HAL
open science

Multigrid dual-time-stepping lattice Boltzmann method

Simon Gsell, Umberto d'Ortona, Julien Favier

To cite this version:

Simon Gsell, Umberto d'Ortona, Julien Favier. Multigrid dual-time-stepping lattice Boltzmann method. Physical Review E , 2020, 101 (2), 10.1103/PhysRevE.101.023309 . hal-02573156

HAL Id: hal-02573156

https://hal.science/hal-02573156v1

Submitted on 14 May 2020

HAL is a multi-disciplinary open access archive for the deposit and dissemination of scientific research documents, whether they are published or not. The documents may come from teaching and research institutions in France or abroad, or from public or private research centers.

L'archive ouverte pluridisciplinaire **HAL**, est destinée au dépôt et à la diffusion de documents scientifiques de niveau recherche, publiés ou non, émanant des établissements d'enseignement et de recherche français ou étrangers, des laboratoires publics ou privés.

Multigrid dual-time-stepping lattice Boltzmann method

Simon Gsell,* Umberto D'Ortona, and Julien Favier
Aix-Marseille Univ., CNRS, Centrale Marseille, M2P2, Marseille, France

(Received 11 September 2019; accepted 28 January 2020; published 18 February 2020)

The lattice Boltzmann method often involves small numerical time steps due to the acoustic scaling (i.e., scaling between time step and grid size) inherent to the method. In this work, a second-order dual-time-stepping lattice Boltzmann method is proposed in order to avoid any time-step restriction. The implementation of the dual time stepping is based on an external source in the lattice Boltzmann equation, related to the time derivatives of the macroscopic flow quantities. Each time step is treated as a pseudosteady problem. The convergence rate of the steady lattice Boltzmann solver is improved by implementing a multigrid method. The developed solver is based on a two-relaxation time model coupled to an immersed-boundary method. The reliability of the method is demonstrated for steady and unsteady laminar flows past a circular cylinder, either fixed or towed in the computational domain. In the steady-flow case, the multigrid method drastically increases the convergence rate of the lattice Boltzmann method. The dual-time-stepping method is able to accurately reproduce the unsteady flows. The physical time step can be freely adjusted; its effect on the simulation cost is linear, while its impact on the accuracy follows a second-order trend. Two major advantages arise from this feature. (i) Simulation speed-up can be achieved by increasing the time step while conserving a reasonable accuracy. A speed-up of 4 is achieved for the unsteady flow past a fixed cylinder, and higher speed-ups are expected for configurations involving slower flow variations. Significant additional speed-up can also be achieved by accelerating transients. (ii) The choice of the time step allows us to alter the range of simulated timescales. In particular, increasing the time step results in the filtering of undesired pressure waves induced by sharp geometries or rapid temporal variations, without altering the main flow dynamics. These features may be critical to improve the efficiency and range of applicability of the lattice Boltzmann method.

DOI: 10.1103/PhysRevE.101.023309

I. INTRODUCTION

The lattice Boltzmann method, issued from the discretization of the Boltzmann equation, has become a popular approach to simulate fluid flows [1,2]. In contrast to the Navier-Stokes equations, the Boltzmann equation statistically describes the dynamics of microscopic fluid particles. It is said that the flow is described at the mesoscopic scale. In the lattice Boltzmann method, fluid particle distribution functions, also called particle populations, are transported on a spatial lattice composed of nodes connected by a discretized velocity space. Despite the differences between the Navier-Stokes and lattice Boltzmann approaches, they have proved to similarly predict the flow behavior at the macroscopic scale [3–5]. In addition, the analogy between the discretized Boltzmann and Navier-Stokes equations can be formally shown by a multiscale Chapman-Enskog analysis [6].

As any explicit method, the lattice Boltzmann method involves restrictions on the numerical time step. These restrictions may, however, be more important than for other methods. In the case of explicit Navier-Stokes solvers, the time step is often set according to a Courant-Friedrichs-Lewy (CFL) stability condition, which can be expressed as [7]

$$\Delta t_{\mathrm{NS}} = \mathcal{C}\,\Delta n/U, \tag{1}$$

where Δt_{NS} is the time step, Δn is the grid size, U is the typical flow velocity, and $\mathcal{C}$ is the CFL number that must satisfy $\mathcal{C} \leqslant 1$. In constrast, the lattice Boltzmann time step Δt_{LB} must scale with the grid size through the acoustic scaling, namely $\Delta n/\Delta t_{\mathrm{LB}} = c$, where c is the lattice speed. In addition, the method relies on a low-velocity assumption, which can be expressed as $U/c = \kappa$, where κ is a small parameter. Substituting c by $\Delta n/\Delta t_{\mathrm{LB}}$, this condition is equivalent to a time-step restriction,

$$\Delta t_{\mathrm{LB}} = \kappa\,\Delta n/U. \tag{2}$$

As $\kappa \ll 1$, it appears that the lattice Boltzmann time step is generally smaller than a Navier-Stokes time step for a given grid size and a given flow velocity. In practice, the value $\kappa = 0.05$ is expected to be small enough to ensure numerical accuracy [2]. In this case, the lattice Boltzmann time step is 20 times smaller than a Navier-Stokes time step with $\mathcal{C} = 1$.

Time-step restrictions of explicit solvers can lead to significant computational costs when simulating flows involving small length scales (i.e., small value of Δn) and relatively large timescales, as in large-eddy simulations of wall-bounded turbulent flows or in many multiscale problems (e.g., porous media, bubble flows, flow over canopies, etc.). Moreover, due to the scaling between the time step and other numerical parameters, as in expressions (1) and (2), it is often difficult to vary physical parameters as the Reynolds number without altering the computational performance. Finally, time-step restrictions prevent the free control of the numerical time

*simon.gsell@univ-amu.fr

 ©2020 American Physical Society

accuracy and generally limit the degrees of freedom of the numerical setup. The present work aims at avoiding these restrictions through a dual-time-stepping (DTS) procedure.

Dual-time-stepping methods, first introduced for Navier-Stokes solvers [8,9], are implicit time-integration methods that avoid any time-step restriction. At each time step, the flow solution is computed by solving a pseudosteady problem, where the flow unsteadiness is taken into account as an external forcing in the steady flow equations. The pseudosteady solution is obtained by integrating the solution over a pseudotime. While the pseudo time step may be subjected to stability restriction, the physical one is only determined by the desired time accuracy of numerical simulations. It allows us to set the time step as a function of the expected flow timescale, regardless of the spatial discretization. This method remains to be extended to lattice Boltzmann solvers; this is addressed in this work.

The efficiency of a dual-time-stepping method is closely related to the performance of the pseudosteady flow solver. Therefore, Navier-Stokes dual-time-stepping methods are often coupled to convergence acceleration techniques as preconditionning [10–12] and multigrid [13,14]. Similar techniques have been developed for lattice Boltzmann methods [15–20]. In the present algorithm, a multigrid technique similar to that proposed by Mavriplis [19] for steady flows is employed to improve the efficiency of the dual-time-stepping algorithm. It is extended to a very generic numerical framework, built using a two-relaxation-time lattice Botzmann method coupled to an immersed-boundary method [21], thus allowing the simulation of a large variety of physical configurations involving moving and/or deformable solid bodies immersed in a fluid.

The present paper is organized as follows. The proposed numerical method is presented in Sec. II. The reliability of the method is analyzed in the cases of the steady and unsteady flows past a circular cylinder, either fixed or towed in the computational domain; these results are presented in Sec. III. The main conclusions of this work are summarized in Sec. IV.

II. NUMERICAL METHOD

The numerical method is described in the following. First, the pseudosteady formulation of the physical problem, based on the Navier-Stokes equations, is presented in Sec. II A. The resulting steady problem is solved using a lattice Boltzmann method, described in Sec. II B, coupled to an immersed-boundary method detailed in Sec. II C. In Sec. II D, the multigrid method employed to accelerate the steady solver is introduced. Finally, the overall algorithm is summarized in Sec. II E.

A. Dual-time-stepping method

In the following, the flow is assumed to be two dimensional and nearly incompressible. It is modeled by the Navier-Stokes equations,

$$\frac{\partial \rho}{\partial t} + \nabla \cdot (\rho \boldsymbol{U}) = 0, \tag{3a}$$

$$\frac{\partial (\rho \boldsymbol{U})}{\partial t} + \nabla \cdot (\rho \boldsymbol{U}\boldsymbol{U}) = -\nabla P + \mu \nabla^2 \boldsymbol{U}, \tag{3b}$$

where ρ, $\boldsymbol{U} = (u, v)^T$, P, and μ designate the fluid density, velocity, pressure, and viscosity, and t is the time. More generally, Eqs. (3) may be rewritten as

$$\frac{\partial \boldsymbol{q}}{\partial t} + \boldsymbol{N}(\boldsymbol{q}) = 0, \tag{4}$$

where $\boldsymbol{q}$ is the solution vector $(\rho, \rho u, \rho v)^T$ and $\boldsymbol{N}(\boldsymbol{q})$ designates the steady Navier-Stokes operator. The dual-time-stepping method is based on a pseudosteady formulation of (4),

$$\frac{\partial \boldsymbol{q}}{\partial t^*} + \boldsymbol{N}(\boldsymbol{q}) = \boldsymbol{S}(\boldsymbol{q}), \tag{5}$$

where t^* is a pseudotime variable and $\boldsymbol{S}(\boldsymbol{q}) = -\partial \boldsymbol{q}/\partial \boldsymbol{t}$. The solution of (4) at time t is found by integrating (5) over the pseudotime t^* until a steady solution is reached, i.e., $\partial \boldsymbol{q}/\partial t^* = 0$. In this steady-state problem, the flow-unsteadiness plays the role of an external forcing. Time and pseudotime spaces are discretized as $t = n\Delta t$ and $t^* = l\Delta t^*$, where n and l are integers designating the time and pseudotime iterations and Δt and Δt^* are the associated time steps. The time-discretized pseudosteady problem reads

$$\left.\frac{\partial \boldsymbol{q}}{\partial t^*}\right|^{l} + \boldsymbol{N}(\boldsymbol{q}^l) = \boldsymbol{S}(\boldsymbol{q}^l), \tag{6}$$

where $\boldsymbol{S}(\boldsymbol{q}^l)$ is computed implicitly as a function of time t through a second-order backward scheme [13,14],

$$\boldsymbol{S}(\boldsymbol{q}^l) = -\frac{3\boldsymbol{q}^l - 4\boldsymbol{q}^n + \boldsymbol{q}^{n-1}}{2\Delta t}. \tag{7}$$

After convergence, the new time-accurate flow solution is obtained by $\boldsymbol{q}^{n+1} \leftarrow \boldsymbol{q}^l$.

The pseudotime step Δt^* is only involved in the integration of (6). Depending on the employed numerical method, it may be subjected to stability or accuracy restrictions. On the other hand, the physical time step Δt, which only affects the computation of the time derivative of the solution vector (7), may be chosen without restrictions. This allows us to set Δt in accordance with the desired time accuracy, regardless of the spatial discretization. Several approaches may be considered to integrate (5). In the present work, a steady lattice Boltzmann solver is employed. It is described in the following.

B. Lattice Boltzmann method

Equation (5), which is equivalent to the Navier-Stokes equations with an external forcing, can be numerically solved using a lattice Boltzmann method [2]. At the mesoscopic level, the flow is described by the flow particle distribution function $f(\boldsymbol{x}, \boldsymbol{\xi}, t^*)$, which represents the density of flow particles with velocity $\boldsymbol{\xi}$ at location $\boldsymbol{x}$ and time t^*. Here only the pseudotime t^* is considered, following (5). The velocity space is discretized on a set of velocity vectors $\{\boldsymbol{c}_\alpha, \alpha = 0, \ldots, Q-1\}$. In the present work, which only focuses on two-dimensional physical configurations, a $D2Q9$ velocity set is used; the velocity space is discretized by nine velocities, namely

$$\boldsymbol{c}_0 = \boldsymbol{0} \qquad \boldsymbol{c}_1 = c\boldsymbol{e}_x,$$

$$\boldsymbol{c}_2 = c\boldsymbol{e}_y, \qquad \boldsymbol{c}_3 = -c\boldsymbol{e}_x,$$

$$\boldsymbol{c_4} = -c\boldsymbol{e_y}, \qquad \boldsymbol{c_5} = c(\boldsymbol{e_x} + \boldsymbol{e_y}),$$
$$\boldsymbol{c_6} = c(-\boldsymbol{e_x} + \boldsymbol{e_y}), \qquad \boldsymbol{c_7} = c(-\boldsymbol{e_x} - \boldsymbol{e_y}),$$
$$\boldsymbol{c_8} = c(\boldsymbol{e_x} - \boldsymbol{e_y}), \tag{8}$$

where c is the lattice speed and $\boldsymbol{e_x}$ and $\boldsymbol{e_y}$ are unit vectors in the x and y directions. The particle densities at velocities $\{\boldsymbol{c_\alpha}\}$ are represented by the discrete-velocity distribution functions $\{f_\alpha(\boldsymbol{x}, t^*)\}$, also called particle populations. The physical space is discretized using a uniform and Cartesian grid; the grid spacing is denoted by Δn. The time discretization ensures that particle populations are transported from a node to a neighboring one during one time step, namely $\Delta n/\Delta t^* = c$. This is the acoustic scaling mentioned in Sec. I. As commonly done in lattice Boltzmann formulations, all numerical quantities are normalized by c and Δt^*, so that $\Delta n = \Delta t^* = 1$. Using this normalization, the lattice Boltzmann equation reads

$$f_\alpha^{l+1}(\boldsymbol{x} + \boldsymbol{c_\alpha}) - f_\alpha^l(\boldsymbol{x}) = \Omega_\alpha^l(\boldsymbol{x}) + F_\alpha^l(\boldsymbol{x}), \tag{9}$$

where the superscript l relates to the pseudotime discretization, Ω_α designates the collision operator, and F_α accounts for external forcing, i.e., the mass and momentum sources related to the dual-time-stepping and immersed-boundary methods. The left-hand side of (9) describes the streaming step, during which the particle populations are transported from one node to a neighboring one; the right-hand side relates to the collision step. Equation (9) is explicit, so the streaming and collision steps can be treated separately.

A simple and commonly used collision model is the Bhatnagar-Gross-Krook (BGK) model [22]. In the present work, a two-relaxation-time (TRT) collision model [23,24] is employed to ensure the viscosity independence of the solver [21]. This model is based on a decomposition of populations into symmetric and antisymmetric parts, namely

$$f_\alpha^+ = \frac{f_\alpha + f_{\overline{\alpha}}}{2}, \quad f_\alpha^- = \frac{f_\alpha - f_{\overline{\alpha}}}{2}, \tag{10a}$$

$$f_\alpha^{(\mathrm{eq})+} = \frac{f_\alpha^{(\mathrm{eq})} + f_{\overline{\alpha}}^{(\mathrm{eq})}}{2}, \quad f_\alpha^{(\mathrm{eq})-} = \frac{f_\alpha^{(\mathrm{eq})} - f_{\overline{\alpha}}^{(\mathrm{eq})}}{2}, \tag{10b}$$

where the index $\overline{\alpha}$ is defined such that $\boldsymbol{c_{\overline{\alpha}}} = -\boldsymbol{c_\alpha}$ and $\{f_\alpha^{(\mathrm{eq})}\}$ are the equilibrium functions given by

$$f_\alpha^{(\mathrm{eq})} = w_\alpha \rho \left[1 + \frac{\boldsymbol{U}\cdot\boldsymbol{c_\alpha}}{c_s^2} + \frac{(\boldsymbol{U}\cdot\boldsymbol{c_\alpha})^2}{2c_s^4} - \frac{\boldsymbol{U}\cdot\boldsymbol{U}}{2c_s^2}\right]. \tag{11}$$

The weights $\{w_\alpha\}$ are specific to the velocity set. In the present case ($D2Q9$ velocity set), $w_0 = 4/9$, $w_1 = w_2 = w_3 = w_4 = 1/9$, and $w_5 = w_6 = w_7 = w_8 = 1/36$. The TRT collision is performed by relaxing symmetric and antisymmetric parts separately,

$$\Omega_\alpha^l = -\frac{1}{\tau}\left[f_\alpha^{l+} - f_\alpha^{l(\mathrm{eq})+}\right] - \frac{1}{\hat{\tau}}\left[f_\alpha^{l-} - f_\alpha^{l(\mathrm{eq})-}\right], \tag{12}$$

where τ is the standard relaxation time, which relates to the fluid kinematic viscosity through $\nu = c_s^2(\tau - \frac{1}{2})$, with c_s denoting the sound speed equal to $1/\sqrt{3}$ using the present normalization. The second relaxation time $\hat{\tau}$ is introduced to relax antisymmetric populations. In contrast to τ, $\hat{\tau}$ does not directly relate to macroscopic fluid properties; it is a free numerical parameter. In practice, $\hat{\tau}$ is often set through the parameter Λ relating both relaxations times [24],

$$\Lambda = \left(\tau - \tfrac{1}{2}\right)\left(\hat{\tau} - \tfrac{1}{2}\right). \tag{13}$$

In the following, Λ is set to $1/4$. The source term F_α involved in the lattice Boltzmann equation (9) is expressed as

$$F_\alpha^l = \left(1 - \frac{1}{2\tau}\right)S_\alpha^{l+} + \left(1 - \frac{1}{2\hat{\tau}}\right)S_\alpha^{l-}, \tag{14}$$

where $S_\alpha^+ = (S_\alpha + S_{\overline{\alpha}})/2$ and $S_\alpha^- = (S_\alpha - S_{\overline{\alpha}})/2$ are the symmetric and antisymmetric parts of S_α and

$$S_\alpha^l = w_\alpha \left\{ S_\rho^l + \left[\frac{\boldsymbol{c_\alpha} - \boldsymbol{U^l}}{c_s^2} + \frac{(\boldsymbol{c_\alpha}\cdot\boldsymbol{U^l})\boldsymbol{c_\alpha}}{c_s^4}\right]\cdot\left(\boldsymbol{S_{\rho U}^l} + \boldsymbol{B^l}\right)\right\}. \tag{15}$$

In Eq. (15), $S_\rho = -\partial\rho/\partial t$ and $\boldsymbol{S_{\rho U}} = -\partial(\rho\boldsymbol{U})/\partial t$ are the mass and momentum sources related to the dual-time-stepping method, and $\boldsymbol{B}$ is the momentum source issued from the immersed-boundary method (see Sec. II C). In the absence of local mass variation ($\partial\rho/\partial t = 0$), Eq. (15) reduces to the forcing scheme proposed by Guo [25]. The additional term related to S_ρ in (15) is a simple local mass source, since $\sum_\alpha w_\alpha S_\rho = S_\rho$ and $\sum_\alpha w_\alpha \boldsymbol{c_\alpha} S_\rho = \boldsymbol{0}$.

The macroscopic flow quantities are moments of the particle populations in the velocity space. In particular, the flow momentum and density are written as follows [2]:

$$(\rho\boldsymbol{U})^l = \sum_{\alpha=0}^{8} f_\alpha^l \boldsymbol{c_\alpha} + \frac{1}{2}(\boldsymbol{S_{\rho U}^l} + \boldsymbol{B^l}), \tag{16a}$$

$$\rho^l = \sum_{\alpha=0}^{8} f_\alpha^l + \frac{1}{2}S_\rho^l. \tag{16b}$$

Expressions (16) are implicit, as the source terms S_ρ^l and $\boldsymbol{S_{\rho U}^l}$ depend on ρ^l and $\boldsymbol{U^l}$. Following Eq. (7), these terms are expressed as

$$\boldsymbol{S_{\rho U}^l} = -\frac{3(\rho\boldsymbol{U})^l - 4(\rho\boldsymbol{U})^n + (\rho\boldsymbol{U})^{n-1}}{2\Delta t}, \tag{17a}$$

$$S_\rho^l = -\frac{3\rho^l - 4\rho^n + \rho^{n-1}}{2\Delta t}. \tag{17b}$$

Consequently, the macroscopic quantities can be computed as follows:

$$(\rho\boldsymbol{U})^l = \frac{\sum_{\alpha=0}^{8} f_\alpha^l \boldsymbol{c_\alpha} + \frac{1}{2}\boldsymbol{B^l} + \frac{(\rho\boldsymbol{U})^n}{\Delta t} - \frac{(\rho\boldsymbol{U})^{n-1}}{4\Delta t}}{1 + \frac{3}{4\Delta t}}, \tag{18}$$

$$\rho^l = \frac{\sum_{\alpha=0}^{8} f_\alpha^l + \frac{\rho^n}{\Delta t} - \frac{\rho^{n-1}}{4\Delta t}}{1 + \frac{3}{4\Delta t}}. \tag{19}$$

It should be emphasized that mass conservation is ensured in expression (19). The computation of the immersed-boundary forcing $\boldsymbol{B}$ is detailed in the next section.

C. Immersed-boundary method

The present numerical method is coupled to an immersed-boundary method that allows us to simulate the flow around arbitrary-shaped and moving solid bodies. The principle of the method is schematized in Fig. 1. A solid boundary Γ, discretized by a set of Lagrangian markers $\{\boldsymbol{X_k}\} = \{(X_k, Y_k)\}$,

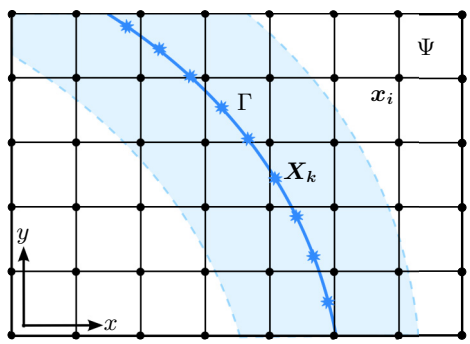

FIG. 1. Schematic view of the immersed-boundary method. A boundary Γ, discretized by Lagrangian markers $\{\boldsymbol{X_k}\}$, is immersed in the fluid domain Ψ, discretized by the lattice nodes $\{\boldsymbol{x_i}\}$. The shaded blue area represents the region where the immersed-boundary forcing is applied, determined by the radius of the employed kernel function (24). Reproduced from Ref. [21].

is immersed in the flow domain Ψ discretized by the Eulerian lattice nodes $\{\boldsymbol{x_i}\} = \{(x_i, y_i)\}$. The no-slip condition on the solid boundary is enforced through a body force $\boldsymbol{B}$ applied on the neighboring lattice nodes. The computation of $\boldsymbol{B}$ is based on a correction of the flow momentum interpolated on the solid boundary. This correction is expressed as a body force $\boldsymbol{B}^*$ on the Lagrangian markers. Based on expression (16), the relation between the body force $\boldsymbol{B}^*$ and the prescribed wall velocity $\boldsymbol{U_b}$ on a boundary marker $\boldsymbol{X_k}$ is expressed as

$$\mathcal{I}[\rho](\boldsymbol{X_k})\boldsymbol{U_b}(\boldsymbol{X_k}) = \mathcal{I}\left[\sum_{\alpha=0}^{8} f_\alpha \boldsymbol{c_\alpha} + \frac{1}{2}\boldsymbol{S}_{\rho U}\right](\boldsymbol{X_k}) + \frac{1}{2}\boldsymbol{B}^*(\boldsymbol{X_k}), \tag{20}$$

where $\mathcal{I}$ denotes the interpolation operator, detailed hereafter. Consequently, a direct computation of the momentum correction $\boldsymbol{B}$ may be expressed as

$$\boldsymbol{B}^* = 2\left(\mathcal{I}[\rho]\boldsymbol{U_b} - \mathcal{I}\left[\sum_{\alpha=0}^{8} f_\alpha \boldsymbol{c_\alpha} + \frac{1}{2}\boldsymbol{S}_{\rho U}\right]\right). \tag{21}$$

However, as analyzed in Ref. [21], expression (21) does not allow to ensure the viscosity independence of the computations, resulting in velocity-slip errors [26] or spurious oscillations when the relaxation time is varied. To avoid these effects, the IB force should be rescaled according to the relaxation time, namely

$$\boldsymbol{B}^* = \frac{2\lambda}{1+\kappa(\lambda-1)}\left(\mathcal{I}[\rho]\boldsymbol{U_b} - \mathcal{I}\left[\sum_{\alpha=0}^{8} f_\alpha \boldsymbol{c_\alpha} + \frac{1}{2}\boldsymbol{S}_{\rho U}\right]\right), \tag{22}$$

where κ is an analytically derived quantity that depends on the interpolation kernel function and $\lambda = 2\tau - 1$. It should be emphasized that expression (22) reduces to expression (21) when $\tau = 1$.

The interpolation of a physical quantity ϕ is performed using a discrete Dirac function δ, namely

$$\mathcal{I}[\phi](\boldsymbol{X}_k) = \sum_{\boldsymbol{x_i}\in\Psi} \phi(\boldsymbol{x_i})\delta(\boldsymbol{x_i} - \boldsymbol{X_k})\Delta S_i, \tag{23}$$

where $\Delta S_i = \Delta x \Delta y = 1$ is the cell surface. The kernel function is expressed as $\delta(\boldsymbol{x}) = \hat{\delta}(x)\hat{\delta}(y)$, with [27]

$$\hat{\delta}(r) = \begin{cases} \frac{1}{2d}\left[1 + \cos\left(\frac{\pi r}{d}\right)\right], & |r| \leqslant d, \\ 0, & |r| > d, \end{cases} \tag{24}$$

and d the radius of the kernel function, set to 3/2 in the following. The quantity κ, involved in expression (22), is equal to $\kappa = 3/(4d)$ using the present kernel function [21].

At each time step, the body force on the immersed boundary is computed on the basis of expression (22). The force is then spread to the lattice nodes, using the same kernel function as that employed for the interpolation. The spread force on a lattice node $\boldsymbol{x_i}$ reads

$$\boldsymbol{B}(\boldsymbol{x}_i) = \mathcal{S}[\boldsymbol{B}^*](\boldsymbol{x}_i) = \sum_{\boldsymbol{X_k}\in\Gamma} \boldsymbol{B}^*(\boldsymbol{X_k})\delta(\boldsymbol{x_i} - \boldsymbol{X_k})\Delta S_k, \tag{25}$$

where $\mathcal{S}$ is the spreading operator and ΔS_k is a local surface element; it is often expressed as $\Delta S_k = \Delta l_k \epsilon$, with Δl_k the local distance between two neighboring boundary markers and ϵ a boundary width generally set to unity—as done in the present work—or in some cases computed explicitly [28].

D. Multigrid method

The efficiency of the above-described lattice Boltzmann dual-time-stepping method is closely related to the convergence rate of the lattice Boltzmann method for steady-state problems. However, as discussed in Sec. III B, the explicit lattice Boltzmann scheme (9) may be poorly efficient for this type of problem. Therefore, a multigrid strategy, similar to that proposed by Mavriplis [19] but coupled for the first time to an immersed-boundary method, is employed in the following to accelerate the steady solver.

The multigrid method consists in computing corrections of the flow solution on coarse grids in order to accelerate the smoothing of low-frequency errors [29]. The numerical domain is discretized on a series of recursively coarsening grids. The finest grid corresponds to the lattice of reference. Consecutive grids are constructed so that the scale ratio between grids is equal to two, with the coarse-grid nodes coinciding with the fine-grid ones, as schematized in Fig. 2. Considering the populations f_α^i on the ith grid, the steady lattice Boltzmann operator $L_\alpha^i[f_\alpha^i(\boldsymbol{x})]$ can be expressed as

$$L_\alpha^i\left[f_\alpha^i(\boldsymbol{x})\right] = f_\alpha^i(\boldsymbol{x}) - f_\alpha^i\left(\boldsymbol{x} - \boldsymbol{c_\alpha^i}\right) - \Omega_\alpha^i\left(\boldsymbol{x} - \boldsymbol{c_\alpha^i}\right) - F_\alpha^i\left(\boldsymbol{x} - \boldsymbol{c_\alpha^i}\right). \tag{26}$$

Note that in the following the exponent i will be generally used to designate variables and operators on the ith grid level. Using this operator, the steady-state problem on each grid reads

$$L_\alpha^i\left[f_\alpha^i(\boldsymbol{x})\right] = D_\alpha^i. \tag{27}$$

The right-hand-side term in (27) is called the defect correction. It vanishes on the first grid, $D_\alpha^1 = 0$, i.e., the solution of (27) is the steady solution of the original physical problem (9). In contrast, grid levels $i \neq 1$ operate on a correction equation (see Ref. [19]) determined by the defect correction, which is computed during the transfer of the solution from a fine

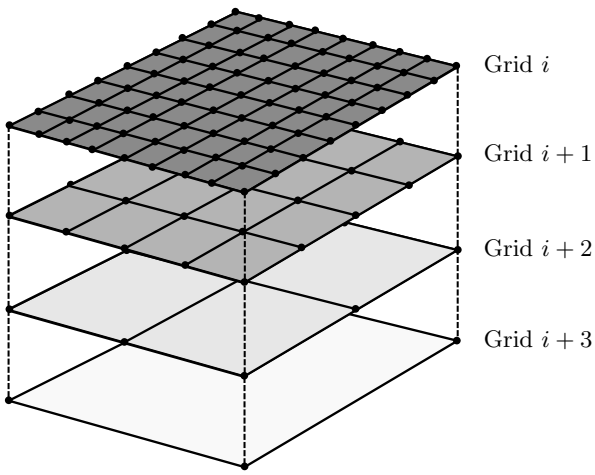

FIG. 2. Schematic view of three consecutive grid levels with a scale factor equal to two and coarse-grid nodes coinciding with the fine-grid ones.

grid to a coarser one, as detailed hereafter. Considering an approximate solution $\tilde{f}^i_\alpha(\boldsymbol{x})$, the residual is defined as

$$R^i_\alpha\big(\tilde{f}^i_\alpha\big) = L^i_\alpha\big(\tilde{f}^i_\alpha\big) - D^i_\alpha. \tag{28}$$

The residual tends to zero as $\tilde{f}^i_\alpha(\boldsymbol{x})$ converges to the exact solution of (27). The iteration procedure on the ith grid is composed of three steps:

(i) collision step:

$$\tilde{f}^i_\alpha(\boldsymbol{x})|^* = \tilde{f}^i_\alpha(\boldsymbol{x})|^l + \tilde{\Omega}^i_\alpha(\boldsymbol{x})|^l + \tilde{F}^i_\alpha(\boldsymbol{x})|^l, \tag{29}$$

(ii) propagation or streaming step:

$$\tilde{f}^i_\alpha(\boldsymbol{x})|^{**} = \tilde{f}^i_\alpha\big(\boldsymbol{x} - \boldsymbol{c}^i_\alpha\big)\big|^*, \tag{30}$$

(iii) relaxation step:

$$\tilde{f}^i_\alpha(\boldsymbol{x})|^{l+1} = \gamma\big[\tilde{f}^i_\alpha(\boldsymbol{x})|^{**} + D^i_\alpha\big] + (1-\gamma)\tilde{f}^i_\alpha(\boldsymbol{x})|^l, \tag{31}$$

where $\tilde{\Omega}^i_\alpha$ and $\tilde{F}^i_\alpha$ are the collision and source terms associated with the approximate solution $\tilde{f}^i_\alpha$ and γ is a relaxation parameter, set to 0.6 in the following. The collision and streaming steps are similar to those performed in a standard lattice Boltzmann method. In addition to expression (30), the streaming step is accompanied by the treatment of boundary conditions, which is not altered by the multigrid method. This iteration procedure, called smoothing sweep, is repeated until the problem is transferred to a coarser grid level $i+1$. The transfer consists in the computation of the defect correction [19],

$$D^{i+1}_\alpha = L^{i+1}_\alpha\big(\hat{I}^{i+1}_i \tilde{f}^i_\alpha\big) - 2I^{i+1}_i R^i_\alpha\big(\tilde{f}^i_\alpha\big), \tag{32}$$

where $\hat{I}^{i+1}_i$ and I^{i+1}_i denote the restriction operators, which respectively transfer solution variables and residuals from the fine grid to the coarse one. In this work, $\hat{I}^{i+1}_i$ is defined as a pointwise injection and I^{i+1}_i is a bilinear interpolation operator [19]. After several sweeps on grid $i+1$, the coarse-grid solution f^{i+1}_α may be used to compute the coarse-grid correction on the finer grid,

$$C^i_\alpha = I^i_{i+1}\big(f^{i+1}_\alpha - \hat{I}^{i+1}_i f^i_\alpha\big), \tag{33}$$

where I^i_{i+1} denotes the prolongation operator, which is a bilinear interpolation from the coarse grid to the fine grid. The fine-grid solution is updated accordingly, $f^i_\alpha \leftarrow f^i_\alpha + C^i_\alpha$, and additional fine-grid sweeps may be performed to smooth the solution.

The convergence of the coarse grid problem can also be improved by performing coarse-grid corrections on an additional coarse-grid level. This nested procedure is the main principle of the multigrid method. Several approaches may be considered to perform a multigrid cycle, i.e., a series of sweeps on different grid levels leading to the computation of the coarse-grid correction of the finest grid. A simple multigrid cycle, called the V cycle, is described in Fig. 3(a). A four-grid method is considered in this example. Steps 1 to 3 are restriction steps. On each grid level, a number of smoothing sweeps are performed and the resulting defect correction D^i_α is transferred to the coarser grid level. At steps 2, 3, and 4, the flow solution f^i_α is initialized by a restriction of the finer-grid solution, namely $f^i_\alpha = \hat{I}^i_{i-1} f^{i-1}_\alpha$. At step 4, a series of sweeps is performed on the coarsest grid level. Then the coarse-grid corrections are recursively transferred up to the finest grid level (steps 4 to 6). A series of final sweeps is performed on each grid level to smooth the solution after correction. Finally, final sweeps are performed on grid 1 before updating the flow solution (step 7). It should be noted that the defect corrections, computed during the restriction steps, remain unchanged during the prolongation steps. Other multigrid cycles have been proposed in prior works; they are based on the same principle as the V cycle but involve additional transfers between coarse-grid levels in order to improve the convergence of the correction problem. In particular, a W cycle is employed in the present work; it is schematized in Fig. 3(b).

On each grid level, the steady-state problem (27) is connected to the forcing term F^i_α, which is composed of two

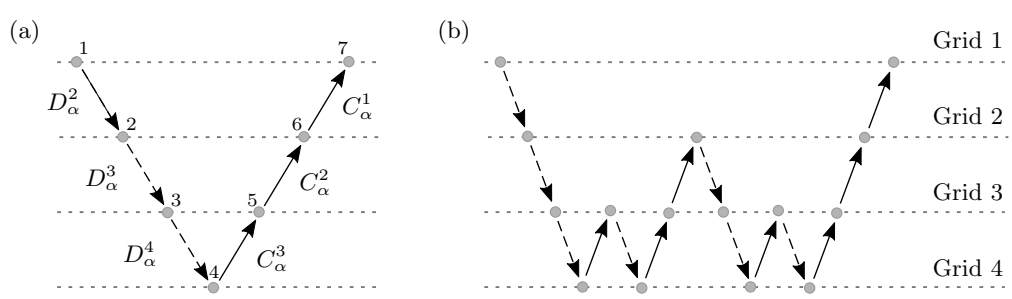

FIG. 3. Schematic view of four-grid (a) V and (b) W cycles. Dashed and solid arrows indicate restriction and prolongation steps.

TABLE I. Summary of the multigrid dual-time-stepping algorithm.

1. Fine-grid solution $f_\alpha^1|^n$ and associated macroscopic quantities are know at time step n
2. Update position and velocity of the immersed-boundary markers
3. Initialize $f_\alpha^1|^{l=0} = f_\alpha^1|^n$, compute macroscopic quantities (16)
4. Loop on pseudo time steps l
 4.1. Loop on grid levels i (multigrid cycle)
  4.1.1. Loop on grid sweeps
   4.1.1.1. Collision (29), streaming (30), and boundary conditions
   4.1.1.2. Relaxation (31)
   4.1.1.3. If $i = 1$, update the IB forcing (25)
   4.1.1.4. Compute macroscopic quantities (18) and (19)
   4.1.1.5. Update dual-time-stepping source terms (17)
  4.1.2. If necessary, perform restriction (32) or prolongation (33)
 4.3. New $f_\alpha^1|^{l+1}$
 4.4. Compute the fine-grid residual $\mathcal{R}$ (34)
 4.5. Exit if $\mathcal{R} < \mathcal{R}_c$ or if $l > l_c$
5. New fine-grid time-accurate solution $f_\alpha^1|^{n+1} = f_\alpha^1|^l$

contributions [see Eq. (15)]: the dual-time-stepping source term and the immersed-boundary source term. The dual-time-stepping forcing is updated at each smoothing sweep, following expressions (18) and (19), on every grid levels. During the restriction steps, it is initialized on coarse-grid levels through a restriction of the finer-grid solution. In contrast, the IB forcing is only updated during the finest-grid sweeps, and it is transferred to the coarse grids through simple pointwise injections. As shown in the following, this procedure is suitable for the coupling of the immersed-boundary and multigrid methods.

Setting the number of grids as well as the number of sweeps performed on each level of the multigrid cycle may be the object of a detailed parametric study in order to optimize the convergence rate of the method. Such optimization, which may depend on the considered physical configuration, is left for future works. These parameters are thus fixed in the following. First, the number of grid levels is set to four. It is generally recommended to perform more sweeps on the coarse grid levels than on the fine ones. Here the number of sweeps performed on grids 1, 2, 3, and 4 are equal to 1, 2, 3, and 4, respectively. It should be noted that these sweeps are performed during the restriction and prolongations phases. Therefore, the total number of sweeps performed on grids 1, 2, 3, and 4, using the V cycle described in Fig. 3(a), is equal to 2, 4, 6, and 8, respectively.

E. Algorithm and implementation

The time-marching algorithm is summarized in Table I. Compared to a standard lattice Boltzmann solver, the new steps are mainly the relaxation steps, the restriction and prolongation steps, and the steps related to the update of the dual-time-stepping source terms. Otherwise, collision, streaming, and immersed boundary steps are very similar to those performed in an explicit lattice Boltzmann solver, allowing us to integrate routines issued from an existing code. In practice, the pseudotime loop can be controlled by setting the maximum number of inner iterations l_c or using a convergence threshold based on the L_2 norm of the fine-grid residuals,

$$\mathcal{R} = \frac{1}{N}\sqrt{\sum_{i=1}^{N}\sum_{\alpha=0}^{8} R_{\alpha,i}^1}, \tag{34}$$

where N is the number of lattice nodes and $R_{\alpha,i}^1$ is the fine-grid residual (28) on the ith node.

It should be mentioned that the immersed-boundary update, corresponding to step 2 in Table I, may be performed at different instants, depending on body motion features. The algorithm proposed in the table corresponds to a forced body motion, as considered in the present paper, i.e., the body position and velocity are known at each physical time step. In flow-structure interaction problems, where body motion is fully coupled to the flow dynamics, the immersed-boundary update may be performed iteratively during the pseudotime loop (step 4), so that both flow and body dynamics are treated implicitly [30]. This aspect, which is beyond the scope of the present study, will be addressed in future works.

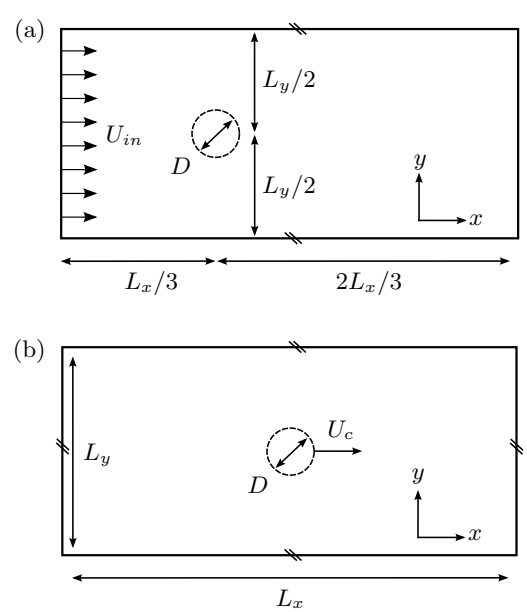

FIG. 4. Schematic view of the considered test cases: (a) Flow past a fixed circular cylinder immersed in a cross flow, and (b) cylinder towed in still fluid. Boundaries of the computational domain are represented by a black rectangle and the dashed line indicates the immersed boundary.

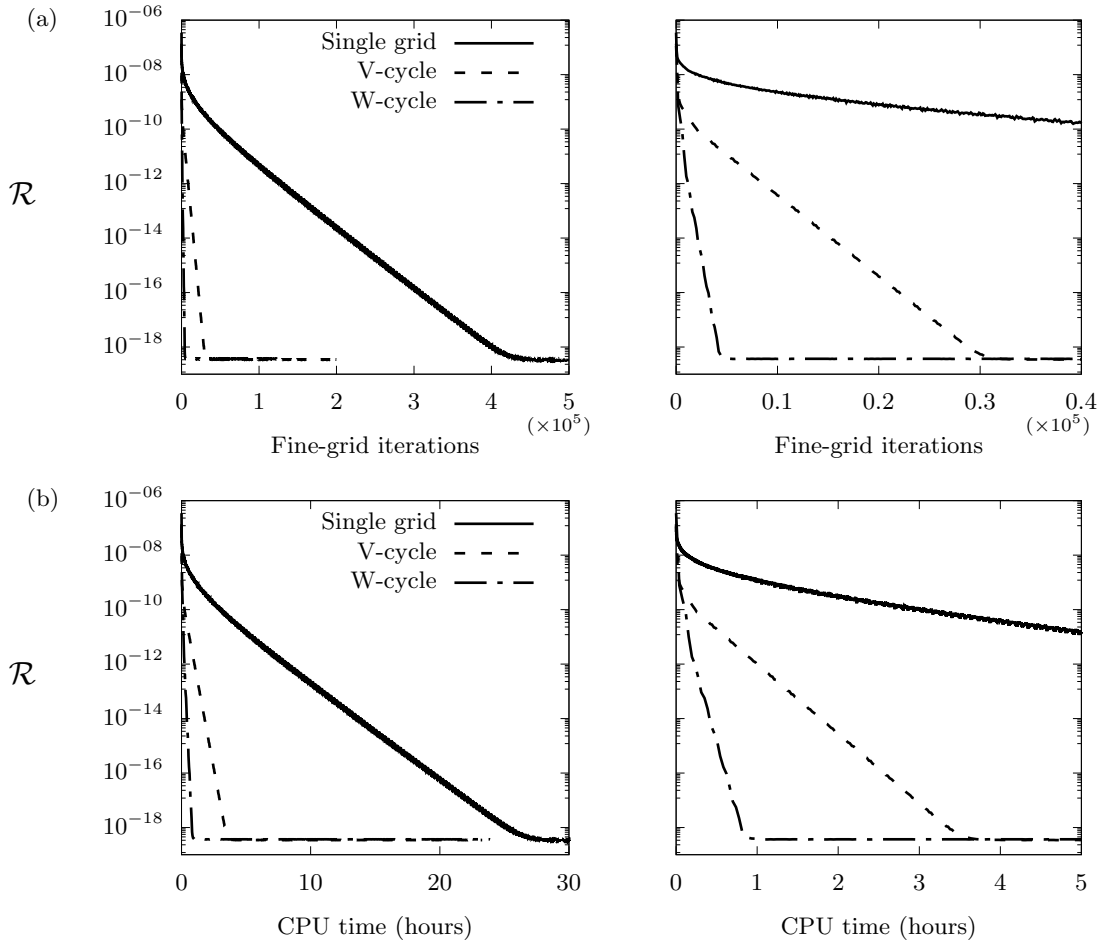

FIG. 5. Simulation of the steady flow past a circular cylinder at Re = 40: Evolution of the residual as a function of (a) fine-grid iterations and (b) CPU time for the single- and multigrid solvers.

III. RESULTS

The above-described method is applied to the simulation of the laminar flow past a circular cylinder. The physical configuration is described in Sec. III A. The accuracy of the immersed-boundary lattice Boltzmann method in this configuration has already been thoroughly examined in previous works [27,31], including Ref. [21], where the same explicit LB solver was employed.

Additional validation results are also provided in Appendix. In the following, the focus is placed on the speedup and possible alteration of the flow solution when using the multigrid and dual-time-stepping techniques, compared to the standard lattice Boltzmann method. Three cases are considered, namely the steady and unsteady flows past a fixed cylinder immersed in a cross flow, and the flow past an impulsively started cylinder in still fluid. The steady flow past a fixed cylinder, addressed in Sec. III B, is simulated using the steady flow solver (multigrid lattice Boltzmann) described in Sec. II D. The reliability of the multigrid solution as well as the associated speedup, compared to an explicit lattice Boltzmann solver, are analyzed. The unsteady cylinder wake is considered in Sec. III C; it is simulated using the unsteady flow solver (multigrid dual-time-stepping lattice Boltzmann, see Sec. II A and Sec. II D). The accuracy of the predicted flow solution is examined in comparison with the solution issued from a standard explicit solver. The influence of the time step, which is not restricted by any stability or accuracy condition, on the flow solution is analyzed. Finally, the ability of the dual-time-stepping method to simulate flows past moving geometries is illustrated in Sec. III D for an impulsively started cylinder.

A. Description of the test cases

Schematic views of the physical configurations are presented in Fig. 4. In the first set-up [Fig. 4(a)], a circular boundary of diameter D, modeled by the immersed-boundary method, is immersed in an oncoming flow U_{in}. A velocity Dirichlet condition U_{in} is set at the left boundary of the domain, using a bounce-back method [32,33]. Periodic boundary conditions are set on the upper and lower boundaries. A pressure Dirichlet condition, ensured by a nonequilibrium bounce-back method [34], is set on the right boundary to model an outflow condition. The flow is initialized with a uniform flow velocity equal to U_{in}. In the second case [Fig. 4(b)], the cylinder is towed in the x direction at velocity U_c. Periodic boundary conditions are set on all boundaries of the computational domain. The flow is initialized with a uniform flow velocity equal to zero, and the body is impulsively started at

the first time step of the computation. The streamwise and cross-flow lengths of the computational domain are denoted by L_x and L_y. In the following, L_x and L_y are set to $L_x/D = 64$ and $L_y/D = 32$, in both configurations.

The Reynolds number, based on the cylinder diameter and a typical flow velocity U_0, is $\mathrm{Re} = U_0 D/\nu$, where ν is the kinematic viscosity of the fluid. In the fixed cylinder case, $U_0 = U_{\mathrm{in}}$; in the case of the towed cylinder, the typical velocity is the towing velocity, $U_0 = U_c$. The streamwise and cross-flow fluid force coefficients are defined as $C_x = 2F_x/(\rho_0 {U_0}^2 D)$ and $C_y = 2F_y/(\rho_0 U_0^2 D)$, where ρ_0 is the reference fluid density and F_x and F_y are the sectional fluid forces in the x and y directions computed on the basis of the immersed-boundary forcing on the Lagrangian markers, namely

$$C_x = \sum_{X_k \in \Gamma} \boldsymbol{B}^*(\boldsymbol{X_k}) \cdot \boldsymbol{e_x} \Delta S_k, \tag{35a}$$

$$C_y = \sum_{X_k \in \Gamma} \boldsymbol{B}^*(\boldsymbol{X_k}) \cdot \boldsymbol{e_y} \Delta S_k. \tag{35b}$$

The flow is expected to remain steady for low values of the Reynolds number. In contrast, an unsteady wake associated with vortex shedding is generally observed for $\mathrm{Re} > 48$, approximately. Both steady ($\mathrm{Re} = 40$) and unsteady ($\mathrm{Re} = 100$) regimes are considered in the following.

B. Steady flow past a fixed cylinder

The steady flow over a circular cylinder is computed for $\mathrm{Re} = 40$, $D/\Delta n = 10$, $U_0/c = 0.05$, and $\tau = 0.5375$. The evolution of the residual (34) during the computation is plotted in Fig. 5(a), for the single- and multigrid solvers. The two multigrid cycles introduced in Sec. II D, namely the V and W cycles, are considered. Moreover, in order to emphasize the effect of the multigrid method on the convergence, the employed single-grid solver follows the same iterative procedure as that described by (29)–(31); in particular, it involves the same relaxation step as that performed in the multigrid solver. In all cases, the residual decreases monotonically until it reaches a plateau associated with round-off errors. Using the single-grid solver, complete convergence is achieved after 4×10^5 iterations, approximately. A reasonably converged state can be obtained before complete convergence; here the entire convergence history is shown for illustrative purpose. Nevertheless, the convergence rate of the single-grid lattice Boltzmann solver remains too small to allow the efficient implementation of a dual-time-stepping method. The effect of coarse-grid corrections on the convergence is well illustrated, since the convergence of the multigrid solvers is drastically faster than that of the single-grid one, in terms of fine-grid iterations. In addition, it is clearly noted that the W cycle performs better than the V cycle. However, the computational cost of the coarse-grid sweeps, even though smaller than fine-grid iterations, is not negligible. The real computational time required to achieve convergence is thus depicted in Fig. 5(b), which represents the evolution of the residual as a function of the CPU time. It is confirmed that the multigrid solvers converge faster than the single-grid one. The convergence acceleration is measured by the speedup $S = t_{\mathrm{mg}}/t_{\mathrm{sg}}$, where t_{sg} and t_{mg} are the CPU times required to decrease the residual from 10^{-10} to 10^{-14} for the single- and multigrid algorithms. In the present case [Fig. 5(b)], the speedup obtained for the V and W cycles are equal to $S \approx 7$ and $S \approx 35$.

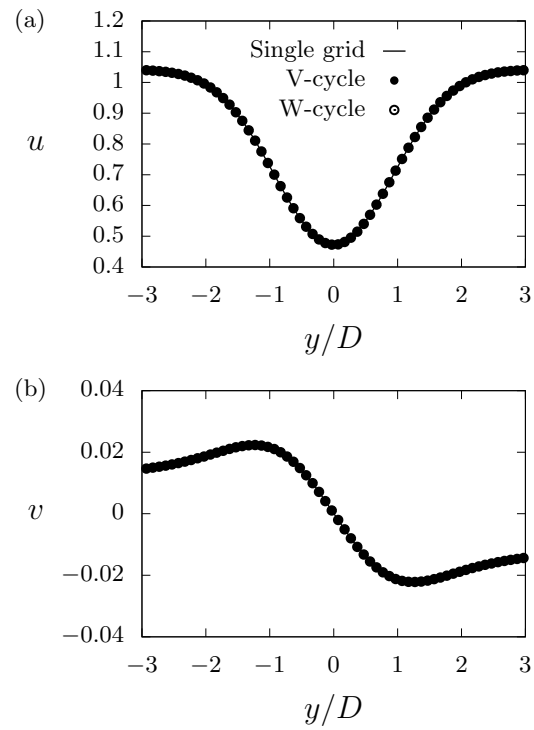

FIG. 6. Simulation of the steady flow past a circular cylinder at $\mathrm{Re} = 40$: Cross-flow evolution of the (a) streamwise and (b) cross-flow flow velocities at $x/D = 1$, predicted by the single- and multigrid solvers.

After convergence, the flow solution predicted by the single- and multigrid solvers are identical. This is illustrated in Fig. 6, which shows the cross-flow evolution of the streamwise and cross-flow flow velocities in the cylinder wake after convergence. In this plot, and in the following, the (x, y) frame has been centered on the cylinder axis. It is noted that single- and multigrid solutions are superimposed. The computed streamwise fluid force C_x is also the same in all cases.

C. Unsteady flow past a fixed cylinder

The unsteady flow past a circular cylinder is simulated for $\mathrm{Re} = 100$, $D/\Delta n = 20$, $U_0/c = 0.05$, and $\tau = 0.53$. As the flow is initialized with a uniform velocity, all computations exhibit a transient associated with the development of the unsteady flow. This transient is not addressed here, and the focus is placed on the flow after statistical convergence of the solution, when the unsteady wake is fully developed. Two algorithms are considered in the following: a standard explicit lattice Boltzmann method and the multigrid dual-time-stepping lattice Boltzmann method described in Sec. II. Two approaches may be considered to control the inner iterations of the dual-time-stepping method: setting the number of inner iterations l_c performed at each time step or fixing a minimum global residual $\mathcal{R}$ that must be reached before updating the physical solution. As done in Ref. [8], the first approach is

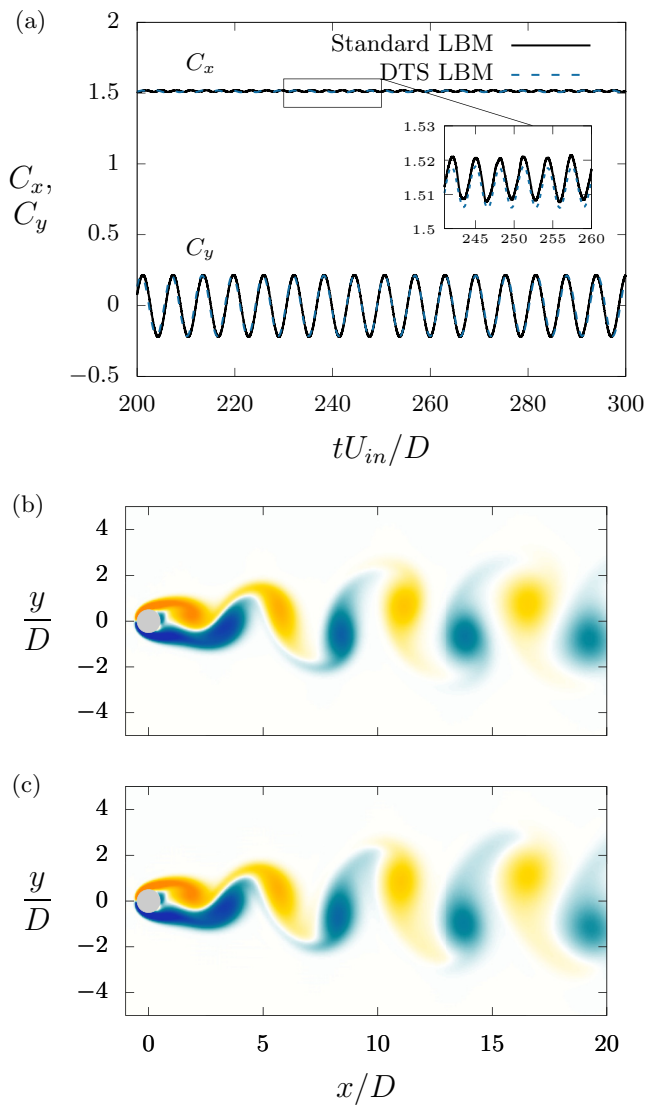

FIG. 7. Overview of the unsteady flow past a circular cylinder at Re = 100 simulated by the standard and dual-time-stepping lattice Boltzmann methods: (a) Time histories of the fluid force coefficients in both cases and [(b) and (c)] instantaneous isocontours of the nondimensional flow vorticity issued from the (b) standard and (c) dual-time-stepping simulations.

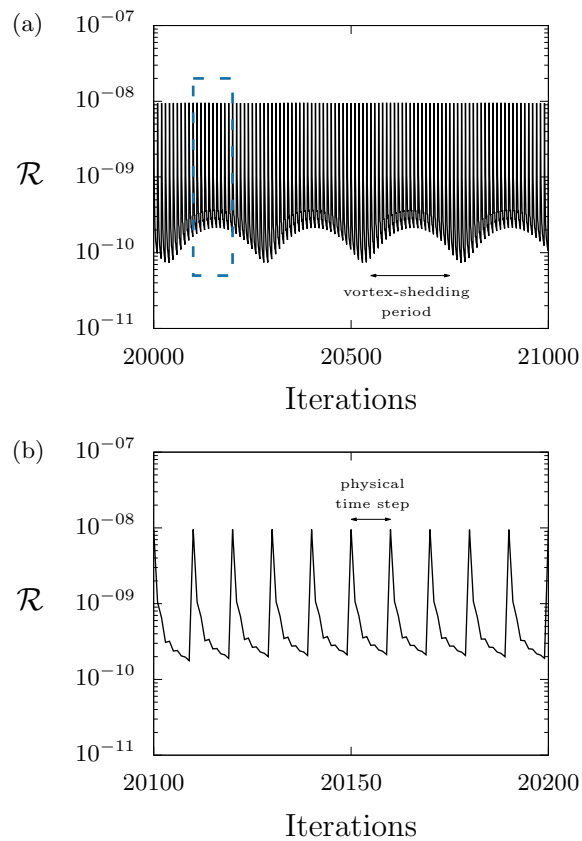

FIG. 8. Dual-time-stepping simulation of the unsteady flow past a circular cylinder at Re = 100: Evolution of the residual (34) as inner iterations are performed. A typical history is presented in (a), and a blue dashed rectangle indicates the region magnified in (b).

employed in the present computations, allowing us to easily control the cost of the simulations. It is recommended that l_c is adjusted as a numerical parameter, according to the desired numerical accuracy, as the time step and mesh size. The effect of l_c will be addressed hereafter.

An overview of the flow predicted by both methods is presented in Fig. 7. In this example, the time step of the dual-time-stepping algorithm is set to $\Delta t/\Delta t^* = 50$, and 10 multigrid W cycles are performed at each time step to solve the pseudosteady problem. It is recalled that Δt^* corresponds to the lattice Boltzmann time step and equals 1 using the present normalization (lattice units). Figure 7(a) shows time histories of the fluid force coefficients in the streamwise and cross-flow directions. The results issued from both methods are very similar. In particular, force magnitudes and frequencies are in agreement, even though a small difference between force frequencies may be noted in the time history of C_y. The fluid force fluctuations are mainly sinusoidal. The Strouhal frequency $f_{\rm st} = f_y D/U$, where f_y is the cross-flow force frequency, is computed on the basis of a fast-Fourier transform of the C_y signal; it corresponds to the vortex-shedding frequency. In both cases, $f_{\rm st}$ is close to 0.16. This frequency corresponds to a time discretization of 2500 and 50 time steps per vortex-shedding period for the standard and dual-time-stepping ($\Delta t/\Delta t^* = 50$) methods. Instantaneous visualizations of the flow are presented in Figs. 7(b) and 7(c). The visualizations are based on instantaneous contours of the nondimensional flow vorticity, $\omega = (\partial u/\partial y - \partial v/\partial x)D/U_{\rm in}$. Overall, the qualitative aspect of the flow is very similar in both cases. The main differences are noted in the far wake ($x/D > 10$), where some variations can be noted in particular in the region of vorticity trails connected to the wake vortices.

The evolution of the solution residual during the computation, for the dual-time-stepping algorithm, is plotted in Fig. 8. The x axis indicates the number of performed inner iterations (i.e., multigrid cycles). The residual exhibits a periodic evolution. Two periods can be noted; the smaller one, equals to 10 iterations, is associated with the inner loop performed at each physical time step. During a physical time step, the residual decreases by approximately two orders of magnitude. The second identified period is associated with the vortex-shedding

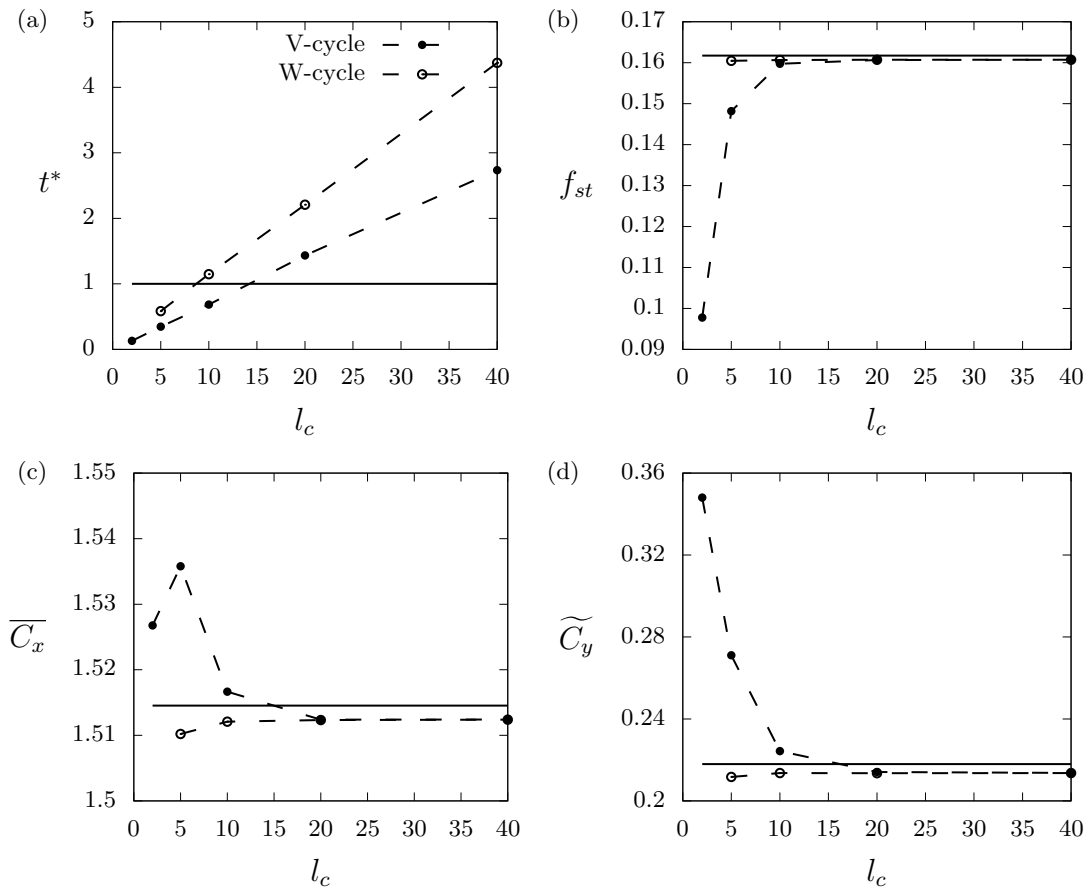

FIG. 9. Dual-time-stepping simulation of the unsteady flow past a circular cylinder at Re = 100: Influence of multigrid cycle and number of inner iterations l_c on (a) the normalized CPU time t^* and on the computed fluid forces, quantified by (b) the Strouhal frequency, (c) the time-averaged streamwise force coefficient, and (d) the amplitude of the cross-flow force coefficient. In all plots, a solid line indicates the value issued from standard explicit method. The computation using a W cycle and $l_c = 2$ is unstable; therefore, the data are not reported for this case. In all cases, the time step is $\Delta t/\Delta t^* = 50$.

frequency, indicating that the mininum residual may vary from one time step to the other. The convergence achieved during each physical time step depends on the multigrid cycle. In this example, the W cycle described in Sec. II D is employed. As expected from the steady analysis performed in Sec. III B, a lower convergence rate is achieved when a V cycle is employed (not shown here). For a given number of inner iterations, this affects the accuracy of the computed flow. The effect of the multigrid cycle on the computational cost and accuracy is examined hereafter.

The numerical cost of the simulations is quantified using the normalized CPU time $t^* = t_{\mathrm{dts}}/t_{\mathrm{exp}}$, where t_{exp} and t_{dts} are the computational times required to advance the flow solution over one convective timescale D/U_{in} for the explicit and dual-time-stepping solvers. For the case presented in Fig. 7 and Fig. 8 ($\Delta t/\Delta t^* = 50$, W cycle, $l_c = 10$), $t^* \approx 1.15$, i.e., the dual-time-stepping simulation is slightly more expensive than the explicit simulation. However, the simulation cost is closely related to the employed multigrid cycle and to the number of inner iterations l_c. This is depicted in Fig. 9(a). The increase of the computational time as a function of l_c is close to linear. In addition, it is noted that simulations employing a W cycle are significantly more expensive than the V-cycle simulations, as expected since the W cycle involves more sweeps of coarse-grid levels as well as more restriction and prolongation operations. The number of inner iterations also affects the accuracy of the computations, as depicted in Figs. 9(b)–9(d), which show the evolutions of the vortex-shedding frequency and fluid force statistics, namely the time-averaged streamwise force coefficient $\overline{C_x}$ and the amplitude of the cross-flow force coefficient $\widetilde{C_y}$. All quantities exhibit converging evolutions: They tend to a plateau value as l_c increases. These plateau values are close to the physical quantities issued from the standard explicit method, supporting the reliability of the dual-time-stepping method. Using the V cycle, it is noted that a reasonable convergence of the physical quantities is obtained for $l_c = 10$; this set-up may thus be employed to perform accurate simulations. However, a lower number of inner iterations is needed if the W cycle is employed, as suggested by the higher convergence rates observed in the steady-flow analysis performed in Sec. III B. In this case, a reasonable convergence is obtained for $l_c = 5$. Based on Fig. 9(a), it can be noted that the normalized CPU time associated with this set-up is $t^* \approx 0.58$, corresponding to a 40% speed-up compared to the explicit method.

A major advantage of the dual-time-stepping method is that it allows us to freely vary the physical time step Δt. As expected, variations of Δt impact the computational cost

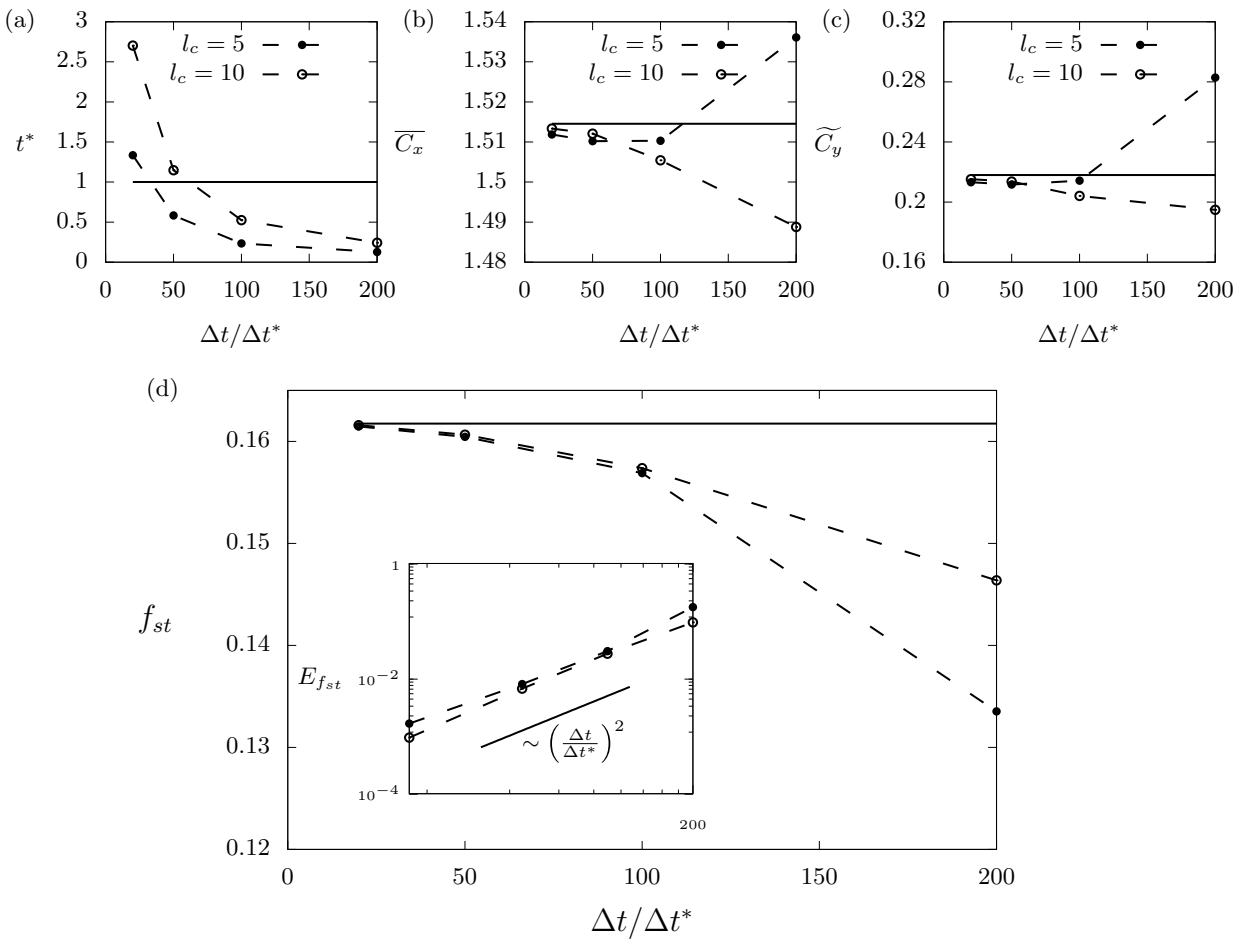

FIG. 10. Dual-time-stepping simulation of the unsteady flow past a circular cylinder at Re = 100, using a multigrid W cycle: Influence of the normalized time step $\Delta t/\Delta t^*$ on (a) the normalized CPU time, (b) the time-averaged streamwise force coefficient, (c) the cross-flow force coefficient amplitude, and (d) the cross-flow force frequency for $l_c = 5$ and $l_c = 10$. In (d), the inset shows the evolution of the frequency relative error, defined by (36).

of the simulations; this is illustrated in Fig. 10(a), which shows the important decrease of t^* as the time step increases. Figures 10(b) and 10(c) show the influence of the time step on the fluid force statistics. Variations of the time-averaged streamwise force remain limited, as its relative difference with the value issued from the explicit computation does not exceed 2% over the considered range of normalized time steps. Yet a converging trend is clearly noted: As the time step decreases, the time-averaged drag tends to a plateau value independent to the number of inner iterations and close to the value issued from the explicit computation. A similar trend is noted in the evolution of $\widetilde{C_y}$, which exhibits larger variations than $\overline{C_x}$. The effect of the number of inner iterations is also clearly noted, as the variation of the error as a function of the time step is less pronounced for $l_c = 10$ than for $l_c = 5$. Finally, the evolution of the wake frequency $f_{\rm st}$ is examined in Fig. 10(d). The observed trend is consistent with the above analysis. In addition, the figure shows the evolution of the relative frequency error,

$$E_{f_{\rm st}} = \frac{f_{\rm st} - f_{\rm st,ref}}{f_{\rm st,ref}}, \tag{36}$$

where $f_{\rm st,ref}$ is the frequency issued from the explicit computation. As $E_{f_{\rm st}}$ relates to the temporal variation of the flow, it is expected to be connected to the accuracy of the dual-time-stepping integration, described by expression (7). The nominal second-order accuracy of the method is supported by the evolution noted in Fig. 10(d): For both $l_c = 5$ and $l_c = 10$, the relative error exhibits a trend $E_{f_{\rm st}} \sim (\Delta t/\Delta t^*)^2$.

Based on Fig. 10, an optimal numerical set-up can be identified. Indeed, using $\Delta t/\Delta t^* = 100$ and $l_c = 5$, the relative errors on $\overline{C_x}$, $\widetilde{C_y}$, and $f_{\rm st}$ are respectively equal to 0.003, 0.017, and 0.030, while the normalized CPU time is equal to 0.23. In summary, the relative variation of the physical quantities is less than 5% while the computation is 4 times faster, compared to the explicit solver, confirming the practical interest of the method. This value of the speed-up is indicative; it corresponds to a straightforward implementation of the proposed method, tested on a standard unsteady test case. Higher speed-up may be achieved by optimizing the convergence rate of the multigrid method as well as its implementation. High speed-up is also expected in configurations involving low-frequency fluctuations, since the physical time step might be further increased in theses cases. In addition to the direct speed-up commented above, a major extra speed-up can be

obtained by increasing the time step during flow transitions (e.g., initial development of an unsteady wake) if temporal accuracy is only required during the fully developed state. Detailed examination of these aspects, which is beyond the scope of the present paper, will be developed in future works.

In this section, the analysis of the dual-time-stepping method has allowed us to emphasize two important points: (i) the DTS method is able to accurately predict the unsteady flow and (ii) it can significantly speed up the computations. The free setting of the time step provides other beneficial features. In particular, using a large time step allows the damping of undesired high-frequency fluctuations, often observed in lattice Boltzmann simulations for configurations involving sharp temporal variations. This important feature is addressed in the next section.

D. Flow past an impulsively started cylinder

So far, simulations have been performed considering fixed geometries. Nevertheless, the extension of the present method to moving geometries is straightforward, since the proposed algorithm (see Table I) involves the update of the position and velocity of the immersed-boundary markers. This is illustrated in the following by considering the flow past a cylinder towed in still fluid. The configuration is similar to that employed in Sec. III B, i.e., $\mathrm{Re} = 40$, $D/\Delta n = 10$, $U_0/c = 0.05$, and $\tau = 0.5375$. Even though the cylinder wake is expected to remain steady for this value of the Reynolds number, this set-up is considered to be unsteady, since the cylinder is moving in the computational frame and attention will be paid to the transient response of the fluid. Therefore, the dual-time-stepping method is employed and compared to the solution predicted by a standard explicit lattice Boltzmann method. The DTS computations are run using a multigrid W cycle, with $\Delta t/\Delta t^* = 40$ and $l_c = 5$. Using this set-up, the DTS and explicit computations have a comparable computational cost. Besides its illustrative purpose concerning simulations of flows around moving bodies, this set-up is chosen to address a common problem in lattice Boltzmann simulations: the emergence of undesired acoustic waves due to sharp variations of the flow solution. Indeed, in this test case, the cylinder is impulsively started, i.e., its velocity instantaneously increases from zero to U_c during the first time step of the simulation. As the simulated flow is weakly compressible, this may generate sound waves that can deteriorate the global solution.

Similar rapid variation of body momentum is expected in other flow-body interaction problems, as in systems involving collision between solid particles, for instance.

The time histories of the streamwise fluid force coefficient, simulated using the explicit and dual-time-stepping methods, are plotted in Fig. 11(a). During the first time steps of the computation, a very large drag is exerted on the cylinder. This can be attributed to the major added mass effect resulting from the impulsive start of the body. Afterward, the streamwise force rapidly converges to a steady value close to $C_x \approx -1.5$. This global evolution is very similar for both methods. However, the drag issued from the explicit simulation also exhibits high-frequency oscillations; these fluctuations relate to undesired pressure waves, as discussed afterward. Even tough the fluctuation amplitude tends to decrease as a function of time, they deteriorate the drag-force signal over large time periods, as their effect remain significant after 60 convective periods in Fig. 11(a). These spurious oscillations are suppressed by the dual-time-stepping method.

Figure 11(b) shows the temporal evolution of the streamwise flow velocity at $(x/D, y/D) = (0, 0)$. Since this point corresponds to the initial position of the cylinder, the flow velocity rapidly increases after the impulsive start of the body. Afterward, the velocity decreases as the cylinder moves away, until it almost vanishes for $tU_0/D > 20$, and it increases again when the cylinder gets back to its initial position due to the periodicity of the computational domain. This global evolution is identical for both methods, supporting that the dual-time-stepping method can accurately describe the flow unsteadiness. It is noted that the high-frequency fluctuations observed in the drag-force signal also alter the flow velocity, especially when using the explicit method. These fluctuations are mostly damped by the dual-time-stepping method. Some residual oscillations, of lower amplitude and lower frequency, are, however, still visible in the velocity signal, as emphasized by the inset in Fig. 11(b).

The high-frequency fluctuations noted in Figs. 11(a) and 11(b) are related to the propagation of pressure waves in the computational domain. This is examined in Figs. 11(c) and 11(h), which show the spatial evolution of the nondimensional fluid density, $\rho^* = (\rho - \rho_0)/\rho_0$, at different times indicated in Figs. 11(a) and 11(b). The explicit simulation is considered first. At the initialization of the computation, a well-defined pressure wave is generated by the impulsive start of the cylinder and propagates toward the boundaries of the computational domain [Fig. 11(c)].

A similar phenomenon has been reported in prior works on lattice Boltzmann simulations of an impulsively started cylinder [35].

At this time, no high-frequency fluctuations are noted in the drag-force signal, since the pressure wave does not interact directly with the body. After some time, the pressure fluctuations cross the periodic boundaries of the domain and propagates toward the body [Fig. 11(e)]. At time t_2, the first interactions between the body and the pressure perturbations is accompanied by the emergence of the high-frequency fluctuations in the drag force signal. Even though the pressure waves tend to dissipate with time, they still alter the global pressure filed in the computation domain after 60 convective periods [Fig. 11(g)]. The solution issued from the dual-time-stepping method exhibits a different behavior. At the initialization of the computation, no clear pressure wave can be identified; instead, a long-range pressure perturbation appears to propagate in the domain [Fig. 11(d)]. Consequently, the solution rapidly tends to a smooth pressure distribution [Fig. 11(f)]. At time t_3, the solution is not altered by any pressure perturbation [Fig. 11(h)].

The damping of the high-frequency pressure waves using the DTS method can be attributed to a filtering effect due to the large value of the time step. An additional computation has been performed, using the DTS method with a time step $\Delta t/\Delta t^* = 1$. The resulting time history of C_x is very close to that issued from the explicit computation (not shown here). In particular, the fluid force exhibits high-frequency fluctuations.

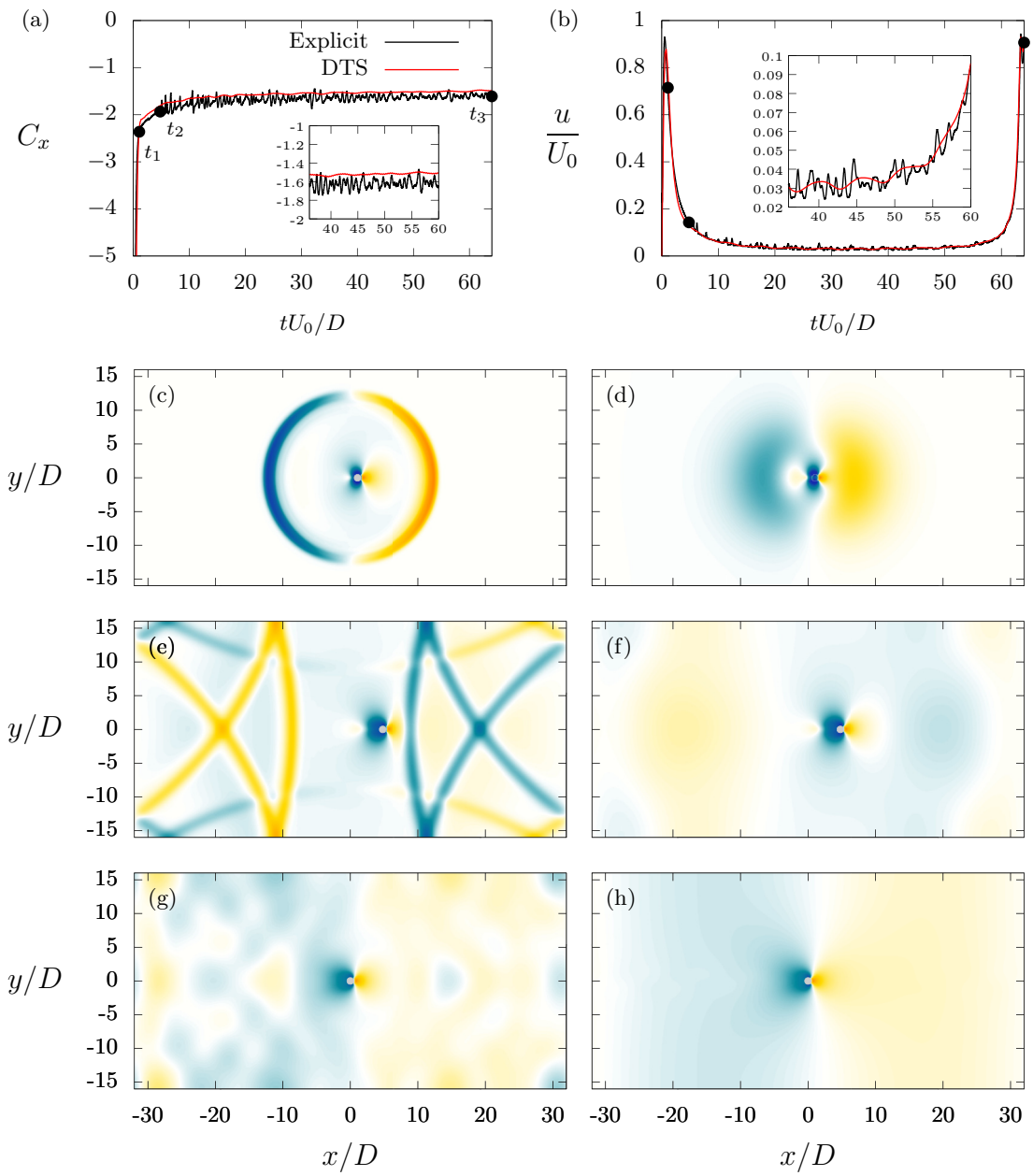

FIG. 11. Flow past an impulsively started cylinder in a periodic domain: [(a) and (b)] Time histories of the (a) streamwise fluid force coefficient and (b) streamwise flow velocity at $(x/D, y/D) = (0, 0)$, and (c)–(h) isocontours of the nondimensional fluid density at times [(c) and (d)] t_1, [(e) and (f)] t_2, and [(g) and (h)] t_3, computed using the [(c), (e), and (g)] explicit and [(d), (f), and (h)] dual-time-stepping methods. The initial position of the cylinder is $(x/D, y/D) = (0, 0)$, and it is moving from left to right. In [(c)–(h)], the body is indicated by a gray circle, and the isocontours are linearly distributed in the range $[-0.004, 0.004]$. Time histories end at $tU_0/D = 64$, when the body has been towed over one domain length.

This supports that the low-pass filtering relates to the time step and that it is not a multigrid effect.

The results depicted in Fig. 11 illustrate several beneficial features of the dual-time-stepping method. Based on the temporal evolution of $\overline{C_x}$ in Fig. 11, two regimes may be identified: the transient regime, for $tU_0/D < 20$, and the steady regime, for $tU_0/D > 20$. In order to analyze the flow during the steady regime, the computation has to be continued until the pressure waves caused by the initialization are damped. As these pressure waves are suppressed when using a large time step, the dual-time-stepping method is therefore more efficient in this context. This is the extra speed-up related to flow transitions, as already mentioned in Sec. III C. On the other hand, the explicit method does not allow to analyze the transient regime, as high-amplitude pressure waves are propagating in the flow domain, affecting the flow velocity and the fluid forces. This illustrates that the standard lattice Boltzmann method may be poorly suited for the study of incompressible flows involving sharp temporal variations. In contrast, the transient regime can be accurately analyzed using the DTS method, without any additional computational cost (it is recalled that the direct speed-up is close to one in this case).

Indeed, if timescales associated with the main flow dynamics are large compared to those related to pressure waves, as expected for low-Mach-number configurations, the time step can be set according to the expected dominant timescales, thus filtering undesired high-frequency fluctuations. As the generation of pressure waves is a critical issue in many simulations, this feature may be useful for future developments, improving the robustness and reliability of the lattice Boltzmann method.

IV. CONCLUSION

A multigrid dual-time-stepping approach has been proposed to avoid time-step restrictions in the lattice Boltzmann method. The multigrid procedure is based on the method proposed by Mavriplis [19], extended here to take into account the presence of immersed boundaries in the flow. The coupling between the dual-time-stepping and lattice Boltzmann methods is ensured by an external forcing in the lattice Boltzmann equation, related to the time derivatives of the macroscopic flow quantities.

In the case of the steady flow past a fixed cylinder (Re = 40), the proposed multigrid lattice Boltzmann solver drastically accelerates the convergence of the flow solution, confirming the efficiency of the multigrid approach and its suitable coupling with the immersed-boundary method. The unsteady flow characterized by the alternate shedding of vortices and developing for a higher value of the Reynolds number (Re = 100) has been simulated using the dual-time-stepping method. The proposed approach is able to accurately simulate the unsteady solution, as supported by the comparison with the solution issued from an standard explicit solver. The effect of the time step and number of inner iterations on the numerical cost and accuracy has been analyzed in detail, and the nominal second-order temporal accuracy of the dual time stepping is supported by the evolution of the simulated vortex-shedding frequency. Finally, the analysis of the flow around an impulsively started cylinder has shown that using a larger time step allows us to efficiently suppress undesired high-frequency pressure waves arising from sharp temporal variations in the system.

In summary, the proposed dual-time-stepping approach is a robust and accurate method allowing to vary the physical time step. Two major advantages resulting from this feature have been identified.

a. Speed-up of the computations. Direct speed-up is achieved by increasing the physical time step. The potential speed-up depends on the physical configuration and on the desired numerical accuracy. In this paper, a direct speed-up equal to 4 has been obtained for the unsteady flow past a cylinder, while keeping the error on fluid forces below 5%. Higher speed-up may be achieved in configurations involving slow flow variations, since larger time steps may be employed in these cases. Major indirect speed-up may also be achieved by accelerating initial flow developments through the increase of the time step. Finally, the computation of fully developed flow regime is drastically accelerated through the damping of high-frequency pressure waves that may arise from initialization.

b. Damping of high-frequency fluctuations. Lattice Boltzmann simulations often involve undesired pressure waves related to sharp geometries or rapid temporal variations. This effect limits the range of application of the lattice Boltzmann method or implies the implementation of artificial damping. In low-Mach-number computations, main flow-dynamics timescales are expected to be large compared to timescales related to pressure waves. By setting the physical time step according to the main flow timescales, pressure waves tend to be *naturally* damped by the computations. This feature is essential for the future development of robust and reliable lattice Boltzmann solvers.

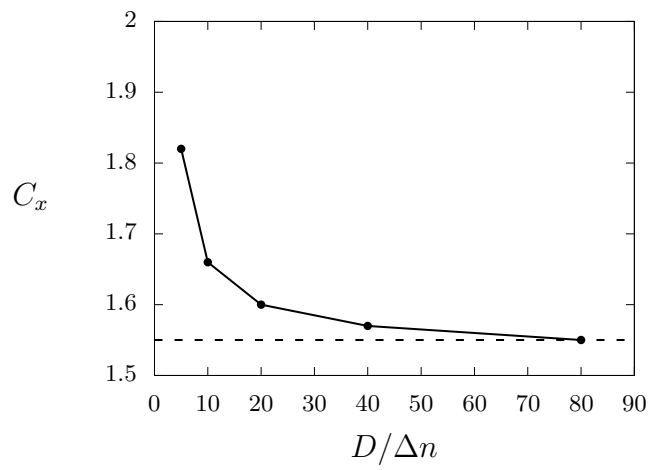

FIG. 12. Mesh convergence analysis for the steady flow past a circular cylinder at Re = 40: Evolution of the drag coefficient C_x as a function of the mesh resolution D/Δ.

ACKNOWLEDGMENTS

The work was supported by the project MACBION. This work received funding from the Excellence Initiative of Aix-Marseille UniversityA*MIDEX, a French Investissements d'Avenir programme, the LABEX MEC, and the SINUMER project (ANR-18-CE45-0009-01) of the French National Research Agency (ANR).

APPENDIX : ADDITIONAL VALIDATION RESULTS ON THE FLOW PAST A CYLINDER AT Re = 40 AND Re = 100

Additional results are provided in the following concerning simulations of the flow past a circular cylinder. First, a focus is placed on simulations performed at Re = 40 with the present multigrid immersed-boundary lattice Boltzmann method. A

TABLE II. Drag coefficient on a cylinder immersed in a flow at Re = 40 from previous works and in the present simulations.

Study	Method	C_x
Tritton [36]	Expt.	1.58
Niu *et al.* [37]	Num.	1.59
Sen *et al.* [38]	Num.	1.51
Bharti *et al.* [39]	Num.	1.53
Posdziech and Grundmann [40]	Num.	1.49
Present	Num.	1.55

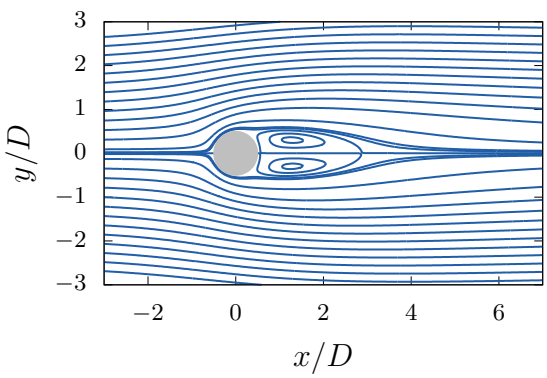

FIG. 13. Streamlines of the flow past a circular cylinder at Re = 40.

mesh convergence analysis is reported in Fig. 12, which shows the evolution of the drag coefficient C_x as a function of $D/\Delta n$. In these simulations, the size of the numerical domain is set to $L_x/D = L_y/D = 64$. The multigrid setup is the same as that employed in Sec. III B: Computations are performed using W cycles on four grids, and the number of sweeps is equal to 1, 2, 3, and 4 on grids 1, 2, 3, and 4, respectively. Figure 12 indicates that the drag coefficient converges to $C_x = 1.55$. This value is in agreement with previous experimental and numerical works, as detailed in Table II.

The flow solution issued from the case $D/\Delta n = 40$ is visualized using streamlines in Fig. 13. The flow pattern is very similar to that issued from the high-resolution finite-element simulations reported by Sen *et al.* [38]. In particular, no flow penetration is observed across the body and well-defined streamlines are computed along the cylinder surface.

TABLE III. Fluid forces on a cylinder immersed in a flow at Re = 100 from previous numerical works and in the present simulations.

Study	$\overline{C_x}$	C_y^m	f_{st}
Braza *et al.* [41]	1.28	0.29	0.16
Saiki and Biringen [42]	1.26	—	0.17
Zhang *et al.* [43]	1.43	—	0.17
Zhou *et al.* [44]	1.48	0.31	0.16
Lai and Peskin [45]	1.45	0.33	0.16
Talley *et al.* [46]	1.34	0.33	0.16
Stansby and Rainey [47]	1.32	0.35	0.17
Kim *et al.* [48]	1.33	0.32	0.16
Le *et al.* [49]	1.37	0.32	—
Shen *et al.* [50]	1.38	0.33	0.17
Present	1.41	0.34	0.17

Then the accuracy of the dual-time-stepping algorithm is illustrated for the unsteady flow past a cylinder at Re = 100. In accordance with the analysis carried out in Sec. III C, the simulation is performed using $\Delta t/\Delta t^* = 100$ and $l_c = 5$, ensuring both time accuracy and computational efficiency. A reasonably fine numerical mesh is employed, corresponding to $D/\Delta n = 80$, in order to achieve high spatial accuracy. The size of the computational domain is set to $L_x/D = 64$ and $L_y/D = 32$, as in Sec. III C. The predicted fluid force statistics are reported in Table III and compared to data reported in prior works. In this table, $\overline{C_x}$ designates the time-averaged drag force coefficient, C_y^m is the amplitude of the fluctuating lift coefficient, and f_{st} is the nondimensional vortex-shedding frequency. Overall, the present results are in agreement with previous studies.

[1] Z. Guo and C. Shu, *Lattice Boltzmann Method and Its Applications in Engineering*, Vol. 3 (World Scientific, Singapore, 2013).
[2] T. Krüger, H. Kusumaatmaja, A. Kuzmin, O. Shardt, G. Silva, and E. M. Viggen, *The Lattice Boltzmann Method: Principles and Practice* (Springer, Berlin, 2017).
[3] S. Chen and G. D. Doolen, Annu. Rev. Fluid Mech. **30**, 329 (1998).
[4] Y. Qian, D. d'Humières, and P. Lallemand, Europhys. Lett. **17**, 479 (1992).
[5] X. Shan, X.-F. Yuan, and H. Chen, J. Fluid Mech. **550**, 413 (2006).
[6] S. Chapman and T. G. Cowling, *The Mathematical Theory of Non-uniform Gases* (Cambridge University Press, Cambridge, 1952).
[7] J. H. Ferziger and M. Peric, *Computational Methods for Fluid Dynamics* (Springer Science & Business Media, New York, 2012).
[8] A. Jameson, in *Proceedings of the 10th Computational Fluid Dynamics Conference* (AIAA, Honolulu, HI, USA, 1991), p. 1596.
[9] J. M. Weiss and W. A. Smith, AIAA J. **33**, 2050 (1995).
[10] A. J. Chorin, J. Comp. Phys. **2**, 12 (1967).
[11] E. Turkel, J. Comp. Phys. **72**, 277 (1987).
[12] Y.-H. Choi and C. L. Merkle, J. Comp. Phys. **105**, 207 (1993).
[13] A. Belov, L. Martinelli, and A. Jameson, in *Proceedings of the 33rd Aerospace Sciences Meeting and Exhibit* (AIAA, Reno, NV, USA, 1995), p. 49.
[14] C. Liu, X. Zheng, and C. Sung, J. Comp. Phys. **139**, 35 (1998).
[15] Z. Guo, T. S. Zhao, and Y. Shi, Phys. Rev. E **70**, 066706 (2004).
[16] S. Izquierdo and N. Fueyo, J. Comp. Phys. **228**, 6479 (2009).
[17] K. N. Premnath, M. J. Pattison, and S. Banerjee, J. Comp. Phys. **228**, 746 (2009).
[18] J. Tölke, M. Krafczyk, and E. Rank, J. Stat. Phys. **107**, 573 (2002).
[19] D. J. Mavriplis, Comp. Fluids **35**, 793 (2006).
[20] D. V. Patil, K. N. Premnath, and S. Banerjee, J. Comp. Phys. **265**, 172 (2014).
[21] S. Gsell, U. D'Ortona, and J. Favier, Phys. Rev. E **100**, 033306 (2019).
[22] P. L. Bhatnagar, E. P. Gross, and M. Krook, Phys. Rev. **94**, 511 (1954).
[23] I. Ginzburg, F. Verhaeghe, and D. d'Humieres, Commun. Comp. Phys. **3**, 519 (2008).

[24] D. d'Humières and I. Ginzburg, Comput. Math. Appl. **58**, 823 (2009).
[25] Z. Guo, C. Zheng, and B. Shi, Phys. Rev. E **65**, 046308 (2002).
[26] G. Le and J. Zhang, Phys. Rev. E **79**, 026701 (2009).
[27] T. Seta, R. Rojas, K. Hayashi, and A. Tomiyama, Phys. Rev. E **89**, 023307 (2014).
[28] J. Favier, A. Revell, and A. Pinelli, J. Comp. Phys. **261**, 145 (2014).
[29] W. L. Briggs, S. F. McCormick, and V. E. Henson, *A Multigrid Tutorial*, 2nd ed., Vol. 72 (SIAM, Philadelphia, 2000).
[30] S. Gsell, R. Bourguet, and M. Braza, J. Fluids Struct. **67**, 156 (2016).
[31] Z. Li, J. Favier, U. D'Ortona, and S. Poncet, J. Comp. Phys. **304**, 424 (2016).
[32] A. J. Ladd, J. Fluid Mech. **271**, 311 (1994).
[33] I. Ginzburg and D. d'Humières, Phys. Rev. E **68**, 066614 (2003).
[34] Q. Zou and X. He, Phys. Fluids **9**, 1591 (1997).
[35] Y. Li, R. Shock, R. Zhang, and H. Chen, J. Fluid Mech. **519**, 273 (2004).
[36] D. J. Tritton, J. Fluid Mech. **6**, 547 (1959).
[37] X. Niu, C. Shu, Y. Chew, and Y. Peng, Phys. Lett. A **354**, 173 (2006).
[38] S. Sen, S. Mittal, and G. Biswas, J. Fluid Mech. **620**, 89 (2009).
[39] R. P. Bharti, R. Chhabra, and V. Eswaran, Can. J. Chem. Eng. **84**, 406 (2006).
[40] O. Posdziech and R. Grundmann, J. Fluids Struct. **23**, 479 (2007).
[41] M. Braza, P. Chassaing, and H. H. Minh, J. Fluid Mech. **165**, 79 (1986).
[42] E. Saiki and S. Biringen, J. Comp. Phys. **123**, 450 (1996).
[43] H.-Q. Zhang, U. Fey, B. R. Noack, M. König, and H. Eckelmann, Phys. Fluids **7**, 779 (1995).
[44] C. Zhou, R. So, and K. Lam, J. Fluids Struct. **13**, 165 (1999).
[45] M.-C. Lai and C. S. Peskin, J. Comp. Phys. **160**, 705 (2000).
[46] S. Talley, G. Iaccarino, G. Mungal, and N. Mansour, Ann. Res. Briefs, CTR, 51 (2001).
[47] P. Stansby and R. Rainey, J. Fluid Mech. **439**, 87 (2001).
[48] J. Kim, D. Kim, and H. Choi, J. Comp. Phys. **171**, 132 (2001).
[49] D.-V. Le, B. C. Khoo, and J. Peraire, J. Comp. Phys. **220**, 109 (2006).
[50] L. Shen, E.-S. Chan, and P. Lin, Comp. Fluids **38**, 691 (2009).