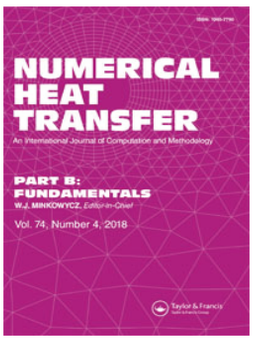

Taylor & Francis
Taylor & Francis Group

Numerical Heat Transfer, Part B: Fundamentals
An International Journal of Computation and Methodology

ISSN: 1040-7790 (Print) 1521-0626 (Online) Journal homepage: https://www.tandfonline.com/loi/unhb20

Applying the free-slip boundary condition with an adaptive Cartesian cut-cell method for complex geometries

Hua Tan

To cite this article: Hua Tan (2018) Applying the free-slip boundary condition with an adaptive Cartesian cut-cell method for complex geometries, Numerical Heat Transfer, Part B: Fundamentals, 74:4, 661-684, DOI: 10.1080/10407790.2018.1562770

To link to this article: https://doi.org/10.1080/10407790.2018.1562770

NUMERICAL HEAT TRANSFER, PART B: FUNDAMENTALS
2018, VOL. 74, NO. 4, 661–684
https://doi.org/10.1080/10407790.2018.1562770

Taylor & Francis
Taylor & Francis Group

Applying the free-slip boundary condition with an adaptive Cartesian cut-cell method for complex geometries

Hua Tan

School of Engineering and Computer Science, Washington State University-Vancouver, Vancouver, WA, USA

ABSTRACT

The slip condition at the interface of a multiphase flow can occur in situations including micro-and nano-fluidic flow, flow over hydrophobic surfaces, rising bubbles in quiescent liquid, and polymer extrusion processes. The aim of this work is to implement the free-slip boundary condition with an adaptive Cartesian grid method. The Navier-Stokes (NS) equations are solved by a cell-centered collocated finite volume method with adaptive mesh refinement. The arbitrarily-shaped solids imbedded in the computational domain are treated by the cut-cell method where the geometric properties of cut-cells near the boundary are computed through robust geometric operations. In discretized NS equations, the second-order-accurate center difference method is used to estimate the surface fluxes of the regular Cartesian cells in the bulk region, whereas the least-squares method is used to estimate the fluxes of the cut-cells near the boundary. A local coordinate system aligned with the normal and tangential directions of the solid boundary is defined for each cut-cell in order to properly implement the free-slip condition. The tangential velocities at the curved solid boundary are obtained using the free-slip condition and the principal curvatures of the solid surface. The proposed numerical method is implemented in the open-source code *Gerris*. Numerical tests have been carried out to validate our method. The tests confirm the excellent performances of the proposed method. Although our work focuses on the free-slip condition, the extension of the proposed method to more general slip conditions is straightforward.

ARTICLE HISTORY
Received 17 September 2018
Accepted 19 December 2018

1. Introduction

Over the past decades, numerical methods based on non-boundary-conforming Cartesian grids to solve inviscid and viscous flows with complex immersed solid boundaries become increasingly popular in the computational fluid dynamics (CFD) community, since they have the potential to truly automate CFD analysis with good accuracy, high efficiency, robust reliability, and minimum human intervention [1–11]. Furthermore, the simplicity of Cartesian mesh facilitates the implementation of adaptive mesh refinement (AMR) techniques [12–14] in numerical methods that allow an automatic adjustment of the grid resolution according to evolution of the characteristics of flow structures as well as geometric features of embedded solids during the solution process. The combination of Cartesian grids and AMR leads to a powerful approach for efficiently solving flow problems with multiple space- and time-scales in complex geometry. It is especially useful when a priori knowledge of the flow phenomena is not even known before the simulation. As a

CONTACT Hua Tan hua.tan@wsu.edu School of Engineering and Computer Science, Washington State University-Vancouver, 14204 NE Salmon Creek Ave, Vancouver, WA, 98686, USA
Color versions of one or more of the figures in the article can be found online at www.tandfonline.com/unhb.

result, numerous AMR based Cartesian grid methods have been proposed to address a wide range of fluid flow problems in the literature in recent years [15–29]. This trend of adopting adaptive Cartesian grid methods for CFD analysis is emerging in the CFD industry too. For example, a number of new commercial CFD codes completely based on adaptive Cartesian grids have been made available in the CFD software market over the past few years [30,31].

As the solid boundaries are usually not aligned with grid lines of the Cartesian mesh, the main challenge of Cartesian grid methods is how to deal with arbitrary boundaries. Depending on how the boundary conditions are imposed on solid surfaces, the existing Cartesian grid methods are often classified into two categories: the immersed boundary methods (IBM) and cut-cell methods [32]. IBM is pioneered by Peskin [1] and has now become an established approach for flow simulations involving complex stationary or moving boundaries through great efforts made by many researchers [3,8,33–37], just to name a few. Generally IBMs can be divided into two groups [32]: (a) continuous forcing approach where the governing equations are modified by adding force terms to account for the effect of the immersed boundaries; and (b) discrete forcing approach where the desired boundary condition is imposed through the use of a layer of ghost cells in the solid without modifying the governing equations. In the first approach the boundary is smeared over several cells (hence a diffuse boundary), whereas a sharp boundary can be represented in the second approach. As Mittal and Iaccarino point out in [32], the inherent disadvantage for these Cartesian grids based IBMs is no strict conservation laws for the cells in the vicinity of the immersed boundary. As a result, the Cartesian cut-cell methods are proposed to strictly conserve mass, momentum, and energy even at the immersed boundaries by adopting the finite-volume approach to discretize the governing equations [4–7, 15, 16, 20–23, 27–29].

In cut-cell methods, the cells intersected by the solid boundaries (usually referred to as cut-cells) are truncated so that they conform to the boundary shape. The discretization in the cut-cells needs special treatments, whereas the discretization in regular cells away from the boundaries follows standard five-point stencil (in 2D) or seven-point stencil (in 3D). The cut-cell methods begin with the geometric calculation (e.g., open volume and area fractions, centroids) of cut cells. Recent advances in computational geometry provide efficient and robust algorithms for such computations [38]. As the Cartesian grids are cut arbitrarily by complex solid boundaries, it is inevitable to have a number of small cut cells after truncation. To avoid a too small time step due to the CFL restriction condition enforced by small cut cells, it is mandatory to properly merge them with their neighboring cells. The cell-merging techniques as well as computations of geometric properties of cut cells often pose a challenge in implementation of the cut-cell methods in computer programs, especially in 3D geometry. In recent years, there has been a growing number of researchers who have successfully developed cut-cell methods for 3D flow simulations involving complex geometries [6,20–23,27–29].

Most published studies on Cartesian grids mentioned early impose the standard no-slip boundary condition at the solid surface. It is well known that the slip condition at the multiphase interface (e.g., liquid-solid, liquid-gas, gas-solid, etc.) of multiphase flows can occur in some cases such as micro- and nano-scales [39], hydrophobic surfaces [40,41], rising bubbles in quiescent liquid [42,43], and polymer extrusion processes [44]. Very limited studies on Cartesian grid methods, however, are reported on imposing slip conditions at the solid boundary. Hartmann *et al.* [21,22] implemented Neumann boundary conditions using a cut-cell/ghost-cell method. In their method, ghost cells are created to implement Dirichlet and Neumann conditions along the solid boundary by reflecting the cut cells along the boundary segments. Recently, Kempe *et al.* [43] developed a numerical method to successfully impose the free-slip boundary condition in the framework of continuous forcing IBM. In fact, the enforcement of the free-slip boundary condition at the curved boundaries is not trivial and requires considerable efforts. To our best knowledge, there has been no report of implementation of free-slip boundary condition using cut-cell methods.

The objective of this paper is to develop a numerical method to impose the free-slip boundary condition at complex solid boundaries with the Cartesian cut-cell method. The open-source code *Gerris* [20,23] is adapted to achieve the objective. In the past, we have successfully modified the original code for simulation of microscopic multiphase flows dominated by surface tension [45–48]. *Gerris* uses the Cartesian grid based finite-volume method with octree-based AMR technique to solve the Navier-Stokes equations. In our cut-cell method for treating the solid geometries, the least-squares method is employed to calculate surface fluxes through the partial faces of the cut-cells and boundary segments inside the cut cells according to the boundary conditions at the solid surface. To implement the free-slip condition at the solid wall, the velocity vectors are projected into a local coordinate systems based on the normal and tangential directions of the local solid surface. The boundary velocities can then be obtained using the free-slip condition and the principal curvatures of the solid surface. Once the boundary velocities are known, surface fluxes in cut-cells can be computed using the face centered gradients obtained by the proposed least-squares method. This paper is organized as follows. First, the governing equations and boundary conditions are given in Section 2. Subsequently, the description of the numerical method of implementing the slip boundary condition is presented in Section 3. Then, three numerical examples are presented to validate the proposed method in Section 4. Finally, the conclusions are drawn in Section 5.

2. Governing equations

We consider unsteady incompressible, Newtonian fluids which are governed by the continuity equation

$$\nabla \cdot \mathbf{u} = 0 \tag{1}$$

and the momentum equation

$$\rho \frac{\partial \mathbf{u}}{\partial t} + \rho(\mathbf{u} \cdot \nabla)\mathbf{u} = -\nabla p + \nabla \cdot \boldsymbol{\tau} \tag{2}$$

where $\mathbf{u} = (u, v, w)$ is velocity vector in Cartesian coordinate system (x, y, z). t is the time, p is the pressure, ρ is the density. ∇ is the gradient operator. $\boldsymbol{\tau}$ is the deviatoric stress tensor which obeys the following law for Newtonian fluids

$$\boldsymbol{\tau} = 2\mu\mathbf{D} \tag{3}$$

with μ the fluid viscosity and $\mathbf{D}$ the rate-of-strain tensor given by

$$\mathbf{D} = \frac{1}{2}(\nabla\mathbf{u} + \nabla\mathbf{u}^T) \tag{4}$$

The specification of boundary conditions is mandatory to solve Eqs. (1) and (2). As mentioned before, most solutions in the literature on Cartesian grid methods are based on Dirichlet type no-slip condition at the solid boundary

$$\mathbf{u} = 0 \tag{5}$$

This imposes that the fluid adheres to the wall, together with the impermeability condition.

The free-slip boundary condition on the impermeable solid wall can be expressed by

$$\mathbf{u} \cdot \boldsymbol{\eta} = 0 \tag{6}$$

$$(\boldsymbol{\tau} \cdot \boldsymbol{\eta}) \times \boldsymbol{\eta} = 0 \tag{7}$$

where $\boldsymbol{\eta}$ is the unit normal vector of the local solid surface. Eq. (6) implies that the flow velocity in the normal direction of the boundary is zero, i.e., impermeable boundary. Eq. (7) means that

the tangential stress of fluid at the boundary is zero, i.e., no shear force is exerted from the fluid to the boundary.

When the shear force at the boundary is not exactly zero, such boundary is often referred to as the slip boundary. The slip condition is generally characterized by a slip length which can be thought as a fictitious distance between the physical solid boundary and an imaginary surface inside the solid where the tangential velocity is zero. The mathematical form of the general slip condition on the impermeable boundary can be written as

$$\mathbf{u} \times \boldsymbol{\eta} = \frac{L_s}{\mu}(\boldsymbol{\tau} \cdot \boldsymbol{\eta}) \times \boldsymbol{\eta} \tag{8}$$

where L_s is the slip length or friction coefficient. Eq. (8), often called the Navier slip law, represents a relationship between slip velocity at the solid boundary and shear stress at the wall. When $L_s = 0$, Eq. (8) implies the velocity vector in tangential direction of the solid boundary is zero, i.e., no-slip condition (Eq. (5)). When L_s approaches to infinity, Eq. (8) becomes identical as Eq. (7), i.e., free-slip condition.

3. Numerical method

As we mentioned early, the proposed numerical method is implemented using the open-source code *Gerris* [20,23]. The code solves the Navier-Stokes equations (Eqs. (1) and (2)) using a cell-centered collocated finite volume method with an octree-based (fully threaded tree data structure [18]) AMR Cartesian grid. The solid geometry embedded in the computational grid is handled by the cut-cell method. In this section, we first give a brief overview of the numerical method of *Gerris* for solving NS equations. Interested readers can find more details in [20,23]. We then explain how to calculate the gradient fluxes through the partial faces of the cut-cells and boundary segments inside the cut cells according to the boundary conditions at the solid surface. The numerical scheme of imposing the free-slip boundary condition is implemented using a local coordinate system based on the orientation of the local solid boundary. Although we describe the numerical implementation using a uniform Cartesian grid, the extension to octree-based adaptive grid is straightforward, following the same approach detailed in [20].

3.1. Solution scheme of governing equations

Eq. (2) is discretized using a time-staggered formulation on the collocated Cartesian grid (primitive variables located at the cell center) as

$$\rho^{n+1/2}\left[\frac{\mathbf{u}^{n+1}-\mathbf{u}^{n}}{\Delta t} + (\mathbf{u}^{n+1/2}\cdot\nabla)\mathbf{u}^{n+1/2}\right] = -\nabla p^{n+1/2} + \nabla\cdot\left[\mu^{n+1/2}(\mathbf{D}^{n}+\mathbf{D}^{n+1})\right] \tag{9}$$

where Δt is the time increment of time step, the superscript denotes the time-step number. Eq. (9) is solved using a classical fractional-step projection method. First, a provisional velocity $\mathbf{u}^*$ is computed using

$$\rho^{n+1/2}\left[\frac{\mathbf{u}^{*}-\mathbf{u}^{n}}{\Delta t} + (\mathbf{u}^{n+1/2}\cdot\nabla)\mathbf{u}^{n+1/2}\right] = \nabla\cdot\left[\mu^{n+1/2}(\mathbf{D}^{n}+\mathbf{D}^{*})\right] \tag{10}$$

and the velocity $\mathbf{u}^{n+1}$ at time $n+1$ can be obtained by correcting provisional velocity $\mathbf{u}^*$

$$\mathbf{u}^{n+1} = \mathbf{u}^{*} - \frac{\Delta t}{\rho^{n+1/2}}\nabla p^{n+1/2} \tag{11}$$

where superscript * denote the provisional quantities. Since $\mathbf{D}^*$ of Eq. (10) is an unknown term depending on $\mathbf{u}^*$ as expressed by Eq. (4), Eq. (10) is re-arranged by moving all provisional terms

to the left hand side of the equation and the rest of terms obtained from previous time step to the right hand side (RHS). Once applying the finite volume method, we can obtain the following integral equation for each computational cell

$$\int_{cv} \frac{\rho^{n+1/2}}{\Delta t}\mathbf{u}^* dV + \oint_{cs} \mu^{n+1/2}(\mathbf{n}\cdot\mathbf{D}^*)dS = \oint_{cs} \mu^{n+1/2}(\mathbf{n}\cdot\mathbf{D}^n)dS + \int_{cv} \frac{\rho^{n+1/2}}{\Delta t}\mathbf{u}^n dV - \oint_{cs} \rho^{n+1/2}\mathbf{u}^{n+1/2}(\mathbf{n}\cdot\mathbf{u}^{n+1/2})dS \tag{12}$$

where *cv* and *cs* denote the cell volume and the cell surface, respectively. $\mathbf{n}$ is the outward normal vector of the cell surface. In Eq. (12), the volume integrals of viscous diffusion and advection terms are converted to the surface integrals using Gauss's theorem. The viscous term in the discretized momentum equation is solved by the implicit Crank–Nicholson scheme with a second-order accuracy in time. The advection term in Eq. (12) is estimated explicitly using a second-order upwind conservative Godunov scheme [20]. Eq. (12) is a Helmholtz-type equation and is solved using a variant of the multilevel Poisson solver [23]. The time step (Δt) is controlled by the Courant-Friendrichs-Lewy (CFL) condition to ensure the numerical stability.

In the second projection step, the pressure $p^{n+1/2}$ at time $n+1/2$ is obtained by taking the divergence of Eq. (11) and using the incompressibility constraint Eq. (1) as follows

$$\nabla\cdot\left[\frac{\Delta t}{\rho^{n+1/2}}\nabla p^{n+1/2}\right] = \nabla\cdot\mathbf{u}^* \tag{13}$$

The resulted Poisson equation is solved using the octree-based multigrid scheme [20]. Once the pressure $p^{n+1/2}$ is known, the divergence-free velocity field $\mathbf{u}^{n+1}$ can be obtained by correcting provisional velocity $\mathbf{u}^*$ with Eq. (11). To avoid the classic problem of decoupling of the pressure and velocity due to collocated grids, an approximate projection method is used here [20]. An auxiliary face-centered velocity $\mathbf{u}_f^*$ is first calculated by averaging the cell-centered velocity field $\mathbf{u}^*$, and then used to calculate the divergence in RHS of Eq. (13). After the cell-centered pressure $p^{n+1/2}$ at time $n+1/2$ is solved using Eq. (13), the face-centered velocity $\mathbf{u}_f^{n+1}$ at time $n+1$ is obtained by correcting auxiliary face-centered velocity $\mathbf{u}_f^*$ using face-centered pressure gradient.

In cut-cell methods, cells in the computational domain can be classified into three groups as shown in Figure 1a: (1) fluid cells that are located completely in the fluid region; (2) solid cells that are located in the solid region and usually removed from the computation; and (3) cut cells that are intersected by the solid boundary. For cut-cells, robust geometric operations are used in *Gerris* to compute geometric properties (e.g., open-volume fraction, area fractions of cell faces, centroid of cell volume, and midpoints of cell faces, etc.). When estimating surface fluxes to solve Eq. (12), the geometric property data of cut-cells allows to reconstruct the boundary using a series of piecewise linear segments similar to the volume-of-fluid approach. The primitive variables are stored at the centroid of the truncated cells, whereas these variables are stored at the cell center of regular Cartesian cells that are not cut by the solid boundary. The surface fluxes through the faces bounding the cut-cells are estimated at the midpoints of these faces. Small cut-cells are inevitable for arbitrarily complex solid boundaries. To avoid too restrictive time step required by the CFL stability due to small cut cells, any small cut cell is combined with its largest neighbor cell into a single merged cell [20]. Therefore, the CFL condition is not controlled by the small cut cell but its merged cell.

3.2. Evaluation of viscous fluxes for cut cells

Solution of Eq. (12) requires the integration of surface fluxes (i.e., momentum and viscous diffusion) over cell surfaces. The upwind conservative Godunov scheme is used to calculate the momentum fluxes as detailed in [20]. For estimation of the viscous fluxes, we use the least-

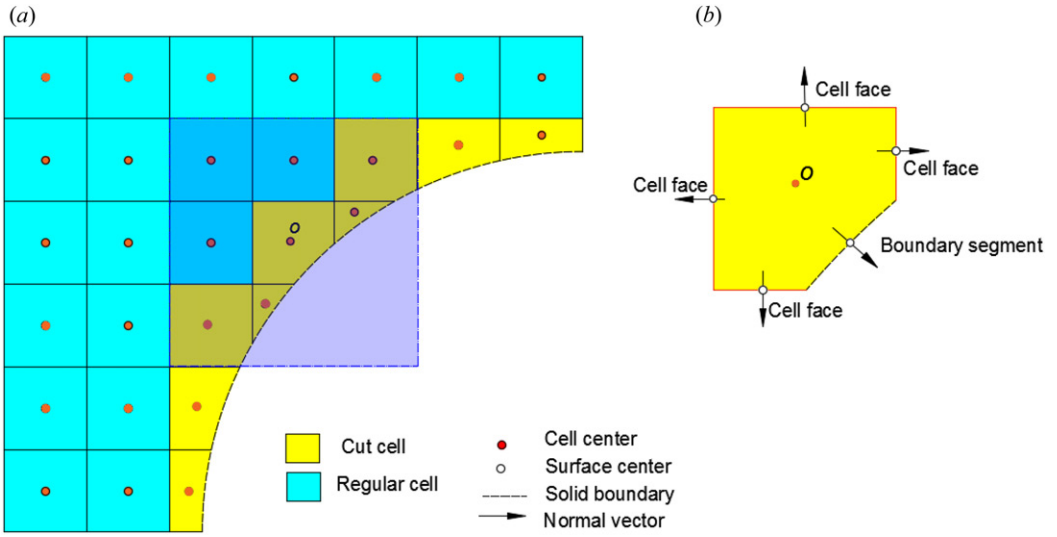

Figure 1. Illustration of computational cells near the immersed solid boundary in the cut-cell method: (*a*) the computational stencil for the least-squares method for estimation of surface fluxes through cell faces of cell *o* (and the stencil is highlighted by the gray shaded region), and (*b*) surfaces enclosing the cut-cell where the arrows represent the outward normal direction of surfaces.

squares method due to its flexibility in terms of the local adaptive mesh topology and the immersed boundary. When a cell face is between two regular Cartesian cells, the viscous fluxes can be estimated by the second-order accurate center difference method. However, special treatment needs to be developed to calculate surface fluxes for a cut-cell. Surfaces enclosing any cut-cell can be classified into two types as shown in Figure 1*b*: Cartesian cell faces (that may be cut by the boundary) and a planar portion of the solid boundary inside the cut cell. Therefore, for cut-cells we can split the computation of integration of viscous fluxes over surfaces into fluxes through cell faces and fluxes through the segment of the solid boundary. If we use the symbol ϕ to represent a component of velocity vector **u**, the viscous term in Eq. (12) can be written for each cut-cell as

$$\oint_{cs} \mu(\mathbf{n}\cdot\nabla\phi)ds = \mu\sum_{d} s_d\nabla_d\phi + \mu\int_{\Gamma}\mathbf{n}\cdot\nabla\phi ds \tag{14}$$

where d is the direction of the Cartesian coordinate, s_d is the surface fraction ratio of the cell face in the direction d that represents the fluid portion of the cell face, $\nabla_d\phi$ represent the gradient evaluated at the midpoint of a cell face in the direction d, and Γ is the solid boundary segment in the cut-cell. In RHS of Eq. (14), the first part corresponds to the viscous fluxes through the faces of the regular Cartesian cell and the second part corresponds to the fluxes through the planar segment of the solid boundary. In the following sections we describe the proposed approaches to estimate the surface fluxes using the two dimensions, however it is straightforward to extend these schemes to three dimensions.

3.2.1. Computation of gradients at cell faces

The least-squares scheme is used to estimate the gradient flux of ϕ at the midpoint of the cell face. For the two-dimensional case, in a local region consisting of a group of cells immediately surrounding the current cut-cell o as shown in Figure 1a, variable ϕ is assumed to vary according to the following function

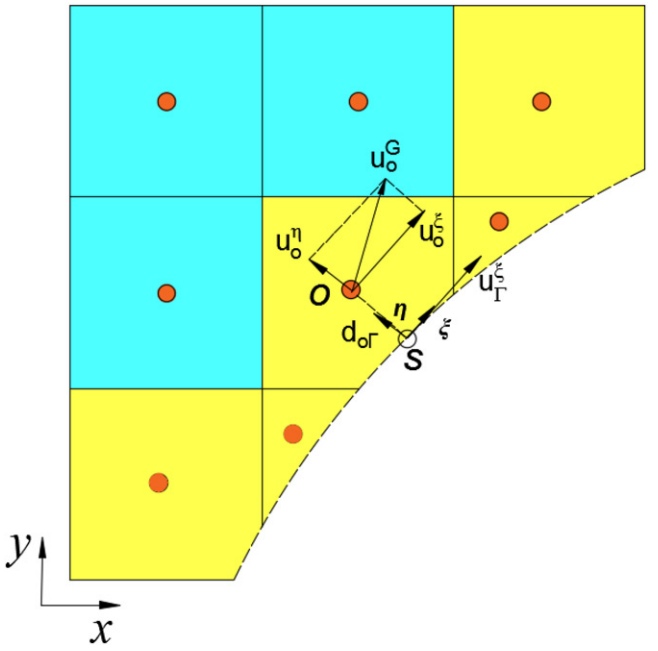

Figure 2. Schematic of estimation of surface fluxes through the solid boundary segment using the least-squares method in the local coordinate system $L = (\boldsymbol{\xi}, \boldsymbol{\eta})$ that is defined by the tangential and normal directions of the local boundary. The velocity vector u_o^G at cell center o needs to be converted into $u_o^L = (u_o^\xi, u_o^\eta)$ in the local system L.

$$\phi(x, y) = a_0 + a_1 x + a_2 y + a_3 xy \tag{15}$$

where a_0 to a_3 are unknown coefficients. These coefficients can be obtained by first minimizing a cost function given by

$$\Pi = \sum_{j \in C_o} \left[\phi_j - \phi(x_j, y_j)\right]^2 \tag{16}$$

where ϕ_j is the value of ϕ at the centroid of cell j, C_o represents a group of immediate neighbors associated with the cut-cell o. x_j and y_j are the coordinates of the centroid of cell j, so $\phi(x_j\ y_j)$ is the value of Eq. (15) at the cell centroid. Π represents the sum of the squares of the errors between values of ϕ of cells in group C_o and values obtained by Eq. (15). The coefficients a_j in Eq. (15) can be obtained by solving $\partial\Pi/\partial a_i = 0$, i.e.,

$$\begin{bmatrix} \sum j & \sum x_j & \sum y_j & \sum x_j y_j \\ \sum x_j & \sum x_j^2 & \sum x_j y_j & \sum x_j^2 y_j \\ \sum y_j & \sum x_j y_j & \sum y_j^2 & \sum x_j y_j^2 \\ \sum x_j y_j & \sum x_j^2 y_j & \sum x_j y_j^2 & \sum x_j^2 y_j^2 \end{bmatrix} \begin{bmatrix} a_0 \\ a_1 \\ a_2 \\ a_3 \end{bmatrix} = \begin{bmatrix} \sum \phi_j \\ \sum \phi_j x_j \\ \sum \phi_j y_j \\ \sum \phi_j x_j y_j \end{bmatrix} \tag{17}$$

Once the coefficients are solved from the above equation, we can easily obtain the gradient of ϕ at the midpoint m of the cell face using Eq. (15)

$$\left.\frac{\partial \phi}{\partial x}\right|_m = a_1 + a_3 y_m \left.\frac{\partial \phi}{\partial y}\right|_m = a_2 + a_3 x_m \tag{18}$$

where x_m and y_m are the coordinates of the midpoint m of the cell face in the direction d. With the gradient of ϕ known, the gradient flux through the cell face the can be calculated using the first part of RHS of Eq. (14).

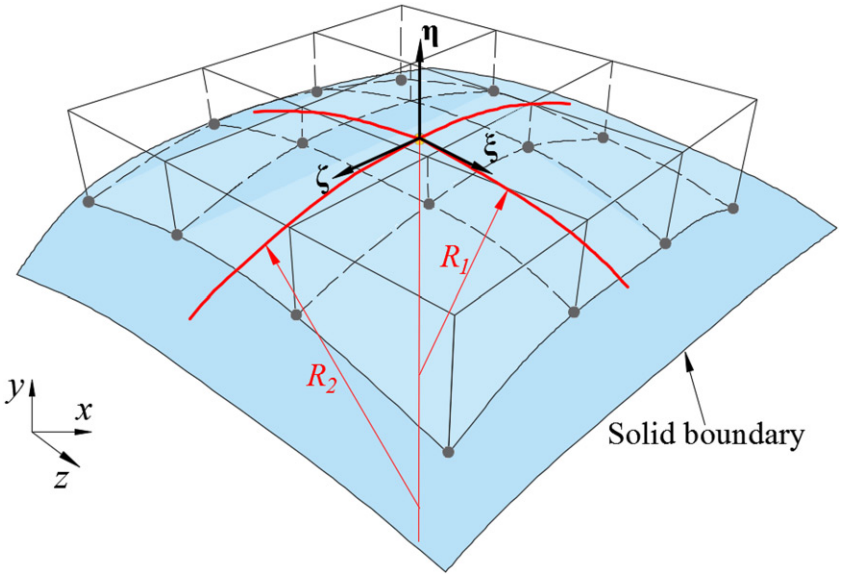

Figure 3. The local coordinate system (**η**, ξ, ζ) is built based on the normal and tangential directions of the solid segment in a cut-cell. The tangential directions are along the principal curvature directions. R_1 and R_2 are curvature radii of the curved solid boundary.

3.2.2. Computation of gradients at boundary segment

We use a different approach to estimate the gradient flux at the midpoint s of the solid boundary segment in the current cut-cell o, as the boundary condition at the solid wall needs to be considered. To facilitate the implementation of the free-slip condition (discussed in Section 3.3), we introduce a local coordinate system L for each boundary segment in cut-cells with two orthonormal basis vectors $\boldsymbol{\xi}$ and $\boldsymbol{\eta}$ being the local tangential and normal directions of the solid wall respectively in 2D case as shown in Figure 2. Suppose the value of ϕ at the midpoint of the boundary segment is known as ϕ_s from the boundary condition. If ϕ represents a component of a vector (i.e., velocity), we need first to convert the vector into the local coordinate system L as shown in Figure 2. So the value of ϕ in the vicinity of the solid boundary can be evaluated in the local system L using the Taylor series expansion about the midpoint s of boundary segment Γ as

$$\phi(\xi,\eta) = \phi_s + \left.\frac{\partial\phi}{\partial\xi}\right|_s \Delta\xi + \left.\frac{\partial\phi}{\partial\eta}\right|_s \Delta\eta \tag{19}$$

where $\Delta\xi$ and $\Delta\eta$ are the distance between the location (ξ, η) and the midpoint s in the local tangential and normal directions, respectively. Note: we need to convert the coordinates of centroids in the cell group C_o from the global system G into the local system L. As in Section 3.2.1, we still use the least-squares scheme to compute the gradients $\partial\phi/\partial\xi$ and $\partial\phi/\partial\eta$ at the point s. After choosing a group of cells immediately neighboring the current cut-cell o, we minimize the following residual function

$$\Pi_s = \sum_{j\in C_o} w_j\left[\phi_j - \phi\left(\xi_j,\eta_j\right)\right]^2 \tag{20}$$

where w_j is a weighting factor for neighbor cell j. We choose $w_j = 1/((\Delta\xi)^2+(\Delta\eta)^2)$. The choice of weighting factor places the greater weight on the neighbors that are nearer the segment midpoint s of the cut-cell o in the stencil. The minimization of Eq. (20) yields

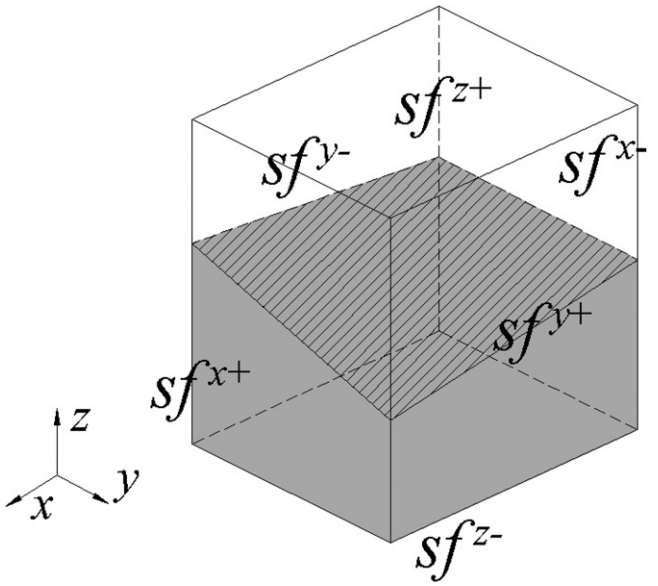

Figure 4. Schematic of the open-surface fractions of cell faces of a cut-cell where the solid volume is highlighted by the gray shaded region.

$$\begin{bmatrix} \sum w_j(\Delta\xi_j)^2 & \sum w_j\Delta\xi_j\Delta\eta_j \\ \sum w_j\Delta\xi_j\Delta\eta_j & \sum w_j(\Delta\eta_j)^2 \end{bmatrix} \begin{bmatrix} \left.\frac{\partial\phi}{\partial\xi}\right|_s \\ \left.\frac{\partial\phi}{\partial\eta}\right|_s \end{bmatrix} = \begin{bmatrix} \sum w_j\Delta\xi_j\Delta\phi_j \\ \sum w_j\Delta\eta_j\Delta\phi_j \end{bmatrix} \tag{21}$$

with $\Delta\phi_j = \phi_j - \phi_s$. Once the derivative $\partial\phi/\partial\eta$ along the wall-normal direction is solved from the above equation, the gradient flux through the boundary segment Γ can be calculated by $\mu(\partial\phi/\partial\eta)\Gamma_A$ (i.e., the second part of RHS of Eq. (14)), where Γ_A is the surface area of Γ.

3.3. Scheme for imposing the free-slip condition

The free-slip boundary condition on the impermeable solid wall requires the velocity vector component in the wall-normal direction is zero and the tangential components of the stress vector $\boldsymbol{\tau}\cdot\boldsymbol{\eta}$ at the wall vanish, as indicated in Eqs. (6) and (7). As we mentioned early, the complex solid wall is represented by a series of piecewise linear segments in our Cartesian cut-cell method. To implement the free-slip condition, we create a local coordinate system L that is aligned with the normal and tangential directions of the boundary segment as shown in Figure 3. L has the orthogonal basis vectors $\boldsymbol{\eta}$, $\boldsymbol{\xi}$, and $\boldsymbol{\zeta}$, where $\boldsymbol{\eta}$ is the outward unit vector normal to the wall, whereas $\boldsymbol{\xi}$ and $\boldsymbol{\zeta}$ are two unit vectors tangential to the wall. For each cut-cell, the open-surface fractions (i.e., ratio of the area open for flow to the total area) of six cell faces are calculated and stored at the beginning of the simulation as shown in Figure 4. The wall-normal direction $\boldsymbol{\eta}$ in the cut-cell can be expressed in terms of the global Cartesian coordinate system G using the surface fractions as following

$$\boldsymbol{\eta} = \frac{1}{A}\left(sf^{x+} - sf^{x-}, sf^{y+} - sf^{y-}, sf^{z+} - sf^{z-}\right)^T \tag{22}$$

with

$$A = \sqrt{(sf^{x+} - sf^{x-})^2 + (sf^{y+} - sf^{y-})^2 + (sf^{z+} - sf^{z-})^2} \tag{23}$$

where *sf* is the open-surface fraction of the cell faces, superscrits +, −, are the positive- and negative-direction of the global Cartesian axis *x*, *y*, *z*, respectively. To determine the two tangential vectors $\boldsymbol{\xi}$ and $\boldsymbol{\zeta}$, we first arbitrarily choose one of the global axis directions as an initial guess of the tangential vector, e.g., $\boldsymbol{\xi}_s = (1, 0, 0)$, and then $\boldsymbol{\xi}$ can be calculated by $\boldsymbol{\eta}\times(\boldsymbol{\xi}_s\times\boldsymbol{\eta})$. Note: if $\boldsymbol{\xi}_s\times\boldsymbol{\eta} = 0$, we choose another axis direction as the initial guess. Once the tangential vector $\boldsymbol{\xi}$ is known, the other tangential vector $\boldsymbol{\zeta}$ can be easily determined by $\boldsymbol{\eta}\times\boldsymbol{\xi}$.

A key to implement the free-slip condition is to determine the tangential velocities u_Γ^ξ and u_Γ^ζ at the solid boundary in terms of the local coordinate system *L*. The local normal velocity$u_\Gamma^\eta = 0$ for the impermeable wall (Eq. (6)). Once $u_\Gamma^\xi$and u_Γ^ζ are determined from the boundary conditions, the velocity vector $\mathbf{u}^L$ at the wall can be expressed by $(0, u_\Gamma^\xi, u_\Gamma^\zeta)^T$. Then, we can use the basis vectors $\boldsymbol{\eta}$, $\boldsymbol{\xi}$, and $\boldsymbol{\zeta}$ to transform the velocity vector $\mathbf{u}^L$ in the local system *L* to $\mathbf{u}^G$ in the global system *G*. Finally, the viscous fluxes through the wall in each cut-cell can be computed as discussed in Section 3.2.2. Therefore, the implementation of the free-slip condition amounts to properly calculate the tangential$u_\Gamma^\xi$and u_Γ^ζin the local coordinate system *L* [43].

For the free shear-stress condition, Eq. (7) can be decomposed into two equations for two tangential directions as

$$(\boldsymbol{\tau}\cdot\boldsymbol{\eta})\cdot\boldsymbol{\xi} = 0 \text{ and } (\boldsymbol{\tau}\cdot\boldsymbol{\eta})\cdot\boldsymbol{\zeta} = 0 \tag{24}$$

If the solid boundary is planar, the shear stress at the $\boldsymbol{\xi}$-$\boldsymbol{\eta}$ plane can be written as the wall-normal derivative of the tangential velocity in the local system *L*, i.e., $\tau_{\xi\eta} = \mu\partial u^\xi/\partial\eta$. So the first equation of (24) becomes $\tau_{\xi\eta} = \mu\partial u^\xi/\partial\eta = 0$, which can be discretized by the one-sided finite differences at the boundary segment of the cut-cell as

$$\tau_{\xi\eta}|_{\eta=0} = \mu\frac{u_o^\xi - u_\Gamma^\xi}{d_{o\Gamma}} = 0 \tag{25}$$

where u_o^ξ is the ξ-component of the velocity vector (i.e., wall-tangential velocity) at the cut-cell centroid *o* in the local coordinate system, $d_{o\Gamma}$ is the distance from the centroid *o* to the boundary segment as shown in Figure 2. We can obtain a similar equation at the $\boldsymbol{\zeta}$-$\boldsymbol{\eta}$ plane. Therefore, the two tangential velocities $u_\Gamma^\xi$and u_Γ^ζat the solid surface can be related to the tangential components of the velocity vector at the cut-cell centroid in the local system *L* as

$$u_\Gamma^\xi = u_o^\xi \text{ and } u_\Gamma^\zeta = u_o^\zeta \tag{26}$$

where u_o^ξ and u_o^ζ are velocity tangential components at the cut cell centroid in the local system *L*. u_o^ξ and u_o^ζ are obtained by transforming the velocity vector in the global system into the vector in the local system.

When the solid boundary is a general curved surface, the shear stresses are related to the curvature radii of the surface. Kempe *et al.* [43] derived the relation between tangential velocites u_Γ^ξ and u_Γ^ζ at the boundary and velocity components $u_o^\xi$and u_o^ζat the cell centroid as

$$u_\Gamma^\xi = \frac{R_1}{R_1 + d_{o\Gamma}}u_o^\xi \text{ and } u_\Gamma^\zeta = \frac{R_2}{R_2 + d_{o\Gamma}}u_o^\zeta \tag{27}$$

where R_1 and R_2 are two principal curvature radii of the boundary surface as shown in Figure 3. It is clear that as R_1 and R_2 approaches to infinity for the planar surface, Eq. (27) reduces to Eq. (26). To find the principal curvatures κ_1 and κ_2, we use the least-squares fitting linear functions to the normal vectors of the boundary segments in current cut-cell and its neighboring cut-cells [49,50]. In this method, the normal vector behavior of the boundary segments in neighboring cells is described by linear functions of two parameters in the local tangential plane (ξ-η plane) defined by the solid boundary in current cut-cell. Once applying the least-squares method to the linear functions, the Weingarten curvature matrix can be obtained. Since the principal curvatures κ_1 and κ_2 are the eigenvalues of the Weingarten matrix, the corresponding raddii of principal

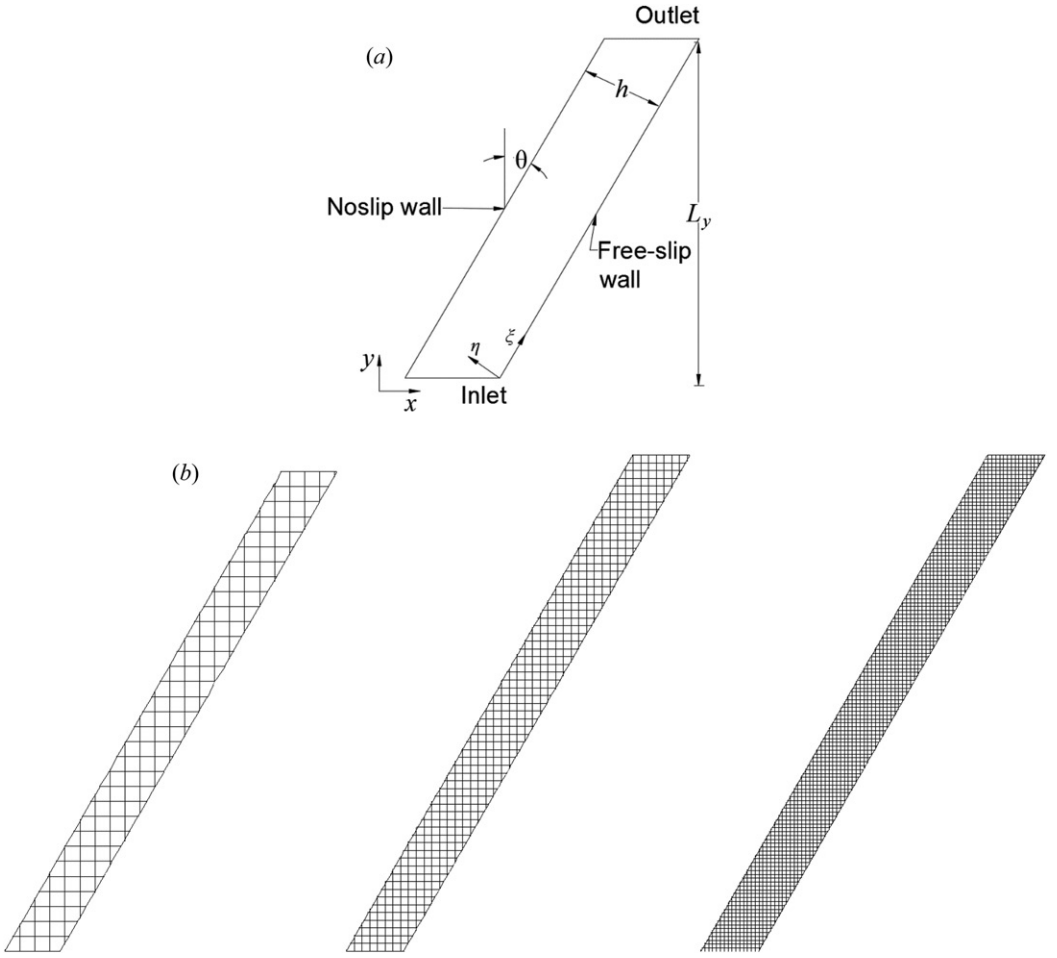

Figure 5. Laminar channel flow between two parallel plates: (*a*) geometry and boundary conditions and (*b*) grids used for the case of skew angle $\theta = 30^\circ$ (from left to right the grid size is 0.0315 m, 0.0156 m, and 0.0078 m, respectively).

curvature R_1 and R_2 can be obtained by inverting the curvatures. As we only consider the rigid solid boundary in the study, calculation of the principal curvature is performed only once after the geometric computation of cut-cells is done at the begnning of the simulation.

Let us summarize the numerical scheme to implement the free-slip boundary condition here. The procedure starts during the solution of the provisional velocity $\mathbf{u}^*$ in Eq. (12) and includes four steps: (a) First we transform the velocity vector $\mathbf{u}_o^G$at the cut-cell centroids in the global coordinate system G into the vector $\mathbf{u}_o^L$in the local system L; (b) Then we can obtain boundary tangential velocities of cut-cells in the local sytem L using Eq. (27) and hence the boundary velocity vector $\mathbf{u}_\Gamma^L = (0, u_\Gamma^\xi, u_\Gamma^\zeta)^T$; (c) Next we transform the boundary velocity vector $\mathbf{u}_\Gamma^L$ in the local system L to the velocity vector$\mathbf{u}_\Gamma^G = (u_\Gamma^x, u_\Gamma^y, u_\Gamma^z)^T$in the global system G; (d) Last the viscous fluxes through the surfaces of cut-cells are estimated using the approach discusssed in Section 3.2. After these steps a new provisional velocity $\mathbf{u}^*$ can be obtained by solving Eq. (12). As the velocity vector $\mathbf{u}_o^G$ in step (a) is obtained from the provisional velocity $\mathbf{u}^*$ of the previous iterative step, steps (a)–(d) are performed in each iteration to solve $\mathbf{u}^*$ until the convergence of $\mathbf{u}^*$. As the general slip condition (Eq. (8)) is similar to the free-slip condition, the proposed method can be extended to implement the general slip condition with little effort.

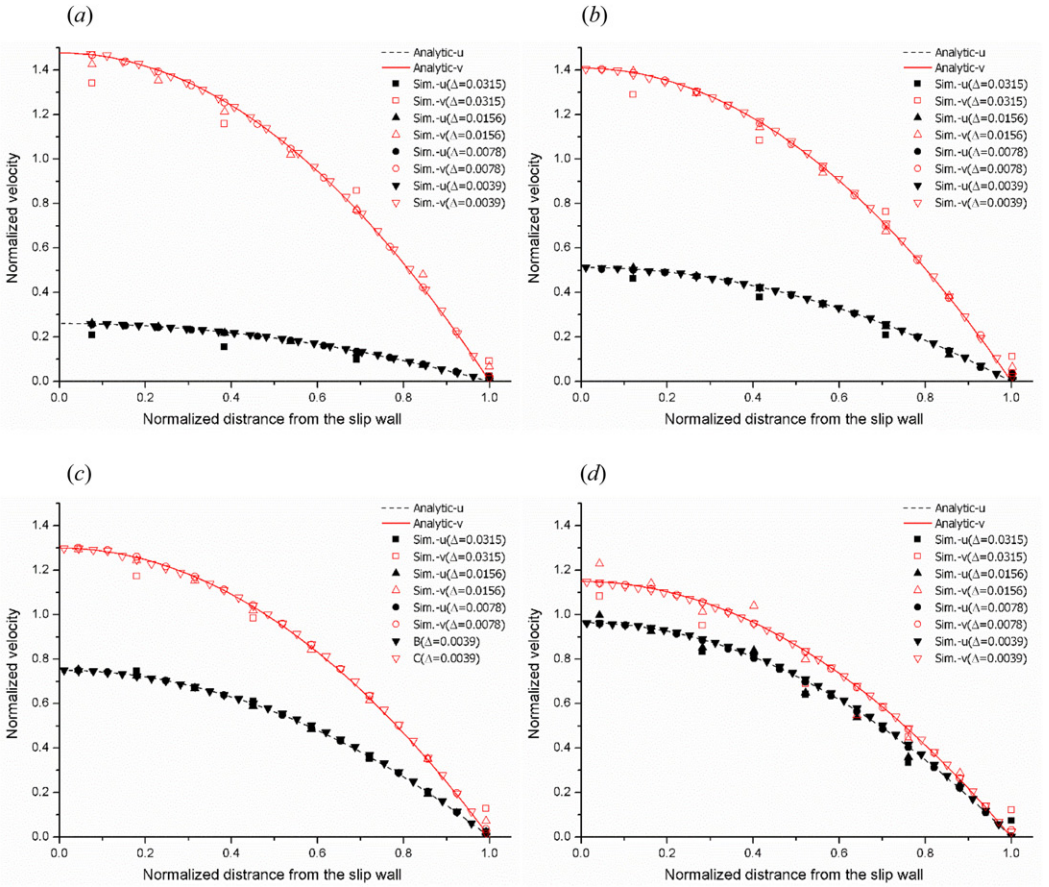

Figure 6. Comparison of velocity components along the line in *x*-direction between numerical simulations and analytical solutions: (*a*) skew angle $\theta = 10^{\circ}$, (*b*) skew angle $\theta = 20^{\circ}$, (*c*) skew angle $\theta = 30^{\circ}$, and (*d*) skew angle $\theta = 40^{\circ}$ (*u* is the normalized velocity component in *x*-axis, *v* is the normalized velocity component in *y*-axis).

4. Results

In this section, we carry out a series of numerical examples to validate the proposed numerical method to impose the free-slip condition using the adaptive cut-cell method. In these tests the numerical results obtained from the proposed method are compared to either analytical solution or published benchmark data, so the following relative error norms are defined

$$L_2 = \sqrt{\frac{\sum_i \left(E_i - E_i^{\text{ref}}\right)^2}{\sum_i \left(E_i^{\text{ref}}\right)^2}} \text{and } L_\infty = \max_i \left(\frac{|E_i - E_i^{\text{ref}}|}{|E_i^{\text{ref}}|}\right) \tag{28}$$

where E_{i} is the numerical solution of the variable E at the point i, $E_{\text{i}}^{\text{ref}}$ is the reference value of variable E.

4.1. Skewed channel flow

In this example, we consider a laminar flow through a channel between two parallel infinite plates that are placed skewed to the Cartesian grid as shown in Figure 5*a*. Four skew angles θ between the channel flow direction ξ and y-axis are considered: $\theta = 10^{\circ}$, $\theta = 20^{\circ}$, $\theta = 30^{\circ}$, and $\theta = 40^{\circ}$. The

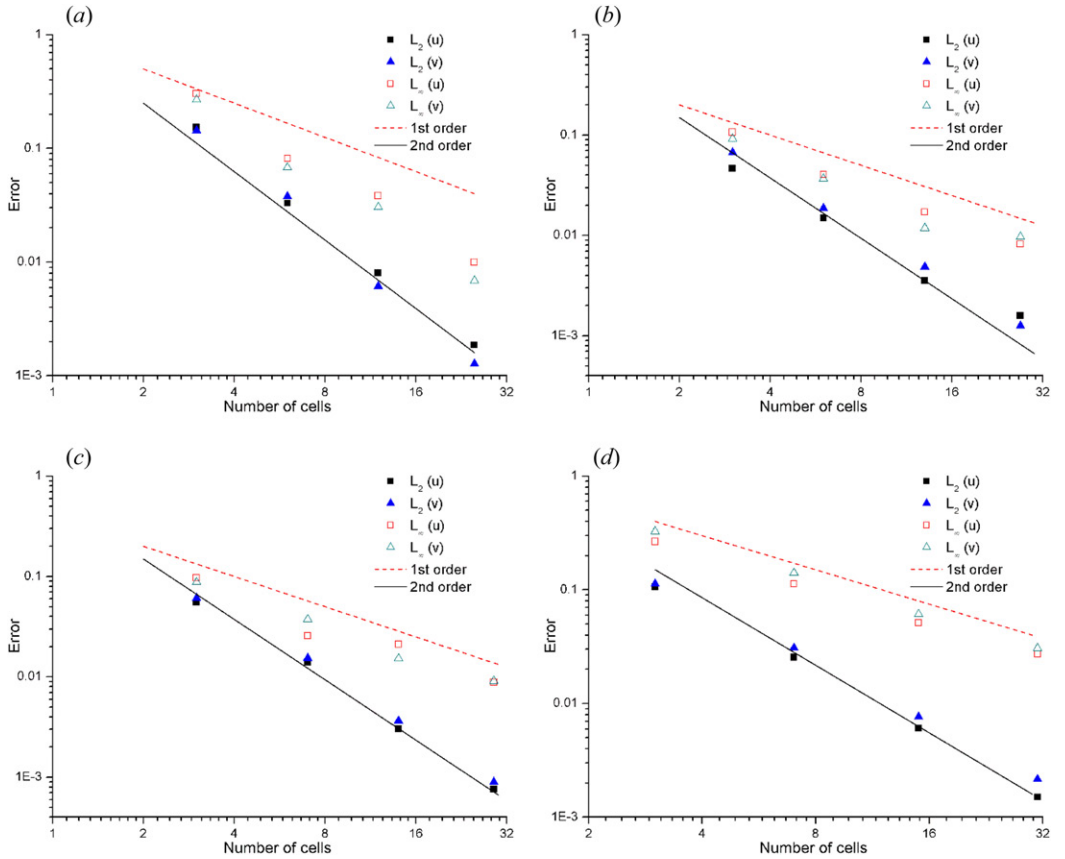

Figure 7. Variation of L_2 and L_∞ norm errors of velocity with the number of cells in the channel height direction: (*a*) skew angle $\theta = 10°$, (*b*) skew angle $\theta = 20°$, (*c*) skew angle $\theta = 30°$, and (*d*) skew angle $\theta = 40°$.

channel height h and channel length L_y in y-axis are $h = 0.1$ m and $L_y = 1$ m, respectively. The fluid density and viscosity $\rho = 1\,\text{kg/m}^3$ and $\mu = 1\,\text{kg/ms}$, respectively. One plate is no-slip boundary, wheras the other one is free-slip boundary. For the fully developed laminar channel flow, the velocity along the channel direction ξ is

$$U(\eta) = \left(\frac{h^2}{2\mu}\right)\frac{\partial p}{\partial \xi}\left(1 - \frac{\eta^2}{h^2}\right)(0 < \eta < h) \tag{29}$$

where η is the distance from the free-slip boundary in the direction perpendicular to the flow direction, $\partial p/\partial \xi$ is the pressure gradient along the flow direction. So the velocity components in x-y coordinate system are $u = \text{U}\cdot\sin\theta$ in x-axis and $v = \text{U}\cdot\cos\theta$ in y-axis. The pressure gradient of 1.0 pa/m is used in the example. The mean velocity $u_{\text{ave}} = (h^2/3)(\partial p/\partial \xi)$.

Since the channel flow is 2D, we carry out 2D simulations with an uniform grid as shown in Figure 5*b*. In order to study the effect of grid size on the accuracy of the numerical solution, we do not turn on the mesh adaptation in simulations. For each skew angle θ, four grid sizes $\Delta = 0.0315$ m, $\Delta = 0.0156$ m, $\Delta = 0.0078$ m, and $\Delta = 0.0039$ m are used to show the sptatial order of accuracy of the proposed method, and hence the correspoding number of grid cells across the channel height is 4, 8, 15, and 30, respectively. A parabolic velocity profile is applied at the inlet boundary with the velocity vector aligned parallel to the channel direction. The outlet boundary condition for velocity is a zero normal gradient. The numerical results are compared with analytic results.

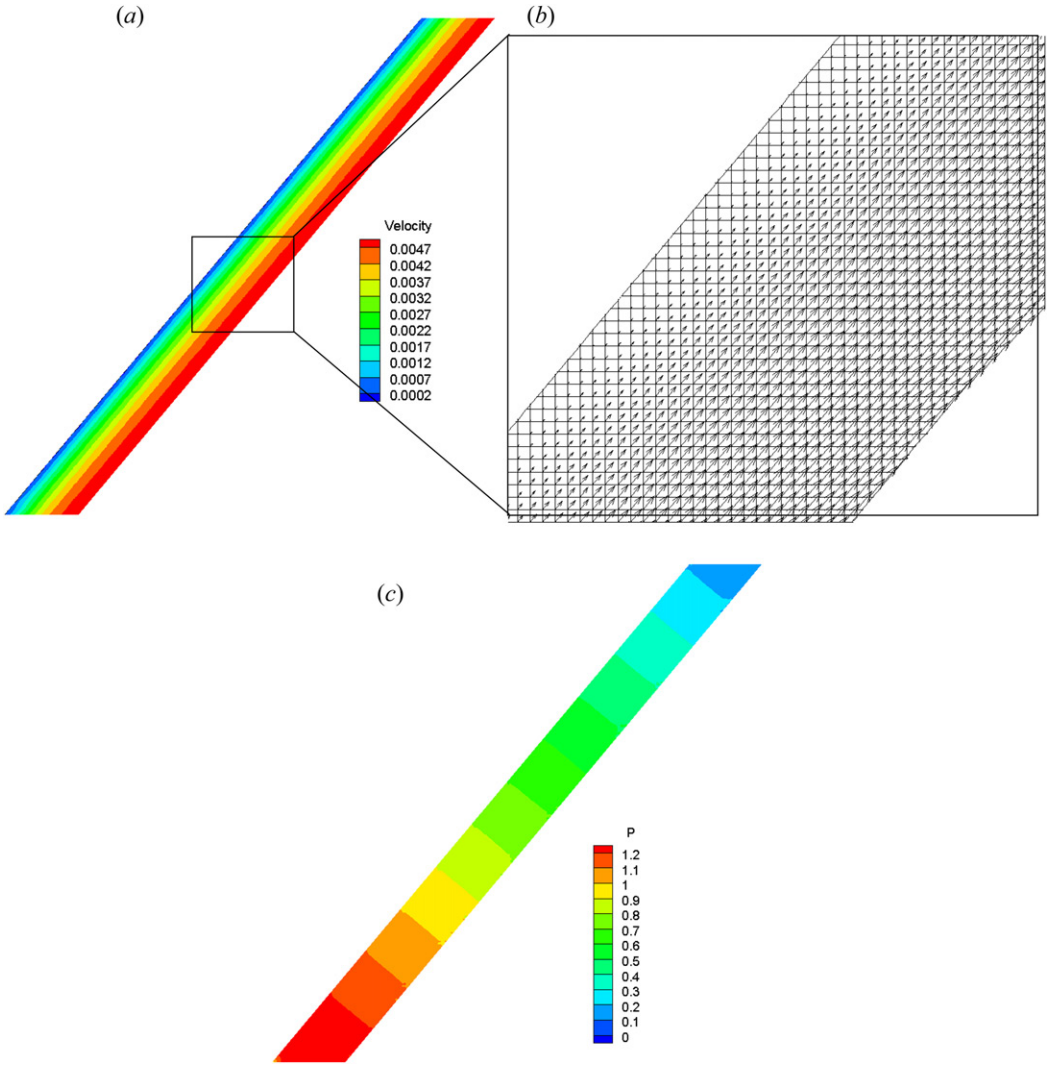

Figure 8. Contour of velocity and pressure in the skewed channel flow with skew angle $\theta = 40°$: (*a*) velocity contour, (*b*) velocity vector in a zoomed-in region, and (*c*) pressure contour.

Figure 6 shows the comparison of the numerical and analytic velocity profiles for the two components u and v along x-axis for channel flows with different skew angle θ. The velocity components are normalized with the mean velocity u_{ave}, and the coordinate position is normalized with distance between two plates in x-direction, i.e., $h/\cos\theta$. It is clear from Figure 6 that the numerical results agree very well with the analytical solutions in the fine mesh. Figure 7 shows the variations of L_2 and L_∞ norm errors of predicted velocity with different grid size. It can be seen that the numerical scheme is approximately second-order accurate in the L_2-norm error. However, the numerical scheme is less than second-order accurate in the L_∞-norm error. It is expected because the L_∞-norm error of the solution is controlled by the error in cut-cells, which is first-order accurate. The velocity and pressure contours are plotted in Figure 8. As can be seen, velocity vectors are parallel to walls. The pressure gradient is aligned with the channel axis direction.

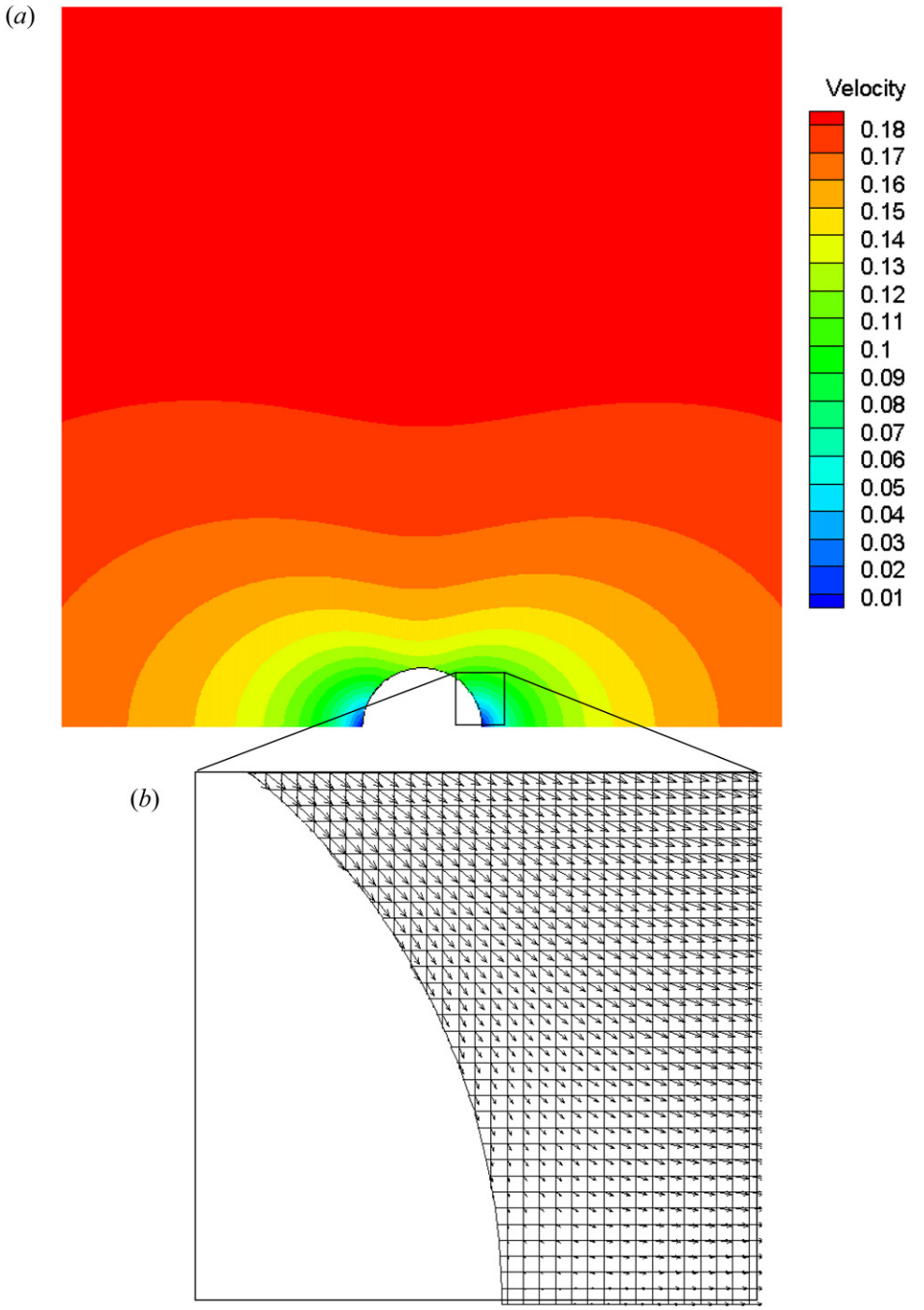

Figure 9. Contour of velocity of the creeping flow through a sphere with the free-slip condition: (*a*) velocity contour and (*b*) velocity vector in a zoomed-in region.

4.2. Flow passing a sphere

A laminar flow around a fixed sphere with a radius R is considered in this example. When the Reynolds number based on the free-stream velocity U_∞ is very small, i.e., creeping flow, an

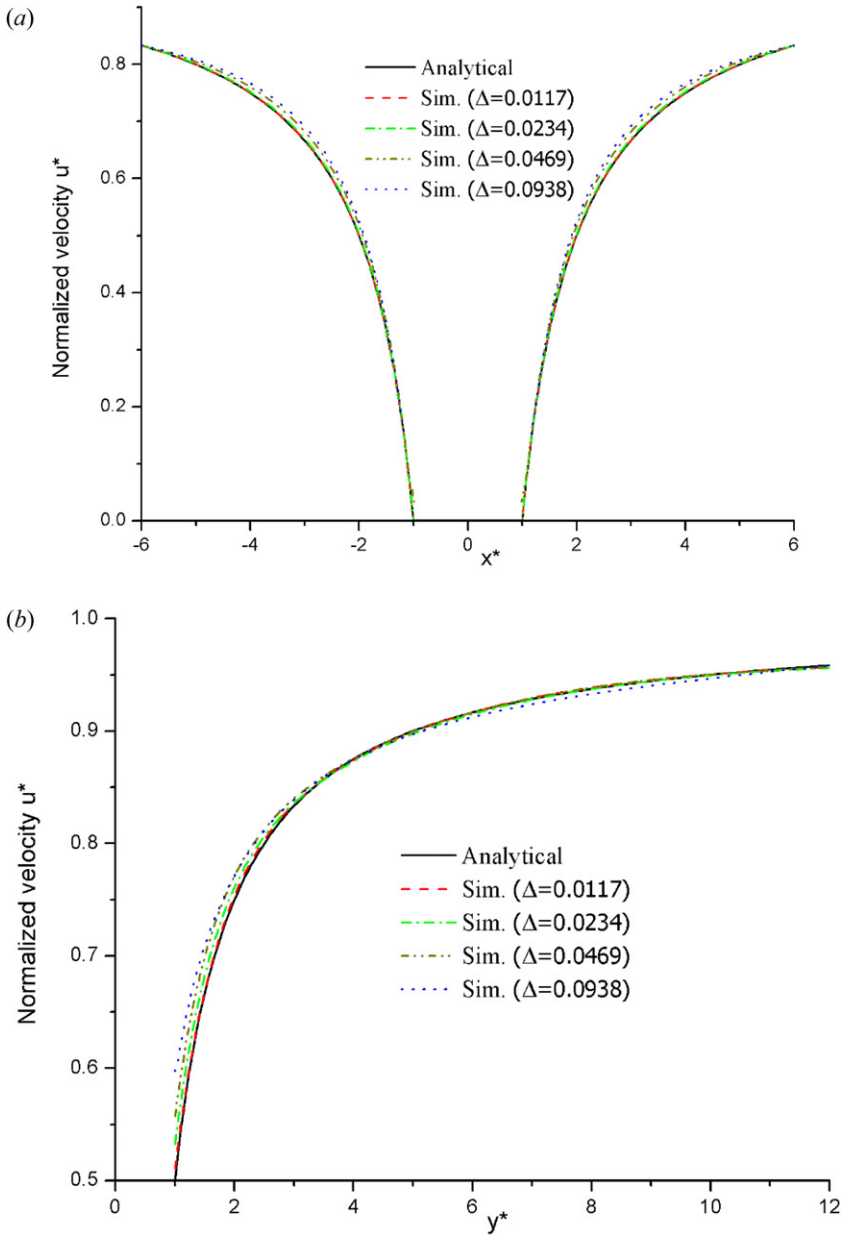

Figure 10. For the creeping flow through a free-slip sphere, comparison of *x*-component of the velocity vector between analytical solutions and numerical simulations of different grid sizes: (*a*) normalized velocity at $y = 0$ along *x*-direction and (*b*) normalized velocity at $x = 0$ along *y*-direction.

analytical solution of the flow can be found in the case of the free-slip condition at the sphere surface [43]. The velocity can be expressed in a spherical coordinate system as

$$u_r = \frac{U_\infty \cos\theta(r - R)}{r}, u_\theta = \frac{U_\infty \cos\theta(R - 2r)}{2r}, \text{and} u_\phi = 0 \tag{30}$$

where u_r u_θ $u\phi$ is radial, polar, and azimuthal velocity, respectively. r is radial distance, θ is polar angle, and ϕ is azimuthal angle. The origin of the spherical coordinate system is located at the center of the sphere.

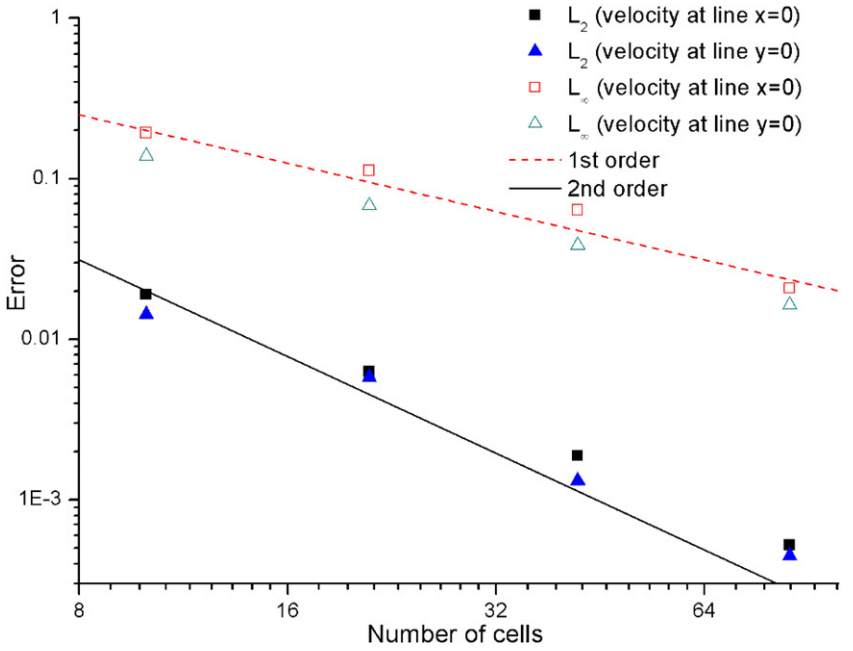

Figure 11. For creeping flow through a sphere with the free-slip condition, variation of L_2 and L_∞ norm errors of velocity with the number of cells in sphere diameter.

In this example, we employ the 2D axisymmetric method to solve the flow with $\rho = 1\,\text{kg/m}^3$, $\mu = 1\,\text{kg/ms}$, $R = 0.5\,\text{m}$, and $U_\infty = 0.2\,\text{m/s}$. The Reynolds number based on U_∞ and the diameter of the sphere is $Re = 0.2$. The computational domain is $12R \times 12R$ with the sphere of radius R positioned in the midpoint of the bottom boundary. As in the example of Section 4.1, the uniform grid with different grid size is still used to check the sptatial order of accuracy. Four grid sizes $\Delta = 0.0938\,\text{m}$, $\Delta = 0.0469\,\text{m}$, $\Delta = 0.0234\,\text{m}$, and $\Delta = 0.0117\,\text{m}$ are used in the test, so the corresponding number of cells along the sphere diameter is 10, 21, 42, and 85, respectively. As the analytical solution is based on the free-stream conditions, to minimize the boundary effect due to the limited spatial extension of the computational domain, the exact solution of Eq. (30) is used as a Dirichlet boundary condition on the boundaries other than symmetric boundary. The same approach has also been adopted by Kempe *et al.* in their numerical test [43].

Figure 9 plots the velocity contour in the entire simulation region as well as the velocity vector in a local region near the solid wall. Figure 10 plot the x-component of velocity vector predicted by analytical solutions and numerical simulations with different grid sizes. In Figure 10, the velocity is normalized with the free-stream velocity U_∞, and the coordinate position is normalized with the sphere radius R. It is clear that the numerical solution is in better agreement with the analytical solution using the finer mesh. Figure 11 shows the variations of L_2 and L_∞ norm errors of velocity with different grid size. It can be seen again that the numerical scheme is the second-order accurate in the L_2-norm error and the first-order accurate in the L_∞-norm error.

4.3. Flow passing an ellipsoid

In this example, we solve the flow passing an oblate ellipsoidal solid with the free-slip boundary condition, which is also used by Kempe *et al.* to validate their method [43]. The outer surface of the oblate ellipsoid can be described by $x^2/a^2 + y^2/b^2 + z^2/b^2 = 1$, with a ratio of semi-major axis to semi-minor axis $b/a = 2$ and $a = 0.075\,\text{m}$. So, the volume-equivalent diameter of a sphere is $D_e = 0.24\,\text{m}$. The computational domain is a cube with the edge length of 1 m. The origin of the

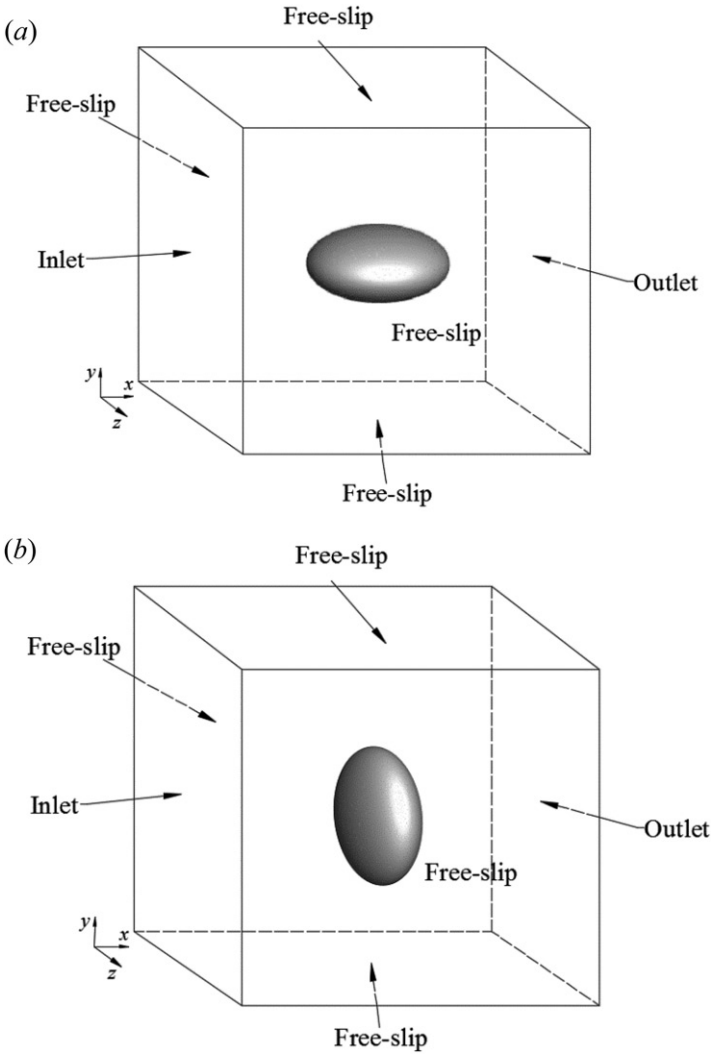

Figure 12. Low Re flow through an oblate ellipsoidal solid with the free-slip condition at solid surface: (*a*) the semi-minor axis of the ellipsoid is aligned is aligned with *y*-axis (Case 1) and (*b*) the semi-minor axis of the ellipsoid is aligned is aligned with *x*-axis (Case 2).

coordinate system is located at the center of the cube. The ellipsoid is positioned at the center of the cube as well. As shown in Figure 12, we consider two scenarios in terms of the ellipsoid orientation, i.e., the semi-minor axis is aligned with *y*-axis (Case 1) or with *x*-axis (Case 2). In both cases, a uniform velocity of 1 m/s is applied at the inlet ($x = -0.5$) and the outlet ($x = 0.5$), and the free-slip boundary condition is applied to the rest of boundaries of the domain. The density and the viscosity are $\rho = 1\,\mathrm{kg/m^3}$ and $\mu = 0.0238\,\mathrm{kg/ms}$, respectively. Therefore, the Reynolds number based on D_e is small Re = 10 with the mean flow in *x*-direction. As there is no analytical solution for such flow, Kempe *et al.* used the commercial code Fluent to solve this flow with a body-fitted mesh [43]. So we use their body-fitted mesh based numerical solution to validate our method as well.

Full 3D simulations with adaptive grids are carried out to solve the flow passing the ellipsoid with two orientations with respect to the mean flow direction (i.e., *x*-axis). The grids are adaptive to the solid geometry and the velocity gradient in the domain. The grid size varies from

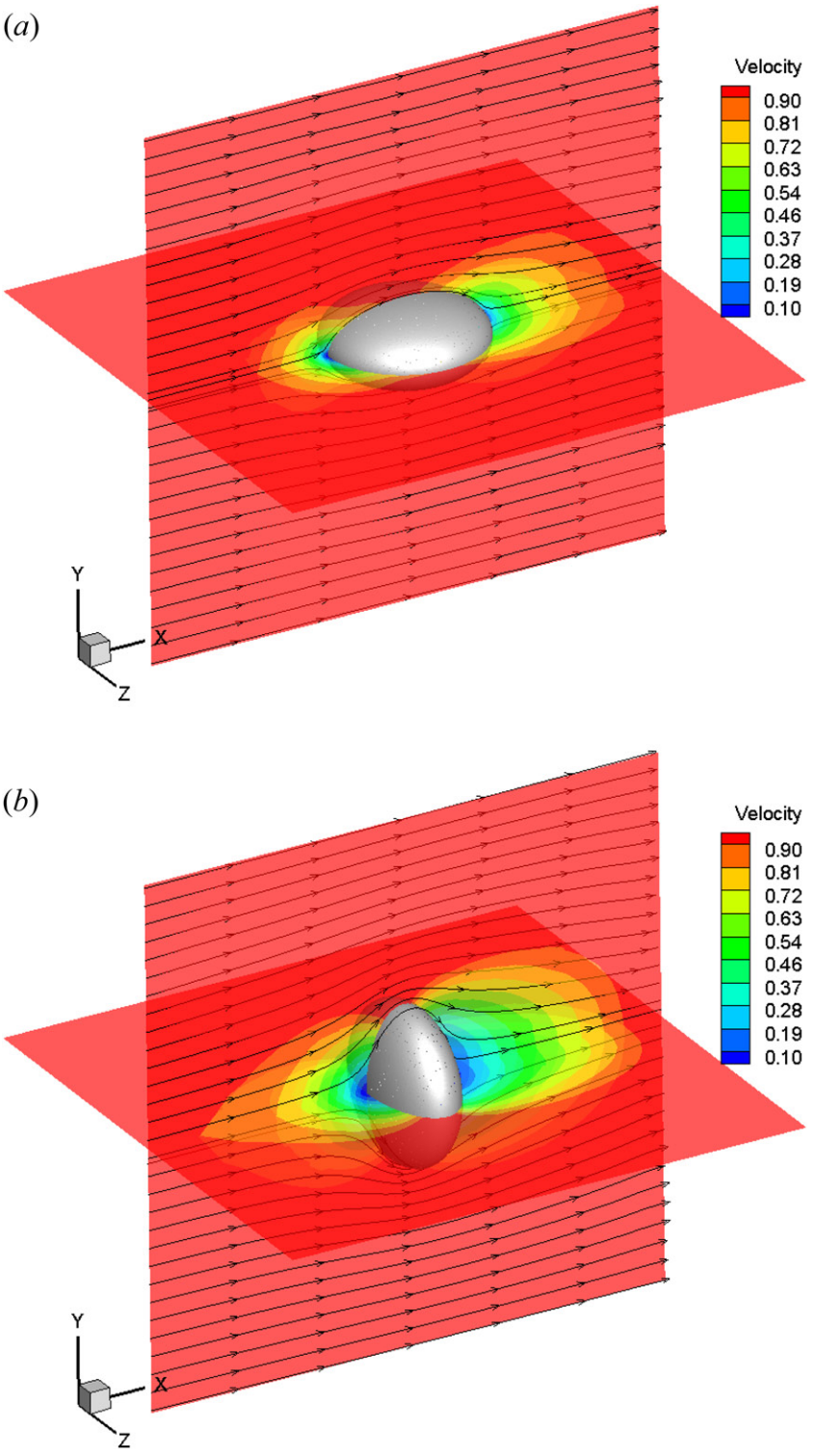

Figure 13. Velocity contour on the planes $y = 0$ and $z = 0$ overlaid with streamlines in the plane $z = 0.5$: (*a*) Case 1 and (*b*) Case 2.

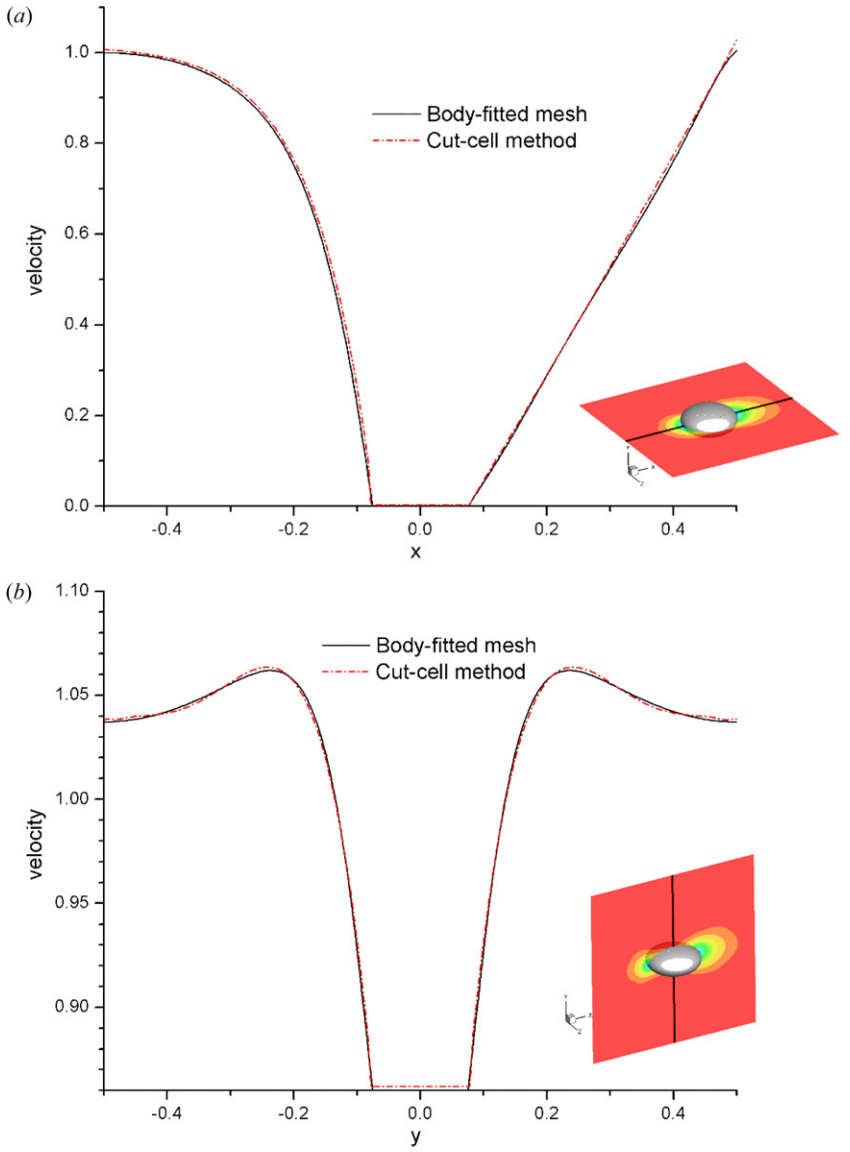

Figure 14. Profile of *x*-component of the velocity vector along different line (which is marked by the black line in the inserted bottom-left-corner figure) for Case 1: (*a*) line $y = z = 0$ and (*b*) line $x = z = 0$.

$\Delta_{\min} = 0.001953$ m to $\Delta_{\max} = 0.03125$ m. Figure 13 plots the velocity contours on the planes $y = 0$ and $z = 0$ as well as the streamlines in the plane $z = 0.5$ for two different ellipsoid orientations. As the Reynolds number is small, the flow is laminar and remains attached to the solid surface of the ellipsoid as shown in Figure 13. The comparisons of velocity profiles along lines $y = z = 0$ and $x = z = 0$ predicted by the body-fitted method and our method for Case 1 and Case 2 are plotted in Figures 14 and 15, respectively. There is a very good agreement of the velocity field computed on the body-fitted mesh and the velocity field obtained with the cut-cell method. The L_2 and L_∞ norm errors of the velocity field predicted by our method in reference to the solutions of the body-fitted mesh are 2.7×10^{-3} and 4.1×10^{-2}, respectively. This highlights the excellent performance of the proposed scheme to implement the free-slip condition at the complex geometry.

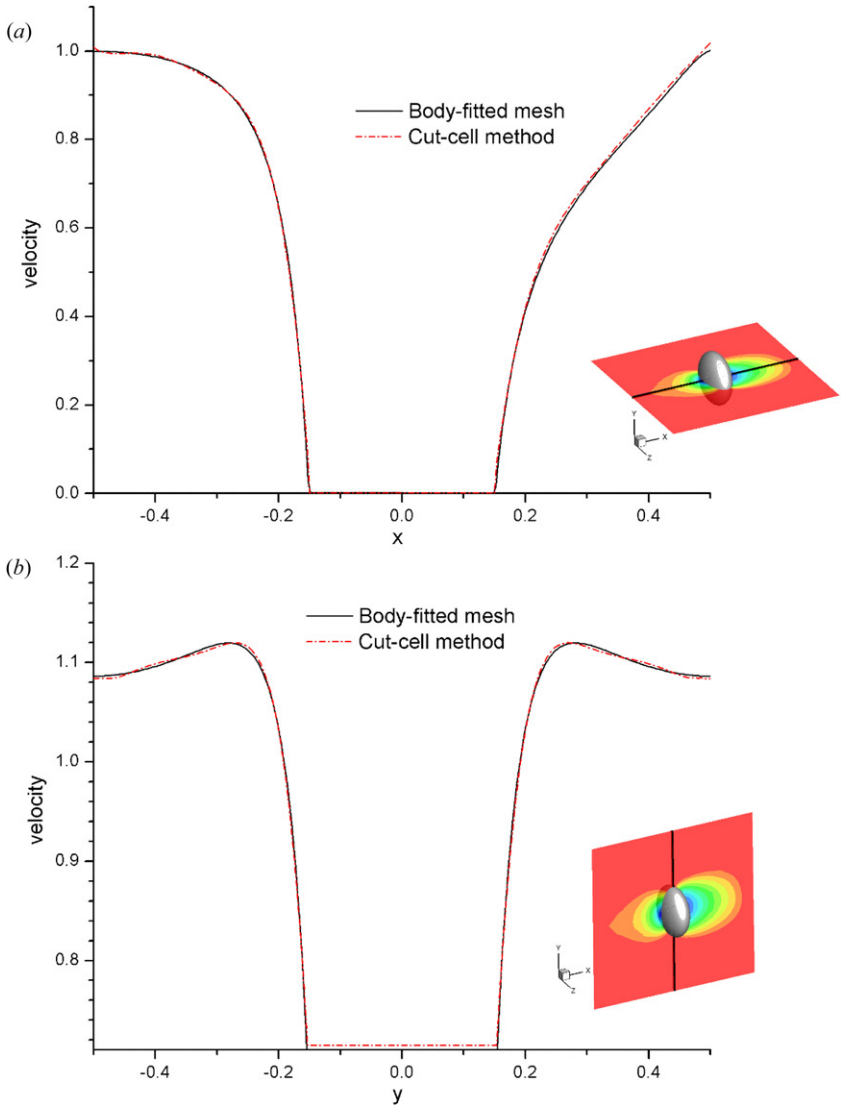

Figure 15. Profile of *x*-component of the velocity vector along different line (which is marked by the black line in the inserted bottom-left-corner figure) for Case 2: (*a*) line $y = z = 0$ and (*b*) line $x = z = 0$.

5. Conclusion

In this study, we have proposed a numerical scheme to impose the free-slip boundary condition in the framework of the Cartesian cut-cell method for the complex geometries. The collocated finite-volume method with an octree-based adaptive Cartesian grid is used to solve the Navier-Stokes equations. The cut-cell method is employed to treat the solids imbedded in the background Cartesian grid. The standard center difference method is used to estimate the surface fluxes in the regular Cartesian cells in the bulk of the computational domain for its efficiencies, whereas the least-squares method is adopted to estimate the fluxes in cut-cells near solid boundaries due to its flexibility in handling the arbitrarily truncated cells and the local adaptive mesh topology. A local coordinate system aligned with the normal and tangential directions of the solid boundary is defined for each cut-cell in order to properly implement the free-slip condition. The velocity vectors of cut-cells in the Cartesian coordinate system are first transformed into the local

system. For a non-planar solid boundary, the descritized equation from the free-slip condition requires the knowledge of the principal curvatures of the boundary that are computed using the least-squares fitting linear functions to the normal vectors of the solid boundary. Once the boundary velocities are obtained, the surface fluxes through the solid segment in each cut-cell can be computed. The proposed numerical algorithms are implemented in the open-source code *Gerris*. To validate our method, we carry out three numerical tests including the skewed channel flow, flow around a sphere, and flow around an ellipsoid. In the first two examples, 2D simulations with the uniform grids are employed to show that the sptatial order of accuracy of our method is generally second order in L_2-norm error, whereas it is first order in L_∞-norm error. In the last example, numerical solutions using the 3D adaptive Cartesian grid are compared against the results predicted by the body-fitted mesh method. Excellent agreement between the two demonstrates the proposed numerical scheme can be applied to flows through complex 3D solid objects with the free-slip boundaries. It is also straightforward to extend our method to impose the general slip conditions at the solid boundaries.

Acknowledgements

The author also would like to thank Dr. Tobias Kempe for his generous share of reference data used in Section 4.3.

Funding

The work is funded by National Science Foundation Grant CBET-1701339. Additional financial support is provided by the Office of Academic Affairs of WSU-Vancouver.

References

[1] C. S. Peskin, "Flow patterns around heart valves: A numerical method," *J. Comput. Phys.*, vol. 10, no. 2, pp. 252–271, 1972.
[2] P. G. Tucker and Z. Pan, "A cartesian cut cell method for incompressible viscous flow," *Appl. Math. Model.*, vol. 24, no. 8-9, pp. 591–606, 2000.
[3] E. A. Fadlun, R. Verzicco, P. Orlandi, and J. Mohd-Yusof, "Combined immersed-boundary finite-difference methods for three-dimensional complex flow simulations," *J. Comput. Phys.*, vol. 161, no. 1, pp. 35–60, 2000.
[4] D. Calhoun and R. J. LeVeque, "A Cartesian grid finite-volume method for the advection-diffusion equation in irregular geometries," *J. Comput.Phys.* vol. 157, no. 1, pp. 143–180, 2000.
[5] T. Ye, R. Mittal, H. S. Udaykumar, and W. Shyy, "An accurate Cartesian grid method for viscous incompressible flows with complex immersed boundaries," *J. Comput. Phys.*, vol. 156, no. 2, pp. 209–240, 1999.
[6] M. P. Kirkpatrick, S. W. Armfield, and J. H. Kent, "A representation of curved boundaries for the solution of the Navier–Stokes equations on a staggered three-dimensional Cartesian grid," *J. Comput. Phys.*, vol. 184, no. 1, pp. 1–36, 2003.
[7] M.-H. Chung, "Cartesian cut cell approach for simulating incompressible flows with rigid bodies of arbitrary shape," *Comput Fluids*, vol. 35, no. 6, pp. 607–623, 2006.
[8] R. Mittal *et al.*, "A versatile sharp interface immersed boundary method for incompressible flows with complex boundaries," *J. Comput. Phys.*, vol. 227, no. 10, pp. 4825–4852, 2008.
[9] Y. Cheny and O. Botella, "The LS-STAG method: A new immersed boundary/level-set method for the computation of incompressible viscous flows in complex moving geometries with good conservation properties," *J. Comput. Phys.*, vol. 229, no. 4, pp. 1043–1076, 2010.
[10] M. Asif Farooq, A. A. Skøien, and B. Müller, "Cartesian grid method for the compressible euler equations using simplified ghost point treatments at embedded boundaries," *Comput. Fluids*, vol. 82, no. Suppl C, pp. 50–62, 2013.
[11] H. Uddin, R. M. J. Kramer, and C. Pantano, "A Cartesian-based embedded geometry technique with adaptive high-order finite differences for compressible flow around complex geometries," *J. Comput. Phys.*, vol. 262, pp. 379–407, 2014.

[12] M. J. Berger and J. Oliger, "Adaptive mesh refinement for hyperbolic partial differential equations," *J. Comput. Phys.*, vol. 53, no. 3, pp. 484–512, 1984.
[13] M. J. Berger and P. Colella, "Local adaptive mesh refinement for shock hydrodynamics," *J. Comput. Phys.*, vol. 82, no. 1, pp. 64–84, 1989.
[14] P. MacNeice, K. M. Olson, C. Mobarry, R. de Fainchtein, and C. Packer, "PARAMESH: A parallel adaptive mesh refinement community toolkit," *Comput. Phys. Commun.*, vol. 126, no. 3, pp. 330–354, 2000.
[15] D. DeZeeuw and K. G. Powell, "An adaptively refined Cartesian mesh solver for the Euler equations," *J. Comput. Phys.*, vol. 104, no. 1, pp. 56–68, 1993.
[16] J. J. Quirk, "An alternative to unstructured grids for computing gas dynamic flows around arbitrarily complex two-dimensional bodies," *Comput. Fluids*, vol. 23, no. 1, pp. 125–142, 1994.
[17] R. B. Pember, J. B. Bell, P. Colella, W. Y. Curtchfield, and M. L. Welcome, "An adaptive Cartesian grid method for unsteady compressible flow in irregular regions," *J. Comput. Phys.*, vol. 120, no. 2, pp. 278–304, 1995.
[18] A. M. Khokhlov, "Fully threaded tree algorithms for adaptive refinement fluid dynamics simulations," *J. Comput. Phys.*, vol. 143, no. 2, pp. 519–543, 1998.
[19] A. S. Almgren, J. B. Bell, P. Colella, L. H. Howell, and M. L. Welcome, "A conservative adaptive projection method for the variable density incompressible Navier–Stokes equations," *J. Comput. Phys.*, vol. 142, no. 1, pp. 1–46, 1998.
[20] S. Popinet, "Gerris: A tree-based adaptive solver for the incompressible Euler equations in complex geometries," *J. Comput. Phys.*, vol. 190, no. 2, pp. 572–600, 2003.
[21] D. Hartmann, M. Meinke, and W. Schröder, "An adaptive multilevel multigrid formulation for Cartesian hierarchical grid methods," *Comput. Fluids*, vol. 37, no. 9, pp. 1103–1125, 2008.
[22] D. Hartmann, M. Meinke, and W. Schröder, "A strictly conservative Cartesian cut-cell method for compressible viscous flows on adaptive grids," *Comput. Methods Appl. Mech. Eng.*, vol. 200, no. 9-12, pp. 1038–1052, 2011.
[23] S. Popinet, "An accurate adaptive solver for surface-tension-driven interfacial flows," *J. Comput. Phys.*, vol. 228, no. 16, pp. 5838–5866, 2009.
[24] D. Zuzio and J. L. Estivalezes, "An efficient block parallel AMR method for two phase interfacial flow simulations," *Comput. Fluids*, vol. 44, no. 1, pp. 339–357, 2011.
[25] D. Aubert, N. Deparis, and P. Ocvirk, "EMMA: An adaptive mesh refinement cosmological simulation code with radiative transfer," *Mon. Not. R. Astron. Soc.*, vol. 454, no. 1, pp. 1012–1037, 2015.
[26] V. Laurmaa, M. Picasso, and G. Steiner, "An octree-based adaptive semi-Lagrangian VOF approach for simulating the displacement of free surfaces," *Comput. Fluids*, vol. 131, no. Suppl C, pp. 190–204, 2016.
[27] L. Schneiders, D. Hartmann, M. Meinke, and W. Schröder, "An accurate moving boundary formulation in cut-cell methods," *J. Comput. Phys.*, vol. 235, pp. 786–809, 2013.
[28] L. Schneiders, C. Günther, M. Meinke, and W. Schröder, "An efficient conservative cut-cell method for rigid bodies interacting with viscous compressible flows," *J. Comput. Phys.*, vol. 311, pp. 62–86, 2016.
[29] H. Ji, F.-S. Lien, and E. Yee, "Numerical simulation of detonation using an adaptive Cartesian cut-cell method combined with a cell-merging technique," *Comput. Fluids*, vol. 39, no. 6, pp. 1041–1057, 2010.
[30] CONVERGE. Available: https://convergecfd.com.
[31] Karalit. Available: https://www.karalit.com.
[32] R. Mittal and G. Iaccarino, "Immersed boundary methods," *Annu. Rev. Fluid Mech.*, vol. 37, no. 1, pp. 239–261, 2005.
[33] J. Mohd-Yusof, "Combined immersed-boundary/B-spline methods for simulations of flow in complex geometries," in *Annual Research Briefs*, Parviz Moin and William C. Reynolds, Eds. Stanford, CA: Center for Turbulence Research, NASA Ames/Standford University, 1997, pp. 317–327
[34] J. Kim, D. Kim, and H. Choi, "An immersed-boundary finite-volume method for simulations of flow in complex geometries," *J. Comput. Phys.*, vol. 171, no. 1, pp. 132–150, 2001.
[35] D. Goldstein, R. Handler, and L. Sirovich, "Modeling a no-slip flow boundary with an external force field," *J. Comput. Phys.*, vol. 105, no. 2, pp. 354–366, 1993.
[36] C. S. Peskin, "The immersed boundary method," *Acta Numer.*, vol. 11, pp. 479–517, 2003.
[37] B. E. Griffith, R. D. Hornung, D. M. McQueen, and C. S. Peskin, "An adaptive, formally second order accurate version of the immersed boundary method," *J. Comput. Phys.*, vol. 223, no. 1, pp. 10–49, 2007.
[38] M. J. Aftosmis, *Solution Adaptive Cartesian Grid Methods for Aerodynamic Flows with Complex Geometries*. Sint-Genesius-Rode, Belgium: von Karman Institute for Fluid Dynamics, 1997.
[39] P. Joseph and P. Tabeling, "Direct measurement of the apparent slip length," *Phys. Rev. E Stat. Nonlin. Soft Matter Phys.*, vol. 71, no. 3 Pt 2A, pp. 035303, 2005.
[40] J. P. Rothstein, "Slip on superhydrophobic surfaces," *Annu. Rev. Fluid Mech.*, vol. 42, no. 1, pp. 89–109, 2010.

[41] I. U. Vakarelski, N. A. Patankar, J. O. Marston, D. Y. C. Chan, and S. T. Thoroddsen, "Stabilization of leidenfrost vapour layer by textured superhydrophobic surfaces," *Nature*, vol. 489, no. 7415, pp. 274, 2012.
[42] Y. Matsumoto, T. Uda, and S. Takagi, "The effect of surfactant on rising bubbles," in *IUTAM Symposium on Computational Approaches to Multiphase Flow: Proceedings of an IUTAM Symposium Held at Argonne National Laboratory, October 4–7, 2004*, S. Balachandar and A. Prosperetti, Ed. Dordrecht, Netherlands: Springer, 2006, p. 311–321
[43] T. Kempe, M. Lennartz, S. Schwarz, and J. Fröhlich, "Imposing the free-slip condition with a continuous forcing immersed boundary method," *J. Comput. Phys*., vol. 282, no. Suppl C, pp. 183–209, 2015.
[44] M. M. Denn, "Extrusion instabilities and wall slip," *Annu. Rev. Fluid Mech*., vol. 33, no. 1, pp. 265–287, 2001.
[45] H. Tan, "Development of an adaptive mesh refinement based flow solver for free-surface in inkjet printing," *Int. J. Multiphase Flow*, vol. 82, pp. 1–16, 2016.
[46] H. Tan, "Three-dimensional simulation of micrometer-sized droplet impact and penetration into the powder bed," *Chem. Eng. Sci*., vol. 153, pp. 93–107, 2016.
[47] H. Tan, "Numerical study on splashing of high-speed microdroplet impact on dry microstructured surfaces," *Comput. Fluids*, vol. 154, pp. 142–166, 2017.
[48] H. Tan, "Absorption of picoliter droplets by thin porous substrates," *AIChE J*., vol. 63, no. 5, pp. 1690–1703, 2017.
[49] Z. Cheng and X. Zhang, "Estimating differential quantities from point cloud based on a linear fitting of normal vectors," *Sci. China F: Inf. Sci*., vol. 52, no. 3, pp. 431–444, 2009.
[50] B. B. M. Kassar, J. N. E. Carneiro, and A. O. Nieckele, "Curvature computation in volume-of-fluid method based on point-cloud sampling," *Comput. Phys. Commun*., vol. 222, pp. 189–208, 2018.