One-point boundary condition for the lattice Boltzmann method

Michael Junk* and Zhaoxia Yang†
FB Mathematik und Statistik, Universität Konstanz, Fach D194, D-78457 Konstanz, Germany
(Received 26 July 2005; published 5 December 2005)

We propose a correction to the bounce back boundary condition for lattice Boltzmann algorithms which improves the accuracy of pressure from zero to first order and the accuracy of velocity from first to second order. Compared to interpolation based corrections, our method has the advantage of being completely local. In fact, methods using interpolation face difficulties at boundary points where not enough neighboring nodes are available. We show that a combination with our method offers a natural solution to this problem.

DOI: 10.1103/PhysRevE.72.066701 PACS number(s): 47.11.+j, 02.70.−c, 05.20.Dd, 51.90.+r

I. INTRODUCTION

In contrast to conventional CFD methods which are based on a direct discretization of the flow equations, the lattice Boltzmann (LB) approach [1,2] uses an intrinsic connection between kinetic theory and the continuum equations. While this trick allows us to change the target equation from a nonlinear system to a semilinear one, it has the disadvantage that the simpler lattice Boltzmann equation involves more unknowns from which the macroscopic fields of interest (like pressure and velocity) are obtained by averaging. In the interior of the flow domain, the increased number of variables results in a larger memory requirement. At boundaries of the domain where, for example, velocity Dirichlet conditions are to be implemented, the disadvantage is more fundamental: one cannot prescribe directly the flow velocity at boundary nodes but one has to set certain variables in the kinetic equation in such a way that the average velocity satisfies the required conditions. Typically, the required number of kinetic conditions exceeds the available conditions from the flow problems. This indicates already that the kinetic conditions have to be chosen carefully in order to avoid the appearance of extra conditions on the macroscopic level which would render the problem ill posed (leading to an unwanted behavior on the grid scale like boundary layers, oscillations, etc.). In view of these remarks, it is not surprising that numerous articles can be found in the literature which deal with the problem of specifying boundary conditions for the lattice Boltzmann method.

One of the most basic algorithms which was already used in connection with the lattice gas ancestor of the lattice Boltzmann method is the bounce back rule. While being a quite robust method, it soon proved not to be accurate enough [3] and modifications have been invented to achieve higher accuracy [4–17]. The idea to complement the given velocity conditions with additional (more or less physically motivated) relations to obtain the kinetic boundary equations has been adopted, for example, in Refs. [4,14,15,17]. Another approach consists in prescribing a suitably modified equilibrium distribution at the boundary [10,16].

A quite popular modification of the bounce back rule is the BFL method given in Ref. [5]. Here, linear interpolation along lattice links is used to improve accuracy. A generalization of this approach can be found in Ref. [9] where also higher order corrections and stability aspects are considered. A common problem of all interpolation based methods is the requirement of a sufficient number of neighboring nodes which may not always be available. Let us consider, for example, the BFL condition which requires one additional neighbor in each velocity direction which enters the domain. For the D2Q9 velocity model (velocities point along the x and y axes and in the diagonal directions) we can face the situation depicted in Fig. 1.

If the classical bounce back rule is used at the boundary nodes where the interpolation is not applicable, the accuracy of the solution generally degrades *everywhere* in the domain (we give examples in Sec. III B). To maintain the order of the interpolation condition, it is therefore mandatory to find accurate additional conditions for certain exceptional boundary nodes.

Interesting candidates for this purpose are sufficiently accurate *local* boundary conditions which do not require any neighboring information. As an example, we mention the method presented in Refs. [6,7]. However, this algorithm also has some deficiencies. In the case shown on the left side of Fig. 1, the algorithm is actually inconsistent (see Ref. [18]) and the method is not uniformly applicable to all choices of the relaxation parameter which determines the collision operator in the lattice Boltzmann equation. Modifications which alleviate the latter problem without removing it have been presented in Refs. [12,13].

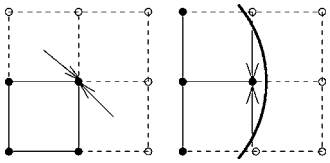

FIG. 1. Examples of boundary nodes where neighbors are not available for all incoming velocities. Nodes of the computational domain are marked with filled circles. At corners one typically finds opposing incoming directions, i.e., the neighbors of the boundary node in these directions are *not* in the computational domain. The same problem can occur in the case of smooth, curved boundaries as shown in the right diagram.

*Electronic address: michael.junk@uni-konstanz.de
†Electronic address: zhaoxia.yang@uni-konstanz.de

In this paper we present a different local boundary algorithm which works without restriction on the relaxation parameter and which can be shown to give a first order accurate pressure and a second order accurate velocity [18]. The scheme is tested for several numerical examples and a comparison with the popular BFL method [5] shows a very similar behavior. We also demonstrate that our method can be used in connection with BFL to remove the problems mentioned above.

II. SETUP OF THE LATTICE BOLTZMANN METHOD

We consider the incompressible Navier-Stokes equation on a domain $\Omega \subset \mathbb{R}^d$, $d \in \{2,3\}$, with initial and Dirichlet boundary values. Our aim is to find numerical approximations of the fields $\boldsymbol{u}:[0,T]\times\Omega\to\mathbb{R}^d$ and $p:[0,T]\times\Omega\to\mathbb{R}$, which satisfy

$$\boldsymbol{\nabla}\cdot\boldsymbol{u}=0, \quad \partial_t\boldsymbol{u}+\boldsymbol{u}\cdot\boldsymbol{\nabla}\boldsymbol{u}+\boldsymbol{\nabla}p=\nu\boldsymbol{\nabla}^2\boldsymbol{u}+\boldsymbol{G}, \tag{1}$$

with $\boldsymbol{u}|_{t=0}=\boldsymbol{\psi}$ and

$$\boldsymbol{u}(t,\boldsymbol{x})=\boldsymbol{\phi}(t,\boldsymbol{x}), \quad t\in[0,T], \quad \boldsymbol{x}\in\partial\Omega, \tag{2}$$

where $\boldsymbol{\psi}$, $\boldsymbol{\phi}$, and $\boldsymbol{G}$ are given functions. To this end we consider a lattice Boltzmann algorithm for the evolution of particle distribution functions $\hat{f}_i(n,\boldsymbol{j})$ which specify the fraction of fluid particles traveling with microscopic velocity $\boldsymbol{c}_i$ at position $\boldsymbol{x}_j=h\boldsymbol{j}$, $\boldsymbol{j}\in\mathbb{Z}^d$ and time $t_n=nh^2$, $n\in\mathbb{N}_0$. The parameter h is the nondimensional grid spacing which is assumed to be small compared to the (scaled) size of the domain. The actual Navier-Stokes fields are recovered in the form of averages over the microscopic velocity,

$$\hat{\rho}(n,\boldsymbol{j})=\sum_i \hat{f}_i(n,\boldsymbol{j}), \quad \hat{\boldsymbol{u}}(n,\boldsymbol{j})=\sum_i \hat{f}_i(n,\boldsymbol{j})\boldsymbol{c}_i. \tag{3}$$

Here, $\hat{\boldsymbol{u}}$ approximates the velocity field and the pressure is related to small deviations of the total mass density $\hat{\rho}$ from the constant value $\rho_0=1$. The discrete evolution of the particles consists of two phases, a transport step and a collision and forcing step. The latter has the form

$$\hat{f}_i^c(n,\boldsymbol{j})=\hat{f}_i(n,\boldsymbol{j})+C_i[\hat{f}](n,\boldsymbol{j})+\hat{g}_i(n,\boldsymbol{j}), \tag{4}$$

where the so called collision operator is of relaxation type

$$C_i[\hat{f}](n,\boldsymbol{j})=\frac{1}{\tau}[f_i^{eq}(\hat{\rho}(n,\boldsymbol{j}),\hat{\boldsymbol{u}}(n,\boldsymbol{j}))-\hat{f}_i(n,\boldsymbol{j})]$$

with $\tau=1/2+3\nu$ and $\hat{\rho}$ and $\hat{\boldsymbol{u}}$ defined through Eq. (3). The equilibrium distribution function [19] is given by

$$f_i^{eq}(\rho,\boldsymbol{u})=\left[\rho+3\boldsymbol{u}\cdot\boldsymbol{c}_i+\frac{9}{2}(\boldsymbol{u}\cdot\boldsymbol{c}_i)^2-\frac{3}{2}|\boldsymbol{u}|^2\right]f_i^*.$$

The weight factors f_i^* depend on the chosen velocity model. For the D2Q9 model which is based on the velocity set $\{-1,0,1\}^2$, the values are $4/9$ for the zero velocity, $1/9$ for the axis parallel velocities of length one, and $1/36$ for the diagonal velocities of length $\sqrt{2}$. The presented scheme also works in connection with the models D3QX, $X\in\{15,19,27\}$ (the weights and velocities can be found, for example, in Ref. [20]). The function $\hat{g}_i$ models the influence of the force term

$$\hat{g}_i(n,\boldsymbol{j})=3h^3f_i^*\boldsymbol{c}_i\cdot\boldsymbol{G}(t_n,\boldsymbol{x}_j). \tag{5}$$

We remark that the relation $\Delta t=h^2=\Delta x^2$ between space and time step and the scaling of the force term $\hat{g}$ with h^3 is necessary to obtain convergence towards the incompressible Navier-Stokes system (1). These issues have been discussed in Ref. [21].

After collision where the original particle distribution function $\hat{f}_i$ transforms into the post-collisional state $\hat{f}_i^c$, the particles simply move undisturbed with their velocities to the neighboring lattice sites,

$$\hat{f}_i(n+1,\boldsymbol{j}+\boldsymbol{c}_i)=\hat{f}_i^c(n,\boldsymbol{j}). \tag{6}$$

This update rule is applied to all nodes $\boldsymbol{x}_j\in\Omega$ for which all neighbors $\boldsymbol{x}_j+h\boldsymbol{c}_i$ are contained in Ω. At the remaining nodes (the so called boundary nodes with labels $\boldsymbol{j}\in J_{bdr}$), a modification is required because neighbors in certain directions are missing. We discuss some standard choices and our method below.

For a complete description, the initial distributions have to be prescribed at all grid points. As in Ref. [16], we take

$$\hat{f}_i(0,\boldsymbol{j})=f_i^{eq}(1+3h^2p(0,\boldsymbol{x}_j),h\boldsymbol{\psi}(\boldsymbol{x}_j))-3h^2\boldsymbol{\tau}\boldsymbol{c}_i(\cdot\boldsymbol{\nabla})\boldsymbol{c}_i\cdot\boldsymbol{\psi}(\boldsymbol{x}_j)f_i^*, \tag{7}$$

where $p(0,.)$ is the initial Navier-Stokes pressure.

III. BOUNDARY CONDITIONS

A. Bounce back rule

The classical method to implement Dirichlet velocity conditions is the bounce back rule. If node $\boldsymbol{j}\in J_{bdr}$ has a missing neighbor in direction $-\boldsymbol{c}_i$ (i.e., if $\boldsymbol{c}_i$ is an incoming direction), the bounce back rule specifies the particle distribution $\hat{f}_i$ according to

$$\hat{f}_i(n+1,\boldsymbol{j})=\hat{f}_{i^*}^c(n,\boldsymbol{j})+6hf_i^*\boldsymbol{c}_i\cdot\boldsymbol{\phi}(t_n,\boldsymbol{x}_{ji}). \tag{8}$$

The velocity index i^* is defined by $\boldsymbol{c}_{i^*}=-\boldsymbol{c}_i$, and the boundary value $\boldsymbol{\phi}$ is evaluated at the boundary point $\boldsymbol{x}_{ji}=\boldsymbol{x}_j-hq_{ji}\boldsymbol{c}_i\in\partial\Omega$ where $q_{ji}\in[0,1)$ represents the distance to the boundary along direction $\boldsymbol{c}_{i^*}=-\boldsymbol{c}_i$ as fraction of $h|\boldsymbol{c}_i|$.

A careful analysis shows that if the lattice Boltzmann algorithm presented in the previous section is used in connection with condition (8), the resulting solution is, in general, a quite inaccurate approximation of the Navier-Stokes fields. The pressure is inconsistent and the error of velocity is proportional to h [18].

Only in some exceptional cases, for example when *all* boundary nodes are located half a grid spacing away from the boundary, the accuracy is one order higher for both pressure and velocity. However, it should be noted that the choice $q_{ji}=1/2$ is only possible for a very restricted class of flow geometries Ω.

To illustrate this behavior, we consider a two-dimensional (2D) stationary linear flow on the unit square $\Omega=(0,1)^2$

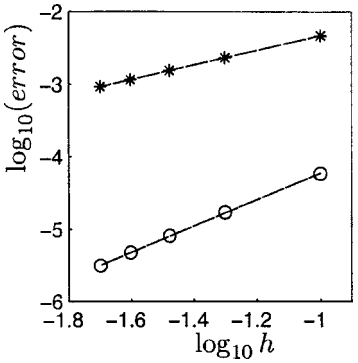

FIG. 2. Logarithmic velocity error versus $\log_{10}h$ in the case of a 2D stationary linear flow with $q_{ji}=0$ (∗) and $q_{ji}=0.5$ (○).

$$\boldsymbol{u}(\boldsymbol{x}) = A\boldsymbol{x}, \quad p(\boldsymbol{x}) = -\frac{1}{2}\boldsymbol{x}^{\mathsf{T}}A^2\boldsymbol{x}, \quad A = \begin{pmatrix} 4 & 1 \\ 1 & -4 \end{pmatrix}. \tag{9}$$

The required boundary and initial values are set to the corresponding values of the exact solution and the numerical tests are carried out on a sequence of grids with grid size $h \in \{\frac{1}{10}, \frac{1}{20}, \frac{1}{30}, \frac{1}{40}, \frac{1}{50}\}$. The termination time is $T=1$. The relaxation time is computed according to $\nu=0.1$ in Eq. (1) and the error between exact and numerical values is measured with the maximum norm in space and time.

Figures 2 and 3 compare the performance of the bounce back rule in the case $q_{ji}=0$ (all boundary nodes are on $\partial\Omega$) with the favorable case $q_{ji}=1/2$ which can be used for this simple set Ω. Note, in particular, that the pressure does not converge in the general case $q_{ji} \neq 1/2$ so that better boundary schemes are required.

B. BFL rule

An improvement of the bounce back rule has been proposed in Ref. [5]. The boundary condition can be viewed as a correction of Eq. (8) in the case $q_{ji} \neq 1/2$. Denoting the right hand side of Eq. (8) as $\hat{f}^b_{i^*}(n,\boldsymbol{j})$, the condition has the form

$$\hat{f}_i(n+1,\boldsymbol{j}) = \hat{f}^b_{i^*}(n,\boldsymbol{j}) + (1-2q_{ji})\Delta^{\pm}_i(n,\boldsymbol{j}),$$

where

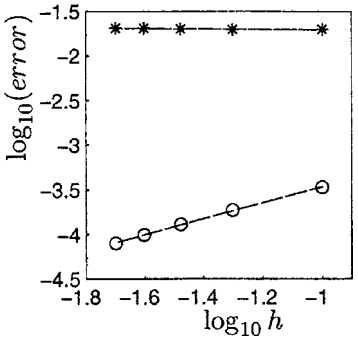

FIG. 3. Logarithmic pressure error versus $\log_{10}h$ in the case of a 2D stationary linear flow with $q_{ji}=0$ (∗) and $q_{ji}=0.5$ (○).

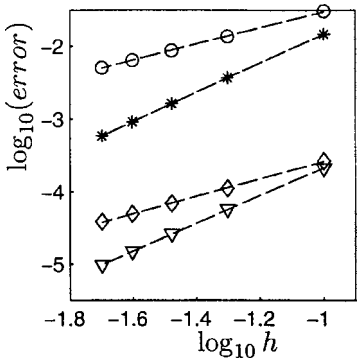

FIG. 4. Logarithmic velocity error versus $\log_{10}h$ in the case of a 2D stationary linear flow with different boundary treatments at the points where neighbors in incoming directions are missing. For circular geometry, ○ and ∗ denote the results with bounce back rule (○) and POP_1 (∗). For the flow in a square, the bounce back result is ◇ and POP_1 yields ▽.

$$\Delta^-_i(n,\boldsymbol{j}) = \hat{f}^c_{i^*}(n,\boldsymbol{j}+\boldsymbol{c}_i) - \hat{f}^c_{i^*}(n,\boldsymbol{j}), \quad q_{ji} \leqslant 1/2,$$

$$\Delta^+_i(n,\boldsymbol{j}) = \hat{f}_i(n+1,\boldsymbol{j}) - \hat{f}^c_i(n,\boldsymbol{j}), \quad q_{ji} > 1/2.$$

Note that the expression Δ^-_i requires a neighbor in the incoming direction. If the neighbor is missing (like in Fig. 1), the condition is undefined and usually replaced by the bounce back rule (8). This choice, however, generally degrades the accuracy. To illustrate this behavior, we consider the previous test case on a square and a circular geometry (reflecting the problematic cases in Fig. 1). Figures 4 and 5 show that the BFL rule supplemented with the bounce back algorithm inherits the low order of Eq. (8) even though it is only used at very few boundary nodes. In contrast to this, a combination with the new local boundary condition POP_1 presented in the next section yields first order pressure and second order accurate velocity at *all* points which is a simple but efficient improvement.

It should be stressed that the low order error is *not* located at the few isolated nodes where the bounce back rule is used

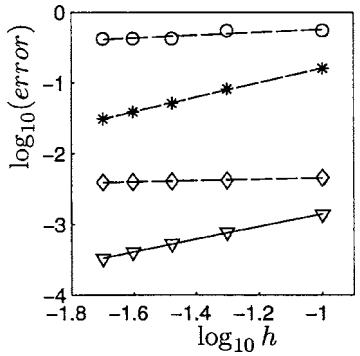

FIG. 5. Logarithmic pressure error versus $\log_{10}h$ in the case of a 2D stationary linear flow with different boundary treatments at the points where neighbors in incoming directions are missing. For circular geometry, ○ and ∗ denote the results with bounce back rule (○) and POP_1 (∗). For the flow in a square, the bounce back result is ◇ and POP_1 yields ▽.

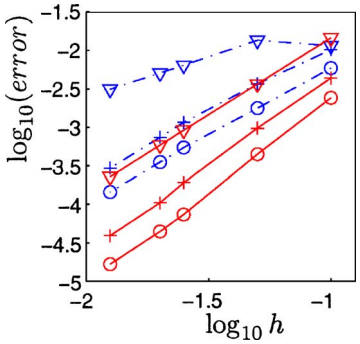

FIG. 6. (Color online) Logarithmic velocity error versus $\log_{10}h$ in the case of a 2D stationary linear flow in a disk with different boundary treatments at the points where neighbors in incoming directions are missing. The dashed lines denote the results with bounce back rule, solid lines stand for the results with POP_1. The symbols ○, ∗, ▽ refer to errors in the concentric subregions R_1, R_3, R_5.

but appears everywhere in the domain. To illustrate this behavior, we consider the linear flow (9) in the unit disk more carefully. In this case, the BFL rule cannot be applied at the four isolated points which are extremal in the coordinate directions. We measure the numerical error within concentric rings R_1, R_3, R_5, where

$$R_i = \{(x,y): r_{i-1}^2 < x^2 + y^2 < r_i^2\},$$

$$r_0 = 0,\ r_1 = 0.5,\ r_2 = 0.7,\ r_3 = 0.8,\ r_4 = 0.9,\ r_5 = 1.0$$

and determine its maximal value during the simulation for each subdomain on several grids. In Figs. 6 and 7, the pressure and velocity errors are shown. One can clearly see that the error is generally lower for subdomains which are futher away from the boundary. In contrast to the results obtained with our method employed at the four isolated points, the bounce back results are less accurate in every ring and the convergence order is generally lower (the orders for velocity and pressure are reported in Table I).

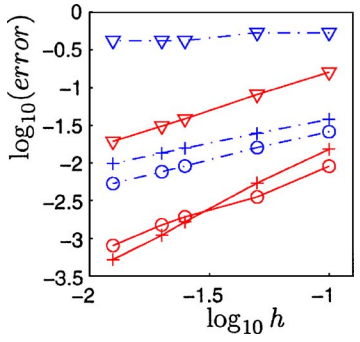

FIG. 7. (Color online) Logarithmic pressure error versus $\log_{10}h$ in the case of a 2D stationary linear flow in a disk with different boundary treatments at the points where neighbors in incoming directions are missing. The dashed lines denote the results with bounce back rule, solid lines stand for the results with POP_1. The symbols ○, ∗, ▽ refer to errors in the concentric subregions R_1, R_3, R_5.

TABLE I. Comparison of numerical convergence orders of velocity/pressure for a linear flow in a disk with different treatment of the isolated points where BFL cannot be used. The error is measured on concentric rings R_1, R_3, R_5.

Ring	Bounce back	POP_1
R_1	1.7730/0.7665	2.4293/1.1231
R_3	1.6968/0.6496	2.2933/1.6396
R_5	0.6752/0.1459	2.0008/1.0246

C. New local boundary condition

The method proposed here is also a correction of the bounce back rule but, in contrast to the BFL approach, it is not link based. To describe the algorithm at a boundary node $\boldsymbol{x}_j$ we first introduce the matrix K_{ik},

$$K_{ik} = -\frac{3}{2}(3 - 6q_{ji}) f_i^* \left((\boldsymbol{c}_i \cdot \boldsymbol{c}_k)^2 - |\boldsymbol{c}_i|^2/3 - c_{i\alpha}^2 \left(|\boldsymbol{c}_k|^2 - \frac{d}{3} \right) \right)$$

which has to be evaluated for all pairs (i,k) of velocity indices at each boundary node. The coordinate index $\alpha \in \{1,\ldots,d\}$ can be chosen arbitrarily (we take $\alpha = d$). After selecting the parameter $\theta \in [0,1]$ which controls the amount of implicitness of the resulting method POP_θ, we set up the matrix $L_{ik} = \delta_{ik} + \theta K_{ik}$ but this time only for indices i,k which refer to incoming velocity directions at $\boldsymbol{x}_j$ (these indices are collected in the set V). Then, the inverse of the small matrix L_{ik} has to be determined. This requirement restricts θ not to be the negative inverse of an eigenvalue of the submatrix K_{ik}, $i,k \in V$. Otherwise, θ can be chosen arbitrarily and, in principle, a different choice is possible for every boundary node. Note that $\theta = 0$ turns L_{ik} into the identity matrix so that no inversion is necessary in that case (explicit method).

For nonmoving boundaries, all steps described so far need to be worked out only once during preprocessing. For the time iteration, the following steps are required. Denoting the right hand side of Eq. (8) again by $f_{i^*}^b(n,\boldsymbol{j})$ and observing that, after the lattice Boltzmann transport step, the densities for the outgoing directions $\hat{f}_k(n+1,\boldsymbol{j})$, $k \notin V$ are available, we compute for all $i \in V$

$$r_i(n,\boldsymbol{j}) = \sum_{k \notin V} \theta K_{ik} \hat{f}_k(n+1,\boldsymbol{j}) + \sum_k K_{ik} \hat{\sigma}_k(n,\boldsymbol{j}),$$

$$\hat{\sigma}_k = \hat{f}_k^c - \hat{g}_k - (1-\theta)\hat{f}_k.$$

Then we determine the required incoming distributions $\hat{f}_k(n+1,\boldsymbol{j})$, $k \in V$ by solving the linear system (here the inverse of the small matrix L_{ik} is needed)

$$\sum_{k \in V} L_{ik} \hat{f}_k(n+1,\boldsymbol{j}) = f_{i^*}^b(n,\boldsymbol{j}) - r_i(n,\boldsymbol{j}), \quad i \in V. \qquad (10)$$

Note that Eq. (10) reduces to the bounce back rule if $q_{ji} = 1/2$ because $K_{ik} = 0$ in that case so that also $r_i = 0$ and $L_{ik} = \delta_{ik}$.

We remark that the boundary condition is also applicable in the case of lattice Boltzmann methods with more general

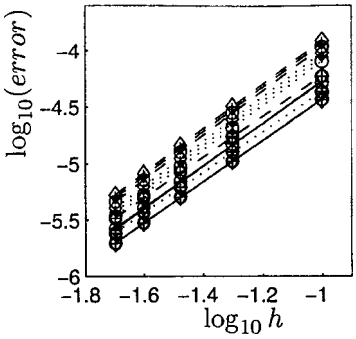

FIG. 8. Logarithmic velocity error versus $\log_{10}h$ in the case of a 3D stationary linear flow. Solid, dashed, and dotted lines correspond to the models D3Q15, D3Q19, and D3Q27, respectively. ○, ◇, and + denote the results of explicit $\theta=0$, mixed $\theta=0.7$, and implicit $\theta=1$ schemes.

collision operators (e.g., multirelaxation time models). In fact, the specific collision model enters only indirectly into Eq. (10) through $f_{i^*}^b(n,\boldsymbol{j})$ which contains the post-collisional state (4).

IV. NUMERICAL TEST OF THE METHOD

In this section the proposed one-point method (10) is carefully tested in two respects which are of practical interest: the accuracy and the applicability in the case of curved boundaries. Moreover, we also report some experiments concerning mass conservation. Generally it is observed that implicit versions of the algorithm are more stable than the explicit one.

Three models which have analytical solutions are used here, a stationary linear flow, a nonstationary linear flow, and a 2D circular flow. All models are used to assess the accuracy and convergence of the proposed boundary treatment. The 2D circular flow has been chosen to test the applicability in the case of curved boundaries. The grid size is $h\in\{1/10,\ 1/20,\ 1/30,\ 1/40,\ 1/50\}$. The parameter θ is chosen from the set $\{0.0,\ 0.7,\ 1.0\}$ and the relaxation time is again set corresponding to $\nu=0.1$ in Eq. (1).

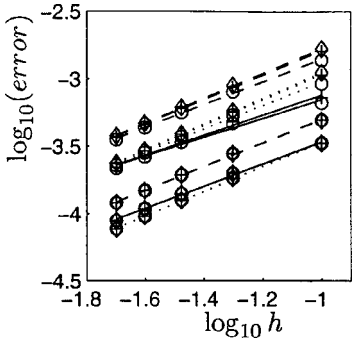

FIG. 9. Logarithmic pressure error versus $\log_{10}h$ in the case of a 3D stationary linear flow. Solid, dashed, and dotted lines correspond to the models D3Q15, D3Q19, and D3Q27, respectively. ○, ◇, and + denote the results of explicit $\theta=0$, mixed $\theta=0.7$, and implicit $\theta=1$ schemes.

TABLE II. Numerical convergence orders of velocity and pressure for a stationary linear flow.

θ	q	D3Q15	D3Q19	D3Q27
0	0	1.846/0.698	1.952/0.847	1.957/0.872
0	0.4	1.814/0.819	1.791/0.875	1.782/0.912
0.7	0	1.875/0.744	1.953/0.909	1.959/0.940
0.7	0.4	1.811/0.818	1.794/0.875	1.786/0.911
1	0	L irregular	1.953/0.926	1.959/0.960
1	0.4	1.807/0.818	1.795/0.874	1.787/0.911

A. A family of 3D linear flows

The velocity and pressure of the 3D linear flows are described as

$$\boldsymbol{u}(t,\boldsymbol{x})=\alpha(t)A\boldsymbol{x},\quad p(t,\boldsymbol{x})=-\frac{1}{2}\alpha(t)\boldsymbol{x}^{\mathrm{T}}A^2\boldsymbol{x},$$

where

$$A=\frac{1}{200}\begin{pmatrix}10 & 5 & 2\\ 5 & 10 & 4\\ 2 & 4 & -20\end{pmatrix},$$

$\alpha(t)$ is a function of time t. It is easy to validate that the fields $\boldsymbol{u}$ and p satisfy the Navier-Stokes equations (1) in the entire space with a body force defined by

$$\boldsymbol{G}(t,\boldsymbol{x})=\alpha'(t)A\boldsymbol{x}+\alpha(t)(\alpha(t)-1)A^2\boldsymbol{x}. \tag{11}$$

The numerical tests for this kind of linear flow will be restricted to the unit cube $\Omega=[0,1]^3$ as computational region and the known values of $\boldsymbol{u}$ are prescribed at the boundary $\partial\Omega$. Because of the simple geometrical structure, the parameters q_{ji} are determined by three values q_i, where $h(q_1,q_2,q_3)$ are the coordinates of the node closest to the origin. In this paper q_i are chosen equally and the common value is denoted q, with values $\{0,0.4\}$ in our test cases.

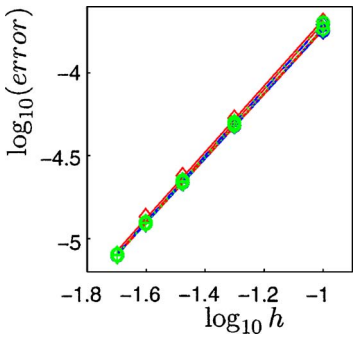

FIG. 10. (Color online) Logarithmic velocity error versus $\log_{10}h$ for nonstationary linear flow. Solid, dashed, and dotted lines correspond to the models D3Q15, D3Q19, and D3Q27, respectively. ○, ◇, and + denote the results of explicit $\theta=0$, mixed $\theta=0.7$, and implicit $\theta=1$ schemes.

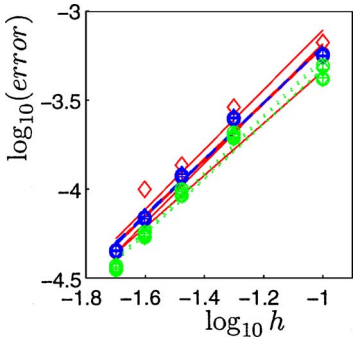

FIG. 11. (Color online) Logarithmic pressure error versus $\log_{10}h$ for nonstationary linear flow. Solid, dashed, and dotted lines correspond to the models D3Q15, D3Q19, and D3Q27, respectively. ○, ◇, and + denote the results of explicit $\theta=0$, mixed $\theta=0.7$, and implicit $\theta=1$ schemes.

1. Stationary linear flow

Under the condition that $\alpha(t)$ is a constant, for example $\alpha(t)=1$, the flow is stationary and the corresponding body force $\boldsymbol{G}$ vanishes. We compute the error of the various methods by comparing the approximate solution with the exact one in the maximum norm over $[0,T]\times\Omega$ with $T=1$. Figure 8 is the logarithmic error plot for velocity. The corresponding results for the pressure is given in Fig. 9. Table II displays all the slopes of the error lines in Figs. 8 and 9. Those of the velocity are around 1.9 and the pressure curves show slopes around 0.9.

These numerical results demonstrate that our boundary scheme produces a second order accurate velocity and at least a first order accurate pressure, which supports the analytical considerations in Ref. [18]. Observing the error size in Figs. 8 and 9, both show that all variants of the method have a very similar performance, and none of the models D3Q15, D3Q19, and D3Q27 produce prominently smaller errors than the others.

2. Nonstationary linear flow

In this case $\alpha(t)$ varies with time t, for example $\alpha(t)=t^2$. This flow is initially at rest and driven by an increasingly stronger body force. The termination time is set to $T=0.1$.

Figures 10 and 11 show the error plots of velocity and pressure versus the grid size. While the error in velocity is essentially identical for all the methods, the D3Q27 model gives slightly better results than the other two models. Table III displays all the slopes of the error lines in Figs. 10 and 11.

TABLE III. Numerical convergence orders for velocity and pressure in the case of a nonstationary linear 3D flow.

θ	q	D3Q15	D3Q19	D3Q27
0	0	1.991/1.589	1.989/1.585	2.000/1.619
0	0.4	1.940/1.589	1.940/1.593	1.957/1.546
0.7	0	1.994/1.681	1.985/1.570	1.992/1.606
0.7	0.4	1.940/1.681	1.941/1.593	1.957/1.549
1	0	L irregular	1.991/1.564	1.999/1.599
1	0.4	1.958/1.460	1.941/1.594	1.957/1.550

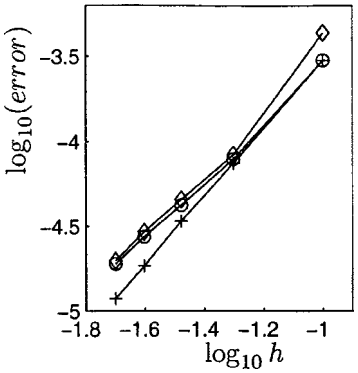

FIG. 12. Logarithmic velocity error versus $\log_{10}h$ for a 2D circular flow. ○, ◇, and + denote the results of implicit $\theta=1$, mixed $\theta=0.7$ schemes, and the BFL method, the best fit slopes are 1.701, 1.889, and 2.005, respectively.

These numerical results demonstrate that our boundary scheme produces a second order accurate velocity and at least a first order accurate pressure.

B. Flow inside a circular cylinder

We consider the so-called impulsively started circular flow which has been described, for example, in Ref. [5]. The radius of the infinitely long circular cylinder is taken as $R=1$. Using dimensional reduction, we are led to a 2D Navier-Stokes problem posed on the unit disk Ω. The exact angular velocity $u_\theta(t,r)$ is given in Ref. [5] based on an infinite series involving Bessel functions of order 0 and 1. In addition, the pressure $p(t,r)$ is centrally symmetric (the origin is the center) and appears as an integration of the angular velocity,

$$p(t,r)=\int_0^r \frac{1}{r}u_\theta^2\,dr.$$

Obtaining the exact velocity and pressure for this model, obviously involves numerical integration and an approxima-

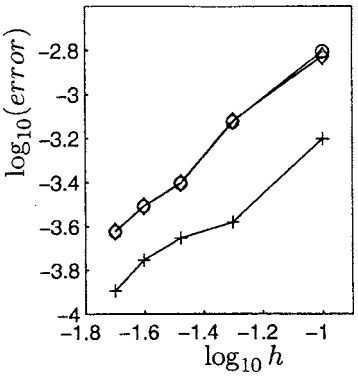

FIG. 13. Logarithmic pressure error versus $\log_{10}h$ for a 2D circular flow. ○, ◇, and + denote the results of implicit $\theta=1$, mixed $\theta=0.7$ schemes, and the BFL method, the best fit slopes are 1.189, 1.159, and 0.928, respectively.

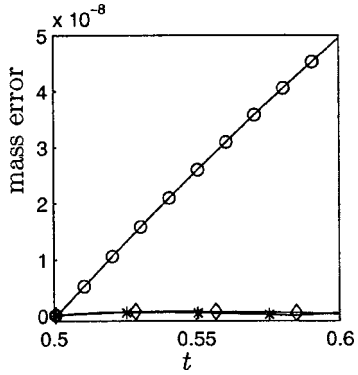

FIG. 14. Absolute difference between total mass and its initial value over time. Implicit scheme $\theta=1$ (∗), mixed scheme $\theta=0.7$ (◇), BFL scheme (○).

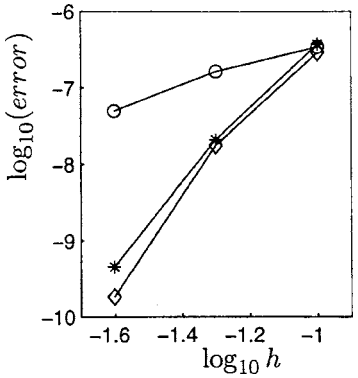

FIG. 15. Grid dependence of the total mass error. Implicit scheme $\theta=1$ (∗), mixed scheme $\theta=0.7$ (◇), BFL scheme (○).

tion of u_θ. For the former we take a second order accurate integration method, the latter is calculated by taking the sum of the first 50 terms in the series only.

Since the boundary curve now is a circle, the distance between boundary nodes and boundary varies arbitrarily between 0 and h, i.e., $q_{ji} \in [0,1)$. In order to concentrate on the errors caused by the proposed boundary schemes, we want to avoid initial errors due to the nonsmooth abrupt start of the cylinder (abrupt start means $\boldsymbol{u}=0$ in the domain but $u_\theta=1$ along the boundary). This can be achieved, for example, by starting at time $t=0.5$ instead of $t=0$. To check the accuracy, the termination time is set to $T=0.6$. In the explicit case $\theta=0$, the scheme is not stable. Figures 12 and 13 show the error of velocity and pressure versus the grid size for implicit and mixed boundary conditions and the BFL method. It demonstrates the second order accuracy of velocity and first order accuracy of pressure. More importantly, these numerical results confirm that the proposed boundary scheme works very well in connection with curved boundaries if $\theta > 0$.

Since the total mass in Ω is a conserved quantity, it is desirable to check to which extent the proposed boundary conditions guarantee conservation. To this end, we calculate and study the deviation of the arithmetic density average (which is a natural approximation of the total mass) from its initial value for various grids.

Figure 14 shows the deviation of total mass from its initial value over time for the implict, mixed, and BFL scheme on lattices with grid size $h=1/40$. We observe that the mass conservation is violated slightly and, in this case, our method performs somewhat better than the BFL rule. With decreasing grid size, the maximal deviation of the total mass also decreases as depicted in Fig. 15 where the mass error is computed on grids with $h \in \{1/10,\ 1/20,\ 1/40\}$.

V. CONCLUSION

We have introduced a family of LB boundary algorithms to solve Dirichlet boundary value problems for the instationary, incompressible Navier-Stokes equation. The accuracy and the stability behavior of our method is similar to the BFL schemes presented in Ref. [5]. However, our scheme has the advantage of being completely local which also distinguishes it from the multireflection condition in Ref. [9]. In contrast to the local condition introduced in Ref. [8], our method does not require derivatives of the Dirichlet data along the boundary and it works exactly in the same way for curved and flat boundaries. However, the two methods are similar in the sense that for every boundary node a small linear system has to be set up and inverted in a preprocessing step. We have shown that our method can also be used to fix the degradation of interpolation based schemes which results from boundary points with missing nodes in incoming directions.

ACKNOWLEDGMENT

The support by the Deutsche Forschungsgemeinschaft (DFG) through Grant No. JU440/1-3 is gratefully acknowledged.

[1] H. Chen, S. Chen, and W. H. Matthaeus, Phys. Rev. A **45**, R5339 (1992).
[2] Y. H. Qian, D. d'Humières, and P. Lallemand, Europhys. Lett. **17**, 479 (1992).
[3] R. Cornubert, D. d'Humières, and D. Levermore, Physica D **47**, 241 (1991).
[4] S. Chen, D. Martìnez, and R. Mei, Phys. Fluids **8**, 2527 (1996).
[5] M. Bouzidi, M. Firdaouss, and P. Lallemand, Phys. Fluids **13**, 3452 (2001).
[6] O. Filippova and D. Hänel, Int. J. Mod. Phys. C **9**, 1271 (1998).
[7] O. Filippova and D. Hänel, J. Comput. Phys. **147**, 219 (1998).
[8] I. Ginzbourg and D. d'Humières, J. Stat. Phys. **84**, 927 (1996).
[9] I. Ginzburg and D. d'Humières, Phys. Rev. E **68**, 066614, (2003).
[10] T. Inamuro, M. Yoshina, and F. Ogino, Phys. Fluids **7**, 2928 (1995).

[11] R. S. Maier, R. S. Bernard, and D. W. Grunau, Phys. Fluids **8**, 1788 (1996).
[12] R. Mei, L.-S. Luo, and W. Shyy, J. Comput. Phys. **155**, 307 (1999).
[13] R. Mei, W. Shyy, D. Yu, and L.-S. Luo, J. Comput. Phys. **161**, 680 (2000).
[14] D. R. Noble, S. Chen, J. G. Georgiadis, and R. O. Buckius, Phys. Fluids **7**, 203 (1995).
[15] D. R. Noble, J. G. Georgiadis, and R. O. Buckius, J. Stat. Phys. **81**, 17 (1995).
[16] P. A. Skordos, Phys. Rev. E **48**, 4823 (1993).
[17] Q. Zou and X. He, Phys. Fluids **9**, 1591 (1997).
[18] M. Junk and Z. Yang, J. Stat. Phys. (to be published).
[19] X. He and L.-S. Luo, J. Stat. Phys. **88**, 927 (1997).
[20] D. Yu, R. Mei, W. Shyy, and L.-S. Luo, J. Comput. Phys. **161**, 680 (2000).
[21] M. Junk, A. Klar, and L.-S. Luo, J. Comput. Phys. (to be published).