AIAA–99–3312

A CARTESIAN GRID METHOD FOR VISCOUS INCOMPRESSIBLE FLOWS WITH COMPLEX IMMERSED BOUNDARIES

T. Ye[1], R. Mittal[2]
Department of Mechanical Engineering
University of Florida
Gainesville, Florida, 32611
H. S. Udaykumar[3] and W. Shyy[4]
Department of Aerospace Engineering, Mechanics and Engineering Science
University of Florida
Gainesville, Florida, 32611

ABSTRACT

A Cartesian grid method has been developed for simulating 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. A new interpolation procedure for accurate discretization of the governing equation in cells that are cut by immersed boundaries is presented which preserves the second-order spatial accuracy of the underlying solver. The convergence of the pressure Poisson equation 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.

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 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[16]. 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 and has been used successfully for diffusion-dominated solidifications problems[19] 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[1,3,6,13] whereas applications to viscous flows are relatively scarce[10,16,18].

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.*[18] 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[4] which results in a fast solution of unsteady flows, (2) adoption of a new compact interpolation scheme near the immersed boundaries that retains the second-order accuracy and conservation property of the solver and (3) use of a preconditioned conjugate gradient method

---

[1] Graduate Student
[2] Assistant Professor, member
[3] Visiting Assistant Professor
[4] Professor, member

*Copyright © 1999 The American Institute of Aeronautics and Astronautics Inc. All rights reserved.

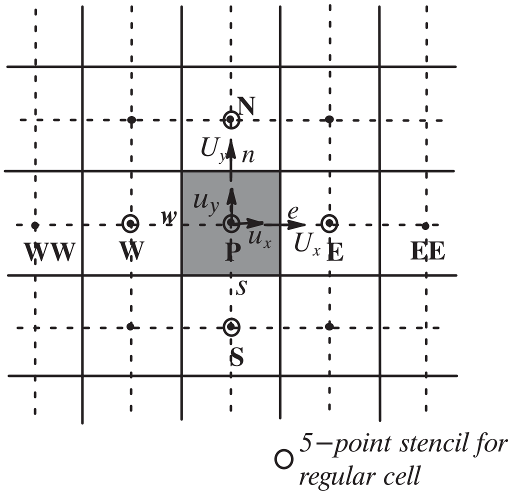


Figure 1. Schematic showing underlying Cartesian mesh and arrangement of cell-center and face-center velocities.

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 colocated (non-staggered) arrangement[8] of the primitive variables which are located at the cell-center. The integral form of the governing equations given by as follows:

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

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

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[4,25] 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 Figure 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 Figure 1 can therefore be written as follows:

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

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 cell-center intermediate velocity.

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

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

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

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

This results in the following equation for pressure

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

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 $\left(\nabla p^{n+1}\right) \cdot \hat{\boldsymbol{N}} = 0$ be used as the boundary condition for equation (6). Once the pressure is obtained by solving this equation, both the cell-center (*cc*) and face-center (*fc*) velocities are updated separately as follows:

$$\begin{aligned} \boldsymbol{u}^{n+1} &= \boldsymbol{u}^{*} - \Delta t\left(\nabla p^{n+1}\right)_{cc} \\ \boldsymbol{U}^{n+1} &= \boldsymbol{U}^{*} - \Delta t\left(\nabla p^{n+1}\right)_{fc} \end{aligned} \tag{7}$$

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. 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.

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, 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 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. Zang et al.[25] have used this procedure in conjunction with a curvilinear mesh solver to perform large-eddy simulations of turbulent flows and have found that the solver performs satisfactorily for high Reynolds number flows.

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 Figure 2a and b. 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, in this paper, we limit our discussion to two-dimensional flows although the algorithm to be described can be extended to three-dimensional flows in a straightforward manner.

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. 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 Figure 2c and 2d. Details of this reshaping procedure can be found in Udaykumar et al.[19].

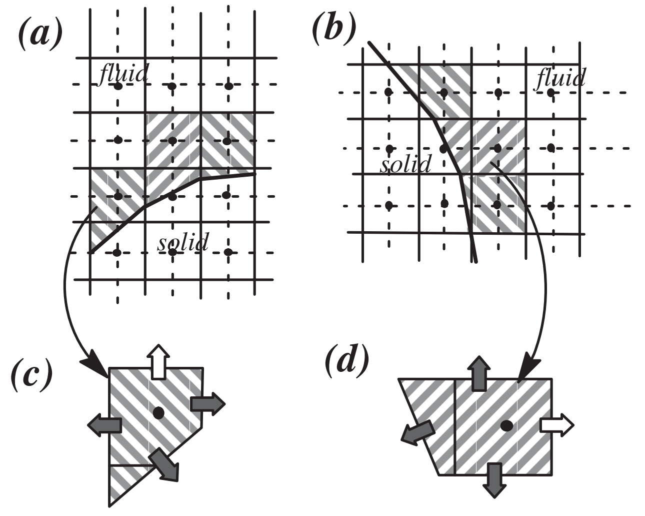


Figure 2. Schematic of computational domain with immersed boundaries. (a) boundary cells with immersed boundary located south of cell center. (b) boundary cells with immersed boundary located west of cell center. (c) typical reshaped trapezoidal boundary cells corresponding to case (a). (d) typical boundary cells corresponding to case (b).

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 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 is however not the case for a trapezoidal boundary cell since the center of some of the faces of such a cell (marked by shaded arrows in Figure 2c and d) 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. In the following, we describe the interpolation function for a typical trapezoidal boundary cell.

As shown in equation (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 $\left(\rho\phi\boldsymbol{v}\cdot\hat{\boldsymbol{n}}\right)$ and of the latter, the diffusive flux given by $\left(\Gamma\nabla\phi\cdot\hat{\boldsymbol{n}}\right)$ 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 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 $\mathcal{A}\mathfrak{B}\mathfrak{C}\mathfrak{D}\mathfrak{E}$ in Figure 3a. The face $\mathcal{A}\mathfrak{B}\mathfrak{C}$ of the trapezoidal cell is composed of two pieces; $\mathcal{A}\mathfrak{B}$ coming from the cell P and $\mathfrak{B}\mathfrak{C}$ coming from cell S. The integral on this face can be decomposed as

$$\int_{\mathcal{A}\mathfrak{C}} f\ dy = \int_{\mathcal{A}\mathfrak{B}} f\ dy + \int_{\mathfrak{B}\mathfrak{C}} f\ dy \tag{8}$$

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

$$\int_{\mathcal{A}\mathfrak{C}} f\ dy \approx f_w\left(y_{\mathcal{A}} - y_{\mathfrak{B}}\right) + f_{sw}\left(y_{\mathfrak{B}} - y_{\mathfrak{C}}\right) \tag{9}$$

where f_w and f_{sw} are computed at the center of segments $\mathcal{A}\mathfrak{B}$ and $\mathfrak{B}\mathfrak{C}$ 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 $\mathfrak{D}\mathfrak{E}$ then the integral can be approximated as

$$\int_{\mathfrak{D}\mathfrak{E}} f\ dy \approx f_e\left(y_{\mathfrak{E}} - y_{\mathfrak{D}}\right) \tag{10}$$

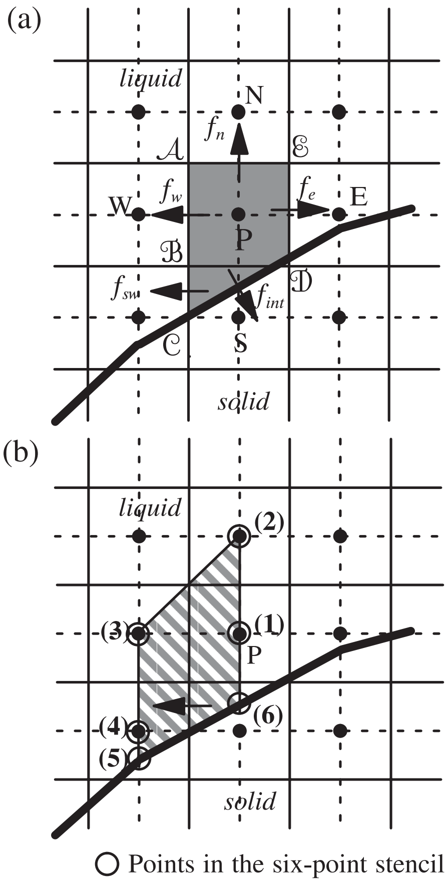


Figure 3. Schematic of interpolation for cell face values and derivative at boundary cells. (a) various fluxes required for trapezoidal boundary cell (b) trapezoidal region and stencil used in computing

where f_e is the flux computed at the center of the segment $\mathfrak{D}\mathfrak{E}$. 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.

Evaluation of f_{sw} or f_e to second-order accuracy is somewhat more complicated. Simple second–order central difference approximations cannot be used since in many instances some of the neighboring nodes can be inside the immersed boundary. For instance, for the situation shown in Figure 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, simple central difference schemes 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 Figure 3b in terms of a function that is linear in x and quadratic in y

$$\phi = c_1xy^2 + c_2y^2 + c_3xy + c_4y + c_5x + c_6 \tag{11}$$

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.

$$\frac{\partial\phi}{\partial x} = c_1y^2 + c_3y + c_5 \tag{12}$$

The rationale for choosing (11) 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 Figure 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 Figure 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 derivative at this location. On the other hand, this situation does not exist in the y-direction for the cell shown in Figure 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 $\mathcal{BC}$.

It can be seen in Figure 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 (11) can be expressed in terms of the values of ϕ at these six locations. Solve for c_n, the value of ϕ at center of $\mathcal{BC}$ can then be expressed in the form of

$$\phi_{sw} = c_1x_{sw}y_{sw}^2 + c_2y_{sw}^2 + c_3x_{sw}y_{sw} + c_4y_{sw} + c_5x_{sw} + c_6 \tag{13}$$

where c_n contains the values of $\phi_j,\ j = 1\ldots 6$, therefore

$$\phi_{sw} = \sum_{j=1}^{6}\alpha_j\phi_j \tag{14}$$

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

$$\left(\frac{\partial\phi}{\partial x}\right)_{sw} = c_1y_{sw}^2 + c_3y_{sw} + c_5 \tag{15}$$

which can be further written as

$$\left(\frac{\partial\phi}{\partial x}\right)_{sw} = \sum_{j=1}^{6}\beta_j\phi_j \tag{16}$$

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 (14) and (16) 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 Figure 4 and a relationship similar to (14) and (16) developed for approximating f_e. The six points contained in this stencil are shown in the Figure 4. 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 Figure 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.

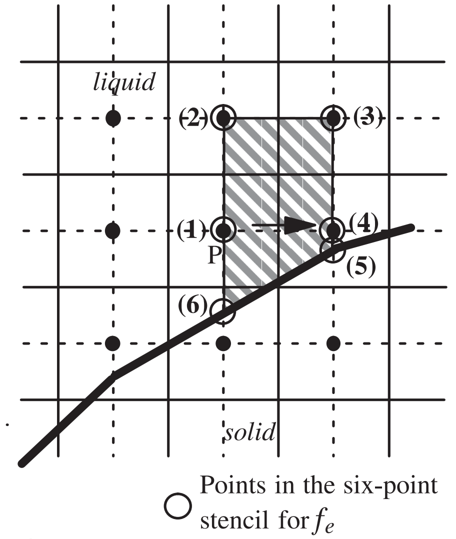


Figure 4. Trapezoidal region and stencil used in computing f_e

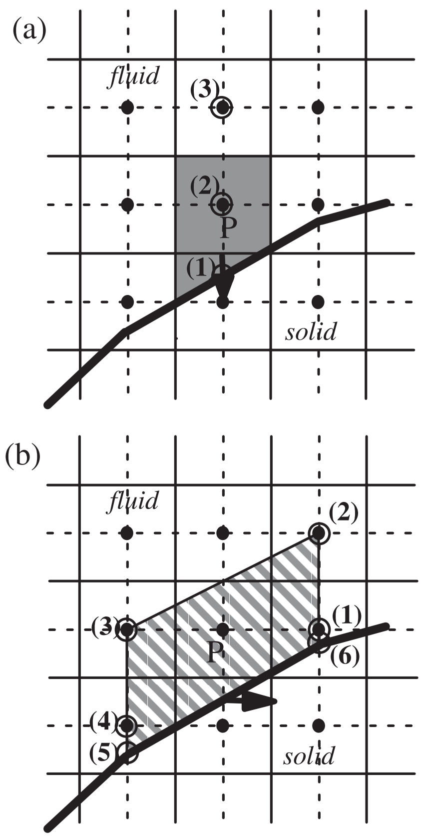
Now we turn to the calculation of the flux on cell face $\mathcal{CD}$ which lies on the immersed boundary as shown in Figure 2a. The integrated flux on this face can again be evaluated to second-order accuracy using the midpoint rule and as before we would like to evaluate the integrand at the center 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 approximation of variables value as well 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

$$\frac{\partial\phi}{\partial n} = \frac{\partial\phi}{\partial x}\hat{n}_x + \frac{\partial\phi}{\partial y}\hat{n}_y \tag{17}$$

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 y as follows

$$\phi = a_1 y^2 + a_2 y + a_3 \tag{18}$$

The coefficients in the quadratic can be expressed in terms of the values of ϕ at the three points indicated in Figure 5a.



○ Points in the stencil.

Figure 5. Stencil for calculation of interface flux. (a) stencil for calculating $\partial\phi/\partial y$ (b) stencil for calculating $\partial\phi/\partial x$

Subsequently, the normal derivative at the center of face $\mathcal{CD}$ is evaluated as:

$$\left(\frac{\partial\phi}{\partial y}\right)_{int} = 2a_1 y_{int} + a_2 = \sum_{j=1}^{3} \tau_j^y \phi_j \tag{19}$$

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 Figure 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:

$$\left(\frac{\partial\phi}{\partial x}\right)_{int} = \sum_{j=1}^{6} \tau_j^x \phi_j \tag{20}$$

where 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

$$\left(\frac{\partial\phi}{\partial n}\right)_{int} = \sum_{j=1}^{9} \tau_j \phi_j \tag{21}$$

for the normal gradient where τ_j can be obtained from (17), (19) and (20). Thus we obtain a nine-point stencil for the flux on the interface and the points in this stencil are shown in Figure 5a and b. 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.

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 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 can quite easily 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.

Inversion of Discrete Operators

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 cells. The main difference is that 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 alternative direction line successive-overrelaxation (SOR) method[14] is used to solve the discretized advection-diffusion equation. 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 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 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. 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 an extremely slow convergence for the pressure equation. Instead, a bi-conjugate gradient stabilized (Bi-CGSTAB) method[2,20] is used. This method, however, only works well in conjunction with a good preconditioner. 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. Thus the structured nature of the grid topology leads to significant benefit in the solution procedure.

This completes the description of the current simulation methodology. Further details regarding the algorithm can be found in the paper[24]. 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[21]. 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. The global error in the numerical solution is computed as:

$$\varepsilon = \frac{1}{N_x N_y} \sum_{j=1}^{N_x N_y} \left| \phi_j^{numerical} - \phi_j^{exact} \right| \tag{22}$$

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 Figure 6 with the exact solution imposed on boundaries. In Figure 7 we show a log-log plot of the error in both velocity components u and v versus N_x. 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 demonstrates that the current approach of treating the fluxes in the boundary cells does indeed result in a solver which is globally second-order accurate.

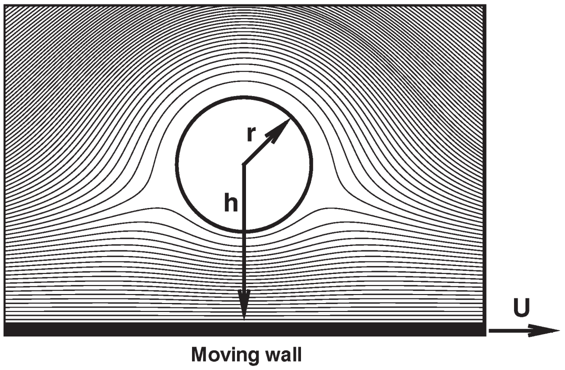


Figure 6. Computational domain and computed streamline pattern of Wannier flow.

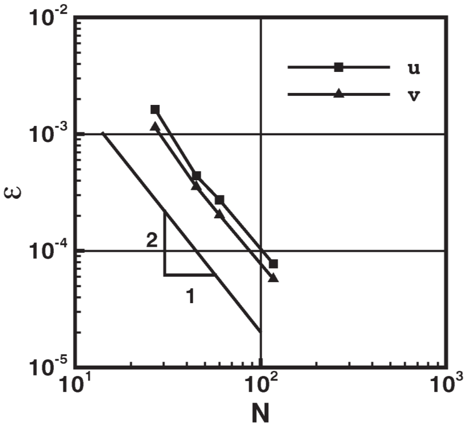


Figure 7. Global error of u and v as a function of mesh points for Wannier flow.

(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 range of Reynolds numbers where the Reynolds number is defined as $Re_d = U_\infty d/\nu$ 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 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 Figure 8 shows the 152×156 non-uniform mesh used in the low Reynolds number simulations. At the inlet and top and bottom boundaries

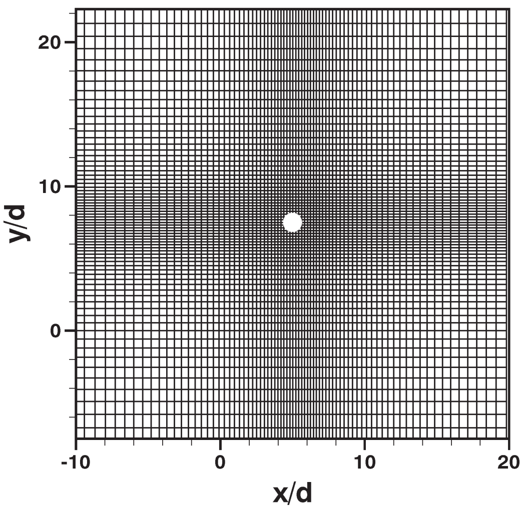


Figure 8. Typical non–uniform mesh for simulation of flow past circular cylinder (only every other grid line is shown).

we specify velocity corresponding to potential flow past a cylinder and a homogeneous Neumann boundary condition is applied at 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 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 $Re_d = 46$ [9,15,23]. Once the flow has reached a steady state we compute the drag coefficient defined by $C_D = \dfrac{drag\ force}{(1/2)\rho U_\infty^2 d}$ and the length of the recirculation zone and compare these with established results.

Re→	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
Ref. [17]	2.2	—	1.5	—	1.3	—	—	—
Ref.[22]	2.1	—	1.7	—	1.5	—	1.2	—
Ref. [5]	2.1	0.9	1.5	2.4	—	—	—	—
Ref. [7]	2.0	0.9	1.5	2.2	—	—	—	—
Ref. [12]	—	—	—	—	—	—	1.4	0.21
Ref. [23]	—	—	—	—	—	0.15	—	0.20
Current	2.0	0.9	1.5	2.3	1.4	0.15	1.4	0.21

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.

The streamline plots in Figure 9a and 9b show the mean recirculation regions behind the circular cylinder at Re_d= 20 and 40 respectively. In this steady flow regime, results using the current method are compared in Table I to numerical simulation by Dennis & Chang[5] as well as experimental measurements of Tritton[17]. It is found that our results compare well with the other numerical simulations and experiments.

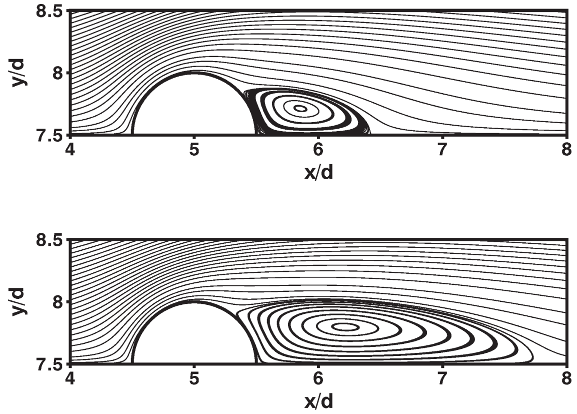


Figure 9. Streamline plot of flow past a circular cylinder. (a)$\mathrm{Re}_d = 20$ (b)$\mathrm{Re}_d = 40$

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 $Re_d = 46$ [9,15]. 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 $\mathrm{Re}_d = 80$. Figure 10 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 1.4 which lies between the two experiments[17,22]. The Strouhal number for vortex shedding is defined as $St = fd/U_\infty$, where f is the shedding frequency and is one of the key quantities that characterizes the vortex

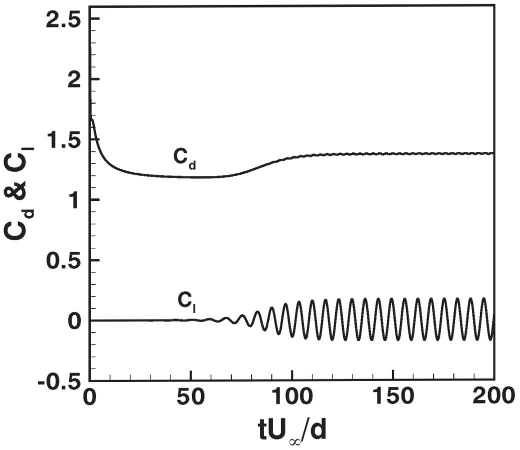


Figure 10. Variation of lift and drag coefficients with time for $\mathrm{Re}_d = 80$

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 experiments [23].

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 simulations is the same as that used 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. The mean drag and Strouhal numbers have been computed and included in Table I. It can be seen that these agree very well with the 2-D spectral simulation of Mittal & Balachandar[12] and the Strouhal number also matches well with the experiments of Williamson[23]. 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 & Balachandar[11] 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 simulations. 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[22].

In Figure 11 we have shown contour plots of spanwise vorticity at one time-instant. Figure 11a shows a view of of the wake that extends to about 10*d* downstream from the cylinder and as expected, this plot shows the formation and evolution of compact Karman vortices in the wake. Figure 11b is 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.

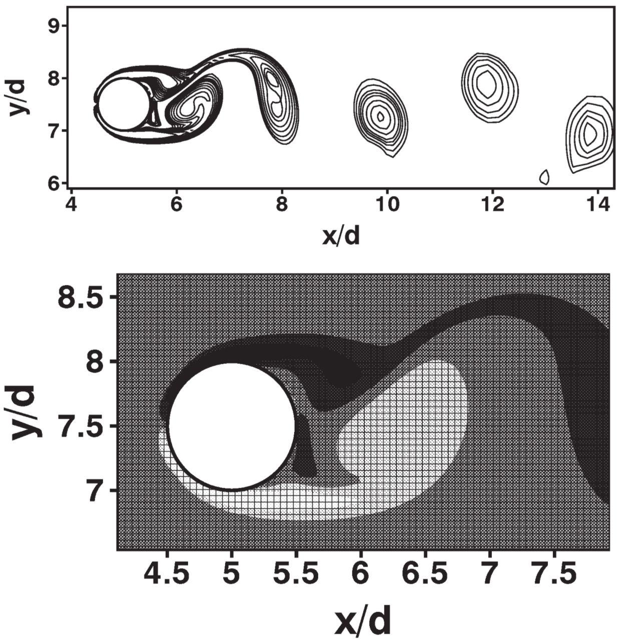


Figure 11. Spanwise vorticity contour plots in the wake of the circular cylinder for $\mathrm{Re}_d = 300$(a) View extending to 9*d* downstream of cylinder. (b) Closeup view showing the resolution provided to the attached boundary layers and separated shear layers.

(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 current solver by presenting results of a numerical simulation of flow past a periodic array of airfoils configured in a way similar to that found in a typical turbine or compressor. The configuration is shown in Figure 12. It should however be pointed out that all the airfoils here are stationary. 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-spacing 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 Figure 12a 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 12b shows the corresponding spanwise vorticity pattern and this plot gives a hint of the 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.

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 procedure 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

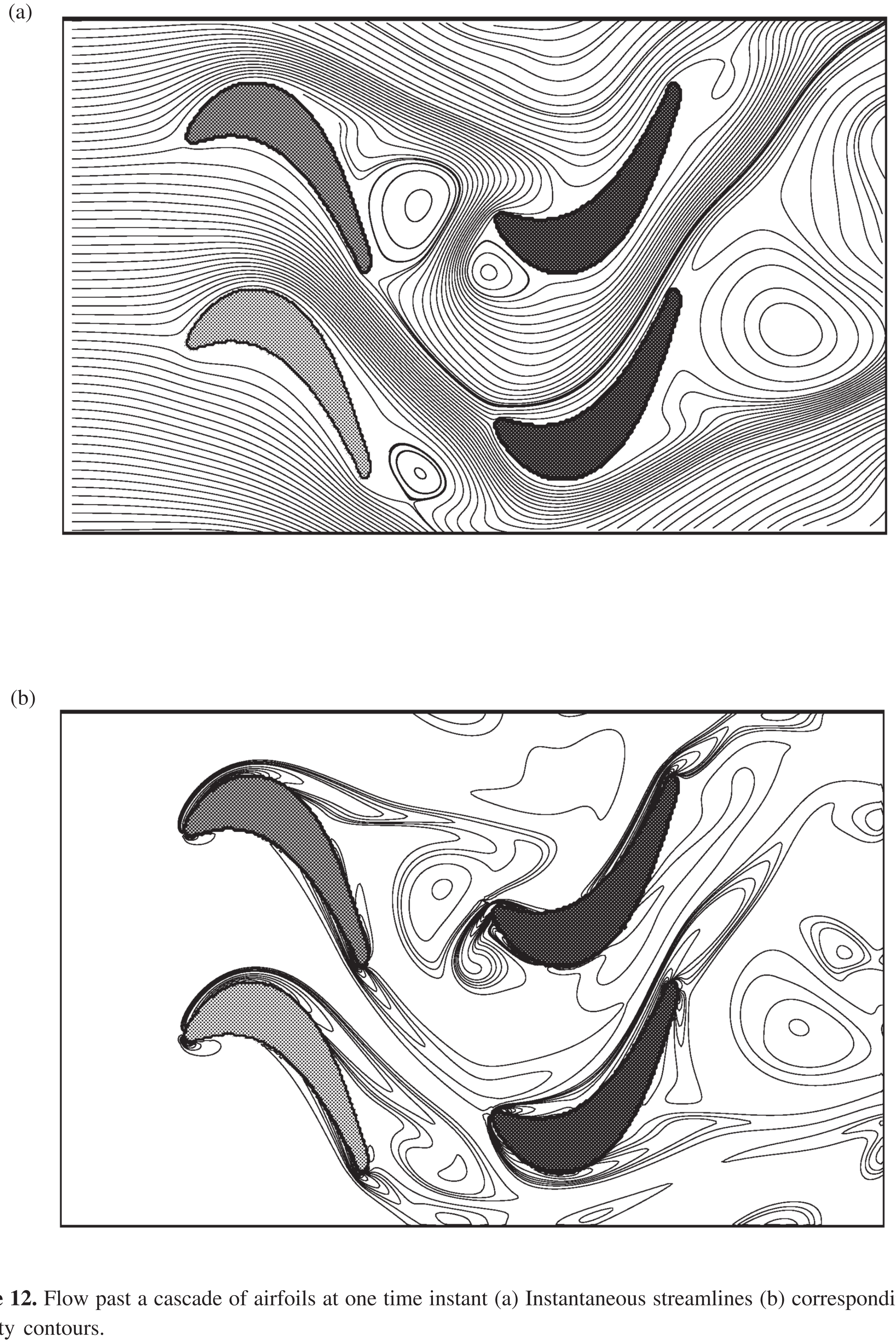


Figure 12. Flow past a cascade of airfoils at one time instant (a) Instantaneous streamlines (b) corresponding spanwise vorticity contours.

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[21]. 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.

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 a relatively complex flow which involves flow through a cascade of airfoils at a relatively high Reynolds number. This flow configuration has been chosen since it would require generation of a relatively complicated mesh if conventional structured/unstructured method were to be used. The current Cartesian grid solver is able to simulate this flow with ease thereby demonstrating the advantage of the current approach for flows with complex immersed boundaries.

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, Pennsylvania, 1993.

3 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).

4 A. J. Chorin, Numerical solution of the Navier–Stokes equations. *Math. Comput.* **22**, 745 (1968).

5 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).

6 D. De Zeeuw and K. G. Powell, An adaptively refined Cartesian mesh solver for the Euler equations. AIAA Paper 91–1542 (1991).

7 B. Fornberg, A numerical study of steady viscous flow past a circular cylinder. *J. Fluid Mech.* **98**, 819 (1980).

8 J. H. Ferziger and M. Peric, *Computational Methods for Fluid Dynamics*. (Springer–Verlag 1996)

9 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).

10 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).

11 R. Mittal and S. Balachandar, Effect of Intrinsic Three-Dimensionality on the Lift and Drag of Nominally Two-Dimensional Cylinders. *Phys. Fluids* 7 (8), 1841-1865 (1995).

12 R. Mittal and S. Balachandar, Inclusion of Three-Dimensional Effects in Simulations of Two-Dimensional Bluff-Body Wake Flows. Proceedings of *1997 ASME Fluids Engineering Division Summer Meeting*.

13 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).

14 W. H. Press, S. A. Teukolsky, W. T. Vetterling and B. P. Flannery, *Numerical Recipes in Fortran The Art of Scientific Computing*, (Cambridge University Press 1992)

15 M. Provensal, C. Mathis and L. Boyer, Benard–von Karman instability: Transient and forced regimes. *J. Fluid Mech.* **182**, 1 (1987).

16 W. Shyy, H. S. Udaykumar, M. M. Rao and R. W. Smith, *Computational Fluid Dynamics with Moving Boundaries*, (Taylor & Francis 1996).

17 D. J. Tritton, Experiments on the flow past a circular cylinder at low Reynolds number. *J. Fluid Mech.* **6,** 547 (1959).

18 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. Numerical Methods in Fluids,* **22,** 691 (1996).

19 H. S. Udaykumar, R. Mittal and W. Shyy, Solid-Liquid phase front computations in the sharp interface limit on fixed grids, to appear in *J. Comput. Phys* (1999).

20 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).

21 G. H. Wannier, A contribution to the hydrodynamics of lubrication. *Quart. Appl. Math.* **8**, 1 (1950).

22 C. Wieselsberger, New Data on the Laws of Fluid Resistance, *NACA TN 84 (*1922*)*.

23 C. H. K. Williamson, Vortex Dynamics in the Cylinder Wake. *Ann. Rev. Fluid Mech.* **28**, 477-539 (1996).

24 T. Ye, R. Mittal, H. S. Udaykumar and W. Shyy, An accurate Cartesian grid method for viscous incompressible flows with complex immersed boundaries, submitted to *J. Comput. Phys* (1999).

25 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).