Journal of Computational Physics **156,** 209–240 (1999)
Article ID jcph.1999.6356, available online at http://www.idealibrary.com on IDEAL®

An Accurate Cartesian Grid Method for Viscous Incompressible Flows with Complex Immersed Boundaries

T. Ye,* R. Mittal,* H. S. Udaykumar,† and W. Shyy†

*Department of Mechanical Engineering, University of Florida, Gainesville, Florida 32611; †Department of Aerospace Engineering, Mechanics and Engineering Science, University of Florida, Gainesville, Florida 32611
E-mail: taoye@grov.ufl.edu, mittal@gollum.me.ufl.edu, ush@confucius.aero.ufl.edu, wss@tiger.aero.ufl.edu

Received March 11, 1999; revised August 10, 1999

A Cartesian grid method has been developed for simulating two-dimensional unsteady, viscous, incompressible flows with complex immersed boundaries. A finite-volume method based on a second-order accurate central-difference scheme is used in conjunction with a two-step fractional-step procedure. The key aspects that need to be considered in developing such a solver are imposition of boundary conditions on the immersed boundaries and accurate discretization of the governing equation in cells that are cut by these boundaries. A new interpolation procedure is presented which allows systematic development of a spatial discretization scheme that preserves the second-order spatial accuracy of the underlying solver. The presence of immersed boundaries alters the conditioning of the linear operators and this can slow down the iterative solution of these equations. The convergence is accelerated by using a preconditioned conjugate gradient method where the preconditioner takes advantage of the structured nature of the underlying mesh. The accuracy and fidelity of the solver is validated by simulating a number of canonical flows and the ability of the solver to simulate flows with very complicated immersed boundaries is demonstrated.

Key Words: viscous incompressible flow; finite volume method; Cartesian grid method; immersed boundaries.

1. INTRODUCTION

The conventional structured-grid approach to simulating flows with complex immersed boundaries is to discretize the governing equations on a curvilinear grid that conforms to the boundaries. The main advantages of this approach are that imposition of boundary conditions is greatly simplified, and furthermore, the solver can be easily designed so as to maintain adequate accuracy and conservation property. However, depending on the geometrical

0021-9991/99 $30.00

complexity of the immersed boundaries, grid generation and grid quality can be major issues and usually, one has to resort to a multi-block approach in order to handle anything but the simplest geometries. Furthermore, transformation of the governing equations to the curvilinear coordinate system results in a complex system of equations and this complexity can adversely impact the stability, convergence and operation count of the solver.

A different approach which is gaining popularity in recent years is the so-called Cartesian grid method where the governing equations are discretized on a Cartesian grid which does not conform to the immersed boundaries. This greatly simplifies grid generation and also retains the relative simplicity of the governing equations in Cartesian coordinates. In addition, this method also has a significant advantage over the conventional body-fitted approach in simulating flows with moving boundaries, complicated shapes, or topological changes [31]. Since the underlying Cartesian grid does not depend on the location of the immersed boundary, there is no need for remeshing strategies. In fact, a moving boundary algorithm has been implemented in conjunction with a Cartesian grid algorithm that has been used successfully for diffusion-dominated solidification problems [37] which involve complex time evolving moving boundaries.

The obvious complication in using Cartesian grid methods is in the imposition of boundary conditions at the immersed boundaries. In particular, since the immersed boundary can cut through the underlying Cartesian mesh in an arbitrary manner, the main challenge is to construct a boundary treatment which does not adversely impact the accuracy and conservation property of the underlying numerical solver. This is especially critical for viscous flows where inadequate resolution of boundary layers which form on the immersed boundaries can reduce the fidelity of the numerical solution. Consequently, Cartesian grid methods have been used extensively for Euler flows [1, 3, 4, 10, 24, 28] whereas applications to viscous flows are relatively scarce [17, 31, 35, 36]. It should be pointed out that there is another related class of methods, the so-called "immersed boundary" methods in which the immersed boundary is represented by a discrete set of body or surface forces. These methods have also been used successfully for viscous flow computations [14, 25, 42]; however, one disadvantage of these methods is that in most cases the immersed boundary is smeared across a few cell-widths. This is mainly due to problems associated with representing a point force on a finite size mesh. As shown by Udaykumar *et al*. [37] this smearing can adversely impact the dynamics of flows where the boundary motion is closely coupled with the evolution of surrounding fluid field. Similarly, in the so-called volume-of-fluid (VOF) method [30] too, the process of interface reconstruction leads to a non-smooth interface. In contrast to these approaches, in Cartesian grid methods the boundary is represented as a sharp interface and this has advantages for high Reynolds number flows as well as flows with strong two-way coupling between the flow and the boundary motion.

Here we have developed a Cartesian grid method which is well suited for simulating unsteady, viscous, incompressible flows. The current solver shares some features with the solver of Udaykumar *et al*. [35] particularly in terms of the description and identification of the immersed boundary and the use of a finite-volume approach. However, a number of key advances have been made in terms of the capability of the solver. These include: (1) A fractional-step scheme [8] which results in a fast solution of unsteady flows, (2) adoption of a new compact interpolation scheme near the immersed boundaries that allows us to retain second-order accuracy and conservation property of the solver, and (3) use of a preconditioned conjugate gradient method for solving the pressure Poisson

equation which takes advantage of the underlying structured nature of the mesh and which substantially accelerates the convergence of the pressure Poisson equation.

The current paper will focus on describing these and other salient features of the numerical methodology, validating the accuracy and fidelity of the approach and demonstrating the capabilities of the solver in some complex configurations.

2. NUMERICAL METHODOLOGY

In this section we will first describe the underlying solver for a Cartesian mesh without considering the immersed boundaries. Following this, we will discuss in detail the modifications that have to be made in the solver to account for immersed boundaries.

Fractional-Step Method

The governing equation is the unsteady, viscous, incompressible Navier–Stokes equation written in terms of the primitive variables. This equation is discretized on a Cartesian mesh using a cell-centered colocated (non-staggered) arrangement [12] of the primitive variables $(\boldsymbol{u}, p)$. The integral form of the non-dimensionalized governing equations is given by

mass conservation
$$\int_{cs} \boldsymbol{u} \cdot \hat{\boldsymbol{n}}\, dS = 0 \tag{1}$$

momentum conservation
$$\frac{\partial}{\partial t}\int_{cv} \boldsymbol{u}\, dV + \int_{cs} \boldsymbol{u}(\boldsymbol{u}\cdot\hat{\boldsymbol{n}})\, dS = -\int_{cs} p\hat{\boldsymbol{n}}\, dS + \frac{1}{\mathrm{Re}}\int_{cs} \nabla\boldsymbol{u}\cdot\hat{\boldsymbol{n}}\, dS. \tag{2}$$

This is used as the starting point for deriving a second-order accurate finite-volume method. In the above equations *cv* and *cs* denote the control-volume and control-surface, respectively, and $\hat{\boldsymbol{n}}$ is a unit vector normal to the control-surface. The above equations are to be solved with $\boldsymbol{u}(\boldsymbol{x}, t) = \boldsymbol{v}(\boldsymbol{x}, t)$ on the boundary of the flow domain where $\boldsymbol{v}$ is the prescribed boundary velocity. A second-order accurate, two-step fractional step method [8, 20, 43] is used for advancing the solution in time. In this time-stepping scheme, the solution is advanced from time level "n" to "$n+1$" through an intermediate advection-diffusion step where the momentum equations without the pressure gradient terms are first advanced in time. A second-order Adams–Bashforth scheme is employed for the convective terms and the diffusion terms are discretized using an implicit Crank–Nicolson scheme. This eliminates the viscous stability constraint which can be quite severe in simulation of viscous flows.

At this stage, in addition to the cell-center velocities which are denoted by $\boldsymbol{u}$, we also introduce face-center velocities $\boldsymbol{U}$. In a manner similar to a fully staggered arrangement, only the component normal to the cell-face is computed and stored (see Fig. 1). The face-center velocity is used for computing the volume flux from each cell in our finite-volume discretization scheme. The advantage of separately computing the face-center velocities will be addressed later in this section. The semi-discrete form of the advection-diffusion equation for each cell shown in Fig. 1 can therefore be written as

$$\int_{cv} \frac{\boldsymbol{u}^* - \boldsymbol{u}^n}{\Delta t}\, dV = -\frac{1}{2}\int_{cs} [3\boldsymbol{u}^n(\boldsymbol{U}^n\cdot\hat{\boldsymbol{n}}) - \boldsymbol{u}^{n-1}(\boldsymbol{U}^{n-1}\cdot\hat{\boldsymbol{n}})]\, dS + \frac{1}{2\,\mathrm{Re}}\int_{cs} (\nabla\boldsymbol{u}^* + \nabla\boldsymbol{u}^n)\cdot\hat{\boldsymbol{n}}\, dS, \tag{3}$$

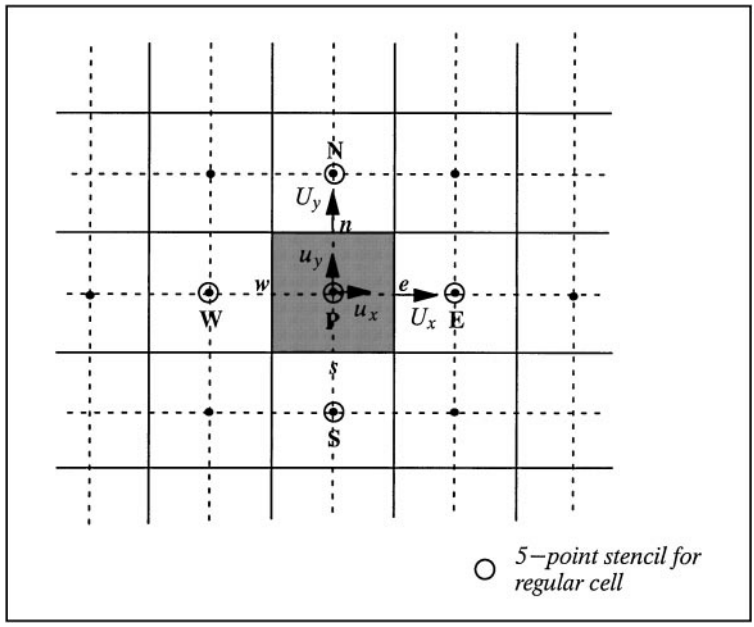

FIG. 1. Schematic showing the underlying Cartesian mesh and arrangement of cell-center and face-center velocities.

where $\boldsymbol{u}^*$ is the intermediate cell-center velocity and *cv* and *cs* denote the volume and surface of a cell, respectively. The velocity boundary condition imposed at this intermediate step corresponds to that at the end of the full time-step, i.e., $\boldsymbol{u}^* = \boldsymbol{v}^{n+1}$. Following the advection-diffusion step, the intermediate face-center velocity $\boldsymbol{U}^*$ is computed by interpolating the intermediate cell-center velocity.

The advection-diffusion step is followed by the pressure-correction step

$$\int_{cv} \frac{\boldsymbol{u}^{n+1} - \boldsymbol{u}^*}{\Delta t}\, dV = -\int_{cv} \nabla p^{n+1}\, dV, \tag{4}$$

where we require that the final velocity field satisfy the integral mass conservation equation given by

$$\int_{cs} (\boldsymbol{U}^{n+1} \cdot \hat{\boldsymbol{n}})\, dS = 0. \tag{5}$$

This results in the following equation for pressure

$$\int_{cs} (\nabla p^{n+1}) \cdot \hat{\boldsymbol{n}}\, dS = \frac{1}{\Delta t} \int_{cs} (\boldsymbol{U}^* \cdot \hat{\boldsymbol{n}})\, dS \tag{6}$$

which is the integral version of the pressure Poisson equation. Note that the pressure-correction step is represented by the inviscid equation (4) and is well posed only if the velocity component normal to the boundary is specified. Therefore the velocity boundary condition consistent with (4) is $\boldsymbol{u}^{n+1} \cdot \hat{\boldsymbol{N}} = \boldsymbol{v}^{n+1} \cdot \hat{\boldsymbol{N}}$ where $\hat{\boldsymbol{N}}$ is the unit normal to the boundary of the flow domain. It can be easily shown that this implies that $(\nabla p^{n+1}) \cdot \hat{\boldsymbol{N}} = 0$ be

used as the boundary condition for Eq. (6). Once the pressure is obtained by solving this equation, both the cell-center (*cc*) and face-center (*fc*) velocities are updated separately as

$$\boldsymbol{u}^{n+1} = \boldsymbol{u}^* - \Delta t(\nabla p^{n+1})_{cc}; \qquad \boldsymbol{U}^{n+1} = \boldsymbol{U}^* - \Delta t(\nabla p^{n+1})_{fc}. \tag{7}$$

It should be pointed out that the pressure gradient computed at the face-center is not simply an interpolated version of the pressure gradient at the cell-center. For instance with reference to Fig. 1 the x-direction pressure gradient at the cell center is computed as

$$(\partial p/\partial x)_P = (p_E - p_W)/2\Delta x \tag{8}$$

whereas the same gradient on the east face is given by

$$(\partial p/\partial x)_e = (p_E - p_P)/\Delta x. \tag{9}$$

It follows that $\boldsymbol{U}^{n+1}$ is not simply an interpolated version of the face-center velocities $\boldsymbol{u}^{n+1}$. In fact the pressure equation (6) is discretized in terms of the pressure gradients on the cell faces and with the separate update of the face-center velocity as shown in (7), exact satisfaction of (5) is guaranteed.

There are many advantages to introducing the face-center velocities into the fractional-step scheme. In conventional colocated methods, the solution of the pressure Poisson equation and satisfaction of the continuity constraint can be quite problematic. As shown clearly in Ferziger and Peric [12] for colocated methods, if a compact stencil is used for pressure then the pressure does not suffer from grid-to-grid oscillation but the final velocity does not satisfy the divergence free constraint exactly. On the other hand, if a consistent non-compact stencil is used for pressure then the divergence constraint can be satisfied to machine precision. However, in this case the pressure is subject to grid-to-grid oscillations. A similar observation has been made in the context of the finite element method by Nonino and Comini [22]. One remedy is to use a fully staggered arrangement of the variables [23]. However, this can increase the complexity of the solver since each of the momentum and pressure equations have to be discretized at different locations. This is more so in our Cartesian grid method since the discretization of the cells near the boundary can become extremely complicated with a fully staggered arrangement. Other remedies have also been proposed for tackling these problems that arise in a colocated arrangement [29, 32, 33] but these have been employed mainly in steady flow computations.

Zang *et al*. [43] have introduced a new colocated approach for solution of incompressible flows which seems especially suitable for unsteady flows. In their approach, the Cartesian velocity components are colocated with pressure at the cell-center and the momentum equation solved only at the cell-center. However, the intermediate cell-center velocity is used to compute the volume fluxes on the cell faces and subsequently, the pressure correction is applied separately to the cell-center velocities and the volume fluxes. In an analogous manner, we define a face-center velocity which when multiplied by the face area gives us the volume flux and this face-center velocity is updated separately in the pressure correction step. This procedure ensures that even with a compact stencil, the integral constraint (6) is satisfied to machine precision at the end of the full time step. The problems of grid-to-grid pressure oscillations and mass conservation error are therefore eliminated simultaneously. Furthermore, this updated face-center velocity is used to compute the convective flux at the

next time step as shown in (3). Since the volume flux is conserved exactly, this ensures that a uniform velocity field will convect on the grid without generating spurious gradients. Zang *et al*. [43] have used this procedure in conjunction with a curvilinear mesh solver for large-eddy simulation of turbulent flows and have found that the solver performs satisfactorily for high Reynolds number flows.

This approach therefore has some of the most desirable features of a fully staggered arrangement. The main advantage of this approach over the fully staggered approach is that the momentum and pressure equations are all solved at the same location. However, as apparent from (8), unlike a fully staggered arrangement, in the current approach the cell-center velocity is not coupled strongly to the pressure gradient over the cell. Furthermore in a fully staggered arrangement, the computed velocity components satisfy both the momentum as well as continuity equations. In contrast, in the the current approach the velocity field is represented by two different but closely related variables, the cell-center velocity which satisfies the momentum equations and the face-center velocity which satisfies the continuity constraint.

Inclusion of Immersed Boundaries

The underlying approach for a Cartesian grid without immersed boundaries has been outlined in the previous subsection. We will now describe how this approach is implemented in a situation where some of the Cartesian cells are cut by immersed boundaries as shown in Figs. 2a and 2b. For the purpose of this discussion we assume that the immersed boundary demarcates a fluid–solid boundary. However, in general, this method is also applicable to flows with fluid–fluid boundaries. Furthermore, this paper focuses only

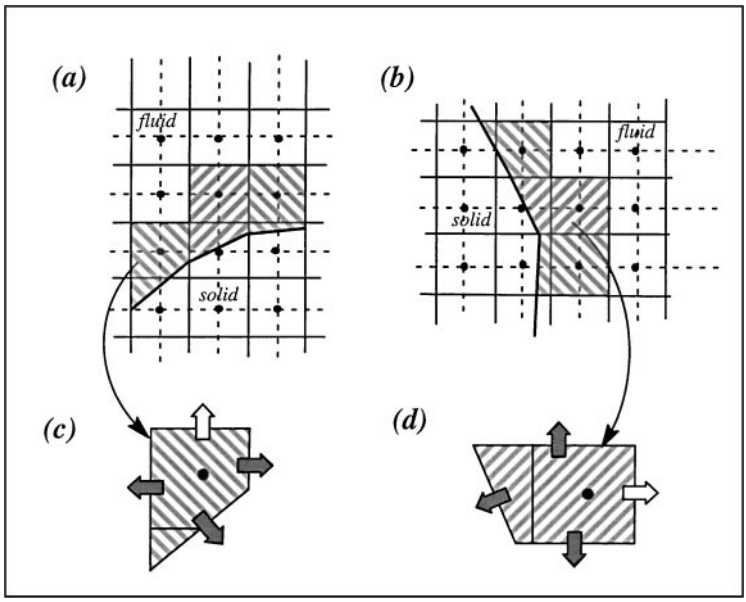

FIG. 2. Schematic of computational domain with immersed boundaries. (a) Boundary cells with immersed boundary located south of cell center. (b) Boundary cells with immersed boundary located west of cell center. (c) Typical reshaped trapezoidal boundary cells corresponding to case (a). (d) Typical boundary cells corresponding to case (b). Shaded arrows indicate fluxes that need special treatment.

on two-dimensional applications. However, the overall methodology to be described here is in principle extendable to three-dimensions.

The immersed boundary is first represented by a series of piecewise linear segments. Based on this representation of the immersed boundary, we identify cells in the underlying Cartesian mesh that are cut by the boundary and determine the intersection of the immersed boundary with the sides of these cut cells. Next, cells cut by the immersed boundary whose cell-center lie in the fluid are reshaped by discarding the part of these cells that lies in the solid. Pieces of cut cells whose center lie in the solid are absorbed by neighboring cells. This results in the formation of control-volumes which are trapezoidal in shape as shown in Figs. 2c and 2d. Details of this reshaping procedure can be found in Udaykumar *et al*. [36, 37].

Depending on the location and local orientation of the immersed boundary, trapezoidal cells of a wide variety of dimensions can be formed. The key issue here is to evaluate mass, convective and diffusive fluxes, and pressure gradients on the cell-faces of these trapezoidal cells from the neighboring cell-center values with adequate accuracy such that global second-order accuracy of the solver will be preserved. Furthermore, the current Cartesian grid method has been developed for unsteady viscous flows at moderately high Reynolds numbers. In such flows we expect that relatively thin boundary layers will be generated in the vicinity of the immersed boundary. These boundary layers are not only regions of high gradients but quite often, they are also the most important features of the flow field. Thus, accurate discretization of the equations is especially important in the boundary layers. Since all the trapezoidal cells are expected to lie within these boundary layer, this is another reason why adequate local accuracy is desirable for these cells.

For a uniform Cartesian mesh, the fluxes and pressure gradients on the face-centers can be computed to second-order accuracy by a simple linear approximation between neighboring cell-centers. This however is not the case for a trapezoidal boundary cell since the center of some of the faces of such a cell (marked by a shaded arrow in Figs. 2c and 2d) may not lie in a location which puts it in the middle of neighboring cell-centers where a linear approximation would give second-order accurate estimate of the gradients. Furthermore, some of the neighboring cell-centers do not even lie on the same side of the immersed boundary and therefore cannot be used in the differencing procedure. Thus, not only do we need a procedure for computing these face-center quantities which is accurate, we also require that the procedure adopted be capable of systematically handling reshaped boundary cells with a wide range of shapes. Our solution has been to use a compact two-dimensional polynomial interpolating function which allows us to obtain a second-order accurate approximation of the fluxes and gradients on the faces of the trapezoidal boundary cells from available neighboring cell-center values. The current interpolation scheme coupled with the finite-volume formulation guarantees that the accuracy and conservation property of the underlying algorithm is retained even in the presence of curved immersed boundaries.

As shown in Eq. (3), a finite-volume discretization of Navier–Stokes equations requires the estimation of surface integrals on the faces of each cell. The integrand (denoted here by f) can either involve the value or the normal derivative of a variable. An example of the former is the convective flux denoted by $(\rho\phi\boldsymbol{v}\cdot\hat{\boldsymbol{n}})$ and of the latter, the diffusive flux given by $(\Gamma\nabla\phi\cdot\hat{\boldsymbol{n}})$ where ϕ is a generic scalar variable. In addition to this, the pressure equation also requires evaluation of the normal pressure gradient. In order to estimate these surface integrals to second-order accuracy, the midpoint rule can be used and this requires accurate evaluation of the integrand at the center of the face. For regular cells which are away from

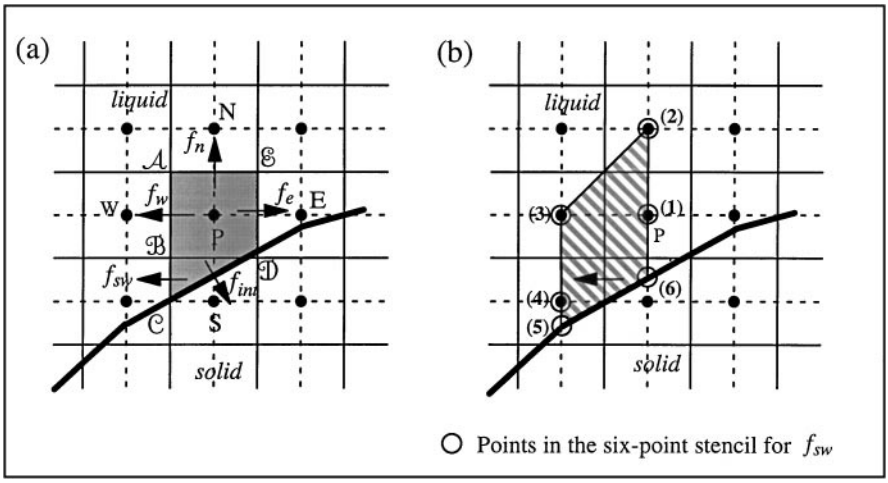

FIG. 3. Schematic of interpolation for cell face values and derivative at boundary cells. (a) Various fluxes required for trapezoidal boundary cell (b) trapezoidal region, and stencil used in computing f_{sw}.

the immersed boundary the integrand can be evaluated at the face-center to second-order accuracy in a straightforward manner by assuming a linear profile between nodes on the either side of the face. This is not the case for the trapezoidal boundary cells. Consider the trapezoidal boundary cell $\mathcal{ABCDE}$ in Fig. 3a. The face $\mathcal{ABC}$ of the trapezoidal cell is composed of two pieces; $\mathcal{AB}$ coming from the cell P and $\mathcal{BC}$ coming from cell S. The integral on this face can be decomposed as

$$\int_{\mathcal{AC}} f\,dy = \int_{\mathcal{AB}} f\,dy + \int_{\mathcal{BC}} f\,dy. \tag{10}$$

A second-order approximation to this integral can then be obtained as

$$\int_{\mathcal{AC}} f\,dy \approx f_w(y_{\mathcal{A}} - y_{\mathcal{B}}) + f_{sw}(y_{\mathcal{B}} - y_{\mathcal{C}}), \tag{11}$$

where f_w and f_{sw} are computed at the center of segments $\mathcal{AB}$ and $\mathcal{BC}$, respectively. If on the other hand, the face is cut by the immersed boundary such that it is smaller than a nominal cell face, as in the case of face $\mathcal{DE}$ then the integral can be approximated as

$$\int_{\mathcal{DE}} f\,dy \approx f_e(y_{\mathcal{E}} - y_{\mathcal{D}}), \tag{12}$$

where f_e is the flux computed at the center of the segment $\mathcal{DE}$. For non-boundary cells, these face-center values can be evaluated to second-order accuracy quite easily by a linear approximation and we would therefore like to evaluate f_w, f_{sw}, and f_e to within second-order accuracy also. Approximation of f_w to second-order accuracy is quite straightforward and is done in the same way as for the face of a non-boundary cell. For instance, if f_w requires the value of ϕ, this can be evaluated to second-order accuracy as

$$\phi_w = \phi_W \lambda_W + \phi_P(1 - \lambda_W), \tag{13}$$

where the linear interpolation factor λ_W is defined as

$$\lambda_W = \frac{x_P - x_w}{x_P - x_W}. \tag{14}$$

Alternatively, if f_w requires the normal gradient of ϕ as it would for the diffusion or pressure gradient terms, this can be approximated by a central-difference scheme as

$$\left(\frac{\partial \phi}{\partial x}\right)_w = \frac{\phi_P - \phi_W}{x_P - x_W}. \tag{15}$$

This approximation is second-order accurate when the cell face is midway between P and W, i.e., when the mesh is uniform.

Evaluation of f_{sw} or f_e to second-order accuracy is somewhat more complicated. Expressions such as (13) or (15) cannot be used since in many instances some of the neighboring nodes lie inside the immersed boundary. For instance, for the situation shown in Fig. 3a, the south node is inside the immersed boundary and cannot be used in the evaluation of f_{sw}. Even if neighboring nodes are available, as they are for the east face, it is not clear how a second-order accurate scheme can be constructed since f_e is not located on the line joining the neighboring cells centers and consequently, expressions such as (13) or (15) cannot approximate this flux to second-order accuracy. Thus, a different approach is needed here for evaluating these fluxes.

Our approach is to express the flow variables in terms of a two-dimensional polynomial interpolating function in an appropriate region and evaluate the fluxes such as f_{sw} or f_e based on this interpolating function. For instance, in order to approximate f_{sw}, we express ϕ in the shaded trapezoidal region shown in Fig. 3b in terms of a function that is linear in x and quadratic in y

$$\phi = c_1 x y^2 + c_2 y^2 + c_3 x y + c_4 y + c_5 x + c_6, \tag{16}$$

where c_1 to c_6 are six unknown coefficients. If f_{sw} involves the normal derivative of ϕ, this can be obtained by differentiating the interpolating function, i.e.,

$$\frac{\partial \phi}{\partial x} = c_1 y^2 + c_3 y + c_5. \tag{17}$$

The rationale for choosing (16) as the interpolating function for evaluating f_{sw} is as follows: the objective here is to evaluate $(\partial\phi/\partial x)$ at the center of $\mathcal{BC}$ to within at least second-order accuracy. Furthermore, we would like to do this with the most compact interpolant so as to minimize the size of the stencil required for the boundary cell. Clearly, a biquadratic interpolating function in the trapezoid shown in Fig. 3b would lead to second-order accurate evaluation of the derivative anywhere inside the trapezoid. However, a biquadratic function has nine unknown coefficients and therefore requires a large nine-point stencil. It turns out however that for the trapezoid shown in Fig. 3b, second-order accurate evaluation of the derivative on the cell face can be achieved by using an interpolating function that is quadratic in y but only linear in x. This is because $\mathcal{BC}$ is midway between the two parallel sides of the trapezoidal and in a manner analogous to central-differencing, linear interpolation in the x-direction leads to second-order accurate evaluation of the derivative at this location. On the other hand, this situation does not exist in the y-direction for the cell shown in Fig. 3b

and therefore a quadratic interpolation is necessary in this direction in order to obtain a second-order accurate approximation to $(\partial\phi/\partial x)$ at the center of $\mathcal{BC}$. In Appendix 1, we have demonstrated numerically that the linear-quadratic interpolating function shown in (16) does indeed result in second-order accurate evaluation of values and derivatives on a line which is located midway between the two parallel sides of a trapezoid.

It can be seen in Fig. 3b that the sides of the trapezoid in which the interpolation is performed pass through four nodal points and two boundary points. Thus, the six unknown coefficients in (16) can be expressed in terms of the values of ϕ at these six locations. To solve for c_n, we obtain the following system of equations by expressing the ϕ at the six locations in terms of the linear-quadratic interpolating function:

$$\begin{Bmatrix} \phi_1 \\ \phi_2 \\ \vdots \\ \phi_6 \end{Bmatrix} = \begin{bmatrix} x_1y_1^2 & y_1^2 & x_1y_1 & y_1 & x_1 & 1 \\ x_2y_2^2 & y_2^2 & x_2y_2 & y_2 & x_2 & 1 \\ \cdots & \cdots & \cdots & \cdots & \cdots & \cdots \\ x_6y_6^2 & y_6^2 & x_6y_6 & y_6 & x_6 & 1 \end{bmatrix} \begin{Bmatrix} c_1 \\ c_2 \\ \vdots \\ c_6 \end{Bmatrix}. \tag{18}$$

The coefficients can now be expressed in terms of values of ϕ at the six points by inverting (18), i.e.,

$$c_n = \sum_{j=1}^{6} b_{nj}\phi_j, \qquad n = 1, 2, \ldots, 6, \tag{19}$$

where b_{nj} are the elements of the inverse of the Vandermonde matrix [15] in (18).

After c_n is obtained, the value of ϕ at center of $\mathcal{BC}$ is expressed in the form of

$$\phi_{sw} = c_1x_{sw}y_{sw}^2 + c_2y_{sw}^2 + c_3x_{sw}y_{sw} + c_4y_{sw} + c_5x_{sw} + c_6 \tag{20}$$

and using (19) this can be rewritten as

$$\phi_{sw} = \sum_{j=1}^{6} \alpha_j\phi_j, \tag{21}$$

where

$$\alpha_j = b_{1j}x_{sw}y_{sw}^2 + b_{2j}y_{sw}^2 + b_{3j}x_{sw}y_{sw} + b_{4j}y_{sw} + b_{5j}x_{sw} + b_{6j}. \tag{22}$$

The value of $\partial\phi/\partial x$ at center of $\mathcal{BC}$ is expressed as

$$\left(\frac{\partial\phi}{\partial x}\right)_{sw} = c_1y_{sw}^2 + c_3y_{sw} + c_5 \tag{23}$$

and using (19) this can be written as

$$\left(\frac{\partial\phi}{\partial x}\right)_{sw} = \sum_{j=1}^{6} \beta_j\phi_j, \tag{24}$$

where

$$\beta_j = b_{1j}y_{sw}^2 + b_{3j}y_{sw} + b_{5j}. \tag{25}$$

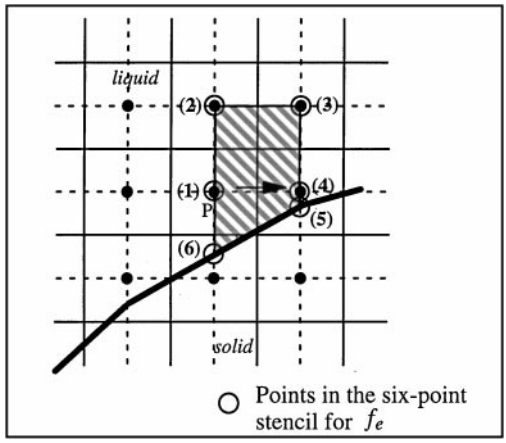

FIG. 4. Trapezoidal region and stencil used in computing f_e.

Note that α are β are coefficients that depend only on the mesh and the location and orientation of the immersed boundary. Therefore these can be computed once at the beginning of the solution procedure. Subsequently, relationships such as (21) and (24) can be used in the spatial discretization of the governing equations (3)–(7).

A similar interpolation procedure is also used for approximating f_e. For this, a linear-quadratic interpolant is used in the trapezoidal region shown in Fig. 4 and a relationship similar to (21) and (24) developed for approximating f_e. The six points contained in this stencil are also shown in the figure. It should be pointed out that the north face of the particular cell being considered here does not need special treatment since face-center values and derivatives can be computed to second-order accuracy using a linear approximation. However, in general there are also boundary cells which have their north or south faces cut by the immersed boundary (as shown in Fig. 2b). For these boundary cells too, the same approach is used to evaluate the fluxes on these cut faces. The only difference here is that the interpolating function is linear in y and quadratic in x.

Now we turn to the calculation of the flux on cell face $\mathcal{CD}$ which lies on the immersed boundary as shown in Fig. 2a. The integrated flux on this face can again be evaluated to second-order accuracy using the midpoint rule and as before we would like to evaluate the integrand at the center of face $\mathcal{CD}$ (denoted here by f_{int}) to second-order accuracy. In general both convective and diffusive fluxes are needed on this face and this requires the approximation of variables value as well as the normal derivative at the center of $\mathcal{CD}$. The value is usually available from a Dirichlet type boundary condition and hence no interpolation is required for this. Here we describe the approximation procedure for the normal derivative. The normal derivative on face $\mathcal{CD}$ can be decomposed as

$$\frac{\partial \phi}{\partial n} = \frac{\partial \phi}{\partial x}\hat{n}_x + \frac{\partial \phi}{\partial y}\hat{n}_y, \tag{26}$$

where $\hat{n}_x$ and $\hat{n}_y$ are the two components of the unit vector normal to face $\mathcal{CD}$. Since we know the shape of the immersed boundary, $\hat{n}_x$ and $\hat{n}_y$ are known. Therefore computation of the normal flux requires estimation of $\partial\phi/\partial x$ and $\partial\phi/\partial y$ at the center of the line segment $\mathcal{CD}$. For the cell being considered here, $\partial\phi/\partial y$ is computed to second-order accuracy with relative ease by expressing the ϕ variation along the vertical line in terms of a quadratic in

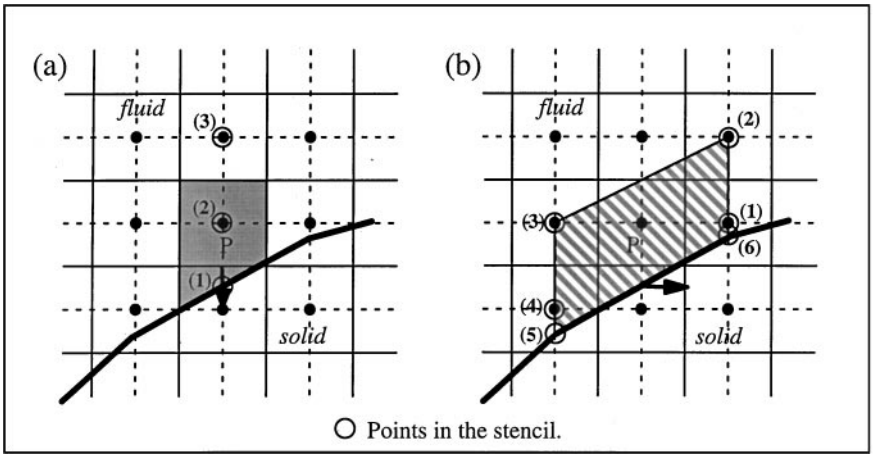

FIG. 5. Stencil for calculation of interface flux f_{int}. (a) Stencil for calculating $\partial\phi/\partial y$, (b) stencil for calculating $\partial\phi/\partial x$.

y as

$$\phi = a_1 y^2 + a_2 y + a_3. \tag{27}$$

The coefficients in the quadratic can be expressed in terms of the values of ϕ at the three points indicated in Fig. 5a. Subsequently, the normal derivative at the center of face $\mathcal{CD}$ is evaluated as

$$\left(\frac{\partial\phi}{\partial y}\right)_{int} = 2a_1 y_{int} + a_2 = \sum_{j=1}^{3} \tau_j^y \phi_j, \tag{28}$$

where again τ_j^y are coefficients which depend solely on the geometry of the boundary cell.

Unlike the calculation of $\partial\phi/\partial y$ for this cell, the calculation of $\partial\phi/\partial x$ is not straightforward. However, an approach consistent with the computation of f_{sw} and f_e can be used to estimate the value of this derivative to desired accuracy. Consider the trapezoid shown in Fig. 5b. Again, because the x-coordinate of the center of $\mathcal{CD}$ is midway between the two parallel sides of this trapezoid, expressing the variable in this trapezoid in terms of an interpolating function which is linear in x and quadratic in y allows us to obtain a second-order accurate approximation to $(\partial\phi/\partial x)_{sw}$ at the center of the line segment $\mathcal{CD}$. The procedure for this follows along lines similar to that shown for $(\partial\phi/\partial x)_{sw}$ and we get the following expression for the x-derivative on the interface,

$$\left(\frac{\partial\phi}{\partial x}\right)_{int} = \sum_{j=1}^{6} \tau_j^x \phi_j, \tag{29}$$

where the coefficients τ_j^x depend on the location and orientation of the immersed boundary in the neighborhood of the cell under consideration. Finally we get an expression of the form

$$\left(\frac{\partial\phi}{\partial n}\right)_{int} = \sum_{j=1}^{9} \tau_j \phi_j \tag{30}$$

for the normal gradient where τ_j can be obtained from (26), (28), and (29). Thus we obtain a nine-point stencil for the flux on the interface and the points in this stencil are shown in Figs. 5a and 5b. As can be seen from these figures, of these nine points, three points lie on the immersed boundary and their values are available from the prescribed boundary condition.

The overall solution procedure is as follows:

1. Determine the intersection of the immersed boundary with the Cartesian mesh.
2. Using this information, reshape the boundary cells.
3. For each reshaped boundary cell, compute and store the coefficients shown in (21), (24), and (30).
4. Use these coefficients to develop discrete expressions and operators for the various terms in the discretized Navier–Stokes equations.
5. Advance the discretized equations in time.

It should be pointed out that although most cells are four-sided trapezoidal cells, some five-sided cells and three-sided triangular cells are also encountered. However, the discretization of the governing equations for these cells can also be handled within the framework of the current interpolation scheme. With all of these features, the current solver can in principle, handle arbitrarily complex geometries. Furthermore, multiple immersed bodies can be handled as easily as a single body. This is in contrast to the body fitted grid where the grid topology can get quite complicated in the presence of multiple bodies. Finally, since the inside of the immersed boundary is also gridded, we also have the capability to solve a different set of equations inside the immersed boundary. For instance, equations of heat conduction could be solved inside the body if the objective were to study conjugate heat transfer.

It is worth pointing out here that our methodology differs significantly from the immersed interface method of Leveque and Li [18]. First, the interpolation scheme used in our method is different from that used by Leveque and Li. Second, their solver is based on a finite-difference method which is altered in the neighborhood of the immersed boundary in order to ensure global second-order accuracy. In contrast, our method is based on a finite-volume method, which is designed to give second-order global *and* local accuracy. Finally, to our knowledge, the method of Li and Leveque has not been extended to unsteady Navier–Stokes flows whereas the current method is primarily designed for such flow applications.

Inversion of Discrete Operators

The finite-volume discretization of (3) or (6) in a given cell P can be written as

$$\sum_{k=1}^{M} \chi_P^k \phi^k = b_P, \tag{31}$$

where the χ's denote the coefficients of the nodal values within a stencil of size M and b_P is the source term that contains the explicit terms as well as the terms involving boundary conditions. The term on the left-hand side represents a discretized Helmholtz operator in the case of the advection-diffusion equation and a Laplacian operator in the case of the pressure Poisson equation. In the present method, M is equal to five for non-boundary cells and this five-point stencil is shown in Fig. 1. For the trapezoidal boundary cells, the linear-quadratic

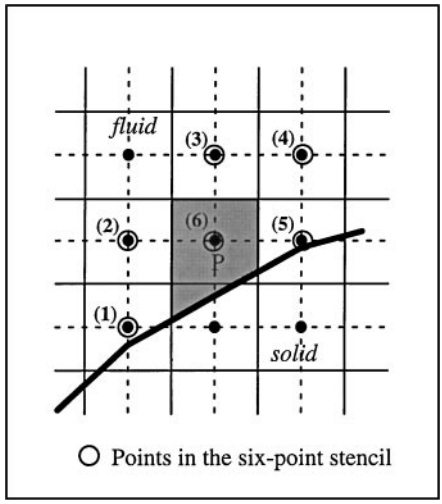

FIG. 6. Six point stencil resulting from discretization on a typical trapezoidal boundary cell.

interpolation described earlier results in a six-point stencil and for the particular boundary cell considered in Fig. 3, the resultant six point stencil is shown in Fig. 6.

The discretized advection-diffusion and pressure Poisson equations result in a coupled system of linear algebraic equations which requires the inversion of a large, sparse, banded matrix. The structure of this matrix is for the most part similar to that obtained on a Cartesian mesh without any immersed boundaries. However, the presence of the immersed boundary alters this matrix since the coefficients in (31) are different for the trapezoidal boundary cells. Furthermore, the rows in the matrix corresponding to the trapezoidal boundary cells also have additional elements since the stencil of the trapezoidal boundary cell is different from regular cells. The alternating direction line successive-overrelaxation (SOR) method [26] is one of the most widely used iterative methods for solving equations resulting from discretization on structured grids. A typical implementation of this technique, which we have adopted here, involves alternating sweeps along the two families of grid-lines. We find that even in the presence of the immersed boundaries, this method is extremely effective for the numerical solution of the discretized advection-diffusion equation and the residual can be reduced to an acceptable level within a few iterations.

The discretized pressure Poisson equation however exhibits slower convergence than the advection-diffusion equation. This is because the time-derivative term in the advection-diffusion equation tends to improve the diagonal dominance of the corresponding discretized operator. In fact due to its slow convergence, the solution of the discretized pressure equation is usually the most time-consuming part of a fractional-step algorithm. In the presence of immersed boundaries this behavior of the pressure equation can be further exacerbated since the stencil for the trapezoidal cells contains dependance on some neighboring cells which are not included in the line-SOR sweeps. For example, in Fig. 6 the coupling of cell P with cells (1) and (4) in the stencil is not included directly in any of the line-sweeps. Furthermore, depending on the aspect ratio of the trapezoidal boundary cell and the angle at which the immersed boundary cuts the cell, diagonal dominance in the pressure operator can be severely weakened. In the various simulations that have been performed using the current method, we have found that the simple line-SOR procedure can result in slow convergence of the pressure equation.

One remedy in such a situation is to resort to a multigrid method [5, 6]. However, the presence of the immersed boundary can complicate the implementation of a multigrid procedure since operations such as prolongation and restriction are difficult to perform near the immersed boundary. In contrast, Krylov subspace methods [15] are an attractive alternative since these are designed for general sparse matrices and therefore do not assume any structure in the matrix operator. Thus, the presence of the immersed boundary does not pose any additional complication for the implementation of these methods. A particularly suitable method in this class is the bi-conjugate gradient stabilized (Bi-CGSTAB) method [2, 38] which is applicable to non-symmetric matrices and provides relatively uniform convergence. However, as with all conjugate gradient methods, the convergence rate of the Bi-CGSTAB procedure depends critically on the choice of the preconditioner. The Jacobi preconditioner which has a trivial construction phase is used routinely in conjunction with unstructured grids. However, this preconditioner only produces marginal improvement in the convergence rate of conjugate gradient type algorithms. Other preconditioners such as those based on incomplete factorization can substantially increase the convergence rate but these usually have a non-trivial and expensive construction phase [2].

The structure of the matrix that results from the current Cartesian grid approach however presents us with another alternative choice of a preconditioner. As pointed out before, the presence of the immersed boundary only alters the underlying matrix operator in the rows corresponding to the boundary cells. Although this alteration slows the convergence rate of the line-SOR procedure, the convergence rate is still significantly faster than what is obtained with a simple point Jacobi method. It follows that the line-SOR procedure would serve as a better preconditioner than the Jacobi preconditioner. A further advantage of using the line-SOR as a preconditioner is that this procedure only requires the solution of tridiagonal systems and this can be accomplished with ease using the Thomas algorithm. Thus, in the solver developed here, we have used the line-SOR procedure as a preconditioner in the Bi-CGSTAB algorithm and find a significant improvement in the convergence rate over a simple line-SOR iterative procedure. In Fig. 7 the convergence rate of the line-SOR and Bi-CGSTAB for solution of the pressure equation on a 100×100 uniform Cartesian mesh has been compared. The flow configuration corresponds to low Reynolds number flow

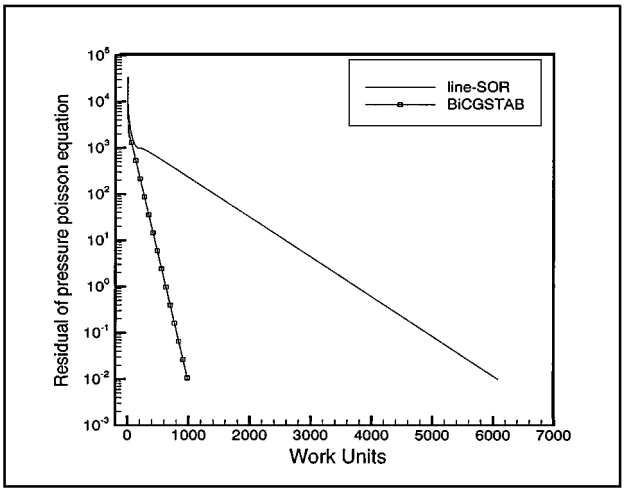

FIG. 7. Convergence rate of line-SOR and Bi-CGSTAB applied to the pressure equation for a typical problem.

past a circular cylinder and Fig. 7 shows the convergence of the pressure equation at the first time-step where the initial residual is extremely large. One iteration of the Bi-CGSTAB procedure requires two applications of the preconditioner and furthermore in addition to the preconditioning steps, the Bi-CGSTAB procedure also incurs a small but non-negligible overhead in its other steps. Therefore, in order to make a fair comparison between the two iterative methods, we have estimated the CPU time taken by one line-SOR iteration and designated this as one work unit. The CPU time used in one Bi-CGSTAB iteration is then estimated in terms of this work unit. In Fig. 7 we have plotted the residual in the pressure Poisson equation against the work units for the two different iterative schemes. The overrelaxation parameter chosen for this calculation is equal to 1.2 and for the geometry and grid in this particular simulation, this value results in the fastest convergence. We have also found that this optimum value of the relaxation parameter varies by less than 10% for all the various test cases that we have simulated. The plot clearly shows that Bi-CGSTAB is about six times faster than the line-SOR procedure. Given this significant convergence acceleration and the fact that Bi-CGSTAB can be implemented with relative ease, we find this to be an ideal methodology for solving the pressure Poisson equation in the current Cartesian grid method.

This completes the description of the current simulation methodology. It should be pointed out that although we have described the methodology only in the context of 2-D geometries, the interpolation procedure developed here is in principle extendable to three-dimensions. The key aspects to be addressed in extending this methodology to 3-D are efficient methods for representing curved 3-D interfaces and reconstructing boundary cells and further refinement of the solution procedure for the pressure equation. In the following sections we will focus on validating this methodology by simulating some canonical flows and demonstrating the capabilities of the method for simulating flows with complex immersed solid boundaries.

3. RESULTS AND DISCUSSIONS

(i) *Wannier Flow*

The most straightforward way of verifying the second-order spatial accuracy of the present method is to compute a flow which has a curved immersed boundary and one for which an analytical solution exists. The flow chosen here corresponds to two-dimensional Stokes flow past a circular cylinder placed next to a moving wall. The exact solution to this flow was given by Wannier [39] and is reproduced in Appendix 2. Here we have simulated this flow using our solver on four different uniform meshes. The meshes have equal spacing in the x and y directions and have N_x and N_y points in these two directions, respectively. In order to simulate Stokes flow, the convection terms have been turned off in our simulation. Computations have been carried out in the domain shown in Fig. 8 with the exact solution imposed on boundaries. Both the L_1 and L_2 norm of the global error have been computed as

$$\varepsilon_1 = \frac{1}{N_x N_y} \sum_{j=1}^{N_x N_y} \left| \phi_j^{numerical} - \phi_j^{exact} \right| \quad \text{and} \quad \varepsilon_2 = \left(\frac{1}{N_x N_y} \sum_{j=1}^{N_x N_y} \left(\phi_j^{numerical} - \phi_j^{exact} \right)^2 \right)^{1/2} \tag{32}$$

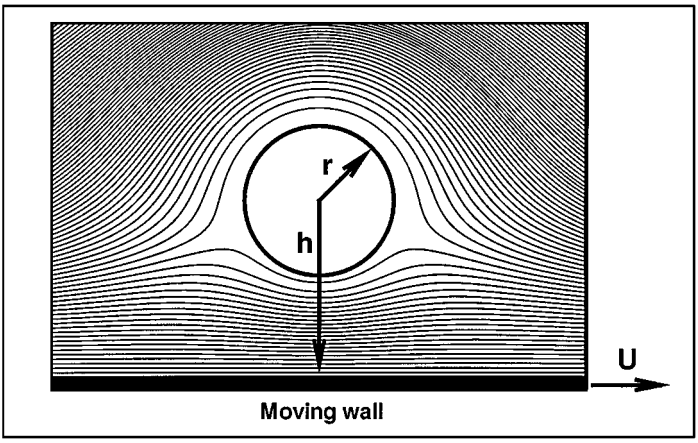

FIG. 8. Computational domain for Wannier flow simulation and the computed streamline pattern.

and in Fig. 9 we show a log–log plot of the errors in both velocity components u and v versus N_y. Also shown is a line with a slope of -2 which corresponds to second-order accurate convergence. The plot clearly shows that the global error in our computed solution decreases in a manner consistent with a second-order accurate scheme. This test therefore proves that the current approach of treating the fluxes in the boundary cells does indeed result in a solver which is globally second-order accurate.

(ii) *Flow Past a Circular Cylinder Immersed in a Freestream*

The exact solution of the Wannier flow allows us to confirm the accuracy of the solver in the Stokes flow regime. Here we validate the solver in the finite Reynolds number regime by simulating steady and unsteady flow past a circular cylinder immersed in an unbounded, uniform flow over a wide range of Reynolds numbers where the Reynolds

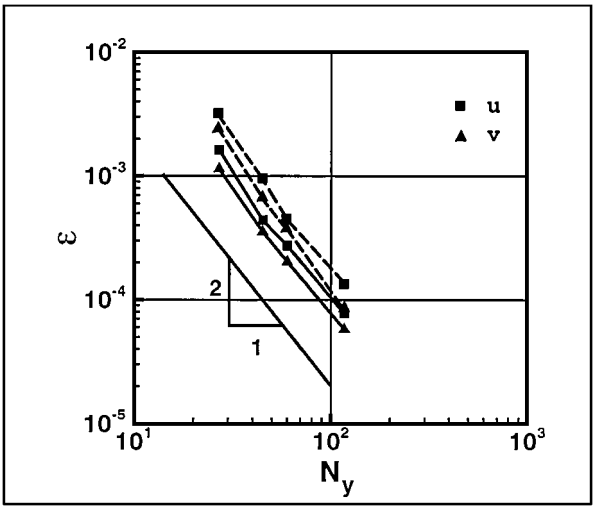

FIG. 9. Global error in u and v versus number of mesh points for Wannier flow. Solid and dashed lines indicate L_1 and L_2 norm of the error, respectively.

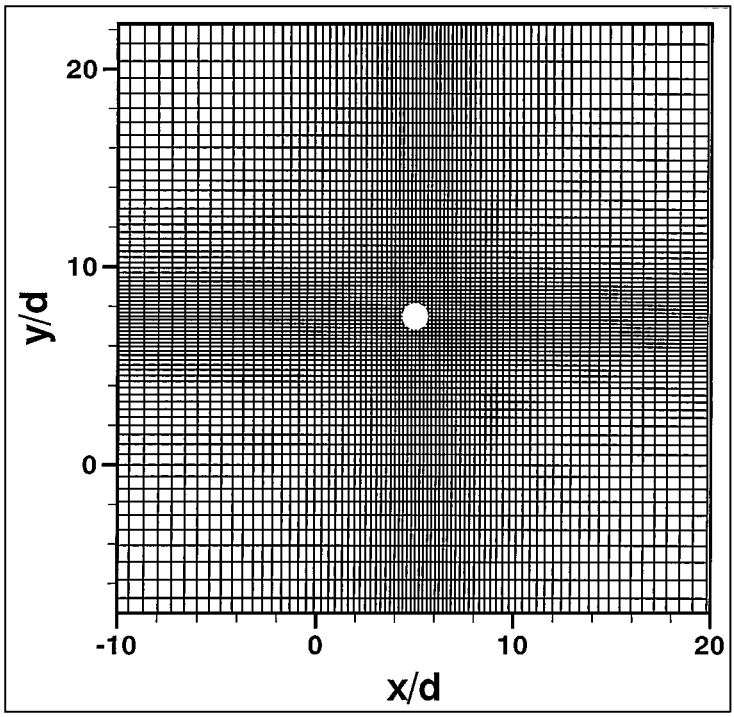

FIG. 10. Non-uniform mesh used for low Reynolds number simulation of flow past a circular cylinder. Only every other grid line is shown in both directions.

number is defined as $\mathrm{Re}_d = U_\infty d/v$ with d the cylinder diameter and U_∞ the freestream velocity. This flow had been studied quite extensively in the past and a number of numerical and experimental datasets exist for this flow which are useful for the purpose of validation. Simulations have been performed at $\mathrm{Re}_d = 20$, 40, 80, and 300 and results compared with established experimental and numerical results. All these simulations have been performed in a large $30d \times 30d$ domain so as to minimize the effect of the outer boundary on the development of the wake and Fig. 10 shows the 152×156 non-uniform mesh used in the low Reynolds number simulations. At the inlet and top and bottom boundaries we specify velocity corresponding to potential flow past a cylinder and a homogeneous Neumann boundary condition is applied at the exit boundary. We have also tested larger domain sizes in order to ensure that the results presented here are independent of the domain size. For all these simulations we first impose a small asymmetric disturbance at the inflow boundary for a short period of time and then allow the flow to evolve naturally after this. For $\mathrm{Re}_d = 20$ and 40 the wake eventually attains a steady symmetric state and this is consistent with the well established result that the cylinder wake is stable to perturbations below $\mathrm{Re}_d = 46 \pm 1$ [16, 27, 41]. Once the flow has reached a steady state we compute the drag coefficient defined by $C_D = drag\ force/(1/2)\rho U_\infty^2 d$ and the length of the recirculation zone and compare these with established results.

The streamline plots in Figs. 11a and 11b show the mean recirculation regions behind the circular cylinder at $\mathrm{Re}_d = 20$ and 40, respectively. In this steady flow regime, results using the current method are compared in Table I to the numerical simulation by Dennis and Chang [9] as well as experimental measurements of Tritton [34]. It is found that our results compare well with the other numerical simulations and experiments.

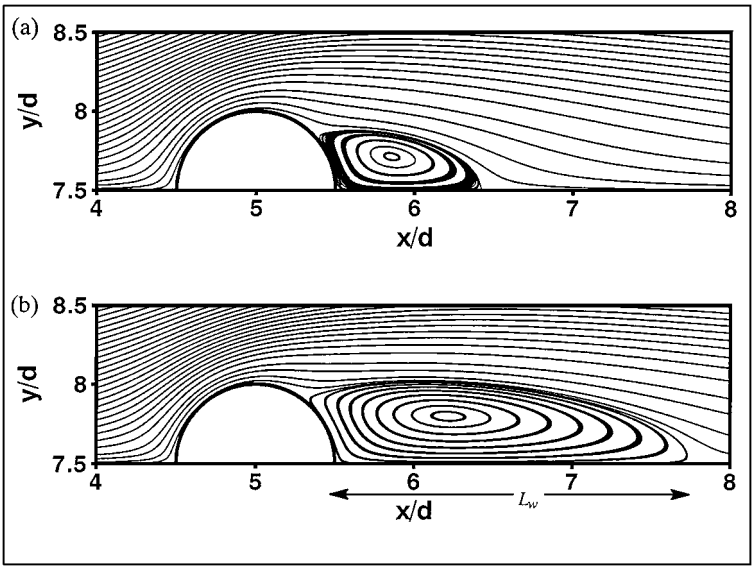

FIG. 11. Streamline plot of flow past a circular cylinder. (a) $\mathrm{Re}_d = 20$, (b) $\mathrm{Re}_d = 40$.

It is generally accepted that the wake of a cylinder immersed in a freestream first becomes unstable to perturbations at a critical Reynolds number of about $\mathrm{Re}_d = 46 \pm 1$ [16, 27]. Above this Reynolds number, a small asymmetric perturbation in the near wake will grow in time and lead to an unsteady wake and Karman vortex shedding. This is indeed what we find for our simulation at $\mathrm{Re}_d = 80$ which has been carried out on a 217×183 non-uniform mesh. Figure 12 shows the variation of the lift and drag coefficient with time and it shows how vortex shedding develops to a periodic state in time. The computed mean drag coefficient from the current simulation is about 1.37 which lies between the two experiments [34, 40]. The vortex shedding Strouhal number defined as $St = f d/U_\infty$, where f is the shedding frequency, is one of the key quantities that characterizes the vortex shedding process. Here we have estimated the Strouhal number from the periodic variation of the lift coefficient and the value comes out to be 0.15 which compares very well with the value obtained from

TABLE I

Comparison of Mean Drag Coefficient, Length of Wake Bubble L_w (Measured from Rear End of Cylinder), and Strouhal Number with Established Results

Reynolds number →	20		40		80		300	
Mesh →	152×156		152×156		217×183		217×183	
Study ↓	C_D	L_w/d	C_D	L_w/d	C_D	St	C_D	St
Tritton [34]	2.22	—	1.48	—	1.29	—	—	—
Weiselsberger [40]	2.05	—	1.70	—	1.45	—	1.22	—
Dennis and Chang [9]	2.05	0.94	1.52	2.35		—	—	—
Fornberg [11]	2.00	0.91	1.50	2.24	—	—	—	—
Mittal and Balachandar [21]	—	—	—	—	—	—	1.37	0.21
Williamson [41]	—	—	—	—	—	0.15	—	0.20
Current	2.03	0.92	1.52	2.27	1.37	0.15	1.38	0.21

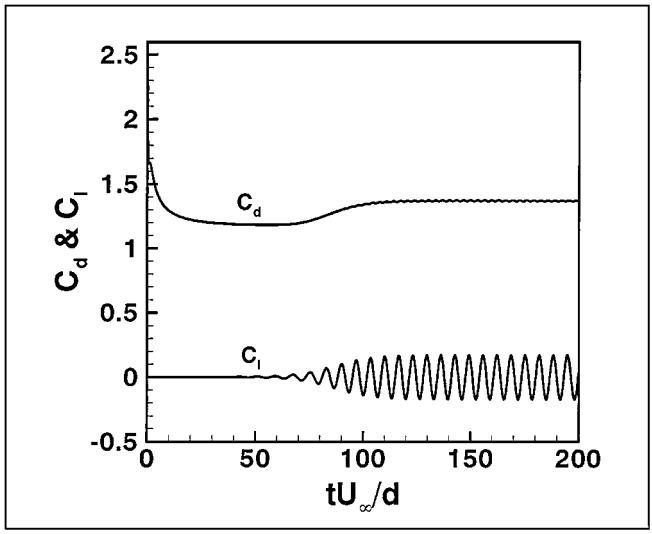

FIG. 12. Variation of lift and drag coefficients with time for $\mathrm{Re}_d = 80$.

experiments [41]. Figure 13 shows a plot of the streamline pattern and spanwise vorticity contour at one time instant and both plots show the classical Karman vortex street.

In addition to the low Reynolds number simulations, we have also carried out a simulation at a moderately high Reynolds number of 300. This simulation serves to demonstrate that the current methodology is capable of resolving thin boundary layers that develop in flows at these Reynolds numbers. The mesh used for this simulation is the same as the one used

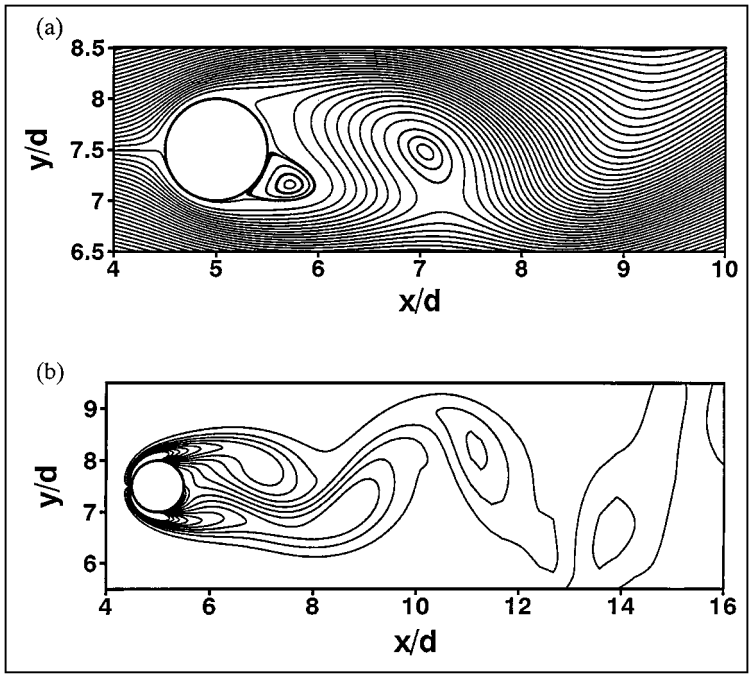

FIG. 13. Instantaneous streamline plot and vorticity contour plot in the near wake of the circular cylinder for $\mathrm{Re}_d = 80$, (a) streamline plot, (b) vorticity contours.

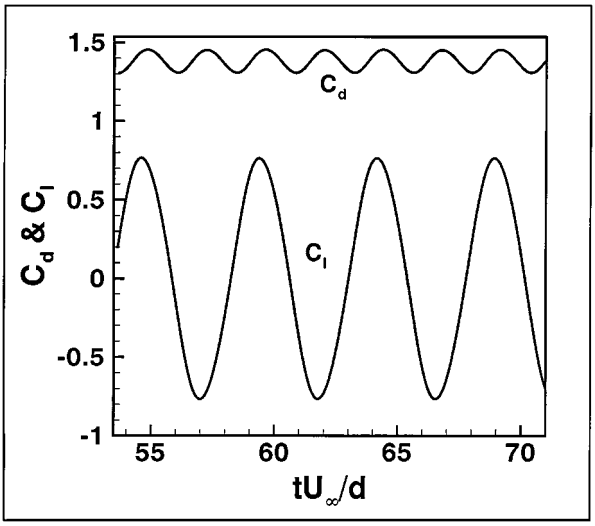

FIG. 14. Variation of lift and drag coefficients with time for $\mathrm{Re}_d = 300$.

for $\mathrm{Re}_d = 80$. This is a relatively coarse mesh and this coarse resolution severely tests the discretization scheme used in our solver for the boundary cells. Figure 14 shows the variation of lift and drag coefficient obtained from our simulation. The mean drag and Strouhal numbers have been computed from these time varying quantities and are included in Table I. It can be seen that these agree very well with the 2-D spectral simulation of Mittal and Balachandar [21] and the Strouhal number also matches well with the experiments of Williamson [41]. It should be pointed out that at this Reynolds number the cylinder wake is intrinsically three-dimensional whereas our simulation is two-dimensional and therefore does not allow spanwise variations. As shown by Mittal and Balachandar [19] one consequence of performing a 2-D simulation in this regime is that the drag is typically over-predicted. This is indeed what we observe for the current simulation. Thus, even though our mean drag matches with the other 2-D simulation, it is about 12% higher than the experimentally determined value of Weiselsberger [40].

In Fig. 15 we have shown contour plots of spanwise vorticity at one time-instant. Figure 15a shows a view of of the wake that extends to about $10d$ downstream from the cylinder and as expected, this plot shows the formation and evolution of compact Karman vortices in the wake. Figure 15b is a closeup view of the flow around the cylinder and the mesh superposed on the greyscale contour plot clearly shows that there are fewer than five points in the attached boundary layer. It can be seen that even with relatively low resolution provided here, the boundary layers on the cylinder surface are smooth indicating that the current treatment of the boundary cells adequately resolves thin boundary layers. It is useful to point out that for the $\mathrm{Re}_d = 300$ simulation the average CPU time required to complete one time step is about 3 s on a DEC Alpha workstation equipped with a 533 MHz processor and one shedding cycle requires about 675 CPU seconds.

(iii) *Flow past A Circular Cylinder in a Channel*

In addition to the case of flow past a cylinder immersed in a freestream, we have also validated our method by computing flow past a cylinder of diameter d placed symmetrically

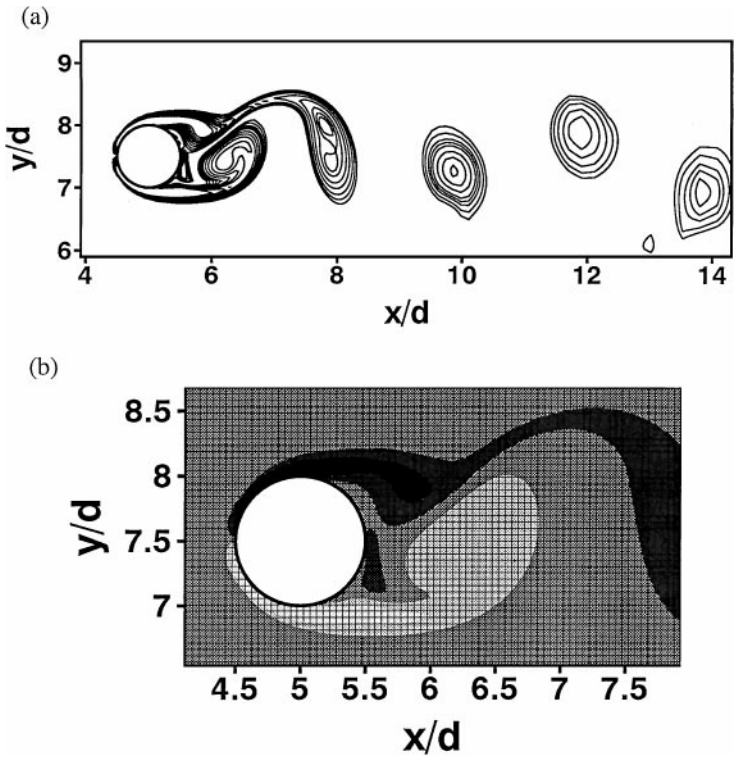

FIG. 15. Spanwise vorticity contour plots in the wake of the circular cylinder for $\mathrm{Re}_d = 300$. (a) View extending to 9*d* downstream of cylinder. (b) Closeup view showing the resolution provided to the attached boundary layers and separated shear layers.

in a planar channel of heigh h. A 195×73 grid is used for these simulations and Fig. 16 shows a schematic of the flow configuration that has been simulated here. A parabolic profile is specified at the channel inlet and the two main parameters in this flow are the blockage ratio $\beta = d/h$ and the Reynolds number defined as $\mathrm{Re} = Q/v$ where Q denotes the inlet volume flux. A systematic numerical study of this flow configuration over a range of parameters has been conducted by Chen *et al.* [7] and results from this study are used to validate our simulations. This flow shares some features with the case of a cylinder immersed in a freestream. In particular, vortex shedding is also observed in this flow beyond a critical Reynolds numbers. However, the critical Reynolds number is a strong function of the blockage ratio. Furthermore, the development of the vortices is also significantly

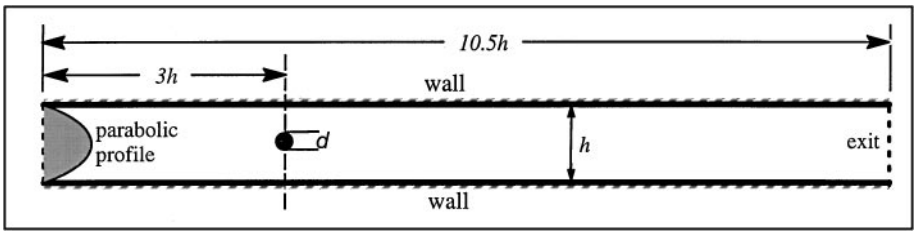

FIG. 16. Flow configuration for simulation of flow past a cylinder placed symmetrically in a planar channel.

affected by the boundary layers that develop on the channel walls. Thus, overall this is a more complicated flow and a good test case for our simulation methodology.

In our simulations we have focussed on predicting the critical Reynolds number for a blockage ratio of 0.2. The bifurcation analysis of Chen *et al.* [7] indicates that for this particular blockage ratio the critical Reynolds number is 231. We have performed a series of simulations ranging from a Reynolds number of 225 to 240 in order to pinpoint the critical Reynolds number. In each of these simulations, we provide a small asymmetric perturbation to the inlet flow for a short time period and subsequently allow the flow to develop naturally to a stationary state. In Fig. 17 we have plotted the temporal variation of the vertical velocity at one point in the near wake of the cylinder which directly indicates the growth or decay of the disturbance in the wake. Figure 17 shows that the perturbation reduces for Reynolds numbers up to 230 whereas it grows for Reynolds numbers greater than 232. Spanwise vorticity contours at a time instant corresponding to $tQ/h^2 = 100$ have

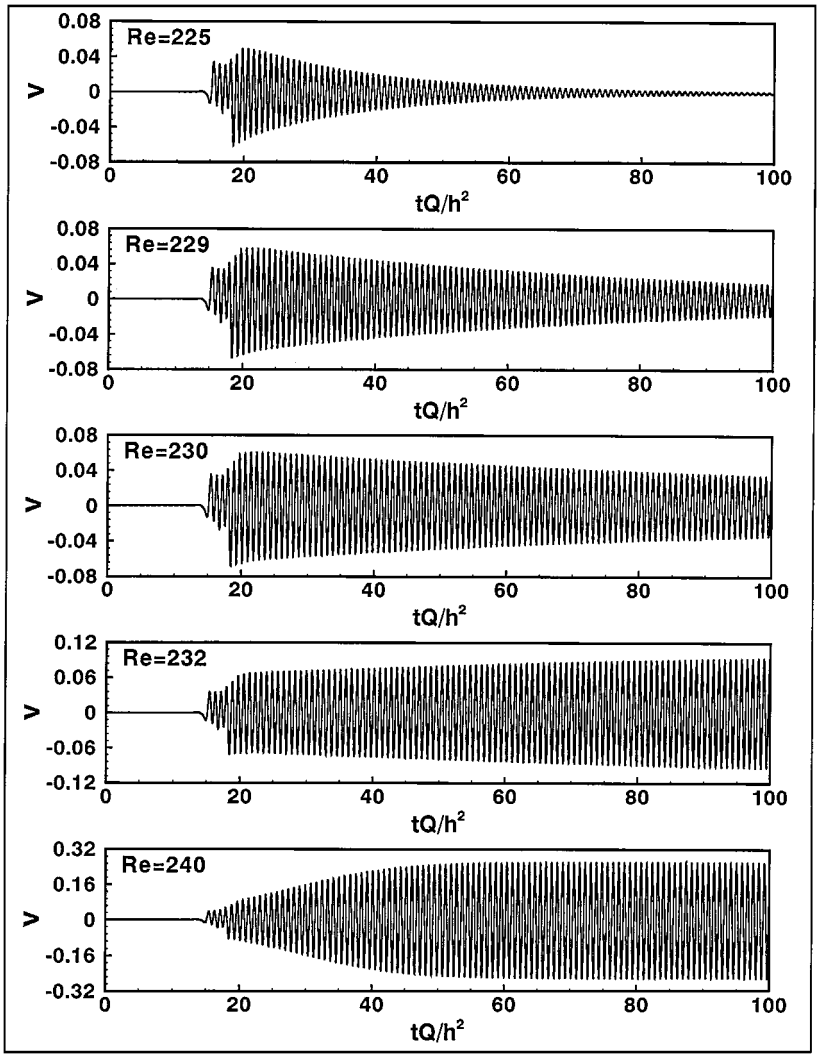

FIG. 17. Temporal variation of vertical velocity at one point in the wake of the cylinder.

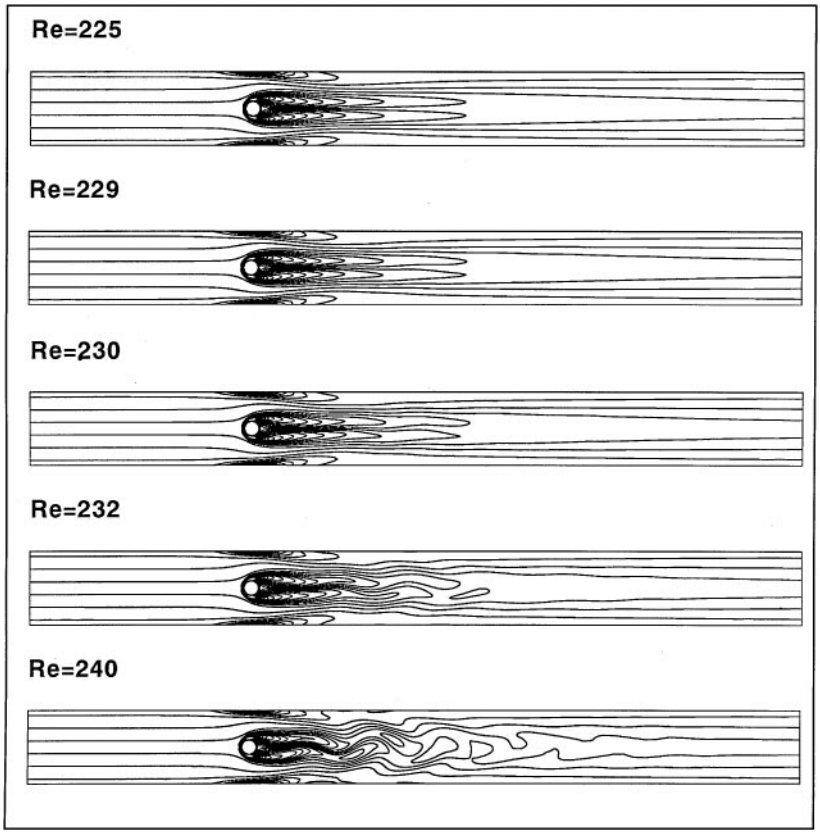

FIG. 18. Spanwise vorticity contours for flow past a cylinder in channel for various Reynolds numbers at $tQ/h^2 = 100$.

been plotted in Fig. 18 for all these five Reynolds numbers and we observe that for $\mathrm{Re} = 225$, 229, and 230 the wake has for the most part recovered its symmetric state. Substantial loss in symmetry can however be observed in the downstream region of the wake at $\mathrm{Re} = 232$ and this is accompanied by a weak flapping of the downstream end of the two attached shear layers and weak vortices are observed forming in the downstream wake. Finally, at a Reynolds number of 240, we observe vigorous vortex shedding in the wake of the cylinder. The critical Reynolds number predicted by our calculation is therefore between 230 and 232 which is in very good agreement with Chen *et al.* [7]. Furthermore, the Strouhal number defined as fdh/Q where f is the shedding frequency can be computed from the temporal variation of the vertical velocity and for $\mathrm{Re} = 240$ we obtain a Strouhal number of 0.17 which is identical to that obtained in the simulations of Chen *et al.* [7]. Thus all results from the current set of simulations match very well with the results of Chen *et al.* [7] which have been obtained using a body-fitted grid.

(iv) *Application to Complex Geometries*

We have verified the accuracy and fidelity of the solver for some relatively simple canonical flows. The main objective of the current work however is to develop an accurate and efficient numerical method that will allow us to simulate flows with extremely complicated

internal boundaries on simple Cartesian grids. We now demonstrate this capability of the solver by presenting results of two numerical simulations which involve flows with highly complex immersed boundaries.

*(*a*) Flow past a random array of cylinders.* The first complex configuration that we have simulated is that of flow past a random array of 95 cylinders. The random array is created in two steps; first we create a regular staggered array of 95 cylinders. Following this, the cylinders are moved in a randomly chosen direction by a fixed distance of $0.2d$ where d is the cylinder diameter. This procedure allows us to obtain a randomized array while maintaining some control over the minimum distance between two cylinders. A uniform inflow velocity is prescribed at the left boundary and a homogeneous Neumann exit boundary condition is applied at the right boundary. Furthermore, periodic conditions are used on the top and bottom boundaries. This configuration which is of relevance to porous media and heat exchanger applications represents a geometry which is extremely complex and one which would pose a serious challenge for a structured or unstructured grid generation routine. Here we have simulated this flow using a 250×250 uniform Cartesian mesh. The Reynolds number based on the inlet flow velocity and cylinder diameter is about 24. This Reynolds number is low enough that the flow eventually attains a steady state. However, at this Reynolds number we do expect to see complex recirculation zones behind the cylinders.

Figures 19a and 19b show the steady state pressure variation and streamlines, respectively. In Fig. 19a dark and light contours indicate high and low pressure, respectively. As expected, there is a significant drop in the pressure as the flow passes through the array and computed results indicate that on average, $(p_{out} - p_{in})/(1/2)\rho U_\infty^2 \approx -66$. The plotting package TECPLOT has been used to compute the streamlines from the velocity field and in addition to the equispaced streamlined introduced at the inlet, we have also introduced additional streamlines in the cylinder wakes in order to clearly show the recirculation zones. The streamlines exhibit extremely tortuous paths and this is typical of flows in randomized arrays [13]. We find that the current method has no difficulty in resolving the complex flow pattern and recirculation regions formed behind the cylinders.

*(*b*) Flow past a cascade of airfoils.* The second flow configuration we have chosen to simulate is of relevance to turbomachinery applications. It consists of flow past a periodic array of airfoils configured in a way similar to that found in a typical turbine or compressor. It should however be pointed out that all the airfoils here are stationary. Similar to the random array simulation, an inflow velocity is provided at the left boundary and a homogeneous Neumann exit boundary condition is applied at the right boundary. Furthermore, periodic conditions are used on the top and bottom boundaries. It is worth mentioning that this is a particularly severe test of the current methodology since the airfoil is only about four grid-spacings wide near the trailing edge and therefore the boundary cell discretization scheme has to contend with large curvatures. The Reynolds number based on the axial chord of the airfoil is 200 and a 400×250 uniform mesh is used for this simulation. The flow at this Reynolds number is inherently unsteady and Fig. 20a shows the streamline pattern at one time instant. It can be observed that a number of recirculation zones are in various stages of formation. Figure 20b shows the corresponding spanwise vorticity pattern and this plot gives a hint of complex vortex–vortex as well as vortex–body interactions that occur in this flow. Despite the relatively coarse resolution of the trailing edge geometry we find that the solver has no difficulty in obtaining the solution for this flow.

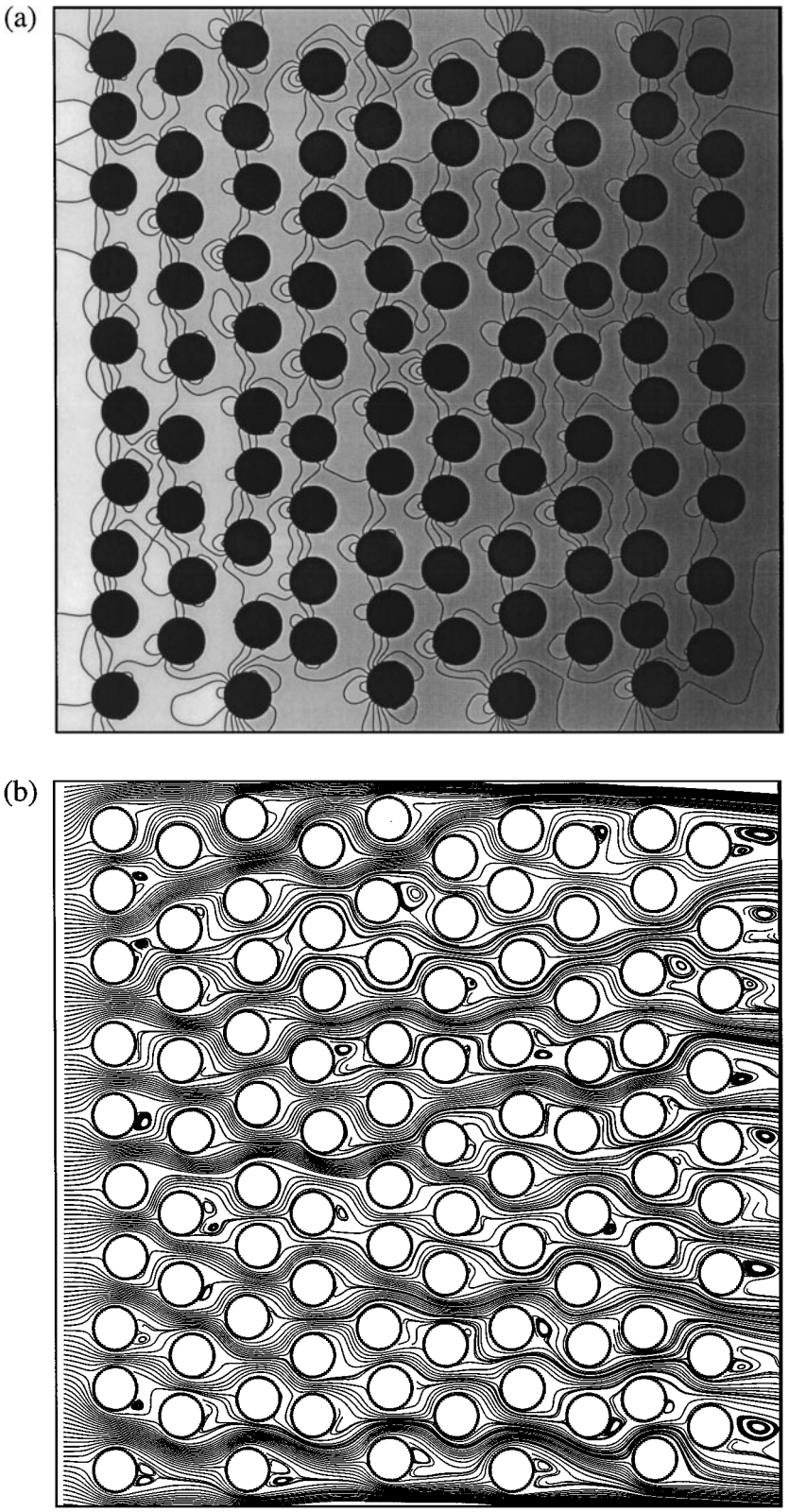

FIG. 19. Flow past a random array of 95 cylinders. (a) Pressure contours. Light and dark shades indicate high and low pressure, respectively. (b) Corresponding streamline pattern.

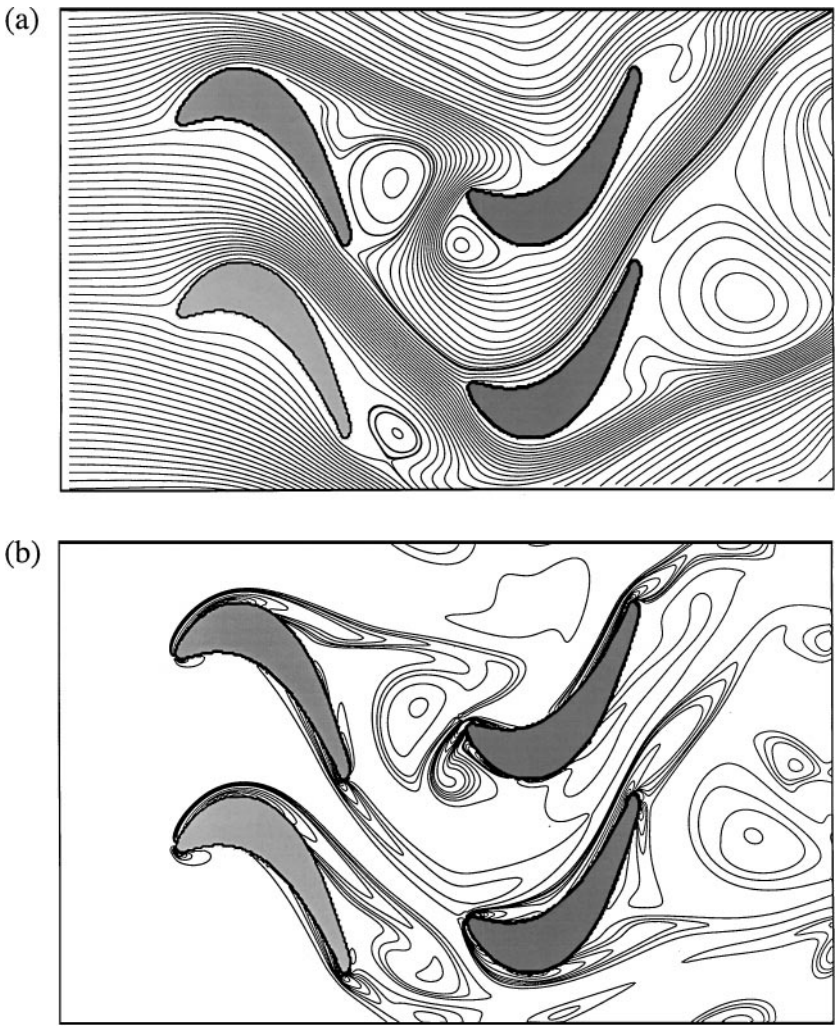

FIG. 20. Flow past a cascade of airfoils at one time instant. (a) Instantaneous streamlines, (b) corresponding spanwise vorticity contours.

4. CONCLUSIONS

A finite-volume based Cartesian grid method has been developed which allows us to simulate unsteady, viscous incompressible flows with complex immersed boundaries. The underlying method is based on a colocated arrangement of variables and a second-order central difference scheme is used for spatial differencing. Furthermore, the solution is advanced in time using a two-step fractional-step scheme. A new interpolation framework has been devised which allows us to systematically develop a spatial discretization for the cells cut by the immersed boundary that preserves the second-order spatial accuracy and conservation property of the underlying solver. This is especially crucial for the current solver since we plan to use it for simulating flows at moderately high Reynolds numbers. In such flows, relatively thin boundary layers are expected to form on the immersed boundaries and these have to be resolved adequately in order to obtain an accurate representation of the flow. We have found that the presence of immersed boundaries alters the conditioning of

the linear operators and slows down the iterative solution of the pressure Poisson equation. In the current solver, the convergence is accelerated by using a preconditioned conjugate gradient method where the preconditioner takes advantage of the structured nature of the underlying mesh.

The second-order global accuracy of the solver has been confirmed by simulating Stokes flow past a circular cylinder placed near a moving wall and comparing with the exact solution provided by Wannier [39]. Simulations of flow past a circular cylinder immersed in a uniform freestream have also been carried out in the Reynolds number range from 20 to 300. Key quantities such as mean drag coefficient, length of recirculation zone, and vortex shedding Strouhal number obtained from our simulations agree well with established experimental and numerical results. We have also simulated flow past a circular cylinder placed in a channel and we find that the solver is able to predict the critical Reynolds number of vortex shedding with a high degree of precision.

The main advantage of the current approach is that flows with extremely complex internal boundaries can be simulated with relative ease on simple Cartesian meshes. In order to demonstrate this capability of the solver, we have simulated two relatively complex flows. The first involves flow through a large array of randomly placed cylinders at finite Reynolds number and the second, flow through a cascade of airfoils at a relatively high Reynolds numbers. These flow configurations have been chosen since they would require generation of extremely complicated meshes if conventional structured/unstructured methods were to be used. The current Cartesian grid solver is able to simulate these flows with ease thereby demonstrating the advantage of the current approach for such complex flows.

APPENDIX 1

Accuracy of Linear-Quadratic Interpolating Function

In this section, we numerically demonstrate that the linear-quadratic interpolation procedure results in second-order accurate approximation of fluxes on the face of the boundary cell. Consider the trapezoid shown in Fig. 21 the size and shape of which is defined by the parameters Δ, ξ, and ζ. Comparison of this figure with Fig. 3b shows that this is a typical situation encountered while computing the flux on the face of a boundary cell. The six points marked on the trapezoid in Fig. 21 can be identified as the cell-enter locations in Fig. 3b. The objective here is to demonstrate that given the value of a variable ϕ at these six points, the value of ϕ and $\partial\phi/\partial x$ at a point (designated by p in Fig. 21) on the line midway between the parallel sides of the trapezoid can be obtained to second-order accuracy using a linear-quadratic interpolation function of the form shown in (16). We first assume a particular functional representation of ϕ in the two-dimensional Cartesian space and the values of ϕ at the six marked locations are computed from this functional representation. For our study we have chosen $\phi(x, y) = \sin(x)\cos(y)$. The greyscale contours in Fig. 21 show the variation of this function and it can be seen that the function has noticeable gradients in the vicinity of point p. The six values of ϕ are now used in conjunction with our linear-quadratic interpolant to obtain ϕ_p and $(\partial\phi/\partial x)_p$. The error in these numerically obtained values is then computed by comparing with the exact values which can be obtained directly from the given functional form of ϕ. Thus the error in the value of ϕ is given by $\varepsilon = |\phi_p - \sin(x_p)\cos(y_p)|$ and for the derivative it is given by $\varepsilon = |(\partial\phi/\partial x)_p - \cos(x_p)\cos(y_p)|$. The size of the trapezoid is now changed and the above procedure repeated for trapezoids of various sizes. In the current procedure, we keep ξ and ζ constant and only change Δ. Thus, all trapezoids

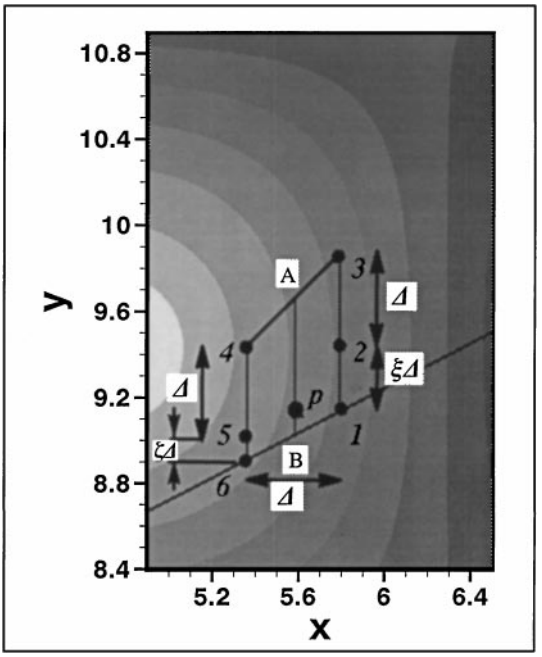

FIG. 21. Schematic showing the geometry of trapezoid used in our analysis of local accuracy. The greyscale contours show the variation in the assumed functional form of ϕ.

are similar in shape and their sizes are defined by the linear dimension Δ. Finally, the error can be plotted as a function of Δ and this allows us to deduce the order of accuracy of our approximation.

In Fig. 22 is shown a log–log plot of the error in ϕ_p and $(\partial\phi/\partial x)_p$ against Δ. Also shown is a line with slope of two which indicates second-order accuracy. Four different values of

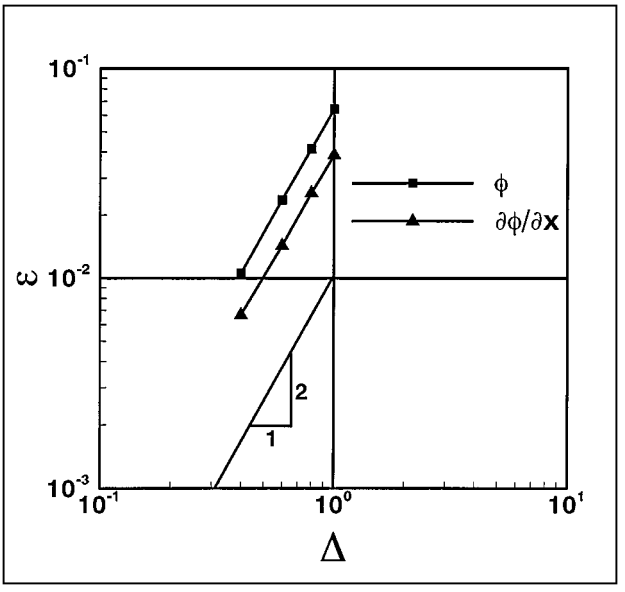

FIG. 22. Variation of local error in the estimation of variable values and derivatives on the face of a trapezoidal boundary cell with the size of the trapezoid.

Δ have been used and the plot clearly shows that the error decreases with Δ in a manner consistent with a second-order accurate scheme. Furthermore, we have also confirmed that a similar reduction in error is obtained at any point on the line AB. This proves that the linear-quadratic interpolation procedure results in second-order accurate estimation of face-center values and derivatives.

APPENDIX 2

Wannier Flow

Wannier [39] obtained the exact solution for Stokes flow past a spinning cylinder located in the vicinity of an infinite wall moving in the horizontal direction. The flow is bounded by the wall on one side and extends to infinity in the other directions. In our validation study we have considered the case where the cylinder is not spinning and the exact solution for this situation is given by

$$u = -\frac{2(A+Fy)}{\alpha}\left[(s+y)+\frac{\alpha}{\beta}(s-y)\right] - D - F\ln\left(\frac{\alpha}{\beta}\right) + \frac{B}{\alpha}\left[(s+2y)-\frac{2y(s+y)^2}{\alpha}\right] - \frac{C}{\beta}\left[(s-2y)+\frac{2y(s-y)^2}{\beta}\right] \tag{33}$$

$$v = \frac{2x}{\alpha\beta}(A+Fy)(\beta-\alpha) - \frac{2Bxy(s+y)}{\alpha^2} - \frac{2Cxy(s-y)}{\beta^2}, \tag{34}$$

where u and v are the horizontal and vertical velocities respectively and the various parameters are defined as

$$A = -\frac{Uh}{\ln(\gamma)}, \qquad B = \frac{2(h+s)U}{\ln(\gamma)} \tag{35}$$

$$C = \frac{2(h-s)U}{\ln(\gamma)}, \qquad D = -U, \qquad F = \frac{U}{\ln(\gamma)} \tag{36}$$

$$\alpha = x^2 + (s+y)^2, \qquad \beta = x^2 + (s-y)^2 \tag{37}$$

$$s^2 = h^2 - r^2, \qquad \gamma = \frac{h+s}{h-s}. \tag{38}$$

In the above expressions, U is the velocity of the wall, r the radius of the cylinder, and h the distance of the wall from the center of the cylinder.

ACKNOWLEDGMENTS

H.S.U., T.Y., and W.S. were supported by grants from Eglin Air Force Base and NASA Langley Research Center. R.M. was supported by ONR under Grant N00014–99–1–0389.

REFERENCES

1. A. S. Almgren, J. B. Bell, P. Colella, and T. Marthaler, A Cartesian grid projection method for the incompressible Euler equations in complex geometries, *SIAM J. Sci. Comput.* **18**(5), 1289 (1997).
2. R. Barrett, M. Berry, T. Chan, T. Demmel, J. Donato, J. Dongarra, V. Eijkhout, R. Pozo, C. Romine, and H. van der Vorst, *Templates for the Solution of Linear Systems: Building Blocks for Iterative Methods* (SIAM, Philadelphia, 1993).

3. S. A. Bayyuk, K. G. Powell, and B. van Leer, A simulation technique for 2-D unsteady inviscid flows around arbitrarily moving and deforming bodies of arbitrary geometry, AIAA Paper 93-3391-CP, 1993.
4. M. J. Berger and R. J. Le Veque, An adaptive Cartesian mesh algorithm for the Euler equation in arbitrary geometries, AIAA Paper 92-0443, 1992.
5. A. Brandt, Multilevel adaptive solutions to boundary value problems, *Math. Comput.* **31**, 333 (1977).
6. W. L. Briggs, *A Multigrid Tutorial* (SIAM, Philadelphia, 1987)
7. J. H. Chen, W. G. Pritchard, and S. J. Tavener, Bifurcation for flow past a cylinder between parallel planes, *J. Fluid Mech.* **284**, 23 (1995).
8. A. J. Chorin, Numerical solution of the Navier–Stokes equations, *Math. Comput.* **22**, 745 (1968).
9. S. C. R. Dennis and G.-Z. Chang, Numerical solution for steady flow past a circular cylinder at Reynolds numbers up to 100, *J. Fluid Mech.* **42**, 471 (1970).
10. D. De Zeeuw and K. G. Powell, An adaptively refined Cartesian mesh solver for the Euler equations, AIAA Paper 91-1542, 1991.
11. B. Fornberg, A numerical study of steady viscous flow past a circular cylinder, *J. Fluid Mech.* **98**, 819 (1980).
12. J. H. Ferziger and M. Peric, *Computational Methods for Fluid Dynamics* (Springer-Verlag, New York/Berlin, 1996).
13. J. G. Georgiadis, D. R. Noble, M. R. Uchanski, and R. O. Buckius, Tortuous micro-flow in large distorted packed beds, *J. Fluid Eng.* **118**, 434 (1996).
14. D. Goldstein, R. Handler, and L. Sirovich, Modeling of a no-slip surface with and external flow field, *J. Comput. Phys.* **105**, 354 (1993).
15. G. H. Golub and C. F. Van Loan, "*Matrix Computations*," 2nd ed. (Johns Hopkins Univ. Press, Baltimore, 1989).
16. C. P. Jackson, A finite element study of the onset of vortex shedding in flow past variously shaped bodies, *J. Fluid Mech.* **182**, 23 (1987).
17. H. C. Kan, H. S. Udaykumar, W. Shyy, and R. Tran-Son-Tay. Hydrodynamics of a compound drop with application to Leukocyte modeling, *Phys. Fluids* **10**(4), 760 (1998).
18. R. J. Leveque and Z. Li, The immersed interface method for elliptic equations with discontinuous coefficients and singular sources, *SIAM J. Numer. Anal.* **31**(4), 1019.
19. R. Mittal and S. Balachandar, Effect of intrinsic three-dimensionality on the lift and drag of nominally two-dimensional cylinders, *Phys. Fluids* **7**(8), 1841 (1995).
20. R. Mittal and S. Balachandar, Direct numerical simulation of flow past elliptic cylinders, *J. Comput. Phys.* **124**, 351 (1996).
21. R. Mittal and S. Balachandar, Inclusion of three-dimensional effects in simulations of two-dimensional bluff-body wake flows, in *Proceedings, 1997 ASME Fluids Engineering Division Summer Meeting*.
22. C. Nonino and G. Comini, An equal-order velocity-pressure algorithm for incompressible thermal flows, *Numer. Heat Transfer Part B* **32**, 1 (1997).
23. S. V. Patankar, *Numerical Heat Transfer and Fluid Flow* (Hemisphere, Washington, DC, 1980).
24. R. B. Pember, J. B. Bell, P. Colella, W. Y. Crutchfield, and M. L. Welcome, An adaptive Cartesian grid method for unsteady compressible flow in irregular regions, *J. Comput. Phys.* **120**, 278 (1995).
25. C. S. Peskin, Numerical analysis of blood flow in the heart, *J. Comput. Phys.* **25**, 220 (1977).
26. W. H. Press, S. A. Teukolsky, W. T. Vetterling, and B. P. Flannery, *Numerical Recipes in Fortran the Art of Scientific Computing* (Cambridge Univ. Press, Cambridge, UK, 1992).
27. M. Provensal, C. Mathis, and L. Boyer, Benard-von Karman instability: Transient and forced regimes, *J. Fluid Mech.* **182**, 1 (1987).
28. J. J. Quirk, An alternative to unstructured grids for computing gas dynamic flows around arbitrarily complex two-dimensional bodies, *Comput. Fluids* **23**(1), 125 (1994).
29. C. M. Rhie and W. L. Chow, Numerical study of the turbulent flow past an airfoil with trailing edge separation, *AIAA J.* **21**(11), 1525 (1983).
30. R. Scardovelli and S. Zaleski, Direct numerical simulation of free surface and interfacial flow, *Ann. Rev. Fluid Mech.* **31**, 567 (1999).

31. W. Shyy, H. S. Udaykumar, M. M. Rao, and R. W. Smith, *Computational Fluid Dynamics with Moving Boundaries* (Taylor & Francis, London, 1996).
32. F. Sotiropoulos and S. Abdallah, The discrete continuity equation in primitive variable solutions of incompressible flow, *J. Comput. Phys.* **95**, 212 (1991).
33. D. Tafti, Alternate formulations for the pressure equation Laplacian on a collocated grid for solving the unsteady incompressible Navier–Stokes equations, *J. Comput. Phys.* **116**, 143 (1995).
34. D. J. Tritton, Experiments on the flow past a circular cylinder at low Reynolds number, *J. Fluid Mech.* **6**, 547 (1959).
35. H. S. Udaykumar, W. Shyy, and M. M. Rao, Elafint: A mixed Eulerian–Lagrangian method for fluid flows with complex and moving boundaries, *Int. J. Numer. Methods Fluids* **22**, 691 (1996).
36. H. S. Udaykumar, H.-C. Kan, W. Shyy, and R. Tran-Son-Tay, Multiphase dynamics in arbitrary geometries on fixed Cartesian grids, *J. Comput. Phys.* **137**, 366 (1997).
37. H. S. Udaykumar, R. Mittal, and W. Shyy, Solid-liquid phase front computations in the sharp interface limit on fixed grids, *J. Comput. Phys.* **153**, 535–574 (1999).
38. H. A. Van der Vorst, Bi-CGSTAB: A fast and smoothly converging variant of Bi-CG for the solution of non-symmetric linear systems, *SIAM J. Sci. Statist. Comput.* **13**, 631 (1992).
39. G. H. Wannier, A. contribution to the hydrodynamics of lubrication, *Quart. Appl. Math.* **8**, 1 (1950).
40. C. Wieselsberger, *New Data on the Laws of Fluid Resistance*, NACA TN 84 (1922).
41. C. H. K. Williamson, Vortex dynamics in the cylinder wake, *Ann. Rev. Fluid Mech.* **28**, 477 (1996).
42. J. M. Yusof, *Interaction of Massive Particles with Turbulence*, Ph.D. Thesis. Department of Mechanical and Aerospace Engineering, Cornell University, 1996.
43. Y. Zang, R. L. Street, and J. R. Koseff, A non-staggered grid, fractional step method for time-dependent incompressible Navier–Stokes equations in Curvilinear coordinates, *J. Comput. Phys.* **114**, 18 (1994).