Solving incompressible fluid flows on unstructured meshes with the lattice Boltzmann flux solver

Nicolas Pellerina, Sébastien Leclaire and Marcelo Reggio

ABSTRACT
An application of the recently developed lattice Boltzmann flux solver (LBFS) is proposed to solve incompressible flows using unstructured meshes with high aspect ratio triangular cells. The capability to solve turbulent flows is also introduced by coupling the method with a turbulence model, for which the viscosity transport equation is solved on the same mesh. The proposed computational approach is validated for the classical lid-driven cavity flow, the flow over a circular cylinder, and the turbulent flow around a NACA0012 airfoil. Overall, the results obtained agree well with reference data, and demonstrate the validity of using the LBFS on directionally refined meshes, providing the advantage of limiting the number of vertices required in boundary layer regions of the fluid flow. An alternate flux construction method derived from a lattice Boltzmann boundary condition model based on equilibrium distribution streaming is also presented.

ARTICLE HISTORY
Received 7 September 2016
Accepted 4 February 2017

KEYWORDS
Lattice Boltzmann method; flux solver; turbulence; Spalart-Allmaras; cylinder; airfoil

SUBJECT CLASSIFICATION CODES
76M28; 76M12

1. Introduction
The lattice Boltzmann method (LBM) for fluids is a flow-solving approach which treats fluids as a probabilistic ensemble of interacting particles, rather than as a continuum, as in Navier-Stokes solvers. This method has become an alternative to established computational strategies for simulating incompressible aerodynamic flows. In fact, commercial packages that can simulate unsteady 3D aerodynamics are available (Holman, Brionnaud, & Modena, 2014; König, Fares, & Nöltling, 2014). Research has also been conducted in an effort to simulate turbulent airfoil flows with 2D lattices or meshes (Filippova et al., 2001; Imamura, Suzuki, Nakamura, & Yoshida, 2005; Li, Shock, Zhang, Chen, & Shih, 2005; Li, Zhong, Zhuo, & Cao, 2012; Pellerin, Leclaire, & Reggio, 2015).

For flows with high Reynolds numbers, in a turbulent regime, the LBM needs to be supplemented with turbulence models. The 3D commercial packages referenced above employ large eddy simulation strategies. For 2D flows, the LBM can be coupled with a turbulent viscosity transport equation solver. Regardless of the approach, fast flows require mesh spacing near walls to be small enough to capture the fluid dynamics within the boundary layers. For this, both commercial packages use regular lattices and multi-domain mesh refinement strategies in order to provide adequate lattice spacing near walls, while limiting the total number of lattice sites. In contrast, the approaches of Imamura et al. (2005) and Li et al. (2012) are not based on step changes in grid size. Imamura et al. (2005) use an orthogonal grid system that conforms to the airfoil geometry, in conjunction with coordinate transformation that allows the LBM to be applied on a non-Cartesian mesh. Li et al. (2012) also use coordinate transformation, however they retain a Cartesian mesh and stretch it to reduce grid spacing in the vicinity of the airfoil. In previous work (Pellerin et al., 2015), the current authors elected to use the multi-domain approach, and coupled it to a finite-difference implementation of the Spalart-Allmaras turbulence model. They identified mesh refinement near the airfoil’s leading edge as a key element in obtaining accurate solutions.

While multi-domain lattices are easy to construct and can be adapted to arbitrary shapes, they do present certain drawbacks. First, their implementation in a solver, such as Pellerin et al.’s (2015) recursive solver, requires expending significant computational effort to transfer information between domains. Second, by definition, regular lattices cannot be refined along one specific direction, which would certainly be an advantage in terms of capturing high velocity gradients normal to walls, while limiting the number of chordwise lattice sites along an airfoil. The method of Imamura et al. (2005) based on a non-uniform orthogonal mesh addresses this issue.
somewhat, but would not be an optimal solution if the shape of the object subjected to the flow were less regular than a clean airfoil, for instance an airfoil with its leading edge covered with irregular ice shapes. In light of these drawbacks, the current work focuses on using the LBFS on unstructured meshes, which allows the solver implementation to be simplified, as well as providing a mesh that can be adapted to any object shape and refined more in the direction normal to a wall than along a wall.

The first unstructured LB algorithm was introduced by Peng, Xi, Duncan, and Chou (1998, 1999), who transformed the lattice Boltzmann equation into a vertex-centered finite-volume formulation, and calculated the particle distribution fluxes through the boundaries of the control volumes with linear interpolations. Noting that the finite-volume LB of Peng et al. (1999) suffers from more numerical instability than the standard LBM, Stiebler, Tölke, and Krafczyk (2006) proposed using an upwind scheme for the discretization of the convection operator. They compared their method to the central discretization scheme of Peng et al. (1999) and achieved stability at higher Reynolds numbers. Observing that central schemes can introduce non-physical oscillations, and that vertex-centered finite-volume formulations require large amounts of memory storage for the boundary fluxes, Patil and Lakshmisra (2009) introduced a cell-centered finite-volume formulation coupled to a Total Variation Diminishing scheme for the flux calculations. They concluded that their method showed better numerical stability than the central scheme. Zarghami, Magrebi, Ghasemi, and Uberini (2012) also use a cell-centered formulation, and address the stability issue by introducing upwind second order pressure-based biasing factors and artificial damping in the flux calculations.

More recently, Shu, Wang, Teo, and Wu (2014) introduced the lattice Boltzmann flux solver (LBFS). This method is based on the correspondence between the LBM and the Navier-Stokes equations, which can be demonstrated through a Chapman-Enskog expansion. Unlike the finite-volume LBM, for which particle distribution fluxes are evaluated and used in distribution conservation equations, the LBFS uses particle distributions to compute mass and momentum fluxes, and then updates macroscopic conservation equations. Shu et al. (2014) validated the LBFS with orthogonal meshes on standard test cases, such as the lid-driven cavity flow and the flow past a circular cylinder, obtaining results that compared favorably with those available in the literature. One interesting aspect of the LBFS is the method used to obtain the macroscopic properties necessary for the flux calculations. Shu et al. (2014) use density and velocity gradients at the cell centers and project them onto the locations near the cell boundaries where the values are required to calculate the usual LBM equilibrium distributions necessary for calculating the fluxes. The method for obtaining the gradients is not specified by these authors, nor do they state that an upwind interpolation scheme is required. Even so, they report a stable solution for the lid-driven cavity flow with a Reynolds number of 10000, using a non-uniform orthogonal grid of $121 \times 121$ with near-wall refinement. This suggests that the LBFS may have interesting stability properties that could be useful for higher Reynolds number flows. Furthermore, the usual collide-and-stream process of the standard LBM is by nature an upwind operator in the lattice speed directions, and the LBFS, by the way it is constructed, takes advantage of this.

The LBFS was subsequently used by Wang, Shu, Teo, and Wu (2015) for studying fluid-structure interactions through the incorporation of immersed boundary conditions. They tested their method on flows with stationary and moving circular cylinders and obtained satisfactory results. The method was further extended to 3D flows by Wang, Shu, Teo, and Yang (2016) and conclusively tested on unsteady flows around a sphere and a torus. Even though only non-uniform orthogonal meshes were used in the works referenced, the LBFS is also applicable to unstructured meshes, because of the way it is constructed. In fact, Wang, Yang, and Shu (2015) extended the applicability of the LBFS to compressible flows and used a Delaunay-type triangular unstructured mesh to test their model on a low Reynolds number flow around two airfoils placed in a biplane configuration. This method is promising in terms of stability, as no special care is given to the interpolations used for the flux calculations. This is not the case for the finite-volume LBM, which is why the LBFS was chosen for the current work. Its applicability will be tested on weakly compressible higher Reynolds flows around an airfoil, using high aspect ratio triangles near the wall to provide for differing mesh refinement in the normal and tangential directions. For the turbulence modeling, a finite-difference implementation of the Spalart-Allmaras model is applied, following the work of Pellerin et al. (2015).

We use three validation cases to verify the accuracy of the proposed unstructured LBFS implementation. We first solve the laminar flows in a lid-driven cavity and over a circular cylinder to confirm that this LBFS implementation is accurate when no turbulence modeling is used. We then solve turbulent flows around the NACA0012 airfoil at moderate Reynolds numbers for the same test case as in Pellerin et al. (2015). This provides a basis for comparing multi-domain and unstructured LBM solvers.

This article is organized as follows. In section 2, we provide a description of the LBFS and a vertex-centered application on an unstructured mesh. In section 3, we
detail the finite-difference application of the Spalart-Allmaras turbulence model. In section 4, we define the boundary conditions implementation and describe a convergence acceleration method. In sections 5 to 7, we present the results for the lid-driven cavity flow, the circular cylinder flow, and the NACA0012 airfoil flow respectively. In the remaining sections, we discuss our results and present our concluding remarks. In the appendix, we describe an alternative flux construction method based on earlier work by Pellerin, Leclaire, and Reggio (2014) and compare it to Shu et al.’s (2014) method.

2. Unstructured LBFS

The lattice Boltzmann method is based on the Boltzmann equation derived from the kinetic theory of gases (Viggen, 2014):

$$\frac{\partial f_i}{\partial t} + v \cdot \nabla f_i = \Omega(f),$$

represented here without a body force term. In this equation, $f = f(x, v, t)$ represents a particle distribution function at a given location $x$ and a given time $t$ for particles moving at a microscopic velocity $v$. The LHS of Equation (1) represents convective motion and the RHS the scattering effect of particle collisions. A simple equation, represented here without a body force term. In this equation, the Maxwell-Boltzmann distribution is used for the equilibrium distribution $f^{eq}$. In order to obtain the lattice Boltzmann equation, the continuous Boltzmann equation must be discretized in time, in physical space, and in velocity space, since the distribution functions depend on these three parameters. For the velocity space discretization, velocity vectors pointing toward the neighboring sites of a regular lattice are introduced, which leads to the discrete velocity Boltzmann equation:

$$\frac{\partial f_{i}}{\partial t} + c_{i} \cdot \nabla f_{i} = \frac{1}{\tau} (f_{i}^{eq} - f_{i}),$$

where $c_{i}$ is a given velocity vector along the direction index $i$. This index covers all the discrete directions according to the chosen lattice configuration. The time and space discretizations are performed by expressing the LHS of Equation (3) with a material derivative and approximating it with a finite time step. This leads to the lattice Boltzmann equation:

$$f_{i}(x + c_{i} \Delta t, t + \Delta t) = f_{i}(x, t) + \frac{\Delta t}{\tau_{R}} (f_{i}^{eq} - f_{i}).$$

In Equation (4), the commonly used relaxation time symbol $\tau$ was replaced with $\Delta t_{R}$ to highlight the fact that the particle distribution relaxation coefficient is a ratio of finite time steps, and to show that $\Delta t$ in lattice units is not necessarily set to unity. Given a velocity space discretization, the equilibrium distributions are obtained by applying a second order Taylor series expansion centered on the macroscopic velocity origin to the Maxwell-Boltzmann distribution (He & Luo, 1997):

$$f_{i}^{eq}(x, t) = \rho w_{i} \left(1 + \frac{c_{i} \cdot u}{c_{s}^{2}} + \frac{1}{2} \left(\frac{c_{i} \cdot u}{c_{s}^{2}}\right)^{2} - \frac{1}{2} \frac{u \cdot u}{c_{s}^{4}}\right),$$

where $\rho$ and $u$ are the macroscopic density and velocity respectively, and $c_{s}$ is the lattice speed of sound. The chosen weights $w_{i}$ ensure that

$$\rho = \sum_{i=1}^{N} f_{i}^{eq} = \sum_{i=1}^{N} f_{i},$$

and

$$\rho u = \sum_{i=1}^{N} c_{i} f_{i}^{eq} = \sum_{i=1}^{N} c_{i} f_{i}.$$

For two-dimensional simulations, the d2q9 lattice model is widely used. Its nine velocity vectors are given by

$$c_{i} = (c_{x}, c_{y})_{i} = \begin{cases} (0, 0) & i = 1 \\ c \cos((i - 2)\pi/2), & i = 2, 3, 4, 5 \\ c \sin((i - 2)\pi/2), & i = 6, 7, 8, 9, \\ c \cos((i - 6)\pi/2 + \pi/4), & i = 2, 3, 4, 5 \\ c \sin((i - 6)\pi/2 + \pi/4), & i = 6, 7, 8, 9. \end{cases}$$

and the associated weights are

$$w_{i} = \begin{cases} 4/9 & i = 1 \\ 1/9 & i = 2, 3, 4, 5 \\ 1/36 & i = 6, 7, 8, 9. \end{cases}$$

The lattice speed $c$ is defined as $\Delta x/\Delta t$ and is usually set to unity. It is related to the lattice speed of sound through $c_{s}^{2} = c^{2}/3$. Equations (4–9) constitute the basis of the lattice Boltzmann method termed LBGK. Through a Chapman-Enskog expansion, it can be demonstrated that this model is equivalent to the Navier-Stokes equations applied to weakly compressible flows (Chen & Doolen, 1998; Shu et al., 2014), and that the link between
the macroscopic scale of the Navier-Stokes equations (Equation 10) and the mesoscopic scale of the LBGK is provided by Equations (11–12).

\[
\frac{\partial (\rho \mathbf{u})}{\partial t} + \nabla \cdot (\rho \mathbf{u} \mathbf{u}^T) = -\nabla \cdot (\rho I) - \nabla \cdot (\nabla \cdot (\rho \mathbf{u})) + (\nabla \cdot \nabla) (\rho \mathbf{u} \mathbf{u}^T) \tag{10}
\]

\[
p = \rho c_s^2 \tag{11}
\]

\[
v = c_s^2 \Delta t \left( \frac{\Delta t}{\Delta t} - \frac{1}{2} \right) \tag{12}
\]

In Equation (12), \( v \) represents the kinematic viscosity in lattice units. Its value is obtained from the Reynolds number of the simulation and the chosen reference speed, also in lattice units.

While the standard approach to solving the flow with the LBGK is to separate Equation (4) into a collision step and a streaming step, the LBFS of Shu et al. (2014) takes a different approach, based on additional insights obtained from the Chapman-Enskog expansion. In fact, the momentum flux tensor appearing in the Navier-Stokes equations expressed in the following form:

\[
\frac{\partial (\rho \mathbf{u})}{\partial t} + \nabla \cdot (\rho \mathbf{u} \mathbf{u}^T) = 0, \tag{13}
\]

is equivalent to the following moment of the distribution functions

\[
\Pi = \sum_{i=1}^{N} c_i c_i^T \left[ f_{i}^{eq} + \left( 1 - \frac{\Delta t}{2\Delta t_{R}} \right) f_{i}^{neq} \right]. \tag{14}
\]

Also, the non-equilibrium distributions \( f_{i}^{neq} \) are related to the equilibrium distributions through

\[
f_{i}^{neq} = -\Delta t_{R} \left( \frac{\partial}{\partial t} + c_i \cdot \nabla \right) f_{i}^{eq}. \tag{15}
\]

The LBFS uses a finite-volume formulation to solve the continuity and Navier-Stokes equations, and Equations (5, 6, 7, 14, and 15) to calculate the cell boundary fluxes. Equation (15) is not actually used in its continuous derivative form, but is recast into a backwards time finite-difference equation. To do so, the RHS of the equation is expressed as a material derivative and then transformed into:

\[
f_{i}^{neq}(x, t) = -\Delta t_{R} f_{i}^{eq}(x, t) - f_{i}^{eq}(x - c_i \Delta t, t - \Delta t) \frac{\Delta t}{\Delta t}. \tag{16}
\]

For a two-dimensional problem, Shu et al. (2014) provide the general finite-volume equations:

\[
W = \left\{ \begin{array}{c} \rho \\ \rho u_x \\ \rho u_y \\ \Pi_{xx} \\ \Pi_{xy} \end{array} \right\}, \quad E = \left\{ \begin{array}{c} \rho u_x \\ \Pi_{xx} \\ \Pi_{yx} \end{array} \right\}, \quad F = \left\{ \begin{array}{c} \rho u_y \\ \Pi_{yy} \end{array} \right\}, \tag{17}
\]

\[
d W_j \frac{dt}{dV_j} = - \frac{1}{dV_j} \sum_{k=1}^{K} (n_k E + n_k F) dS_k, \tag{18}
\]

where \( d V_j \) is the control volume, and \( d S_k \) is the \( k \)th control surface. Since the computations are performed in 2D, unit depth is assumed for the control volume and surfaces. In the original presentation of the LBFS (Shu et al., 2014), and in following work (Wang, Shu, et al., 2015; Wang et al., 2016), a cell-centered approach is taken. The present implementation differs in that a vertex-centered scheme for the finite-volume computations is used. This method was selected because the macroscopic quantity interpolations performed to compute the boundary fluxes can be simply accomplished with linear interpolations, whereas in the cell-centered approach the quantities are obtained with projections based on spatial gradient approximations. Furthermore, the vertex-centered approach provides more points for discretizing a control volume boundary, which is advantageous when high aspect ratio triangular cells are used, because the fluxes in the elongated direction can be better approximated. In fact, some care is generally required when using slender cells, and the aspect ratio should be limited when high gradients are expected in all directions. In the present work, wake recirculation is expected for the cylinder simulations, therefore the aspect ratio will be limited in order to accurately capture the separation angle. On the other hand, the NACA 0012 airfoil flows are expected to remain attached, therefore higher aspect ratios will be used on the mid-chord wall section, where the gradients are much larger in the wall normal direction than in the tangential direction.

A typical unstructured mesh vertex-centered control volume is represented in Figure 1(a). Each triangle bounding the vertex provides two surfaces through which fluxes may pass, calculated at points C and D shown in Figure 1(b). These points are placed halfway between the triangle’s geometric center and its side mid-points. Additional surfaces can be included if the triangle is bounded by the domain limit or an object wall, as indicated by the G and H flux calculation points. For adjacent triangles, both of which are located in the flow domain, the fluxes calculated at these points are internal
to the control volumes and cancel out, and therefore are omitted.

For each flux calculation point, nine macroscopic interpolation points are included, as depicted in Figure 1. These latter points correspond to the mirrored d2q9 lattice configuration and are used for constructing particle distributions, which will be streamed onto the flux calculation points. Each interpolation point is located at 
\[
\mathbf{x}_{\text{flux}} - \Delta \mathbf{x}_i = \mathbf{x}_{\text{flux}} - c_i \Delta t,
\]
which is, in fact, equivalent to 
\[
\mathbf{x}_{\text{flux}} - c_i \Delta \mathbf{x},
\]
since we assume a unitary lattice speed 
\[
c = \frac{\Delta \mathbf{x}}{\Delta \mathbf{t}} = 1.
\]
The \(
\Delta \mathbf{x}
\) value is chosen to be identical for all flux calculation points within a particular triangle, and is set to ensure that all nine macroscopic interpolation points linked to the core flux calculation points, for instance \(C\) or \(D\), are within the triangle. Since the interpolations are to be performed linearly, the rationale behind this choice for the interpolation point locations is to use only local triangle information in order to ensure local continuity of the macroscopic fields, while avoiding extrapolations outside the triangle. Obviously, for boundary points, such as \(G\) or \(H\), some macroscopic interpolation points will lie outside the triangle, and so their location will be approximated to that of the center point of the d2q9 stencil.

Within the entire domain, each flux calculation point has its own \(\Delta t_R\), which is calculated via Equation (12) and again using \(\Delta t = \Delta x\), or \(c = 1\).

The procedure for calculating the fluxes and updating the macroscopic quantities within a control volume is described below. It is directly inspired by the work of Shu et al. (2014), however a different macroscopic time stepping method is used.

1. Interpolate \(\rho\) and \(\mathbf{u}\) linearly on the macroscopic interpolation points, using the values at the vertices of the local triangle.
2. Use the interpolated macroscopic quantities in Equation (5) to construct equilibrium distributions.
3. Stream the equilibrium distributions obtained in step 2 onto the associated flux calculation points, and compute \(\rho_{\text{flux}}\) and \(u_{\text{flux}}\) using Equations (6–7).
4. Calculate the equilibrium distributions at the flux calculation points using Equation (5), and \(\rho_{\text{flux}}\) and \(u_{\text{flux}}\) obtained in step 3.
5. Calculate the non-equilibrium distributions at the flux calculation points using Equation (16), in which the equilibrium distributions \(f_{\text{eq}}^i(\mathbf{x} - c_i \Delta t, t - \Delta t)\) and \(f_{\text{eq}}^i(\mathbf{x}, t)\) come from steps 2 and 4 respectively.
6. Compute the components of the momentum flux tensor \(\Pi\) using the equilibrium and non-equilibrium distributions of steps 4 and 5, and Equation (14).
7. Build the \(\mathbf{E}\) and \(\mathbf{F}\) vectors of Equation (17) with \(\rho_{\text{flux}}\), \(u_{\text{flux}}\), and \(\Pi\).
8. Update \(W\), that is, the vertex \(\rho\) and \(\mathbf{u}\), using Equation (18) and the Euler forward method for time stepping.

The magnitude of the time step used in step 8 is not the same as the time step associated with the flux calculation points. In order for the time evolution of the solution in every control volume to follow the same transient time scale, this time step must be identical for all control volumes. It is set to the \(\Delta t\) of the smallest triangle (or \(\Delta x\) since \(c\) is set to 1) in the entire domain. This condition is later relaxed to increase the convergence rate for
assumed steady flows. Shu et al. (2014) use a fourth-order Runge-Kutta scheme for time integration, but they do not specify the value they use for $\Delta t$. While a higher-order integration scheme is expected to be more accurate, for the present LBFS implementation, a first-order scheme was selected for its simplicity and because we consider the chosen time step to be small enough for the integration scheme to retain sufficient accuracy.

3. Spalart-Allmaras turbulence model

For the third test case, we simulate turbulent airfoil flows. Since these simulations are two-dimensional, we need to use a turbulence model of the Reynolds Averaged Navier-Stokes type, for which all turbulent scales are modeled. Based on previous work by Pellerin et al. (2015), we chose the Edwards version of the Spalart-Allmaras model. The model equations are not repeated here. The current work also uses a finite-difference scheme. However, since the present method uses an unstructured mesh, the velocity and viscosity gradient calculation stencil is different and should be detailed. Therefore, Figure 2 displays the interpolation points $N$, $S$, $E$, and $W$ used to compute the gradients at a given vertex. The interpolations are linear and use the vertices at each end of the local triangle edge.

The first-order gradients of the Spalart-Allmaras viscosity $\tilde{\nu}$ are calculated with an upwind scheme:

$$\frac{\partial \tilde{\nu}}{\partial x} = \begin{cases} (\tilde{\nu}_V - \tilde{\nu}_W)/\Delta x_W & u_{x,V} \geq 0 \\ (\tilde{\nu}_E - \tilde{\nu}_V)/\Delta x_E & u_{x,V} < 0, \end{cases}$$

(19)

$$\frac{\partial \tilde{\nu}}{\partial y} = \begin{cases} (\tilde{\nu}_V - \tilde{\nu}_S)/\Delta y_S & u_{y,V} \geq 0 \\ (\tilde{\nu}_N - \tilde{\nu}_V)/\Delta y_N & u_{y,V} < 0, \end{cases}$$

(20)

and the second-order gradients are calculated with a central-difference scheme:

$$\frac{\partial^2 \tilde{\nu}}{\partial x^2} = \frac{(\tilde{\nu}_E - \tilde{\nu}_V)/\Delta x_E - (\tilde{\nu}_V - \tilde{\nu}_W)/\Delta x_W}{0.5(\Delta x_E + \Delta x_W)},$$

(21)

$$\frac{\partial^2 \tilde{\nu}}{\partial y^2} = \frac{(\tilde{\nu}_N - \tilde{\nu}_V)/\Delta y_N - (\tilde{\nu}_V - \tilde{\nu}_S)/\Delta y_S}{0.5(\Delta y_N + \Delta y_S)}.$$  

(22)

The velocity component gradients used for calculating $\tilde{\nu}$ in the viscosity production term are calculated with a central-difference scheme:

$$\frac{\partial u_x}{\partial x} = \frac{u_{x,E} - u_{x,W}}{\Delta x_E + \Delta x_W}, \quad \frac{\partial u_y}{\partial x} = \frac{u_{y,E} - u_{y,W}}{\Delta x_E + \Delta x_W},$$

(23)

$$\frac{\partial u_x}{\partial y} = \frac{u_{x,N} - u_{x,S}}{\Delta y_N + \Delta y_S}, \quad \frac{\partial u_y}{\partial y} = \frac{u_{y,N} - u_{y,S}}{\Delta y_N + \Delta y_S}. $$

(24)

The time step used for the forward time-stepping of the Spalart-Allmaras equation is the same as for the finite-volume method in the LBFS. A turbulent viscosity $\nu_t$ is obtained at every vertex from the updated $\tilde{\nu}$. It is then interpolated linearly at the flux calculation points, and the relaxation times $\Delta t_R$ are updated with Equation (12) by replacing $\nu$ with $\nu + \nu_t$. On solid walls, the turbulent viscosity should be null, and so $\tilde{\nu}$ is set to 0 on the associated vertices.

Some explanation should be provided for the choice of a finite-difference scheme for solving the Spalart-Allmaras equation. Since a finite-volume scheme is adopted for solving the flow, it would have been logical to apply a similar scheme for the turbulence transport equation. However, the current approach was chosen to limit memory usage. In this vertex-centered finite-volume implementation, each vertex can have up to 10 bounding triangles, depending on the local mesh topology. This may result in as many as 20 flux points to compute, which would have a significant impact on memory requirements. The proposed finite-difference scheme requires far less memory. Furthermore, it is also simpler to compute the upwind viscosity gradients with the finite-difference scheme.

4. Boundary conditions and convergence acceleration

The boundary conditions are applied after the macroscopic properties have been updated through Equation (18) in step 8 of the LBFS procedure detailed above. For the lid-driven cavity flow, the no-slip condition on the walls is applied by setting $u = (u_x, u_y)$ to $(0, 0)$ for all vertices located on those walls. For the top boundary, $u$ is set to $(U_0, 0)$ and $\rho$ is set to $\rho_0$. For the free stream boundary conditions on a rectangular domain, the velocity is set to $(U_\infty, 0)$ on the left boundary, defined as the
inlet. On the top and bottom boundaries, \( u_y \) is left free, and an approximate null derivative normal to the boundary is applied to \( u_x \) and \( \rho \). This is done by setting the streamwise velocity and density of the boundary vertices to their values interpolated at a distance equivalent to half the local boundary mesh spacing from the boundary. A natural choice for the right boundary, defined as the outlet, would be to impose constant back pressure through the density \( \rho_0 \). However, this approach leads to undesired pressure wave reflections in the computational domain. So instead, a convective condition is imposed for all macroscopic properties, such as

\[
\frac{\partial q}{\partial t} + u_x \frac{\partial q}{\partial x} = 0, \tag{25}
\]

where \( q \) can represent \( \rho \), \( u_x \), or \( u_y \). In Equation (25), the spatial derivative is calculated with a first-order finite difference using interpolated values from the previous time step. The temporal derivative is calculated with a forward first-order Euler integration. Also, in this implementation of the convective boundary condition, the convective speed \( u_x \) is actually set to \( U_0 \). Since the density is not strictly imposed on any domain boundary, we ensure that the entire flow field density remains near the nominal \( \rho_0 \) level by adding the following quantity to the density of all the vertices:

\[
\Delta \rho = \rho_0 - \bar{\rho}_L, \tag{26}
\]

where \( \bar{\rho}_L \) is the average density on the inlet. This approach is similar to that of Wang, Shu, et al. (2015), in which the density is left as a free parameter that varies in time. The difference in the present approach is the continuous correction intended to obtain a true steady-state solution of the flow field.

For a flow that is expected to be steady, such as the flow around an airfoil for which the LBFS is supplemented with a turbulence model that treats all turbulence scales in a RANS fashion, the solution convergence can be accelerated using local time-stepping. In this method, the global time step \( \Delta t \) introduced in section 2 is replaced with a local time step \( \Delta t_L \) that is unique for every vertex. It is actually set to the smallest \( \Delta t \), or \( \Delta x \), of all the triangles comprised in a vertex’s control volume. This means that the physical solution will evolve faster where the mesh cells are larger. Since the cells are much larger far from the airfoil than near its wall, the advantage of this method is to accelerate the influence of the domain boundary conditions on the area near the airfoil. Furthermore, delaying the application of local time-stepping on the continuity equation by a few tens of thousands of iterations also improves convergence time, because it reduces solution oscillations introduced by the initial pressure waves generated at the airfoil wall. The use of a local time-stepping strategy in the context of lattice Boltzmann methods on unstructured meshes is also suggested by Patil and Lakshmisha (2009).

5. Lid-Driven cavity flow

The first validation case consists of the classical lid-driven cavity flow, and its purpose is to validate the proposed unstructured LBFS model on a laminar flow and a mesh that does not contain large cell area variations. The flows for Reynolds numbers 100 and 1000 are simulated on three meshes. The coarsest mesh, built with \( N = 65 \) vertices on each of the square domain sides, is shown in Figure 3. The other two meshes were built with 129 and 257 side vertices, respectively. All the meshes were generated with the Gmsh software of Geuzaine and Remacle (2009). The same software was used to build the meshes presented in the sections that follow.

The simulations were initialized with the fluid at rest, and calculation was stopped when the normalized momentum sampled every ten thousand time steps did not vary by more than \( 10^{-5} \) for any vertex:

\[
\max (|\Delta (\rho |\mathbf{u}|/\rho_0 U_0)|)_{\text{vertices}} \leq 10^{-5}. \tag{27}
\]

This convergence criterion was sufficient for reaching steady-state on all meshes. The results are presented in terms of the velocity profiles on domain centerlines, the center location of the main vortex, and the stream function and vorticity magnitude at the vortex center. We compare these results to those of Ghia, Ghia, and Shin.
which were obtained by solving a vorticity stream function formulation of the Navier-Stokes equations with a finite-difference method on a \(129 \times 129\) uniform grid. The normalized velocity profiles for the computations performed with all mesh levels are shown in Figures 4–5, where

\[
x' = -1 + 2x/L_x, y' = -1 + 2y/L_y.
\] (28)

The agreement between our results and those of Ghia et al. (1982) is satisfactory for both Reynolds numbers. The velocity profiles for the two finest mesh levels are almost superimposed, which indicates that mesh convergence is nearly attained. Further insight into our solutions is presented in Table 1. For the three meshes used, and for both Reynolds numbers, the locations calculated for the main vortex center agree with the reference data. The stream function and vorticity also agree reasonably well. The results for \(Re = 1000\) and the \(N = 129\) meshes are all within 1% of each other.

![Figure 4.](image1.png) **Figure 4.** Normalized velocity profile comparison for \(Re = 100\). The circles represent the results of Ghia et al. and the full line represents the present solution, with \(N = 257\).

![Figure 5.](image2.png) **Figure 5.** Normalized velocity profile comparison for \(Re = 1000\). The circles represent the results of Ghia et al. and the full line represents the present solution, with \(N = 257\).

### 6. Circular cylinder in a free stream

The second validation case involves the circular cylinder in a free stream flow, and is a case that has been well studied. It was chosen to validate the proposed unstructured LBFS model on a laminar flow and a mesh that contains large cell area variations throughout the computational domain. Cells with larger aspect ratios are also included near the solid wall. The flows for Reynolds numbers ranging between 20 and 200 are simulated on three meshes. The cylinder is located in the center of a square computational domain of side length 200 m, and its diameter is set to 1 m. For all meshes, 101 vertices are used along each domain side. For the coarsest mesh, represented in Figure 6, the height of the first vertex layer is set to \(h_1 = D/256\), and the distance between the vertices located on the cylinder wall is constant, at \(l_w \approx 3h_1\). The total number of vertices is 60021. The second mesh uses the same aspect ratio for \(l_w\), with \(h_1 = D/512\). This yields a total of 63541 vertices. The third mesh also uses the same \(l_w\), and with \(h_1 = D/1024\), yielding 70494 vertices.

| Re  | \(N\) | \(x_{ctr}\) | \(y_{ctr}\) | \(\psi\) | \(\omega\) |
|-----|------|-------------|-------------|---------|---------|
| 100 | 65   | 0.6138      | 0.7359      | -0.1032 | 3.1362  |
|     | 129  | 0.6146      | 0.7360      | -0.1028 | 3.1270  |
|     | 257  | 0.6150      | 0.7363      | -0.1027 | 3.1345  |
| 1000| 65   | 0.5297      | 0.5652      | -0.1212 | 2.1102  |
|     | 129  | 0.5305      | 0.5658      | -0.1182 | 2.0584  |
|     | 257  | 0.5306      | 0.5656      | -0.1179 | 2.0573  |

Table 1. Main vortex characteristics for the lid-driven cavity flow.

The results for \(Re = 1000\) and the \(N = 129\) meshes are all within 1% of each other.
First, in Table 2, we present the drag coefficients, recirculation lengths, and separation angles obtained for the steady flows at Reynolds numbers 20 and 40. We compare them to the results of other works based on a combination of the Boltzmann equation and the finite-volume approach. Zarghami et al. (2012) use a finite-volume LBM formulation on a structured non-uniform mesh. Shu et al. (2014) and Wang, Shu, et al. (2015) use the LBFS, although the former use a structured curvilinear O-type mesh, and the latter use a structured non-uniform mesh. Yuan, Zhong, and Zhang (2015) use a gas kinetic scheme on a Cartesian mesh with local grid refinement. Their results are included in this comparison because their gas kinetic approach utilizes the BGK (Bhatnagar et al., 1954) collision model and the Maxwellian equilibrium distributions to compute the cell boundary fluxes, and so it is related to the LBFS. More details on this method can be found in Tian, Xu, Chan, and Deng (2007). The results of Sun, Shu, Wang, Teo, and Chen (2016) are presented because they also use a gas kinetic flux solver on a refined Cartesian mesh.

The overall agreement with the selected reference results is good, although our drag coefficients are lower. Also, there is some dispersion in the drag results selected for comparison, which is most likely due to the fact that, for free stream flow problems, domain size, domain boundary conditions, and the mesh selected are all factors that have a large impact on the solution. These parameters were not the same for all the works presented. In the present simulations, it was found that the domain size had a larger impact on the lower Reynolds number flows, and that using smaller domains increased the drag coefficients. This partially explains the lower drag coefficients that were obtained, since our 200D × 200D domain is much larger than that for all the other results presented. In fact, Zarghami et al. (2012) use the smallest domain height, 16D, and obtain a much larger drag coefficient for Re = 20.

In the present results, grid convergence is not fully attained, and further grid refinement to, perhaps, $h_1 = D/2048$ or $h_1 = D/4096$ would be necessary. Nevertheless, the results are consistent and the drag coefficient difference between successive meshes is small.

Unsteady flow fields were also simulated, at Reynolds numbers of 100 and 200, for which there is regular vortex shedding in the wake of the cylinder. It is therefore important, when comparing results, to include force coefficient variation amplitudes and a non-dimensional measure of the frequency of vortex shedding, namely the Strouhal number:

$$St = \frac{f D}{U_\infty}. \quad (29)$$

Our results are presented in Table 3. Comparison results from the same sources as in Table 2 are included. The
Table 3. Force coefficients for the unsteady flow over a circular cylinder in a free stream.

| Source                  | $Re = 100$ | $Re = 200$ |
|-------------------------|------------|------------|
|                         | $C_{d,m}$  | $C_{d,a}$  | $C_{l,a}$ | St  | $C_{d,m}$  | $C_{d,a}$  | $C_{l,a}$ | St  |
| Zarghami et al.         | 1.310      | 0.009      | 0.33      | 0.164 | 1.338      | 0.045      | 0.69      | 0.197 |
| Shu et al.              | 1.334      | 0.012      | 0.37      | 0.163 | 1.43       | 0.051      | 0.75      | 0.195 |
| Wang, Shu, Teo and Wu   | 1.352      | 0.012      | 0.37      | 0.163 | 1.370      | 0.049      | 0.714     | 0.195 |
| Yuan et al.             | 1.367      | 0.01      | 0.348     | 0.164 | 1.325      | 0.044      | 0.673     | 0.195 |
| Sun et al.              | 1.325      | 0.009      | 0.324     | 0.164 | 1.327      | 0.044      | 0.677     | 0.195 |
| Present method, mesh 1  | 1.326      | 0.009      | 0.325     | 0.164 | 1.327      | 0.044      | 0.677     | 0.195 |
| Present method, mesh 2  | 1.326      | 0.009      | 0.325     | 0.164 | 1.327      | 0.044      | 0.677     | 0.195 |
| Present method, mesh 3  | 1.326      | 0.009      | 0.325     | 0.164 | 1.327      | 0.044      | 0.677     | 0.195 |

Figure 7. Strouhal number for the flow over a circular cylinder at the coarsest mesh level.

subscripts $m$ and $a$ designate the mean and amplitude values respectively.

The mean drag coefficients and Strouhal numbers agree very well with the selected reference data. Our results also vary less with mesh level increments than for the lower Reynolds number simulations, therefore grid convergence requires less mesh refinement.

For the coarsest mesh level, simulations for $Re = 45$, 47, and 50 were also conducted. The intent was to verify whether or not the transition to unsteady flow near $Re = 46$ is captured correctly with our LBFS implementation. The results are displayed in Figure 7. The flow remains steady for $Re = 45$, and is unsteady for $Re = 47$, with a Strouhal number of 0.117.

7. NACA0012 airfoil in a free stream

The third validation case involves a turbulent flow over a NACA0012 airfoil at a Reynolds number of $5 \times 10^5$. This case was chosen for two reasons: first, to validate the combining of the LBFS method on an unstructured mesh with a finite-difference implementation of the Spalart-Allmaras turbulence model; and second, to provide a basis for comparing this case with earlier work by the current authors (Pellerin et al., 2015), which is based on a lattice Boltzmann solver applied to a multi-domain lattice. The computational results of Lockard, Luo, Milder, and Singer (2002) obtained with CFL3D are used as the baseline for this comparison.

The computational domain is a 100 m square with 51 vertices along each side. The airfoil is placed in the center of the domain and has a 1-m chord. In contrast with the cylinder meshes, the maximum aspect ratio of the cells near the wall is much larger. Furthermore, only one mesh is used for each of the angles of attack simulated. Although grid refinement necessarily has an impact on the results, the intention was not to study its influence, but rather validate the proposed method. Therefore, the meshes were created with the intention of providing a normal direction refinement near the wall similar to that of the lattices used by Pellerin et al. (2015), and also of requiring fewer vertices along the wall. Actually, the fact that regular lattices cannot be refined more in a given direction was one of the motivations for this work. Simulations were performed for three angles of attack: 0°, 3°, and 7°. The associated meshes were all constructed with two zones. The first zone, near the wall, contains thirty layers of structured triangles. The thickness of the triangles touching the wall is $1/16384$th of a chord. The second zone is a fully unstructured mesh joining the structured layers and the domain boundaries. These two zones are depicted in Figure 8 for the 0° mesh. Along the wall, the distances between vertices is $1/8192$th of a chord near the leading edge, and gradually increases to $1/1024$th of a chord near the trailing edge. For the 0° mesh, 1001 vertices are located on both the upper and lower surfaces of the airfoil. For the 3° and 7° meshes, 1201 vertices are located on the upper side to provide for more concentration near the leading edge, and 1001 vertices are located on the lower side. The resulting total number of vertices is 131049 for the 0° mesh, 139568 for the 3° mesh, and 139156 for the 7° mesh.

The simulations were run on a desktop computer with a Matlab program implementation of the proposed LBFS method. Local time-stepping was used, and the minimum time step was $6.36 \times 10^{-6}$ for all the meshes.
units are not specified because the time steps are in lattice units and stem from the $c = \Delta x / \Delta t = 1$ definition. They do not represent actual physical time, even though they are applied to the macroscopic conservation equations. Convergence of the force coefficients to the fourth decimal place required $2.4 \times 10^5$ iterations for the $0^\circ$ simulation and a total elapsed clock time of 5.6 hours. The $3^\circ$ simulation required $4.1 \times 10^5$ iterations and 11.4 hours, and the $7^\circ$ simulation $5.3 \times 10^5$ iterations and 13.2 hours. The results are presented in terms of drag and lift coefficients, pressure coefficients, and velocity profiles. The force coefficients are provided in Table 4. There is good agreement for lift at $3^\circ$ and $7^\circ$, with a difference of less than 0.7% from the results of the earlier work of Pellerin et al. (2015) and the CFL3D data of Lockard et al. (2002). The drag is in good agreement with the previous work of the current authors, however it still underpredicts the CFL3D results slightly. The pressure coefficients are presented in Figures 9–10. Good overall agreement with the CFL3D results is observed, with slight differences near the leading edge. The velocity profiles are depicted in Figures 11–12. The normalized coordinates $u_\infty / U_\infty$ and $y_\infty / l$ are defined in Pellerin et al. (2015). There is a close agreement with the authors’ earlier work, and the velocity

Table 4. Lift and drag coefficients for the NACA0012 airfoil at $Re = 5 \times 10^5$.

| AOA [°] | $C_l$ | CFL3D | Pellerin et al. | Present | CFL3D | Pellerin et al. | Present |
|---------|-------|-------|----------------|---------|-------|----------------|---------|
| 0       | 0.0000| 0.0000| 0.0000         | 0.0128  | 0.0100| 0.0095         |
| 3       | 0.3237| 0.3257| 0.3259         | 0.0130  | 0.0104| 0.0099         |
| 7       | 0.7449| 0.7457| 0.7418         | 0.0157  | 0.0128| 0.0122         |

Figure 8. Unstructured mesh for the NACA0012 airfoil at a $0^\circ$ angle of attack.
Finally, a contour plot of the ratio of turbulent viscosity to nominal fluid viscosity obtained with the present method is provided in Figure 13 for the 7° angle of attack. This plot provides an indication of the turbulence level in the boundary layer and the near wake of the airfoil. The maximum value of this ratio is approximately 75 and occurs downstream of the trailing edge, after the boundary layers of the pressure and suction sides of the airfoil have merged.

8. Discussion

In this article, we extended the application of the recently developed LBFS to turbulent flows around an airfoil. To do this, we applied a vertex-centered formulation of the method to unstructured meshes and coupled it to a finite-difference implementation of the Spalart-Allmaras turbulence model. Before solving the airfoil flows, the program was validated on standard test cases, without turbulence modeling. First, the lid-driven cavity flow was solved for Reynolds numbers of 100 and 1000 using a Delaunay-type unstructured mesh, the purpose being to verify the accuracy of the method on a mesh that contained low aspect ratio triangles. The resulting centerline velocity profiles and main vortex centers agree well with the data of Ghia et al. (1982), commonly used as a baseline. Second, low Reynolds number steady and unsteady flows around a circular cylinder were investigated, by including larger aspect ratio triangles in the mesh near the cylinder wall, in order to deviate from the Delaunay triangulation near a solid boundary. The program was tested on three meshes with increasing refinement, but in all cases the same aspect ratio was kept for the triangular cells near the wall. The resulting force coefficients and Strouhal numbers show good convergence with mesh refinement. They also compare well with the data selected from other works using finite-volume formulations in conjunction with either the lattice Boltzmann method or a gas kinetic approach.

To test the proposed methodology on turbulent airfoil flows, we constructed unstructured meshes that included larger aspect ratio triangular cells than for the circular cylinder flow test case, especially near the airfoil's mid-chord. The rationale for this mesh design was the necessity of correctly capturing the high-velocity gradients in the wall’s normal direction, while limiting the number of vertices in the streamwise direction where the velocity gradients are smaller – see Figure 8(d). Since the LBFS is an explicit unsteady solver, simulations were run until the flow became steady through the effect of the added turbulent viscosity, and the lift and drag coefficients converged to stable values. The resulting lift coefficients compare favorably with the selected reference data obtained with CFL3D, and earlier work of the current authors using a multi-domain LBM solver. The drag coefficients are in excellent agreement with the LBM solver, but still under-predict the CFL3D results slightly. The velocity profiles, too, are in close agreement with their LBM counterparts, and are still fuller than for CFL3D. This demonstrates a consistency between the finite-difference implementation of the Spalart-Allmaras turbulence model in the LBM and LBFS frameworks. In fact, aside from the differences in meshes, the velocity profile differences with CFL3D can most likely be explained by the choice of turbulence model and its computational implementation. For instance, the CFL3D results were produced with a finite-volume solver and the standard S-A model, whereas the Edwards variant was used in the present work. Also, since the contribution of the wall shear stress to the drag coefficient is larger than that of pressure, and the pressure coefficient profiles that we obtained matched the CFL3D profiles well, it is likely that the small differences in drag are also attributable to the turbulence...
Figure 11. Velocity profiles for the NACA0012 airfoil at a 0° angle of attack at chord coordinates $x^*/l = 0.0, 0.5, \text{ and } 1.0$. 
Figure 12. Velocity profiles for the NACA0012 airfoil at a 7° angle of attack at chord coordinates $x^*/l = 0.0, 0.5,$ and 1.0.
modeling and its impact on the velocity gradients near the wall. Nevertheless, the purpose of the current work was to test the combination of a turbulence model with an unstructured LBFS, and the results obtained indicate that the proposed method produces valid results. Further simulations could be run with the standard S-A model to verify the impact on the velocity profiles and force coefficients produced, but that is beyond the scope of the current work. For the same reason, no mesh sensitivity analysis was performed, and it is acknowledged that the mesh we constructed may not be optimal, even though its refinement in the wall’s normal direction is on a par with that of earlier work (Pellerin et al., 2015) using approximately one quarter the number of vertices. A goal of this work in fact was to use a smaller number of vertices while still providing for an adequate mesh.

While the simulation results that were obtained are satisfactory overall, the present LBFS implementation has two drawbacks. First, the clock time required to achieve steady airfoil flows is high. Even though our Matlab program may not perform computations as quickly as a higher language would, such as C++, the main reason for the long duration is the very small time steps. Near the airfoil wall, the mesh is very fine in the normal direction, and consequently the time steps chosen based on the local LBM lattice stencil size used for computing the cell fluxes scale accordingly. Therefore, even though the flow far from the airfoil stabilizes quickly, especially through the use of local time-stepping, the information travels much more slowly in the vicinity of the airfoil. One remedy for this problem could be the use of wall functions that would limit the normal mesh refinement required. Convergence time improvements could also be sought by using solution adaptive dynamically varying local time steps. The second drawback of the present LBFS derives from its vertex-centered formulation. While this formulation has the advantage of eliminating the need to calculate cell center gradients for calculating boundary fluxes, as in cell-centered formulations, it requires the use of more control surfaces for each control volume. Furthermore, at each control surface, the flux calculations require nine interpolation points for the d2q9 LBM. Overall, this makes the method memory-heavy because interpolation coefficients must be stored for all of these interpolation points.

The LBFS method has a number of advantages. First, the boundary conditions are implemented directly onto the macroscopic quantities, rather than onto the distribution functions as with standard LBM solvers. This simplifies the computational treatment of boundary conditions, since there are no missing post-streaming distributions to be determined. Also, when compared to conventional Navier-Stokes solvers for incompressible flows, the LBFS method does not require the solution of a separate equation for pressure. Moreover, since the density and velocity units used in the macroscopic conservation equations are the same as for the LBM used for the flux calculations, and because the flow physics are, in fact, based on the LBM, the macroscopic pressure is recovered through the \( p = \rho c_s^2 \) relationship. This was the equation used to calculate the pressure coefficients presented in Figures 9–10. To recover true macroscopic physical units, appropriate scaling should be applied through the Reynolds number and given free stream density and velocity. A further advantage of the LBFS is its numerical stability. While improving the stability of finite-volume Navier-Stokes solvers may rely on first or higher order upwind schemes for calculating convective term fluxes, no special scheme is required for the proposed LBFS. This is because the LBM streaming process is by definition applied from an upwind lattice speed direction, and the location of the interpolation points presented in Figure 1 is chosen such that the local d2q9 stencils cover large areas within their associated triangular cell. Finally, it is interesting to note that, even though the LBFS is based on the BGK collision operator, through the Chapman-Enskog expansion that makes the bridge with the Navier-Stokes equations, it does not suffer from the known instability observed when the relaxation coefficient approaches its maximum value of 2.
is this inherent instability that drove the derivation of more complex LBM collision operators. In fact, for the multi-domain LBM simulation of turbulent airfoil flows, Pellerin et al. (2015) resorted to the cascaded collision operator to maintain numerical stability. In that sense, the LBFS presented here constitutes a simpler computational implementation.

9. Conclusion

In this work, two elements were proposed for extending the applicability of the LBFS. Unstructured meshes that included large aspect ratio triangles for providing directional mesh refinement near walls were used. The method was also coupled to a turbulence model, which made it possible to solve two-dimensional turbulent flows around an airfoil. The results obtained are promising and open the way to possible future validations and developments. For instance, a very interesting test case would be the flow around iced airfoils. Also, validating the method on three-dimensional airfoil flows would broaden its applicability spectrum.

Disclosure statement

No potential conflict of interest was reported by the authors.

References

Bhatnagar, P. L., Gross, E. P., & Krook, M. (1954). A model for collision processes in gases. I. Small amplitude processes in charged and neutral one-component systems. Physical Review, 94(3), 511–525. https://doi.org/10.1103/PhysRev.94.511

Chen, S., & Doolen, G. D. (1998). Lattice Boltzmann method for fluid flows. Annual Review of Fluid Mechanics, 30, 329–364. https://doi.org/10.1146/annurev.fluid.30.1.329

Filippova, O., Succi, S., Mazzocco, F., Arrighetti, C., Bella, G., Hanel, D., & Dieter, H. (2001). Multiscale lattice Boltzmann schemes with turbulence modeling. Journal of Computational Physics, 170(2), 812–829. https://doi.org/10.1006/jcph.2001.6764

Geuzaine, C., & Remacle, J.-F. (2009). Gmsh: A 3-D finite element mesh generator with built-in pre- and post-processing facilities. International Journal for Numerical Methods in Engineering, 79(11), 1309–1331. https://doi.org/10.1002/nme.2579

Ghia, U., Ghia, K., & Shin, C. (1982). High-Re solutions for incompressible flow using the Navier-Stokes equations and a multigrid method. Journal of Computational Physics, 48, 387–411. https://doi.org/10.1016/0021-9991(82)90058-4

He, X., & Luo, L.-S. (1997). A priori derivation of the lattice Boltzmann equation. Physical Review E, 55(6), R6333–R6336. https://doi.org/10.1103/PhysRevE.55.R6333

Holman, D. M., Brionnand, R. M., & Modena, M. C. (2014). Aerodynamic analysis of the 2nd high lift prediction workshop by a lattice-Boltzmann method solver. In 32nd AIAA applied aerodynamics conference. Atlanta, GA: AIAA. https://doi.org/10.2514/6.2014-2568

Imamura, T., Suzuki, K., Nakamura, T., & Yoshida, M. (2005). Flow simulation around an airfoil by lattice Boltzmann method on generalized coordinates. AIAA Journal, 43(9), 1968–1973. https://doi.org/10.2514/1.7554

König, B., Fares, E., & Nöltting, S. (2014). Lattice-Boltzmann flow simulations for the HiLiftPW-2. In A.SciTech (Ed.), 52nd aerospace sciences meeting (pp. 1–19). American Institute of Aeronautics and Astronautics. https://doi.org/10.2514/6.2014-0911

Li, K., Zhong, C., Zhuo, C., & Cao, J. (2012). Non-body-fitted Cartesian-mesh simulation of highly turbulent flows using multi-relaxation-time lattice Boltzmann method. Computers and Mathematics with Applications, 63(10), 1481–1496. https://doi.org/10.1016/j.camwa.2012.03.080

Li, Y., Shock, R., Zhang, R., Chen, H., & Shih, T. (2005). Simulation of flow over an iced airfoil by using a lattice-Boltzmann method. In 43rd AIAA aerospace sciences meeting and exhibit (pp. 1–8). Reno, NV: American Institute of Aeronautics and Astronautics. http://dx.doi.org/10.2514/6.2005-1103

Lockard, D. P., Luo, L., Milder, S. D., & Singer, B. A. (2002). Evaluation of Powerflow for aerodynamic applications. Journal of Statistical Physics, 107(April), 423–478. https://doi.org/10.1023/A:1014539411062

Patil, D. V., & Lakshmisha, K. N. (2009). Finite volume TVD formulation of lattice Boltzmann simulation on unstructured mesh. Journal of Computational Physics, 228(14), 5262–5279. https://doi.org/10.1016/j.jcp.2009.04.008

Pellerin, N., LeClaire, S., & Reggio, M. (2014). Equilibrium distributions for straight, curved, and immersed boundary conditions in the lattice Boltzmann method. Computers and Fluids, 101, 126–135. http://dx.doi.org/10.1016/j.compfluid.2014.06.007

Pellerin, N., LeClaire, S., & Reggio, M. (2015). An implementation of the Spalart–Allmaras turbulence model in a multi-domain lattice Boltzmann method for solving turbulent airfoil flows. Computers and Mathematics with Applications, 70(12), 3001–3018. https://doi.org/10.1016/j.camwa.2015.10.006

Peng, G., Xi, H., Duncan, C., & Chou, S.-H. (1998). Lattice Boltzmann method on irregular meshes. Physical Review E, 58(4), R4124–R4127. https://doi.org/10.1103/PhysRevE.58.R4124

Peng, G., Xi, H., Duncan, C., & Chou, S.-H. (1999). Finite volume scheme for the lattice Boltzmann method on unstructured meshes. Physical Review E, 59(4), 4675–4682. https://doi.org/10.1103/PhysRevE.59.4675

Shu, C., Wang, Y., Teo, C. J., & Wù, J. (2014). Development of lattice Boltzmann flux solver for simulation of incompressible flows. Advances in Applied Mathematics and Mechanics, 6(4), 436–460. https://doi.org/10.4208/aamm.2014.4.s2

Stiebler, M., Tölke, J., & Krafczyk, M. (2006). An upwind discretization scheme for the finite volume lattice Boltzmann method. Computers & Fluids, 35(8–9), 814–819. https://doi.org/10.1016/j.compfluid.2005.09.002

Sun, Y., Shu, C., Wang, Y., Teo, C. J., & Chen, Z. (2016). An immersed boundary-gas kinetic flux solver for simulation of incompressible flows. Computers & Fluids. https://doi.org/10.1016/j.compfluid.2016.01.007
Tian, C. T., Xu, K., Chan, K. L., & Deng, L. C. (2007). A three-dimensional multidimensional gas-kinetic scheme for the Navier-Stokes equations under gravitational fields. *Journal of Computational Physics*, 226(2), 2003–2027. https://doi.org/10.1016/j.jcp.2007.06.024

Viggen, E. M. (2014). *The lattice Boltzmann method: Fundamentals and acoustics*. Trondheim, Norway: Norwegian University of Science and Technology.

Wang, Y., Shu, C., Teo, C. J., & Wu, J. (2015). An immersed boundary-lattice Boltzmann flux solver and its applications to fluid-structure interaction problems. *Journal of Fluids and Structures*, 54, 440–465. https://doi.org/10.1016/j.jfluidstructs.2014.12.003

Wang, Y., Shu, C., Teo, C. J., & Yang, L. M. (2016). An efficient immersed boundary-lattice Boltzmann flux solver for simulation of 3D incompressible flows with complex geometry. *Computers & Fluids*, 124, 54–66. https://doi.org/10.1016/j.compfluid.2015.10.009

Wang, Y., Yang, L., & Shu, C. (2015). From lattice Boltzmann method to lattice Boltzmann flux solver. *Entropy*, 17, 7713–7735. https://doi.org/10.3390/e17117713

Yuan, R., Zhong, C., & Zhang, H. (2015). An immersed-boundary method based on the Gas kinetic BGK scheme for incompressible viscous flow. *Journal of Computational Physics*, 296(C), 184–208. https://doi.org/10.1016/j.jcp.2015.04.052

Zarghami, A., Maghrebi, M. J., Ghasemi, J., & Ubertini, S. (2012). Lattice Boltzmann finite volume formulation with improved stability. *Communications in Computational Physics*, 12(1), 42–64. https://doi.org/10.4208/cicp.151210.140711a

**Appendix – Alternate Flux Construction**

The alternate approach we use here to construct the flux terms of Equation (18) is based on the method of Pellerin et al. (2014) for calculating missing post-streaming distributions applicable to boundary conditions. In that work, the missing distributions were approximated by equilibrium distributions streamed from ghost nodes placed at specific distances relative to the nominal lattice grid spacing; more precisely, these ghost nodes were positioned at a distance $\Delta x/(\Delta t/\Delta R)$ from the lattice site onto which the distributions were streamed. This process allowed us to avoid approximating the non-equilibrium part of the distributions, and, in turn, enabled us to effectively replace the collide- and-stream sequence with the sole streaming of the equilibrium distributions. The density and velocity necessary for calculating the equilibrium distributions were interpolated or extrapolated at the ghost nodes. This concept is introduced into the LBFS by assuming that the macroscopic interpolation points used in step 1 of the LBFS procedure are the ghost nodes. Steps 1 to 4 remain unchanged. In step 5, the original LBFS method uses Equation (16) to obtain the non-equilibrium part of the distributions, whereas in our alternate construction, they are simply obtained by subtracting the equilibrium distributions obtained in step 4 from those calculated in step 2. This is because the distributions from step 2 actually replace streamed distributions that include both an equilibrium and a non-equilibrium part. The relaxation coefficient of the BGK collision model is no longer present in the non-equilibrium calculation of step 5. This coefficient only appears in step 6 for calculating $\Pi$; however, it is replaced with $\Delta t^{\text{eff}}/\Delta R^{\text{eff}}$, which constitutes an effective relaxation. This change is necessary because the nominal $\Delta x$ now represents the distance to the ghost node, which makes the effective lattice spacing for the d2q9 stencil, $\Delta x^{\text{eff}}$, different from $\Delta x$. It is important to point out that $\Delta x^{\text{eff}}$ is only useful in the construction of the alternate model and does not specify any macroscopic interpolation point. It is calculated with the relationship given above as

$$
\Delta x^{\text{eff}} = \Delta x x^{\text{eff}} / \Delta R^{\text{eff}}. 
$$

(A.1)

Then, using Equation (12) and $c = \Delta x^{\text{eff}}/\Delta t = \Delta x^{\text{eff}} / \Delta t^{\text{eff}} = 1$, the effective relaxation coefficient is obtained from

$$
\nu = \frac{\Delta t^{\text{eff}}}{\Delta R^{\text{eff}}} \left( \frac{\Delta R^{\text{eff}} - 1}{2} \right),
$$

(A.2)

which is equivalent to

$$
\frac{\Delta t^{\text{eff}}}{\Delta R^{\text{eff}}} = 2 \left( 1 - \frac{\nu}{c^3/\Delta t} \right).
$$

(A.3)

Once the momentum fluxes have been computed in step 6 using the above relaxation coefficient, the macroscopic properties are updated at the vertices through step 8, as in the original LBFS, using the same macroscopic time step value. This time step, which is used in the forward Euler integration, could actually be different and is not strictly restricted to the time step defined in section 2. In fact, it only needs to conform to a CFL condition to maintain numerical stability.

We tested this alternate approach on the NACA0012 flow at $7^\circ$ in section 7, and obtained identical results to those obtained with the original LBFS. At first this may appear surprising; however, the algebraic development of Equation (14) in terms of $\nu$, $c^3$, $\Delta x$, and $f_{\text{eq}}^{\text{eff}}$ following the two approaches yields identical coefficients for $f_{\text{eq}}^{\text{eff}}(x, t)$ and $f_{\text{eq}}^{\text{eff}}(x - c^3 \Delta t, t - \Delta t)$. Although the method of Pellerin et al. (2014) was developed more empirically, actually it can also be tied to Equation (15) stemming from the Chapman-Enskog expansion. In fact, the absence of a relaxation coefficient in its finite-difference form, given by Equation (16) and used in step 5, is equivalent to

$$
f_{\text{eq}}^{\text{eff}}(x, t) = -\Delta R^{\text{eff}} f_{\text{eq}}^{\text{eff}}(x, t) - f_{\text{eq}}^{\text{eff}}(x - c^3 \Delta t, t - \Delta t) / \Delta t,
$$

(A.4)

where $\Delta R^{\text{eff}}$ is set to be equal to $\Delta t$. Using this equality, the standard kinematic viscosity-relaxation coefficient relationship of Equation (12) expressed in terms of the effective time step becomes

$$
\nu = c^3 \Delta t^{\text{eff}} / \left( \frac{\Delta R^{\text{eff}} - 1}{2} \right) = c^3 \Delta t^{\text{eff}} / \left( \Delta t^{\text{eff}} - 1 \right),
$$

(A.5)

which implies that

$$
\Delta t^{\text{eff}} = 2 \left( \Delta t - \frac{\nu}{c^3} \right).
$$

(A.6)

Therefore,

$$
\frac{\Delta t^{\text{eff}}}{\Delta R^{\text{eff}}} = 2 \left( \frac{\Delta t - \nu / c^3}{\Delta t} \right) = 2 \left( 1 - \frac{\nu}{c^3 / \Delta t} \right).
$$

(A.7)

reverses Equation (A.3), which did not initially originate from Equation (15). Furthermore, since the lattice speed $c$ is defined...
as unity regardless of the lattice time step,

\[
\frac{\Delta x}{\Delta x^{\text{eff}}} = \frac{\Delta t}{\Delta t^{\text{eff}}} = \frac{\Delta t^{\text{eff}}}{\Delta t^{\text{eff}}}, \quad \text{or} \quad \Delta x = \frac{\Delta x^{\text{eff}}}{\Delta t^{\text{eff}}/\Delta t_R^{\text{eff}}} \quad \text{(A.8)}
\]

Equation (A.8) confirms the choice of the ghost node location. There is no loss of generality in this derivation, even though \(\Delta t\) was chosen over \(\Delta t^{\text{eff}}\) in Equation (A.4) for approximating the material derivative of Equation (15) with a finite difference. The derivative is simply approximated over a shorter time span.

Overall, this alternate flux construction scheme only provides a marginal gain in computational time, as there is one less multiplication performed in step 5 of the LBFS procedure. Nonetheless, we chose to present it here to demonstrate that the applicability of the equilibrium streaming concept proposed in earlier work (Pellerin et al., 2014) is not limited to boundary conditions.