pieraccini and G. Puppo, *Implicit–explicit schemes for BGK kinetic equations*, J. Sci. Comput., 32 (2007), pp. 1–28.
[34] Y. H. Qian, D. d'Humières, and P. Lallemand, *Lattice BGK models for Navier-Stokes equation*, EPL (Europhysics Lett.), 17 (1992), pp. 479–484.
[35] Y. H. Qian, S. Succi, and S. A. Orszag, *Recent advances in lattice Boltzmann computing*, Annu. Rev. Comput. Phys., 3 (1995), pp. 195–242.
[36] M. B. Reider and J. D. Sterling, *Accuracy of discrete-velocity BGK models for the simulation of the incompressible Navier-Stokes equations*, Comput. Fluids, 24 (1995), pp. 459–467.
[37] D. R. Reynolds, R. Samtaney, and C. S. Woodward, *A fully implicit numerical method for single-fluid resistive magnetohydrodynamics*, J. Comput. Phys., 219 (2006), pp. 144–162.
[38] Y. Saad and M. H. Schultz, *GMRES: A generalized minimal residual algorithm for solving nonsymmetric linear systems*, SIAM J. Sci. Stat. Comput., 7 (1986), pp. 856–869.
[39] J. N. Shadid, R. S. Tuminaro, K. D. Devine, G. L. Hennigan, and P. T. Lin, *Performance of fully coupled domain decomposition preconditioners for finite element transport/reaction simulations*, J. Comput. Phys., 205 (2005), pp. 24–47.
[40] C. Shu, Y. T. Chew, and X. D. Niu, *Least-squares-based lattice Boltzmann method: a meshless approach for simulation of flows with complex geometry*, Phys. Rev. E, 64 (2001), pp. 045701: 1–4.
[41] P. A. Skordos, *Initial and boundary conditions for the lattice Boltzmann method*, Phys. Rev. E, 48 (1993), pp. 4823–4842.
[42] S. Succi, *The Lattice Boltzmann Equation: for Fluid Dynamics and Beyond*, Oxford University Press, 2001.
[43] A. Toselli and O. B. Widlund, *Domain Decomposition Methods: Algorithms and Theory*, vol. 34, Springer, 2005.
[44] E. Vergnault, O. Malaspinas, and P. Sagaut, *A lattice Boltzmann method for nonlinear disturbances around an arbitrary base flow*, J. Comput. Phys., (2012), pp. 8070–8082.
[45] C. Völcker, J. B. Jørgensen, P. G. Thomsen, and E. H. Stenby, *Adaptive stepsize control*

in implicit Runge-Kutta methods for reservoir simulation, in Proceedings of the 9th Intl. Symp. on Dynamics and Control of Process Systems (DYCOPS 2010), 2010, pp. 509–514.

[46] Y. Wang, Y. L. He, J. Huang, and Q. Li, *Implicit–explicit finite-difference lattice Boltzmann method with viscid compressible model for gas oscillating patterns in a resonator*, Int. J. Numer. Methods Fluids, 59 (2009), pp. 853–872.

[47] Y. Wang, Y. L. He, T. S. Zhao, G. H. Tang, and W. Q. Tao, *Implicit-explicit finite-difference lattice Boltzmann method for compressible flows*, Int. J. Mod. Phys. C, 18 (2007), pp. 1961–1983.

[48] H. W. Xi, G. W. Peng, and S. H. Chou, *Finite-volume lattice Boltzmann method*, Phys. Rev. E, 59 (1999), pp. 6202–6205.

[49] C. Yang and X.-C. Cai, *Parallel multilevel methods for implicit solution of shallow water equations with nonsmooth topography on the cubed-sphere*, J. Comput. Phys., 230 (2011), pp. 2523–2539.

[50] C. Yang, X.-C. Cai, D. E. Keyes, and M. Pernice, *Parallel domain decomposition methods for the 3D Cahn-Hilliard equation*, in Proceedings of the 2011 annual meeting of DOEs Scientific Discovery through Advanced Computing, 2011.

[51] C. Yang, J. W. Cao, and X.-C. Cai, *A fully implicit domain decomposition algorithm for shallow water equations on the cubed-sphere*, SIAM J. Sci. Comput., 32 (2010), pp. 418–438.

[52] Q. S. Zou and X. Y. He, *On pressure and velocity boundary conditions for the lattice Boltzmann BGK model*, Phys. Fluids, 9 (1997), pp. 1591–1598.