A NONLINEARLY PRECONDITIONED INEXACT NEWTON ALGORITHM FOR STEADY STATE LATTICE BOLTZMANN EQUATIONS

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

Abstract. Most existing methods for calculating the steady state solution of the lattice Boltzmann equations are based on pseudo time stepping, which often requires a large number of time steps especially for high Reynolds number problems. To calculate the steady state solution directly without the time integration, in this paper we propose and study a nonlinearly preconditioned inexact Newton algorithm with a domain decomposition based linear solver for parallelization. More precisely, the proposed algorithmic framework involves an implicit, second-order discretization, a two-level inexact Newton method, and a nonlinear elimination preconditioner to accelerate the convergence of Newton iteration. A nonstandard, pollution removing, coarse space is introduced for the two-level method. Numerical experiments are presented to demonstrate the robustness and efficiency of the algorithm, especially for problems at a high Reynolds number. A comparison is also included to show the superiority of the proposed approach over other explicit and implicit methods in terms of the total compute time measured on a parallel computer.

Key words. steady state lattice Boltzmann equations, inexact Newton algorithm, nonlinear preconditioning, pollution removing coarse space, parallel scalability

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

DOI. 10.1137/15M1028078

1. Introduction. In the past decades, the lattice Boltzmann equations (LBEs) have been applied with significant success for the simulation of a variety of complex flows, ranging from steady flows to fully developed turbulent flows [19, 29, 42]. The LBEs are based on the kinetic theory in which the particle interactions are studied based on the particle density distribution function. The macroscopic variables and the macroscopic quantities (velocity and density) are obtained by taking the summation of the particle density distribution functions. The approximate Navier–Stokes equations can be derived from the LBEs through a Chapman–Enskog expansion procedure in the incompressible limit of a low Mach number [8]. The conventional LBEs are nonlinear hyperbolic equations consisting of a transient term, a collision term, and a streaming term. To solve the time dependent LBEs, lattice Boltzmann methods (LBM) [2, 20, 29], finite difference methods [21, 30, 35], finite volume methods [43], and finite element methods [38] have been developed. Among them, LBM, which is the most popular approach for solving the LBEs, is a semi-Lagrangian scheme that comprises a collision and a streaming step at each time step. The Courant–Friedrichs–Lewy (CFL) number for LBM is forced to be 1, because the particle distribution function can only

*Submitted to the journal’s Methods and Algorithms for Scientific Computing section June 30, 2015; accepted for publication (in revised form) March 18, 2016; published electronically June 7, 2016. This work was supported in part by NSFC 11501554, 91530323, 91530103, 91330111, 6151101105, 61120106005, 863 program 2015AA01A302, and NSF CCF-1216314.

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

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

Unlike LBM, the finite difference methods, finite volume methods, and finite element methods aim to directly discretize the temporal and spatial derivatives in the time dependent LBEs. In these methods, the consistency requirement between the mesh points and lattice points are removed and the discretization of the temporal and spatial derivatives are separately chosen according to the accuracy requirement of the problem. It is important to note that most of those methods are explicit or semi-implicit and the stability of the discretization depends on the CFL number. To remove the CFL limitation, a fully implicit method is presented in our previous work [21] for time dependent LBEs. In the fully implicit approach, a large sparse nonlinear algebraic system is constructed at every time step and a parallel, highly scalable, inexact Newton–Krylov–Schwarz (INKS) algorithm is applied to solve the system. An adaptive algorithm which changes the time step size as the system evolves is employed to reduce the computational cost of the fully implicit approach.

To obtain the steady state solution of the problem by solving the time dependent LBEs using the abovementioned methods, a large number of time steps is usually required, especially for high resolution simulations with high Reynolds number. This often leads to a rather large computing time and sometimes a large accumulated error due to the large number of time steps. As shown in [21], to obtain the steady state solution of a two-dimensional (2D) lid-driven cavity problem at a high Reynolds number, the total compute times of a fully implicit method and explicit method both are quite large. When the steady state of the equations is of interest, the steady state system is believed to be a better choice than the time dependent one. As mentioned in [2], the steady state LBEs solver significantly outperforms the time dependent LBEs at low grid resolutions and the advantage of steady state LBEs grows with the increasing size of the problem. In [2], a direct-to-steady state calculation saves about 90% of the total compute time for a Poiseuille flow problem, as compared to solving the time dependent LBEs. Nevertheless, it remains to be assessed whether the aforementioned conclusions still hold for the case of high Reynolds number flows with strong nonlinear effects. Other related work can be found in [44]. As far as we known, there is no published work for the direct-to-steady state calculation for problems with strong nonlinear effect.

In this work, we present a nonlinearly preconditioned inexact Newton (IN) algorithm to directly solve the steady state LBEs with strong nonlinear effects. The classic IN method [13, 41] is popular for solving large sparse nonlinear system of equations arising from discretization of partial differential equations and is quite robust and efficient for smooth nonlinear problems. But if the steady state LBEs with strong nonlinear effects are considered, the convergence rate of IN degrades, and the method may fail to converge even when used together with a globalization technique, such as line search or trust region [25]. To overcome the difficulties, we develop a two-level IN algorithm with a local nonlinear elimination (NE) preconditioner in this paper. On the coarse level, we apply an IN method with a Reynolds number continuation approach and a pollution removing interpolation technique to generate a good initial guess for the fine level IN solver. On the fine level, the local high nonlinearity of the system is eliminated by the NE preconditioner before calculating the global Newton update.

A nonlinear preconditioner can be applied on the left or on the right of the nonlinear function. The basic idea of the left nonlinear preconditioner, such as the additive Schwarz preconditioned inexact Newton algorithm (ASPIN) [3, 22], is to transform the function of the system to a more balanced function and then solve the new system by...
the standard IN. ASPIN has been applied successfully to incompressible Navier–Stokes equations with high Reynolds number flows [3, 4, 6, 22, 23], transonic compressible flows [5, 24], and flows in porous media [39]. On the other hand, the right preconditioner such as the NE algorithm is applied by modifying the variables of the nonlinear function. The key idea of the NE algorithm is to implicitly remove the components that cause trouble for the standard IN. NE has been applied successfully to the quasi one-dimensional shocked duct flow [24] and the transonic full potential equation [25]. Compared with the Navier–Stokes equations, the steady state LBEs are often more challenging to solve due to the less balanced distribution of the unknowns and the possible nonsmoothness of the solution near the moving boundary. As a result, the straightforward extension of NE [24, 25] doesn’t work for LBEs. In this paper, we extend the applicability of NE in a two-level framework with a pollution removing interpolation to the steady state LBEs.

The remainder of the paper is organized as follows. In section 2, we present the steady state LBEs and introduce a second-order finite difference scheme on general curvilinear coordinates. The details of the two-level nonlinearly preconditioned IN solver are described in section 3, in which a pollution removing interpolation technique is also introduced. Two sets of numerical experiments with parallel performance results are reported in section 4, and some concluding remarks are given in section 5.

2. The steady state LBEs and discretization. In this section, we first describe the steady state LBEs [2] based on the D2Q9 model [34] without external force,

\begin{equation}
\mathbf{e}_\alpha \cdot \nabla f_\alpha(x) = -\frac{1}{\tau}(f_\alpha(x) - f_\alpha^{eq}(x)), \quad \alpha = 0, 1, \ldots, 8, \quad x \in \Omega,
\end{equation}

where \(f_\alpha(x)\) is the particle distribution function, and \(f_\alpha^{eq}(x)\) is the local equilibrium distribution function (EDF). The relaxation time \(\tau\) is defined by \(\tau = \nu/c_s^2\), where \(\nu\) is the shear viscosity and \(c_s\) is the speed of sound of the fluid. The discrete velocities \(\mathbf{e}_\alpha\) are given by \(\mathbf{e}_0 = 0\), and \(\mathbf{e}_\alpha = \lambda_\alpha(\cos \theta_\alpha, \sin \theta_\alpha)\) with \(\lambda_\alpha = 1\) for \(\alpha = 1, 2, 3, 4\) and \(\lambda_\alpha = \sqrt{2}\) for \(\alpha = 5, 6, 7, 8\). In the D2Q9 model [34], the EDF is chosen as

\begin{equation}
f_\alpha^{eq}(x) = w_\alpha \rho(x) \left[ 1 + \frac{1}{c_s^2} \mathbf{e}_\alpha \cdot \mathbf{u}(x) + \frac{1}{2c_s^4} (\mathbf{e}_\alpha \cdot \mathbf{u}(x))^2 - \frac{1}{2c_s^4} |\mathbf{u}(x)|^2 \right],
\end{equation}

where \(c_s = 1/\sqrt{3}\), and the weighting factors are defined as \(\omega_0 = 4/9\), \(\omega_\alpha = 1/9\) for \(\alpha = 1, 2, 3, 4\) and \(\omega_\alpha = 1/36\) for \(\alpha = 5, 6, 7, 8\). The macroscopic density \(\rho\), pressure \(p\), and velocity \(\mathbf{u} = (u_1, u_2)\) of the fluid are, respectively, induced from the particle distribution function by

\begin{equation}
\rho(x) = \sum_\alpha f_\alpha(x), \quad p(x) = c_s^2 \rho(x), \quad \mathbf{u}(x) = \frac{1}{\rho(x)} \sum_\alpha f_\alpha(x) \mathbf{e}_\alpha.
\end{equation}

In traditional computational fluid dynamics calculations, the boundary conditions are often given in terms of macroscopic variables \(\mathbf{u}\) and \(p\). To determine the particle distribution function \(f_\alpha\) on the boundary, several algorithms, such as the bounce-back scheme [28], the extrapolation method [9], and the nonequilibrium extrapolation method [17] are developed. Due to the absence of an external force, the given steady state LBEs coupled with any of these boundary conditions is not a well-posed problem. If we assume \(f_\alpha(x)\) is a solution of (2.1), then \(\beta f_\alpha(x)\) is also a solution of (2.1) for
A1704  JIZU HUANG, CHAO YANG, AND XIAO-CHUAN CAI

Fig. 1. (a) Physical domain with curvilinear nonuniform mesh points and ghost points \((x_{i1}^1, x_{j2}^2)\). (b) Transformed domain with uniform mesh points and ghost points \((\xi_{i1}^1, \xi_{j2}^2)\). Here solid circles represent the mesh points and hollow ones represent the ghost points.

any constant \(\beta \neq 0\). To overcome the difficulty, we introduce a normalized particle distribution function \(\tilde{f}_\alpha(x) = f_\alpha(x)/\rho(x)\) and rewrite (2.1) as

\[
\begin{align*}
\left\{ \begin{array}{l}
\mathbf{e}_\alpha \cdot \nabla \tilde{f}_\alpha(x) + \tilde{f}_\alpha(x) \nabla \rho(x) \rho(x) = -\frac{1}{\tau} (\tilde{f}_\alpha(x) - \tilde{f}_\alpha^{(eq)}(x)), \\
\sum_{\alpha=0}^{8} \tilde{f}_\alpha(x) = 1, \quad x \in \Omega,
\end{array} \right.
\end{align*}
\]

(2.4)

where

\[
\tilde{f}_\alpha^{(eq)}(x) = w_\alpha \left[ 1 + \frac{1}{c_s^2} \mathbf{e}_\alpha \cdot \mathbf{u}(x) + \frac{1}{2c_s^2} (\mathbf{e}_\alpha \cdot \mathbf{u}(x))^2 - \frac{1}{2c_s^2} |\mathbf{u}(x)|^2 \right].
\]

(2.5)

Now the macroscopic density \(\rho\) is an unknown in (2.4) and the velocity is defined as \(\mathbf{u} = \sum_\alpha f_\alpha(x) \mathbf{e}_\alpha\). It is important to note that problem (2.4) is well-posed and the approximate Navier–Stokes equation can be derived from it through a Chapman–Enskog expansion procedure.

In this paper, we consider the discretization of (2.4) on general curvilinear coordinates in two dimensions. As shown in Figure 1(a), \(\Omega\) is covered by a curvilinear nonuniform mesh \((x_{i1}^1, x_{j2}^2), i = 0, 1, \ldots, N_1, j = 0, 1, \ldots, N_2\). The mesh is transformed to a uniform mesh \((\xi_{i1}^1, \xi_{j2}^2), \xi_{i1}^1 = h_1^i, \xi_{j2}^2 = h_2^j\) (see Figure 1(b)) with mesh size \(h_1 = (\xi_{i1}^{N_1} - \xi_{i1}^{0})/N_1, h_2 = (\xi_{j2}^{N_2} - \xi_{j2}^{0})/N_2\) through a coordinates transform

(2.6)

\[
\xi_1 = \xi_1(x_1, x_2), \quad \xi_2 = \xi_2(x_1, x_2).
\]

To construct a second-order accurate discretization, a layer of ghost points outside the computational domain is added, denoted as \((\xi_{i1}^{-1}, \xi_{j2}^1), (\xi_{i1}^{N_1+1}, \xi_{j2}^1), (\xi_{i1}^1, \xi_{j2}^{-1}),\) and \((\xi_{i1}^1, \xi_{j2}^{N_2+1})\), respectively. To discretize (2.4) on the transformed uniform mesh, we
first rewrite (2.4) in the coordinates \( \xi = (\xi_1, \xi_2) \) as
\[
\begin{align*}
\mathbf{e}_\alpha \cdot \mathbf{A}(\tilde{f}_\alpha(\xi)) + \tilde{f}_\alpha(\xi) \mathbf{e}_\alpha \cdot \mathbf{A}(\rho(\xi)) &= \frac{1}{\tau} (\tilde{f}_\alpha(\xi) - \tilde{f}_\alpha^{eq}(\xi)), \\
\alpha &= 0, 1, \ldots, 8, \quad \xi \in (\xi_1^0, \xi_1^N) \times (\xi_2^0, \xi_2^N),
\end{align*}
\]
(2.7)
\[
\sum_{\alpha=0}^{8} \tilde{f}_\alpha(\xi) = 1, \quad \xi \in (\xi_1^0, 1) \times (\xi_2^0, \xi_2^N),
\]
with
\[
\mathcal{A} = J^{-T} \left( \frac{\partial}{\partial \xi_1} \right),
\]
(2.8)
where \( J = [J_1; J_2] \) is the Jacobian matrix of the coordinate transform between \( x \) and \( \xi \). To discretize \( \frac{\partial \tilde{f}_\alpha}{\partial \xi_i} \), \( k = 1, 2 \), we apply a family of second-order finite difference schemes as follows:
\[
\frac{\partial \tilde{f}_\alpha}{\partial \xi_i} \bigg|_u = \epsilon \frac{\partial \tilde{f}_\alpha}{\partial \xi_i} \bigg|_c + (1 - \epsilon) \frac{\partial \tilde{f}_\alpha}{\partial \xi_i} \bigg|_e, \quad k = 1, 2, \quad i = 1, 2, \ldots, N_k - 1,
\]
(2.9)
where \( \frac{\partial \tilde{f}_\alpha}{\partial \xi_i} \bigg|_u \) is an upwinding scheme, \( \frac{\partial \tilde{f}_\alpha}{\partial \xi_i} \bigg|_c \) 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 \( \frac{\partial \tilde{f}_\alpha}{\partial \xi_i} \bigg|_u \) and \( \frac{\partial \tilde{f}_\alpha}{\partial \xi_i} \bigg|_c \) are, respectively, defined by
\[
\frac{\partial \tilde{f}_\alpha}{\partial \xi_i} \bigg|_u = \begin{cases} 
\frac{1}{2h_k} \left[ 3\tilde{f}_\alpha(\xi_{i-1}^k) - 4\tilde{f}_\alpha(\xi_{i}^k) + \tilde{f}_\alpha(\xi_{i+1}^k) \right] & \text{if } \mathbf{e}_\alpha \cdot J^k \geq 0, \ 0 < i < N_k, \\
-\frac{1}{2h_k} \left[ 3\tilde{f}_\alpha(\xi_{i}^k) - 4\tilde{f}_\alpha(\xi_{i+1}^k) + \tilde{f}_\alpha(\xi_{i+2}^k) \right] & \text{if } \mathbf{e}_\alpha \cdot J^k < 0, \ 0 < i < N_k,
\end{cases}
\]
and
\[
\frac{\partial \tilde{f}_\alpha}{\partial \xi_i} \bigg|_c = \frac{1}{2h_k} \left[ \tilde{f}_\alpha(\xi_{i+1}^k) - \tilde{f}_\alpha(\xi_{i-1}^k) \right], \quad 1 \leq i \leq N_k - 1,
\]
where \([J^1; J^2] = J^{-T}\). To make the schemes second-order accurate, we introduce a third-order scheme for the ghost points. For brevity, we only show the approximation for the ghost points below the bottom wall. For the ghost point \((\xi_1^i, \xi_2^{-1})\), \(i = 0, 1, \ldots, N_1\), a third-order scheme is given by
\[
\tilde{f}_\alpha(\xi_1^i, \xi_2^{-1}) = 2\tilde{f}_\alpha(\xi_1^i, \xi_2^0) - 2\tilde{f}_\alpha(\xi_1^i, \xi_2^2) + \tilde{f}_\alpha(\xi_1^i, \xi_2^3), \quad i = 0, 1, \ldots, N_1.
\]
It is worth pointing out that the discretization scheme for the macroscopic density \( \rho \) is the same as that of the normalized particle distribution function \( \tilde{f}_\alpha \).

The boundary conditions for the normalized particle distribution function \( \tilde{f}_\alpha \) are given based on the second-order nonequilibrium extrapolation method [21], which is second-order accurate on the boundary. In the second-order nonequilibrium extrapolation method, the particle distribution function \( f_\alpha \) of the 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 the EDF (2.2) with the macroscopic boundary conditions, and the nonequilibrium part \( f_\alpha^{neq} \) is approximately obtained by an second-order extrapolation approximation. Let us take a point \((\xi_1^1, \xi_2^0)\) on the bottom
wall as an example; the boundary condition for \( f_0(\xi_1, \xi_2) \) is set to be

\[
(2.10) \quad f_0(\xi_1, \xi_2) \approx f^{eq}_0(\xi_1, \xi_2) + 2f^{(neq)}_0(\xi_1, \xi_2) - f^{(neq)}(\xi_1, \xi_2).
\]

Combining (2.10) and \( \tilde{f}_\alpha(\xi) = f_\alpha(\xi)/\rho(\xi) \), the boundary condition for \( \tilde{f}_\alpha(\xi_1, \xi_2) \) is defined as

\[
(2.11) \quad \tilde{f}_\alpha(\xi_1, \xi_2)\rho(\xi_1, \xi_2) = \tilde{f}^{eq}_\alpha(\xi_1, \xi_2)\rho(\xi_1, \xi_2) + 2\tilde{f}^{(neq)}_\alpha(\xi_1, \xi_2)\rho(\xi_1, \xi_2) - \tilde{f}^{(neq)}(\xi_1, \xi_2)\rho(\xi_1, \xi_2),
\]

where \( \tilde{f}^{(neq)}(\xi_1, \xi_2) = \tilde{f}_\alpha(\xi_1, \xi_2) - \tilde{f}^{eq}_\alpha(\xi_1, \xi_2) \) for \( j = 1, 2 \). The boundary conditions for the normalized particle distribution function \( \tilde{f}_\alpha \) on the other three walls are similar.

The boundary condition of the macroscopic density \( \rho \) is determined by the local pressure boundary condition [27, 32] given as follows:

\[
(2.12) \quad c_s^2 \frac{\partial \rho(x)}{\partial n} = \frac{\partial p(x)}{\partial n} = \frac{1}{Re} \mathbf{n} \cdot \nabla^2 \mathbf{u}(x) \quad \text{on } \partial \Omega,
\]

where \( \mathbf{n} = (n_1, n_2) \) is the outward unit normal vector. The boundary condition in the curvilinear coordinates \((\xi_1, \xi_2)\) is rewritten as

\[
(2.13) \quad c_s^2 \mathbf{n} \cdot \mathbf{A}(\rho(\xi)) = \frac{1}{Re} \mathbf{n} \cdot \mathbf{JH}
\]

with \( \mathbf{u} = (u_1, u_2)^T = J(u^1, u^2)^T \) and

\[
(2.14) \quad H_k = \sum_{i,j,l} \left( \frac{1}{\lambda \partial \xi_i} \left[ g^{ij}_l \frac{\partial u^k}{\partial \xi_j} \right] + \frac{1}{\lambda \partial \xi_j} \left[ \Lambda g^{ij}_k \frac{\partial u^l}{\partial \xi_i} \right] + \sum_m g^{ij}_m \Gamma^{ik}_m \Gamma^{jl} u^l \right),
\]

where \( g^{ij} \) is the \( ij \)th component of the inverse metric tensor for the curvilinear coordinates, \( \Lambda \) is the square root of the determinant of the metric tensor, and \( \Gamma^{ij}_m \) is the \( mij \)th Christoffel symbol. It is clear that the values of \( J, \mathbf{H}, g^{ij}, \Lambda, \) and \( \Gamma^{ij}_m \) are dependent on the coordinates transform. We present the discretizations of the pressure boundary condition (2.13) in detail in the appendix.

3. **A nonlinearly preconditioned inexact Newton algorithm with a Krylov–Schwarz Jacobian solver.** After discretizing (2.7) with the proposed second-order scheme, a large sparse nonlinear system of equations

\[
(3.1) \quad \tilde{F}(\mathbf{X}) = 0
\]

is constructed. For both the unknowns and the equations, a point-block (field-coupled) ordering is used to organize the algebraic system on the mesh. We denote the approximate solution as

\[
\mathbf{X} = (\tilde{f}_{01}, \tilde{f}_{11}, \ldots, \tilde{f}_{81}, \rho^{11}, \tilde{f}_{02}, \tilde{f}_{12}, \ldots, \tilde{f}_{82}, \rho^{21}, \ldots)^T,
\]

where \( \tilde{f}_{ij} = \tilde{f}_\alpha(\xi_i, \xi_2) \) and \( \rho^{ij} = \rho(\xi_i, \xi_2) \). Similarly, the point-block ordering is used for \( \tilde{F} \). This ordering helps in improving not only the cache performance but also the parallel efficiency in load and communication balance; see [15], for example.

For the steady state LBEs, there are ten unknowns at each mesh point, corresponding to the normalized particle distribution functions \( \tilde{f}_\alpha, \alpha = 0, 1, \ldots, 8 \), and the density \( \rho \). The scales between the ten equations are not well-balanced for a given
mesh point. We rescale the nonlinear system (3.1) as
\begin{equation}
\mathcal{F}_i(X) = \begin{cases} 
\tilde{F}_i(X) & \text{if } (i+1) \text{ is divided exactly by 10,} \\
\tau \tilde{F}_i(X) & \text{otherwise.}
\end{cases}
\end{equation}

In the rest of this section, we propose a two-level IN algorithm with nonlinear preconditioner to solve the rescaled nonlinear system \( \mathcal{F}(X) = 0 \).

For such a large sparse nonlinear system, an IN algorithm such as INKS is an efficient and popular solver [6, 45]. We give a brief review of INKS as follows. Let \( X^n \) be the approximate solution at the \( n \)th iteration. The approximate solution at the \( (n+1) \)th Newton iteration is given by
\begin{equation}
X^{n+1} = X^n + \lambda^n S^n, \quad n = 0, 1, \ldots,
\end{equation}
where \( \lambda^n \) is the step-length obtained by a line search procedure [11] and \( S^n \) is the search direction which is an approximate solution of the Jacobian system
\begin{equation}
\mathcal{J}^n S^n = -\mathcal{F}(X^n),
\end{equation}
where the Jacobian matrix \( \mathcal{J}^n = \mathcal{F}'(X^n) \). In the INKS algorithm, the Jacobian system is solved inexactly by using a Krylov subspace method with a preconditioner. The stopping conditions for the Krylov iteration and Newton iteration are
\begin{align*}
\| \mathcal{J}^n S^n + \mathcal{F}(X^n) \| &\leq \max \{ \eta_r \| \mathcal{F}(X^n) \|, \eta_a \} \\
\| \mathcal{F}(X^{n+1}) \| &\leq \max \{ \gamma_r \| \mathcal{F}(X^0) \|, \gamma_a \},
\end{align*}
respectively. Here \( \eta_r, \eta_a \geq 0 \) are linear tolerances and \( \gamma_r, \gamma_a \geq 0 \) are nonlinear tolerances. All tolerances will be chosen in the numerical results section.

### 3.1. A nonlinearly preconditioned INKS

The convergent behavior of INKS is problem dependent. When the nonlinearities in the system are well-balanced and a good initial guess is available, a near quadratic convergence is usually observed. However, if the nonlinearities of the system are strong and not well-balanced, such as the nonlinear system arising from the discretization of the steady state LBEs at a high Reynolds number, the convergence of INKS is problematic. In this paper, we propose an inexact two-level Newton algorithm with a NE preconditioner to solve the nonlinear system \( \mathcal{F}(X) = 0 \). A coarse mesh is introduced to generate the initial guess for the IN algorithm on the fine mesh and the NE preconditioner is proposed to balance the nonlinearities of the system. Below, we present a high level description of the basic algorithm for a general problem and then discuss the details of each component.

**Algorithm 1.** INKS with a coarse mesh \( Re \)-continuation and NE based nonlinear preconditioning.

**Phase 1: On the coarse mesh** \( (N_{1,c} + 1) \times (N_{2,c} + 1) \): Solve the nonlinear system by a Reynolds number continuation approach.
- Choose an increasing sequence of Reynolds numbers such as \( Re = 10, 100, 400, 1000, 2000, 3200, \ldots \).
- Set the initial guess on the coarse mesh to \( u = 0 \) and solve the nonlinear system with a relatively small Reynolds number by the classical INKS algorithm.
- Increase the Reynolds number step by step, until the nonlinear system with the given Reynolds number is solved. At each step, the approximate solution with the previous Reynolds number is taken as the initial guess for the current Reynolds number.
Phase 2: Pollution removing interpolation: We obtain the initial guess $X_0$ on the fine mesh by interpolating the approximate solution on the coarse mesh to the fine mesh with the pollution removing technique.

Phase 3: On the fine mesh $(N_1 + 1) \times (N_2 + 1)$: Solve the nonlinear system by INKS with the NE preconditioner. Set $X_n$ to be the approximate solution at the $n$th global Newton iteration.

Step 1 (The Nonlinearity Checking Step): Check the local and global stopping conditions.
- If the global condition is satisfied, stop.
- If the local conditions indicate that nonlinearities are unbalanced and the residual is reduced slowly, go to Step 2; otherwise, go to Step 3.

Step 2 (The NE Step): Construct and solve the local nonlinear system.
- Mark the points with relatively large residuals as bad-points.
- Construct the local nonlinear system $F^n_{b, \delta}(R^{n,\delta}(X))$.
- Solve the local nonlinear system $F^n_{b, \delta}(R^{n,\delta}(X))$ by the classical INKS algorithm. Assume $X_n = R^{n,\delta}(X^*) + R^{n,\delta}(X^n)$, where $R^{n,\delta}(X^*)$ is the approximate solution of the local nonlinear system. If $\|F(X_n)\| > \|F(\tilde{X}_n)\|$, set $X_n = \tilde{X}_n$. Go to Step 3.

Step 3 (The global Newton Step): Compute the $(n + 1)$th global Newton iteration by $X^{n+1} = X^n + \lambda^n S^n$, and go to Step 1.

3.2. Detailed explanations of the new algorithm.

3.2.1. The pollution removing interpolation. For solving a linear or nonlinear system of equations arising from the discretization of PDEs, a coarse mesh correction often leads to the reduction of iterations. However, for some problems, the linear system of equations arising from the discretization of PDEs, a coarse mesh the LBEs.

In this paper, we follow the same idea but define a new interpolation more suitable for this phenomenon is the coarse mesh pollution. In other words, the location or the number of iterations may increase when a coarse correction is involved. The reason for this phenomenon is the coarse mesh pollution. To remove the coarse mesh pollution, a modified coarse-to-fine interpolation procedure is proposed in [33] for certain flow control problems. In this paper, we follow the same idea but define a new interpolation more suitable for the LBEs.

3.2.1. The pollution removing interpolation. For solving a linear or nonlinear system of equations arising from the discretization of PDEs, a coarse mesh correction often leads to the reduction of iterations. However, for some problems, the number of iterations may increase when a coarse correction is involved. The reason for this phenomenon is the coarse mesh pollution. In other words, the location or the sharpness of the jump of the solution on the fine mesh is dislocated or smoothed out by the coarse correction. To remove the coarse mesh pollution, a modified coarse-to-fine interpolation procedure is proposed in [33] for certain flow control problems. In this paper, we follow the same idea but define a new interpolation more suitable for the LBEs.

Let $I : \mathbb{R}^{N_c} \to \mathbb{R}^N$ denote a linear interpolation operator from the coarse to fine mesh, where $N_c = M(N_1,c + 1) \times (N_2,c + 1)$ and $N = M(N_1 + 1) \times (N_2 + 1)$ are the total number of unknowns on the coarse mesh and fine mesh, respectively. Here, $M$ is the total number of unknowns at a mesh point. The modified interpolation is expressed as

$$I_{\text{modif}} = I - Z(I_c - Z_c),$$

where $I_c : \mathbb{R}^{N_c} \to \mathbb{R}^{N_c}$ denotes the identity operator, $Z_c : \mathbb{R}^{N_c} \to \mathbb{R}^{N_c}$ and $Z : \mathbb{R}^{N} \to \mathbb{R}^{N}$ are given in [33]. To describe the pollution removing interpolation more clearly, we take a one-dimensional case with only one unknown at each mesh point as an example. Let us assume $\xi_{N_c}$ is a given polluted point (see Figure 2(a)). The operator $Z_c$ and $Z$ are, respectively, defined as $Z_c(X_c) = X_c$ and $Z(X) = X$, where the $i$th component of $X_c$ and $X$ are, respectively, defined as

$$\tilde{X}_i = \begin{cases} 0 & \text{if } i = N_c - 1, \\ X_i & \text{if } i = 0, 1, \ldots, N_c - 2, \end{cases}$$

$$\tilde{X}_i = \begin{cases} 0 & \text{if } i = N - 1, \\ X_i & \text{if } i = 0, 1, \ldots, N - 2. \end{cases}$$

For problems with sharp jumps, the modified interpolation is efficient. However, for the LBEs the pollution removing technique based on the above interpolation is
too excessive. To remove the pollution more accurately, we propose a new modified interpolation as

$$I_{new}^{\text{modify}} = I - ZI_{modify}(I_c - \tilde{Z}_c)$$

with operator $\tilde{Z}_c(X^c) = \bar{X}^c$, where the $i$th component of $\bar{X}^c$ is

$$\bar{X}^c_i = \begin{cases} 2X^c_{N^-c-2} - X^c_{N^-c-3} & \text{if } i = N^-c - 1, \\ X^c_i & \text{otherwise.} \end{cases}$$

The “clean” fine mesh solutions given by the modified interpolation operators $I_{modify}$ and $I_{new}^{\text{modify}}$ are shown in Figure 2(c), (d), respectively. It is clear that the new modified interpolation operator $I_{new}^{\text{modify}}$ leads to a more reasonable solution. A detailed comparison between the two modified interpolation operators is given in section 4.

### 3.2.2. The nonlinear elimination preconditioner.

When the nonlinearities at the $n$th Newton iteration are not well-balanced and the residual is reduced slowly, an NE step is introduced as a nonlinear preconditioner to accelerate the Newton iteration. We consider $\frac{\|F(X_n)\|}{\|F(X_{n-1})\|}$ as a measure of the relative reduction of the residual at the $n$th Newton iteration. We say that the residual reduces slowly if $\frac{\|F(X_n)\|}{\|F(X_{n-1})\|} > \varrho_0$, where $0 < \varrho_0 < 1$ is a prechosen constant. To check the balance of the nonlinearity, we first introduce some notation. Let $I$ be an index set of $N/M$ mesh points, and each index corresponds to $M$ unknown components $X_{im}$ and $M$ nonlinear residual components $F_{im}$, $m = 0, 1, \ldots, M-1$. At each Newton iteration, we decompose the set $I$ into a bad subset $I^n_b$ with $M^n$ mesh points and a good subset $I^n_g$ with $(N/M - M^n)$ mesh points, where $I^n_g = I \setminus I^n_b$. We say that the $i$th mesh point is a bad point if $\max_m \{\|F_{im}\|\} > \varrho_1 \|F\|_{\infty}$, where $\varrho_1$ is a prechosen constant. The nonlinearities at the $n$th Newton iteration are not well-balanced if $M^n < \varrho_2 N/M$, where $0 < \varrho_2 < 1$ is another prechosen constant.

Next let us introduce an index set $I^{n,\delta}_b$ as an extension of $I^n_b$, which is given as

$$I^{n,\delta}_b = \{ i | \text{if the } i\text{th mesh point is in the } \delta\text{-neighborhood of a bad point } j\text{th}\};$$

here the $j$th mesh point belongs to the bad subset $I^n_b$. In this paper, $\delta$ is set to be $2h_1$ in the $\xi_1$ direction and $2h_2$ in the $\xi_2$ direction, which is consistent with the
stencil width of the proposed finite difference scheme. With this subset, we define two subspaces
\[ V_b^{n,\delta} = \{ v | v = (v_0, \ldots, v_{N-1})^T \in \mathbb{R}^N, v_{i_0} = 0 \text{ if } i \notin I_b^{n,\delta} \} \]
and
\[ V_g^{n,\delta} = \{ v | v = (v_0, \ldots, v_{N-1})^T \in \mathbb{R}^N, v_{i_0} = 0 \text{ if } i \notin I_g^{n,\delta} \}, \]
respectively. The corresponding restriction operators are denoted as \( R_b^{n,\delta} \) and \( R_g^{n,\delta} \), both map vectors from \( \mathbb{R}^N \) to \( V_b^{n,\delta} \) and \( V_g^{n,\delta} \), respectively. Then we define the local nonlinear function \( F_b^{n,\delta} : V_b^{n,\delta} \rightarrow V_b^{n,\delta} \) as
\[
F_b^{n,\delta}(R_b^{n,\delta}(X)) = R_b^{n,\delta}(F(R_b^{n,\delta}(X) + R_g^{n,\delta}(X^n))).
\]

The local nonlinear system \( F_b^{n,\delta}(R_b^{n,\delta}(X)) = 0 \) is solved by using the classical INKS algorithm. The initial guess for the local nonlinear system is chosen as \( R_b^{n,\delta}(X^n) \). \( R_b^{n,\delta}(X^n) \) is accepted as the approximate solution of the local nonlinear system if the stopping condition \( \| F_b^{n,\delta}(R_b^{n,\delta}(X^n)) \| \leq \max\{\gamma_r^{NE}\|R_b^{n,\delta}(X^n)\|, \gamma_a^{NE}\} \) is satisfied, where \( \gamma_r^{NE}, \gamma_a^{NE} \) are the relative tolerance and absolute tolerance for the local nonlinear solver. With this approximate solution, we define a temporary solution \( \tilde{X}_n \) for the global nonlinear system at the \( n \)th iteration as
\[
\tilde{X}_n = R_b^{n,\delta}(X^n) + R_g^{n,\delta}(X^n).
\]
If \( \| F(\tilde{X}_n) \| < \| F(X_n) \| \), we set \( X_n = \tilde{X}_n \) and accept the temporary solution \( \tilde{X}_n \) as the solution of the global nonlinear system at the \( n \)th iteration.

The tolerances control the accuracy of the solution of the local nonlinear system. If the local nonlinear system is solved too accurately, the residuals corresponding to the bad points in the global nonlinear system are also small, which may introduce further unbalances of the nonlinearities. Therefore the relative tolerance \( \gamma_r^{NE} \) is set to be something not too small. In this work, we set the absolute and relative tolerances to \( 10^{-8} \) and \( 10^{-2} \), respectively. The local nonlinear iteration is also stopped if the maximum allowable iteration number is reached.

4. Numerical experiments. In this section, we investigate the numerical behavior and parallel performance of the newly proposed algorithm. We consider two test cases, namely, a flow around a circular cylinder and a lid-driven cavity flow. We focus on (1) the verification of the correctness of the proposed discretization scheme, (2) a comparison of the proposed solution algorithm with the classical INKS method, (3) a comparison of the performance of an explicit method, a fully implicit method with adaptive time stepping, and the new algorithm, and (4) the parallel scalability of the new algorithm.

We implement the algorithms described in the previous sections based on PETSc [1]. The numerical tests are carried out on a parallel computer with 282 nodes. Each node consists of two 4-core Intel X5550 CPUs with 24GB local memory and is interconnected via the Infiniband high performance network. In the numerical experiments we use all 8 cores in each node and assign one subdomain to each core.

The stopping conditions for the nonlinear iterations are as follows:
- The relative tolerance on the fine mesh is \( 10^{-6} \).
- The absolute tolerance on the fine mesh is \( 10^{-5} \).

Copyright © by SIAM. Unauthorized reproduction of this article is prohibited.
The relative tolerance on the coarse mesh is $10^{-2}$.

The absolute tolerance on the coarse mesh is $10^{-6}$.

In all experiments, the Jacobian matrix is analytically calculated. The linear system at each Newton iteration is solved by GMRES (restarts at every 50 iterations) [36] with the right restricted additive Schwarz (RAS) preconditioner [7]. In the RAS preconditioner, we apply a sparse point-block LU factorization as the subdomain solver and set the overlapping size between subdomains to be $h, 2h, 3h$, where $h$ is the fine mesh size. The block size is $10 \times 10$. The absolute and relative tolerances for the linear solver are $10^{-8}$ and $10^{-5}$, respectively.

4.1. Flow around a circular cylinder. The first test case is a 2D flow around a circular cylinder, which is used to validate the proposed discretization scheme. In the test, we assume the flow has a free stream velocity $U_0 = 0.1$. The Reynolds number is defined as $Re = DU_0/\nu$, where $D = 1$ is the diameter of the cylinder. This problem has been studied extensively as a benchmark case by many researchers, and thus there are numerous experimental and numerical results available in the literature. Since the proposed method is aimed at solving the steady state LBEs directly, we only focus on the steady state solution of the problem at Reynolds numbers $Re = 10, 20, 40$. Due to the relatively small Reynolds numbers, the nonlinear effects of the system are not too strong. So, the standard INKS without a coarse mesh and the NE-based nonlinear preconditioner are good enough to solve the sparse nonlinear equations. This test case focuses on validating the discretization of the proposed algorithm while the pollution removing interpolation and the NE preconditioner will be examined in the second test case.

The geometry and the curvilinear nonuniform mesh used for the numerical simulation of the steady flows around the circular cylinder are shown in Figure 3. The radius of computational domain is set to be 50 times of the diameter of the cylinder, and only the upper half domain is considered. The mesh employed in the simulation is generated based on the following coordinate transform:

$$x_1 = q(\xi_1) \cos(\xi_2), \quad x_2 = q(\xi_1) \sin(\xi_2), \quad \xi_1 \in (R, R), \quad \xi_2 \in (0, \pi)$$

with $q(\xi_1) = \bar{R} + (R - \bar{R}) \frac{\tanh(3\xi_1 - 3)}{\tanh 3}$, where $\bar{R} = 0.5, \sqrt{\bar{R}} = 50$. In the $\xi_2$ direction, the symmetrical boundary condition [37] is implemented to define the normalized distribution function $\tilde{f}_n$ and density $\rho$. On the wall, we impose a no-slip boundary condition. On the inlet boundary, the values of the macroscopic velocity components...
Fig. 4. Streamlines of the flow over a circular cylinder at different Reynolds numbers. Here “present method” (upper half) denotes the solution of the steady state LBEs computed by using the proposed method, and “implicit method” (lower half) denotes the solution obtained by solving the time dependent LBEs with an implicit method [21].

are assumed to be given. For the calculation of the density $\rho$ on the wall and the inlet boundary, the pressure boundary condition is used. For the outlet boundary, the value of pressure is assumed to be known and the values of the macroscopic velocity components are obtained from the stress-free boundary conditions. With the given macroscopic pressure and velocity, the values of normalized distribution function $\tilde{f}_\alpha$ on the wall, on the inlet boundary, and on the outlet boundary are obtained by using the second-order nonequilibrium extrapolation method (2.11).

The steady state solutions on a $200 \times 200$ mesh for $Re = 10, 20, 40$ are shown in Figures 4 and 5. It is observed that a pair of stationary recirculating eddies appears behind the cylinder for the three Reynolds numbers. We compare the solutions of the steady state LBEs computed by using the proposed method and the solutions obtained by solving the time dependent LBEs using an implicit method [21] in these
Fig. 5. Vorticity contours of the flow over a circular cylinder at different Reynolds numbers. Here, "present method" (upper half) denotes the solution of the steady state LBEs computed by using the proposed method, and "implicit method" (lower half) denotes the solution obtained by solving the time dependent LBEs by using an implicit method [21].

Figures. Note that the computed results using the new algorithm match quite well with the results from a well established algorithm. As the Reynolds number increases, the length of the recirculating region $L$ (the distance from the rearmost point of the cylinder to the end of the wake) and the separation angle $\theta_s$ become larger.

In Table 1, we present the numerical solution obtained by solving the steady state LBEs in the curvilinear coordinates with several experimental and numerical methods [10, 12, 16, 18, 19, 26, 30, 31, 40] for the hydrodynamic coefficients and the locations of the separation and reattachment points. The drag coefficient $C_d$ computed by solving the steady state LBEs with the proposed IN algorithm agrees well with the values obtained by other methods. The stagnation pressure coefficients at the front of the cylinder $C_p(\pi)$ and at its rear $C_p(0)$ also agree well with the results in previously published works. The length of the recirculating region normalized by the radius of
Table 1
Flow around a circular cylinder. Comparisons between the results obtained by solving the steady state LBEs in the curvilinear coordinates and the previously published experimental and numerical results. Here “-” means that the value is not available.

<table>
<thead>
<tr>
<th>Re</th>
<th>Reference</th>
<th>Method</th>
<th>$C_d$</th>
<th>$C_p(0)$</th>
<th>$C_p(\pi)$</th>
<th>$2\pi \theta_s$</th>
</tr>
</thead>
<tbody>
<tr>
<td>10</td>
<td>[40]</td>
<td>Experiment</td>
<td>3.06</td>
<td>-</td>
<td>-</td>
<td>0.68</td>
</tr>
<tr>
<td></td>
<td>[10]</td>
<td>Experiment</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>0.68</td>
</tr>
<tr>
<td></td>
<td>[12]</td>
<td>Navier–Stokes</td>
<td>2.846</td>
<td>-0.742</td>
<td>1.489</td>
<td>0.53</td>
</tr>
<tr>
<td></td>
<td>[31]</td>
<td>Navier–Stokes</td>
<td>2.828</td>
<td>-0.692</td>
<td>1.500</td>
<td>0.434</td>
</tr>
<tr>
<td></td>
<td>[18]</td>
<td>ISLBM</td>
<td>3.170</td>
<td>-0.687</td>
<td>1.393</td>
<td>0.474</td>
</tr>
<tr>
<td></td>
<td>[30]</td>
<td>FDLBM</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>0.489</td>
</tr>
<tr>
<td></td>
<td>[16]</td>
<td>FDLBM</td>
<td>3.049</td>
<td>-0.661</td>
<td>1.476</td>
<td>0.486</td>
</tr>
<tr>
<td></td>
<td>[26]</td>
<td>GILBM</td>
<td>2.848</td>
<td>-0.733</td>
<td>1.403</td>
<td>0.478</td>
</tr>
<tr>
<td></td>
<td>[19]</td>
<td>CFDLBM</td>
<td>2.801</td>
<td>-0.673</td>
<td>1.487</td>
<td>0.489</td>
</tr>
<tr>
<td></td>
<td>Present</td>
<td>IN–SSLBEs</td>
<td>2.893</td>
<td>-0.680</td>
<td>1.476</td>
<td>0.496</td>
</tr>
<tr>
<td>20</td>
<td>[40]</td>
<td>Experiment</td>
<td>2.2</td>
<td>-</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td></td>
<td>[10]</td>
<td>Experiment</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>1.86</td>
</tr>
<tr>
<td></td>
<td>[12]</td>
<td>Navier–Stokes</td>
<td>2.045</td>
<td>-0.589</td>
<td>1.269</td>
<td>1.88</td>
</tr>
<tr>
<td></td>
<td>[31]</td>
<td>Navier–Stokes</td>
<td>2.053</td>
<td>-0.582</td>
<td>1.274</td>
<td>1.786</td>
</tr>
<tr>
<td></td>
<td>[18]</td>
<td>ISLBM</td>
<td>2.152</td>
<td>-0.567</td>
<td>1.233</td>
<td>1.842</td>
</tr>
<tr>
<td></td>
<td>[30]</td>
<td>FDLBM</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>1.804</td>
</tr>
<tr>
<td></td>
<td>[16]</td>
<td>FDLBM</td>
<td>2.048</td>
<td>-0.512</td>
<td>1.289</td>
<td>1.824</td>
</tr>
<tr>
<td></td>
<td>[26]</td>
<td>GILBM</td>
<td>2.051</td>
<td>-0.589</td>
<td>1.251</td>
<td>1.852</td>
</tr>
<tr>
<td></td>
<td>[19]</td>
<td>CFDLBM</td>
<td>2.021</td>
<td>-0.547</td>
<td>1.266</td>
<td>1.848</td>
</tr>
<tr>
<td></td>
<td>Present</td>
<td>IN–SSLBEs</td>
<td>2.050</td>
<td>-0.570</td>
<td>1.265</td>
<td>1.868</td>
</tr>
<tr>
<td>40</td>
<td>[40]</td>
<td>Experiment</td>
<td>1.65</td>
<td>-</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td></td>
<td>[10]</td>
<td>Experiment</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>4.26</td>
</tr>
<tr>
<td></td>
<td>[12]</td>
<td>Navier–Stokes</td>
<td>1.522</td>
<td>-0.509</td>
<td>1.144</td>
<td>4.69</td>
</tr>
<tr>
<td></td>
<td>[31]</td>
<td>Navier–Stokes</td>
<td>1.550</td>
<td>-0.554</td>
<td>1.117</td>
<td>4.357</td>
</tr>
<tr>
<td></td>
<td>[18]</td>
<td>ISLBM</td>
<td>1.499</td>
<td>-0.497</td>
<td>1.113</td>
<td>4.490</td>
</tr>
<tr>
<td></td>
<td>[30]</td>
<td>FDLBM</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>4.38</td>
</tr>
<tr>
<td></td>
<td>[16]</td>
<td>FDLBM</td>
<td>1.475</td>
<td>-0.448</td>
<td>1.168</td>
<td>4.168</td>
</tr>
<tr>
<td></td>
<td>[26]</td>
<td>GILBM</td>
<td>1.538</td>
<td>-0.514</td>
<td>1.156</td>
<td>4.454</td>
</tr>
<tr>
<td></td>
<td>[19]</td>
<td>CFDLBM</td>
<td>1.515</td>
<td>-0.481</td>
<td>1.154</td>
<td>4.510</td>
</tr>
<tr>
<td></td>
<td>Present</td>
<td>IN–SSLBEs</td>
<td>1.528</td>
<td>-0.496</td>
<td>1.152</td>
<td>4.588</td>
</tr>
</tbody>
</table>

the cylinder $2L/D$ and the separation angle are both close to the results obtained by other published works for the three Reynolds numbers. It is worth pointing out that all of existing numerical methods obtain the steady state solution by pseudo time stepping, and the proposed IN algorithm is able to obtain the steady state solution without time stepping. The experiments confirm the correctness of the proposed algorithm.

Now let us compare the proposed method with a fully implicit method with adaptive time stepping and an explicit method [21]. For the explicit method, we use a fixed time step size $\Delta t = 5.03 \times 10^{-3}$ which is determined by the maximum allowable CFL number. For the implicit method, we apply the adaptive time stepping algorithm and the first-order discretization-based RAS preconditioner. The stopping condition for the implicit method is given by $\|G(X^n)\| \leq 2 \times 10^{-6}$, and the explicit method stops at the same physical time with the implicit method. Here $G(\cdot)$ is a nonlinear function of $X^n$ which depends on the spatial discretization and the collision term of the time dependent LBEs (see the definition in [21]). More details about the explicit method and implicit method are given in [21]. From Table 2, it is clear that the total compute time of the proposed method is smaller than that of the explicit method and the implicit method.

4.2. A lid-driven cavity flow. The second test case is a 2D lid-driven cavity flow problem which has been used as a benchmark problem for many numerical
A comparison of the new method, an implicit method, and an explicit method for the 2D flow around a cylinder. A nonuniform mesh of size $200 \times 200$ and 32 CPU processors are used.

<table>
<thead>
<tr>
<th>Reynolds number</th>
<th>Present method</th>
<th>Implicit method</th>
<th>Explicit method</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Newton</td>
<td>Implicit time steps</td>
<td>Explicit time steps</td>
</tr>
<tr>
<td></td>
<td>GMRES/Newton</td>
<td>Newton</td>
<td>GMRES/Newton</td>
</tr>
<tr>
<td></td>
<td>Total compute time(s)</td>
<td>Total compute time(s)</td>
<td>Total compute time(s)</td>
</tr>
<tr>
<td>10</td>
<td>17.66</td>
<td>5</td>
<td>368</td>
</tr>
<tr>
<td>20</td>
<td>13.88</td>
<td>4</td>
<td>409</td>
</tr>
<tr>
<td>40</td>
<td>14.32</td>
<td>4</td>
<td>514.5</td>
</tr>
<tr>
<td></td>
<td></td>
<td>58.8</td>
<td>49.5</td>
</tr>
<tr>
<td></td>
<td></td>
<td>58.0</td>
<td>51.45</td>
</tr>
<tr>
<td></td>
<td></td>
<td>60.8</td>
<td>51.18</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Fig. 6. Velocity profiles of the cavity flow for 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.

Due to its simple geometry and complicated flow behaviors, in the test, the top wall 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 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, 10,000, 20,000, 50,000, and 100,000. Due to the large Reynolds number and the strong nonlinear effect of the system, we use the proposed preconditioned IN algorithm to solve the problem. In this case, an identity coordinate transform $x_1 = \xi_1, x_2 = \xi_2$ is considered. We use a uniform $(N_{1,c} + 1, N_{2,c} + 1)$ coarse mesh and a uniform $(N_1 + 1, N_2 + 1)$ fine mesh. The second-order nonequilibrium extrapolation method is applied as boundary conditions for $\tilde{f}_\alpha$ on all walls, and the pressure boundary condition is used for $\rho$. The density boundary condition on the bottom wall is discretized as

$$-(3\rho(\cdot, \xi_2^0) - 4\rho(\cdot, \xi_2^1) + \rho(\cdot, \xi_2^2)) = \frac{1}{c_s^2 Re h_1} \left(4u_1(\cdot, \xi_2^2) - \frac{1}{2} u_1(\cdot, \xi_2^2)\right).$$

The density boundary conditions on the other three walls are similar.

To verify the proposed algorithm, the two velocity components $u_1$ and $u_2$ along the vertical and horizontal lines through the cavity center are shown in Figure 6, together with the benchmark solutions obtained by solving the Navier–Stokes equations [14].
and the numerical solutions given in [21] by solving the time dependent LBEs using a fully implicit method. Agreements are clearly reached which validate the newly proposed discretization scheme. We also compare the vorticity contours and the streamline patterns obtained by the new method with that of the other two methods. The consistency between those contours is obvious.

We next compare the new modified interpolation operator with the modified interpolation operator and the linear interpolation operator. For the lid-driven cavity flow problem, the solutions \( f_\alpha, \alpha = 0, 1, \ldots, 8 \), have a small jump on the top wall which is shown in Figure 7(d). The fine mesh solution obtained by the linear interpolation operator is polluted due to the jump on the wall and the corresponding residual (as shown in Figure 7(a)) is also polluted. The pollution of the fine mesh solution obtained by the modified interpolation operator \( I_{\text{modify}} \) is overly corrected, and the residual is still relatively large (as shown in Figure 7(b), (e)). As shown in Figure 7(f), the proposed modified interpolation is able to remove the pollution cleanly and the corresponding residual is smaller than that of the other two approaches (see Figure 7(c)). In the rest of the section, we use the new modified interpolation operator to obtain the initial guess \( \mathbf{X}_0 \) on the fine mesh.

Next we present a comparison of the INKS-NE algorithm with the classical INKS algorithm. In Figure 8(a), we show the history of the norm of the residuals of several tests with different Reynolds numbers on a \( 257 \times 257 \) mesh with mesh size \( h = 1/256 \) by using the INKS-NE algorithm and the classical INKS algorithm. Both methods use the same initial guess which is obtained from a coarse mesh solution corrected by the pollution removing interpolation. In the INKS-NE algorithm, the prechosen
constants \( \rho_0 = 0.9, \rho_1 = 0.125, \) and \( \rho_2 = 0.25 \) are used for all Reynolds numbers. As the Reynolds number increases, the nonlinear system becomes harder to solve. The number of Newton iterations of the classical INKS algorithm increases rapidly once the Reynolds number passes a certain value. However, the number of global Newton iterations of the INKS-NE algorithm does not change much as we increase the Reynolds number as shown in Figure 8(a). The total compute time of the INKS-NE algorithm and the classical INKS algorithm are shown in Figure 8(b). It is important to note that the total compute times of the INKS-NE algorithm are always less than that of the INKS algorithm, especially for a high Reynolds number. In Figure 9, we show the surface plots for the sixth component of \( \mathcal{F}(X^n) \) corresponding to \( f_5 \). The other components are similar. In Figure 9(a), we observe that the dominant part of the residual is near the top right corner of the computational domain. After the local nonlinear system is solved, the dominant components are reduced as shown in Figure 9(b). It is clear that the NE preconditioner improves the performance of the global Newton iteration by balancing the nonlinearities.
Fig. 9. The 2D lid-driven cavity flow. A comparison of the residual surfaces obtained from the INKS method (left) and the INKS-NE algorithm (right). Both figures correspond to the sixth component of the residual function. The results are for the sixth iteration of both methods. The calculation is carried out for $Re = 5 \times 10^4$ on a $257 \times 257$ mesh.

Table 3
The numbers of iterations of the INKS-NE. Mesh size $129 \times 129$, $193 \times 193$, $257 \times 257$ on 64 processors. The overlapping size is $3h$. Note that the relative and absolute tolerances for the linear solver for $Re = 10^5$ are $10^{-2}$ and $10^{-7}$, respectively.

<table>
<thead>
<tr>
<th>Reynolds number</th>
<th>$10^3$</th>
<th>$5 \times 10^3$</th>
<th>$10^4$</th>
<th>$2 \times 10^4$</th>
<th>$5 \times 10^4$</th>
<th>$10^5$</th>
</tr>
</thead>
<tbody>
<tr>
<td>Global Newton iterations</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>$129 \times 129$</td>
<td>3</td>
<td>5</td>
<td>7</td>
<td>5</td>
<td>11</td>
<td>12</td>
</tr>
<tr>
<td>$193 \times 193$</td>
<td>3</td>
<td>4</td>
<td>8</td>
<td>11</td>
<td>9</td>
<td>11</td>
</tr>
<tr>
<td>$257 \times 257$</td>
<td>3</td>
<td>4</td>
<td>6</td>
<td>11</td>
<td>6</td>
<td>10</td>
</tr>
<tr>
<td>Average GMRES iterations</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>$129 \times 129$</td>
<td>638</td>
<td>497</td>
<td>298</td>
<td>383</td>
<td>3/6</td>
<td>16</td>
</tr>
<tr>
<td>$193 \times 193$</td>
<td>1,085</td>
<td>586</td>
<td>1,267</td>
<td>451</td>
<td>567</td>
<td>21</td>
</tr>
<tr>
<td>$257 \times 257$</td>
<td>1,491</td>
<td>778</td>
<td>1,007</td>
<td>256</td>
<td>353</td>
<td>37</td>
</tr>
<tr>
<td>The total Newton iterations for NE preconditioner</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>$129 \times 129$</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>8</td>
<td>27</td>
</tr>
<tr>
<td>$193 \times 193$</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>14</td>
<td>17</td>
</tr>
<tr>
<td>$257 \times 257$</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>28</td>
<td>24</td>
</tr>
</tbody>
</table>

In Table 3, we summarize the number of Newton iterations on three different meshes $129 \times 129$, $193 \times 193$, and $257 \times 257$ with six different Reynolds numbers. As shown in Table 3, the number of global Newton iterations is almost independent of the mesh size and increases slowly with the increase of the Reynolds number. For the case with Reynolds number less than $2 \times 10^4$, the classical INKS algorithm with coarse mesh correction converges in several iterations. For the case with Reynolds number larger than $2 \times 10^4$, the NE preconditioner plays an important role in the INKS algorithm. We also observe that the total number of Newton iterations for the NE preconditioner increases as the Reynolds number becomes larger. As shown in Figure 8, the total compute time of the NE preconditioner is smaller than over 70% of the total compute time on the fine mesh. Based on this together with the number of iterations reported in Table 3, we conclude that the computational cost per Newton iteration for the NE preconditioner is lower than that of the global Newton iteration, and the NE preconditioner improves the convergence of the classical INKS and thus decreases the total compute time. On the other hand, the number of GMRES iterations per Newton step is sensitive to the mesh size and the Reynolds number.

To understand the impact of the prechosen constants $\varrho_0$, $\varrho_1$, and $\varrho_2$ on the performance of the proposed method, a test with $Re = 50,000$ on a $129 \times 129$ coarse
Table 4

Results on the number of iterations and the total compute time of the INKS-NE method on a 257 × 257 mesh using 64 processors. The overlapping size is 3h, Re = 50,000.

<table>
<thead>
<tr>
<th>ϱ1 = 0.125 and ϱ2 = 0.25</th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>ϱ0</td>
<td>0.84</td>
<td>0.86</td>
<td>0.88</td>
<td>0.9</td>
<td>0.92</td>
</tr>
<tr>
<td>Global Newton iterations</td>
<td>10</td>
<td>10</td>
<td>10</td>
<td>10</td>
<td>14</td>
</tr>
<tr>
<td>Average GMRES iterations</td>
<td>353</td>
<td>353</td>
<td>353</td>
<td>353</td>
<td>338</td>
</tr>
<tr>
<td>Newton iterations for NE preconditioner</td>
<td>28</td>
<td>28</td>
<td>28</td>
<td>28</td>
<td>34</td>
</tr>
<tr>
<td>Total compute time(s)</td>
<td>226.8</td>
<td>223.1</td>
<td>227.0</td>
<td>223.7</td>
<td>295.5</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>ϱ0 = 0.9 and ϱ1 = 0.25</th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>ϱ1</td>
<td>0.095</td>
<td>0.11</td>
<td>0.125</td>
<td>0.14</td>
<td>0.155</td>
</tr>
<tr>
<td>Global Newton iterations</td>
<td>9</td>
<td>16</td>
<td>10</td>
<td>8</td>
<td>11</td>
</tr>
<tr>
<td>Average GMRES iterations</td>
<td>419</td>
<td>312</td>
<td>353</td>
<td>259</td>
<td>295</td>
</tr>
<tr>
<td>Newton iterations for NE preconditioner</td>
<td>38</td>
<td>52</td>
<td>28</td>
<td>21</td>
<td>42</td>
</tr>
<tr>
<td>Total compute time(s)</td>
<td>220.8</td>
<td>365.9</td>
<td>223.7</td>
<td>131.3</td>
<td>227.8</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>ϱ0 = 0.9 and ϱ2 = 0.25</th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>ϱ2</td>
<td>0.22</td>
<td>0.23</td>
<td>0.24</td>
<td>0.25</td>
<td>0.26</td>
</tr>
<tr>
<td>Global Newton iterations</td>
<td>11</td>
<td>11</td>
<td>11</td>
<td>10</td>
<td>10</td>
</tr>
<tr>
<td>Average GMRES iterations</td>
<td>354</td>
<td>354</td>
<td>354</td>
<td>353</td>
<td>353</td>
</tr>
<tr>
<td>Newton iterations for NE preconditioner</td>
<td>29</td>
<td>29</td>
<td>29</td>
<td>28</td>
<td>28</td>
</tr>
<tr>
<td>Total compute time(s)</td>
<td>240.7</td>
<td>237.2</td>
<td>240.5</td>
<td>223.7</td>
<td>227.7</td>
</tr>
</tbody>
</table>

Fig. 10. Strong scaling results for solving the driven cavity flow problem. Re = 5 × 10^4 on a 393 × 393 mesh, and the overlapping size is 3h. The coarse mesh is 197 × 197. When 128 or 256 processors are used, the absolute tolerance on the fine mesh is set to be 2 × 10^−5.
Fig. 11. Weak scaling results for solving the driven cavity flow problem. \( Re = 5 \times 10^4 \) on different meshes with a coarse mesh \((N_1/2 + 1) \times (N_2/2 + 1)\), and the overlapping size is \(3h\).

Table 5

<table>
<thead>
<tr>
<th>Reynolds number</th>
<th>10^4</th>
<th>5 \times 10^4</th>
<th>10^5</th>
<th>2 \times 10^5</th>
<th>5 \times 10^4</th>
<th>10^5</th>
</tr>
</thead>
<tbody>
<tr>
<td>Mesh size</td>
<td>129</td>
<td>257</td>
<td>257</td>
<td>257</td>
<td>257</td>
<td>257</td>
</tr>
<tr>
<td>Number of processors</td>
<td>128</td>
<td>128</td>
<td>128</td>
<td>128</td>
<td>128</td>
<td>128</td>
</tr>
<tr>
<td>a Newton time steps</td>
<td>175</td>
<td>1,683</td>
<td>5,974</td>
<td>–</td>
<td>–</td>
<td>–</td>
</tr>
<tr>
<td>GMRES/Newton</td>
<td>382</td>
<td>481</td>
<td>285</td>
<td>1,196</td>
<td>944</td>
<td>346</td>
</tr>
<tr>
<td>Total compute time(s)</td>
<td>7.6</td>
<td>59.4</td>
<td>72.4</td>
<td>141.6</td>
<td>155.3</td>
<td>170.1</td>
</tr>
<tr>
<td>b Implicit time steps</td>
<td>1,633</td>
<td>14,782</td>
<td>39,674</td>
<td>–</td>
<td>–</td>
<td>–</td>
</tr>
<tr>
<td>Newton time steps</td>
<td>88</td>
<td>136</td>
<td>133</td>
<td>–</td>
<td>–</td>
<td>–</td>
</tr>
<tr>
<td>GMRES/Newton</td>
<td>88</td>
<td>136</td>
<td>133</td>
<td>–</td>
<td>–</td>
<td>–</td>
</tr>
<tr>
<td>Total compute time(s)</td>
<td>260.1</td>
<td>8.7e+3</td>
<td>4.3e+4</td>
<td>–</td>
<td>–</td>
<td>–</td>
</tr>
<tr>
<td>c Explicit time steps</td>
<td>1.8e+6</td>
<td>3.4e+7</td>
<td>5.1e+8</td>
<td>–</td>
<td>–</td>
<td>–</td>
</tr>
<tr>
<td>Total compute time(s)</td>
<td>642.6</td>
<td>4.8e+4</td>
<td>7.2e+5</td>
<td>–</td>
<td>–</td>
<td>–</td>
</tr>
</tbody>
</table>

Now let us compare the proposed method with a fully implicit method and an explicit method [21] by solving a lid-driven cavity flow problem with strong nonlinear effect. As shown in Table 5, the total time steps and the total compute time of the fully implicit method and the explicit method increase quickly as the Reynolds number grows. With the NE preconditioner, the proposed method solves the problem in fewer than 10 Newton iterations for all Reynolds numbers, which saves about 99% of the total compute time in comparing with the given time stepping approaches. The results show that the new method is a clear winner, especially at high Reynolds number. Moreover, the new method shows a robust performance with a wide range of Reynolds numbers.

5. Concluding remarks. In this paper, a nonlinearly preconditioned inexact Newton algorithm was developed to directly compute the steady state solution of LBEs. A normalized particle distribution function is introduced to convert the steady state LBEs to a well-posed problem, which is then discretized on a general curvilinear mesh by a mixed second-order finite difference scheme. The numerical solution obtained by solving the steady state LBEs with the proposed method are in good agreement with existing numerical methods and the solution obtained by experimental studies. With the NE preconditioner and the coarse correction, the proposed
method is able to obtain a steady state solution of the LBEs for high Reynolds numbers. The proposed algorithm exhibits a linear speedup in terms of strong scaling and a reasonably good parallel efficiency in terms of weak scaling with up to 2048 processors. Compared to the explicit method and the fully implicit method, the new method saves a large amount of compute time, even for the case of high Reynolds flows with strong nonlinear effects. More importantly, the convergence of the proposed method is not sensitive with respect to the high Reynolds numbers, which is a property not shared by any existing algorithms.

**Appendix A.** In the first test case, the coordinate transform is defined as

\[ x_1 = q(\xi_1) \cos(\xi_2), \quad x_2 = q(\xi_1) \sin(\xi_2) \]

with \( q(\xi_1) = \bar{R} + (R - \bar{R}) \frac{\tanh \left( \frac{\xi_1}{2} \right)}{\tanh \left( \frac{1}{2} \right)} \), where \( 0 < \bar{R} < \bar{R} \) are given constants. For this transform, the Jacobian matrix \( J \) and its transpose inverse \( J^{-T} \) are

\[
J = \begin{bmatrix}
q'(\xi_1) \cos(\xi_2) & -q(\xi_1) \sin(\xi_2) \\
q'(\xi_1) \sin(\xi_2) & q(\xi_1) \cos(\xi_2)
\end{bmatrix},
\quad J^{-T} = \begin{bmatrix}
\cos(\xi_1) & -\sin(\xi_1) \\
\sin(\xi_1) / q(\xi_1) & \cos(\xi_1) / q(\xi_1)
\end{bmatrix}.
\]

The metric tensor and the inverse metric tensor are defined by

\[
G = \begin{bmatrix}
[q'(\xi_1)]^2 & 0 \\
0 & q(\xi_1)^2
\end{bmatrix}, \quad G^{-1} = \begin{bmatrix}
[q'(\xi_1)]^{-2} & 0 \\
0 & q(\xi_1)^{-2}
\end{bmatrix},
\]

and \( \Lambda = q'(\xi_1)q(\xi_1) \). The Christoffel symbols \( \Gamma^m \) are given by

\[
\Gamma^1 = \begin{bmatrix}
0 & -q(\xi_1) / q'(\xi_1) \\
q(\xi_1) / q'(\xi_1) & 0
\end{bmatrix}, \quad \Gamma^2 = \begin{bmatrix}
0 & q'(\xi_1) / q(\xi_1) \\
q'(\xi_1) / q(\xi_1) & 0
\end{bmatrix}.
\]

Then \( \mathcal{A} = \frac{\cos(\xi_1)}{q'(\xi_1) q(\xi_1) q(\xi_1)} \begin{bmatrix}
\frac{\partial}{\partial \xi_1} & \frac{\partial}{\partial \xi_2} & \frac{\partial}{\partial \xi_2} \\
\frac{\partial}{\partial \xi_1} & \frac{\partial}{\partial \xi_2} & \frac{\partial}{\partial \xi_2}
\end{bmatrix} \), and

\[
H_1 = \frac{1}{qq'} \frac{\partial}{\partial \xi_1} \left( \frac{q du^1}{q' \partial \xi_1} \right) + \frac{1}{q^2} \frac{\partial^2 u^1}{\partial \xi_2^2} + \frac{q''}{(q')^3} \frac{\partial u^1}{\partial \xi_1} - \frac{1}{qq'} \frac{\partial u^2}{\partial \xi_2}
\]

(5.1)

\[
H_2 = \frac{1}{qq'} \frac{\partial}{\partial \xi_1} \left( \frac{q du^2}{q' \partial \xi_1} \right) + \frac{1}{q^2} \frac{\partial^2 u^2}{\partial \xi_2^2} + \frac{q'}{q^3} \frac{\partial u^1}{\partial \xi_2} - \frac{1}{qq'} \frac{\partial u^2}{\partial \xi_1}
\]

(5.2)

On the wall (\( \xi_1 = \bar{R} \)), a no-slip boundary condition \( (u^1 = u^2 = 0) \) is considered. The divergence-free condition in the given coordinates \((\xi_1, \xi_2)\) is defined as \( \frac{\partial u^1}{\partial \xi_1} + \frac{\partial u^2}{\partial \xi_2} = 0 \),
which implies \( \frac{\partial u_1}{\partial \xi_1} = 0 \). Thanks to this, we simplify \( H_1 \) and \( H_2 \) as

\[
H_1 = \frac{1}{(q')^2} \frac{\partial^2 u_1}{\partial \xi_1^2},
\]
\[
H_2 = \frac{1}{(q')^2} \frac{\partial^2 u_2}{\partial \xi_1^2} - \frac{q''}{q'} \frac{\partial u_2}{\partial \xi_1}.
\]

Therefore the pressure boundary condition on the wall \( \xi_1 = R \) is given by

\[
\frac{\partial p(\xi)}{\partial \xi_1} = \frac{1}{c_s^2 Re} \frac{\partial^2 u_1}{\partial \xi_1^2}.
\]

Using the ghost points, the boundary condition (5.5) is discretized as

\[
-\frac{3(\rho(\cdot, \xi_2^0) - 4\rho(\cdot, \xi_2^1) + \rho(\cdot, \xi_2^2))}{2} = \frac{1}{c_2^2 Re h_1} (u^1(\cdot, \xi_2^1) + u^1(\cdot, \xi_2^{-1}))
\]

It is worth noting that \( \nabla \cdot \mathbf{u} = \sum_i \frac{1}{K} \frac{\partial u_i}{\partial \xi_i} = 0 \) holds on \( \partial \Omega \), which gives a high order approximation at the ghost points as \( u^1(\cdot, \xi_2^1) = -\frac{3}{2}u^1(\cdot, \xi_2^0) + 3u^1(\cdot, \xi_2^{-1}) - \frac{1}{2}u^1(\cdot, \xi_2^2) \).

Based on this, the pressure boundary condition on the wall is discretized as

\[
-\frac{3(\rho(\cdot, \xi_2^0) - 4\rho(\cdot, \xi_2^1) + \rho(\cdot, \xi_2^2))}{2} = \frac{1}{c_2^2 Re h_1} (4u^1(\cdot, \xi_2^1) - \frac{1}{2}u^1(\cdot, \xi_2^2)).
\]

REFERENCES


