Available online at www.sciencedirect.com
ScienceDirect
Journal of Computational Physics 227 (2008) 4825–4852
JOURNAL OF COMPUTATIONAL PHYSICS
www.elsevier.com/locate/jcp

A versatile sharp interface immersed boundary method for incompressible flows with complex boundaries

R. Mittal [a,*], H. Dong [a], M. Bozkurttas [a], F.M. Najjar [b], A. Vargas [a], A. von Loebbecke [a]

[a] Department of Mechanical and Aerospace Engineering, The George Washington University, Washington, DC 20052, United States
[b] Center for Simulation of Advanced Rockets, University of Illinois at Urbana-Champaign, Urbana, IL 61801, United States

Received 15 February 2007; received in revised form 4 January 2008; accepted 7 January 2008
Available online 1 February 2008

Abstract

A sharp interface immersed boundary method for simulating incompressible viscous flow past three-dimensional immersed bodies is described. The method employs a multi-dimensional ghost-cell methodology to satisfy the boundary conditions on the immersed boundary and the method is designed to handle highly complex three-dimensional, stationary, moving and/or deforming bodies. The complex immersed surfaces are represented by grids consisting of unstructured triangular elements; while the flow is computed on non-uniform Cartesian grids. The paper describes the salient features of the methodology with special emphasis on the immersed boundary treatment for stationary and moving boundaries. Simulations of a number of canonical two- and three-dimensional flows are used to verify the accuracy and fidelity of the solver over a range of Reynolds numbers. Flow past suddenly accelerated bodies are used to validate the solver for moving boundary problems. Finally two cases inspired from biology with highly complex three-dimensional bodies are simulated in order to demonstrate the versatility of the method.

Keywords: Computational fluid dynamics; Immersed boundary method; Ghost-cell; Body non-conformal grid methods

1. Introduction

Immersed boundary methods have emerged in recent years as a viable alternative to conventional body-conformal grid methods especially in problems involving complex stationary and/or moving boundaries. For such flows, the elimination of the need to establish a new body-conformal grid at each time-step can significantly simplify and speedup the solution procedure and also eliminates issues associated with regridding such as grid-quality and grid-interpolation errors. Immersed boundary methods can broadly be characterized under two categories [34]; first is the category of methods that employ "continuous forcing" wherein a forcing

* Corresponding author. Tel.: +1 202 994 9394.
E-mail address: mittal@gwu.edu (R. Mittal).

0021-9991/$ - see front matter © 2008 Elsevier Inc. All rights reserved.
doi:10.1016/j.jcp.2008.01.028

term is added to the continuous Navier–Stokes equations before they are discretized. Methods such as those of Peskin [41], Goldstein et al. [16] and Saiki and Biringen [47] fall in this category.

The second category consists of methods that employ discrete forcing where the forcing is either explicitly or implicitly applied to the discretized Navier–Stokes equations. These include methods of Udaykumar et al. [54], Ye et al. [63], Fadlun et al. [11], Kim et al. [19], Gibou et al. [15], You et al. [64], Balaras [3], Marella et al. [26], Ghias et al. [14] and others. The key advantage of the first category of methods is that they are formulated relatively independent of the spatial discretization and therefore can be implemented into an existing Navier–Stokes solver with relative ease. However, one of their drawbacks is that they produce a "diffuse" boundary. This means that the boundary condition on the immersed surface is not precisely satisfied at its actual location but within a localized region around the boundary. For the methods in the second category, the forcing scheme is very much dependent on the spatial discretization. However, one key advantage of the second category of methods is that for certain formulations, they allow for a sharp representation of the immersed boundary.

In the current paper, we describe a finite-difference based immersed boundary method that allows us to simulate incompressible flows with complex three-dimensional stationary or moving immersed boundaries on Cartesian grids that do not conform to the immersed boundary. The current method is based on the calculation of the variables on "ghost-cells" inside the body such that the boundary conditions are satisfied precisely on the immersed boundary. There are a number of features that distinguish this method from previously developed methods. First, unlike some past immersed boundary methods [16,23], there are no *ad-hoc* constants introduced in this procedure and neither is any momentum forcing term [41,23] employed in any of the fluid cells. Consequently, the method results in a "sharp" representation of the immersed boundary. This implies that the boundary conditions on the immersed boundary are imposed at the precise location of the immersed body and there is no spurious spreading of boundary forcing into the fluid as what usually occurs with diffuse interface methods [34].

Second, unlike the ghost-fluid method (GFM) [15], the interpolation scheme used here (and described in 2.2.2) stays well-conditioned in all cases and there is no need to resort to lower-order fixes for ill-conditioned situations. Furthermore, unlike GFM where interpolations are performed along the Cartesian directions, the interpolation operators in the current method are constructed in a direction normal to the immersed boundary and this significantly simplifies the implementation of the Neumann boundary conditions on the immersed boundary. Finally, in comparison to previous cut-cell based sharp interface methods [53,55], the current method provides the same spatial order of accuracy but is easily extended to complex 3D geometries. As noted in [34], successful implementation of the cut-cell method to 3D geometries has not yet been accomplished.

The method is designed from the ground up for simulations of flow with complex, moving, three-dimensional boundaries such as those encountered in bio-fluid mechanics and this paper describes the salient features of the methodology. The solver is validated by simulating a number of canonical two- (2D) and three-dimensional (3D) flows with stationary and moving boundaries, and comparing with established computed and/or experimental data. We also verify the spatial accuracy of the solver through a grid refinement study. Finally in order to showcase the capability of the method for handling general immersed boundaries, we present some computed results for flow with highly complex, non-canonical geometries.

The paper is organized as follows: Section 2 describes the numerical methodology including a detailed description of underlying flow solver and the overlaid immersed boundary methodology. In Section 3 we present computed results for a variety of cases that are intended to validate the solver and to firmly establish its accuracy. In this section we also include qualitative results from two biologically inspired simulations that involve highly complex moving/deforming bodies and these are meant to demonstrate the ability of the solver to handle complex immersed bodies. Finally, conclusions are presented in Section 4.

2. Numerical method

2.1. Governing equations and discretization scheme

The governing equations considered are the 3-D unsteady Navier–Stokes equations for a viscous incompressible flow with constant properties given by

$$\frac{\partial u_i}{\partial x_i} = 0 \tag{1}$$

$$\frac{\partial u_i}{\partial t} + \frac{\partial (u_i u_j)}{\partial x_j} = -\frac{1}{\rho}\frac{\partial p}{\partial x_i} + \nu \frac{\partial}{\partial x_j}\left(\frac{\partial u_i}{\partial x_j}\right) \tag{2}$$

where $i; j = 1, 2, 3$, u_i are the velocity components, p is the pressure, and where ρ and ν are the fluid density and kinematic viscosity.

The Navier–Stokes Eq. (2) are discretized using a cell-centered, collocated (non-staggered) arrangement of the primitive variables (u_i, p). In addition to the cell-center velocities (u_i), the face-center velocities, U_i, are computed (see Fig. 1). The equations are integrated in time using the fractional step method of Van-Kan [56] which consists of three sub-steps. In the first sub-step of this method, a modified momentum equation is solved and an intermediate velocity u^* obtained. A second-order, Adams–Bashforth scheme is employed for the convective terms while the diffusion terms are discretized using an implicit Crank–Nicolson scheme which eliminates the viscous stability constraint. In this sub-step, the following modified momentum equation is solved at the cell-nodes

$$\frac{u_i^* - u_i^n}{\Delta t} + \frac{1}{2}\left[3N_i^n - N_i^{n-1}\right] = -\frac{1}{\rho}\frac{\delta p^n}{\delta x_i} + \frac{1}{2}\left(D_i^* + D_i^n\right) \tag{3}$$

where $N_i = \frac{\delta(U_j u_i)}{\delta x_j}$ and $D_i = \nu \frac{\delta}{\delta x_j}\left(\frac{\delta u_i}{\delta x_j}\right)$ are the convective and diffusive terms respectively, and $\frac{\delta}{\delta x}$ corresponds to a second-order central difference. This equation is solved using a line-SOR scheme [1]. Subsequently, face-center velocities at this intermediate step U^* are computed by averaging the corresponding values at the grid nodes. Similar to a fully staggered arrangement, only the face velocity component normal to the cell-face is calculated and used for computing the volume flux from each cell. The following averaging procedure is followed:

$$\widetilde{u}_i = u_i^* + \Delta t \frac{1}{\rho}\left(\frac{\delta p^n}{\delta x_i}\right)_{\mathrm{cc}} \tag{4}$$

$$\widetilde{U}_1 = \gamma_{\mathrm{w}} \widetilde{u}_{1P} + (1 - \gamma_{\mathrm{w}})\widetilde{u}_{1W} \tag{5}$$

$$\widetilde{U}_2 = \gamma_{\mathrm{s}} \widetilde{u}_{2P} + (1 - \gamma_{\mathrm{s}})\widetilde{u}_{2S} \tag{6}$$

$$\widetilde{U}_3 = \gamma_{\mathrm{b}} \widetilde{u}_{3P} + (1 - \gamma_{\mathrm{b}})\widetilde{u}_{3B} \tag{7}$$

$$U_i^* = \widetilde{U}_i - \Delta t \frac{1}{\rho}\left(\frac{\delta p^n}{\delta x_i}\right)_{\mathrm{fc}} \tag{8}$$

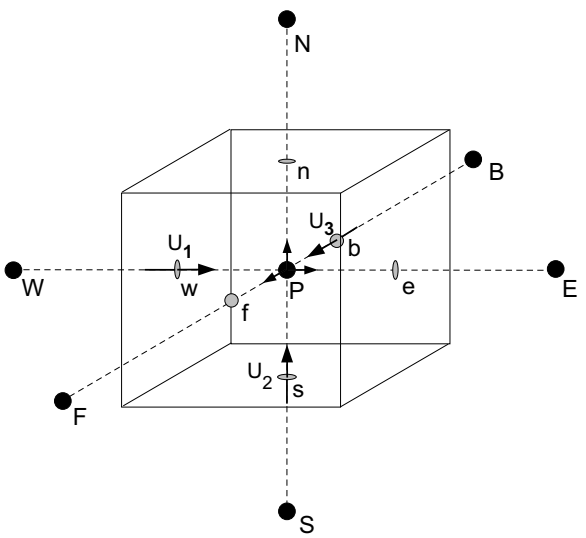

Fig. 1. Schematic describing the naming convention and location of velocity components employed in the spatial discretization of the governing equations.

where γ_w, γ_s and γ_b are the weights corresponding to linear interpolation for the west, south and back face velocity components respectively. Furthermore, cc and fc denote gradients computed at cell-centers and face-centers, respectively. The above procedure is necessary to eliminate odd–even decoupling that usually occurs with non-staggered methods and which leads to large pressure variations in space. The second sub-step requires the solution of the pressure correction equation

$$\frac{u_i^{n+1} - u_i^*}{\Delta t} = -\frac{1}{\rho}\frac{\delta p'}{\delta x_i} \tag{9}$$

which is solved with the constraint that the final velocity u_i^{n+1} be divergence-free. This gives the following Poisson equation for the pressure correction

$$\frac{1}{\rho}\frac{\delta}{\delta x_i}\left(\frac{\delta p'}{\delta x_i}\right) = \frac{1}{\Delta t}\frac{\delta U_i^*}{\delta x_i} \tag{10}$$

and a Neumann boundary condition imposed on this pressure correction at all boundaries.

This Poisson equation is solved with a highly efficient geometric multigrid method [4] which employs a Gauss–Siedel line-SOR smoother [43]. The ability to employ such methods is another key advantage of the current Cartesian grid approach over body-conformal unstructured grid approaches. Geometrical multigrid methods are relatively simple to implement and have very limited memory overhead. Furthermore, when coupled with powerful smoothers like line-Gauss–Siedel, they can lead to a numerical solution of the pressure Poisson equation which scales almost linearly with the number of grid points. In contrast, for unstructured body-conformal methods, one has to either resort to algebraic multigrid methods [45,46] or other more complex methods such as agglomeration multigrid [28]. Another choice for solving the pressure Poisson equation would be Krylov subspace based methods (such as conjugate gradient or GMRES) but these require effective preconditioners to provide good performance. Our past experience with both stationary and non-stationary iterative methods [63,55] indicates that geometric multigrid methods are very well suited for sharp interface immersed boundary methods and we have therefore used this method in the current solver.

Once the pressure correction is obtained, the pressure and velocity are updated as

$$p^{n+1} = p^n + p' \tag{11}$$

$$u_i^{n+1} = u_i^* - \Delta t \frac{1}{\rho}\left(\frac{\delta p'}{\delta x_i}\right)_{cc} \tag{12}$$

$$U_i^{n+1} = U_i^* - \Delta t \frac{1}{\rho}\left(\frac{\delta p'}{\delta x_i}\right)_{fc} \tag{13}$$

These separately updated face-velocities satisfy discrete mass-conservation to machine accuracy and use of these velocities in estimating the non-linear convective flux in Eq. (3) leads to a more accurate and robust solution procedure. The advantage of separately computing the face-center velocities was initially proposed by Zang et al. [65] and discussed in the context of the Cartesian grid methods in Ye et al. [63]. The above collocated scheme is simpler to implement than a conventional staggered mesh scheme [65] and when coupled with a central-difference spatial scheme, it leads to a numerical discretization that has good discrete kinetic energy conservation properties [12] making it suitable and robust for simulating relatively high (up to at least $O(10^4)$) Reynolds number flows without the need for artificial dissipation or upwinding.

2.2. Immersed boundary treatment

The current immersed boundary method employs a multi-dimensional ghost-cell methodology to impose the boundary conditions on the immersed boundary and is similar in spirit to the methodology proposed by Majumdar et al. [25] and employed by Ghias et al. [13,14] and Tseng and Ferziger [52]. However, unlike these previous efforts, the current solver is designed from the start for fast, efficient and accurate solution of flows with complex *three-dimensional*, *moving* boundaries. In Ghias et al. [14] in particular, similar ideas have been used for simulating *compressible* flows with 2D stationary immersed boundaries. Within the context of the categorization put forth by Mittal and Iaccarino [34], the current method employs a "discrete forcing" method wherein the effect of

the immersed boundary is incorporated into the discretized governing equations. In the rest of this section we provide an overview of the salient features of this method.

2.2.1. Geometric representation of immersed boundary

The current method is designed to simulate flows over arbitrarily complex 2D and 3D immersed stationary and moving boundaries and the approach chosen to represent the boundary surface should be flexible enough so as not to limit the type of geometries that can be handled. In addition, the surface representation method should be such as not to lead to excessive computational overhead (in terms of both memory and CPU time) for geometric operations associated with the immersed boundary (IB) surface. Finally, another factor to be considered here is the compatibility between the current solver and CAD programs which oftentimes provide the geometry of the immersed boundary and other pre- and post-processing softwares such as Alias MAYA [10] (animation), Rhino [9] (surface modeling and modification) and Tecplot (visualization) which are useful in the modeling and articulation of the geometries and analysis of these complex flows.

A number of different approaches are available for representing the surface of the immersed boundary including level-sets [40], NURBS [42] and unstructured surface meshes [7]. In the current solver we choose to represent the surface of the IB by a unstructured mesh with triangular elements. This approach is very well suited for the wide variety of engineering and biological flow configurations that we are interested and as will be described in Section 2.2.2, can be integrated into the current immersed boundary solver quite seamlessly.

Many fast and efficient algorithms exist for generation of such meshes [6,27] and accurate and efficient representation of surfaces can be obtained through the use of non-uniform, non-isotropic meshes. Fig. 2 shows the surface mesh over a porpoise which is being used to examine the fluid dynamics of drafting in cetaceans and the figure shows how the surface mesh non-uniformity allows us to provide enhanced resolution in regions such as the flippers that have finer geometric features. Because of these capabilities such meshes are ubiquitous in computational mechanics and virtually all commercial solid modeling, CAD, rapid prototyping, animation, graphics and visualization softwares are able to take input and/or provide output in this format. Such surface

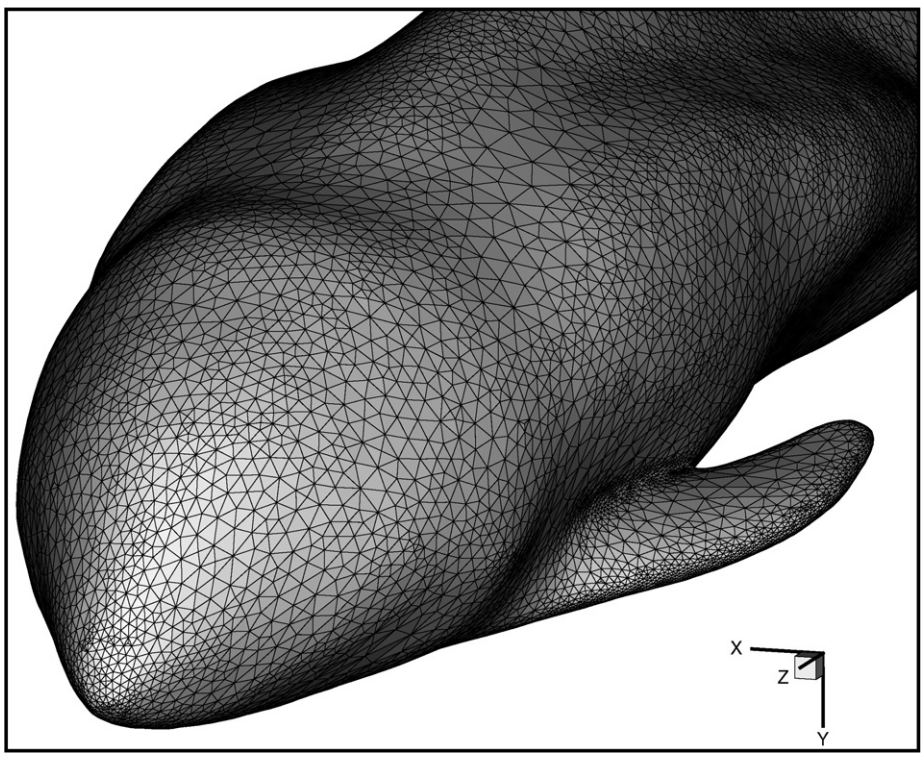

Fig. 2. Example of the type of surface mesh with triangular elements used to represent all immersed bodies in the current solver. This particular body is based on a CT scan of a harbor porpoise (*Phocoena phocoena*).

meshes are also easy to modify through operations such as smoothing, triangle subdivision/decimation and surface properties such as areas and normals are also available through simple operations. Finally, level-set representations can be obtained from the surface mesh quite easily if needed.

2.2.2. Ghost-cell formulation

The unstructured surface mesh is "immersed" into the Cartesian volume grid and Fig. 3 shows this for the particular case of the body in Fig. 2. The next step is to develop all the computational machinery that is needed to implement the ghost-cell methodology for such an immersed boundary. The method proceeds by first identifying cells whose nodes are inside the solid boundary (termed "solid cells") and cells that are outside the body (termed "fluid cells"). A straightforward method for this as depicted in Fig. 4 is to determine the surface element closest to a given node and taking a dot-product of the vector $\vec{p}$ extending from this element to the node, with the surface normal of this element $\hat{n}$. A positive(negative) value of the dot-product $(\vec{p} \cdot \hat{n})$ then indicates that the node is outside(inside) the body. For stationary boundaries, this determination needs to be done just once at the beginning of the simulation and therefore represents only a small fraction of the total turnaround time. For moving boundaries, this determination needs to be done at every time-step. However given that the immersed boundary can only travel a distance of the order of the nominal grid spacing in one time-step, the solid–fluid demarcation from the previous time-step can be used to minimize the number of grid nodes for which the above process has to be carried out. Thus, even in the moving boundary case, the solid–fluid demarcation only takes a very small fraction of the total CPU time. Consequently, very fine surface meshes can be used to provide highly accurate representations of the immersed geometry without any significant implications for the overall computational processing time.

Once the solid–fluid demarcation has been accomplished, the next step is to determine the so-called "ghost-cells". These are cells whose nodes are inside the solid but have at least one north, south, east, west, front or back neighbor in the fluid. This is easily depicted for a 2D case and the schematic in Fig. 5 shows the various types of cells for a 2D boundary cutting through a Cartesian grid. The overall approach now is to determine an appropriate equation for these ghost-cells which leads to the implicit satisfaction of the boundary condition

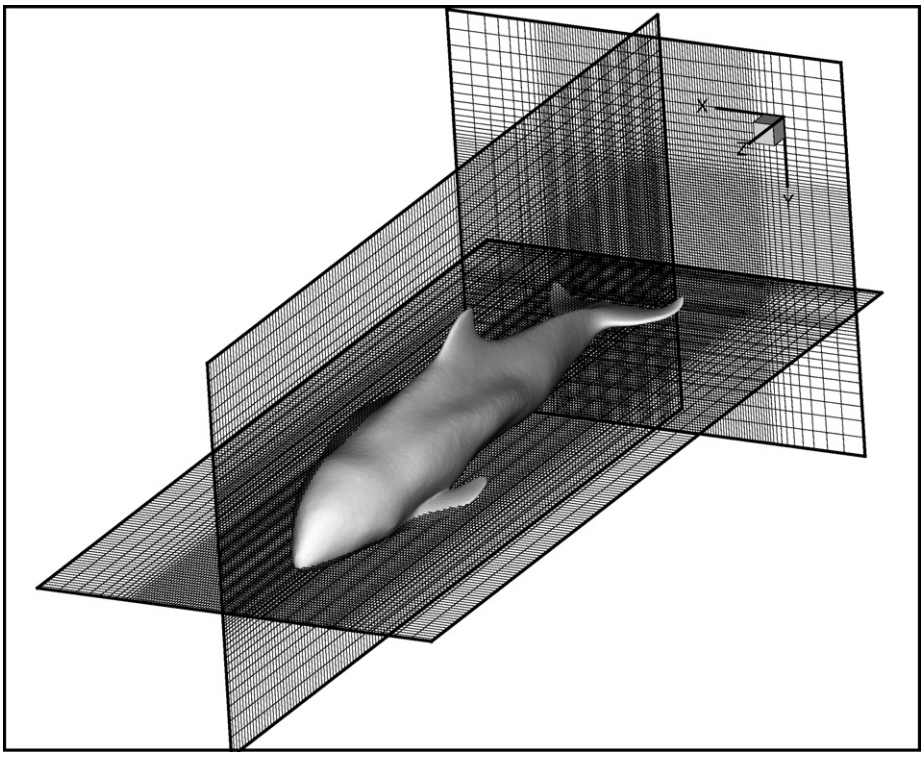

Fig. 3. Representative example showing the harbor porpoise of Fig. 2 immersed in a non-uniform Cartesian grid.

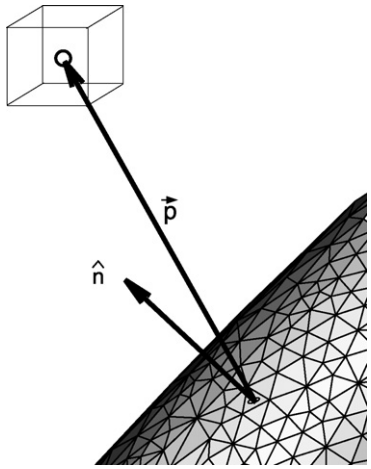

Fig. 4. Schematic showing the procedure for computing whether a node is inside or outside the body. The cube represents a fluid cell and its node for which this determination is to be made. Vector $\vec{p}$ is the position vector between this node and the surface triangle closest to the node and $\hat{n}$ is the outward pointing surface normal of this triangular element.

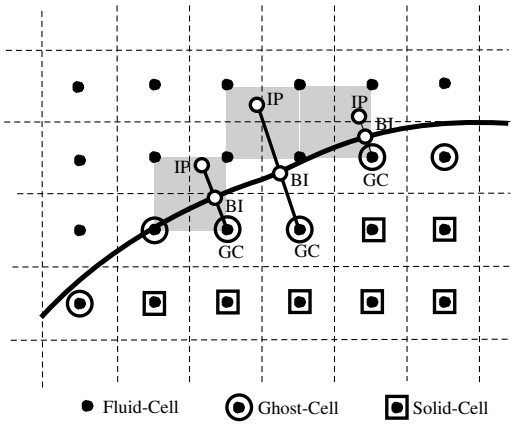

Fig. 5. 2D schematic describing ghost-cell methodology used in the current solver. Schematic depicts an immersed boundary cutting through a Cartesian grid and identifies three particular ghost-cells (GC) that form the basis for discussion in this section. BI and IP denote boundary intercept and image-point respectively.

on the immersed boundary in the vicinity of each ghost-cell. In order to accomplish this, we extend a line segment from the node of these cells into the fluid to an "image-point" (denoted by *IP*) such that it intersects normal to the immersed boundary and the boundary intercept (denoted by BI) is midway between the ghost-node and the image-point.

The identification of the boundary intercept (BI) point, although conceptually simple, presents significant complications during implementation. The type of immersed bodies that are of interest to us can have highly complex shapes and we would like the solver to be robust even in situations where the resolution of the surface mesh and/or the Cartesian volume grid is not high enough to adequately resolve the geometrical features of the surface. The BI has a crucial link to the robustness of the solution algorithm. In principle, the BI is the point on the immersed surface which has the minimum distance to the immersed boundary. In most cases, this is uniquely determined by the normal-intercept from the ghost-cell to the immersed boundary. However, as shown in Fig. 6, even for a simple 2D case, one can encounter degenerate situations where determination of a unique BI which represents the closest point on the surface is not straightforward. The situation is significantly more complicated for 3D boundaries. Correct identification of BI is crucial since an incorrectly identified

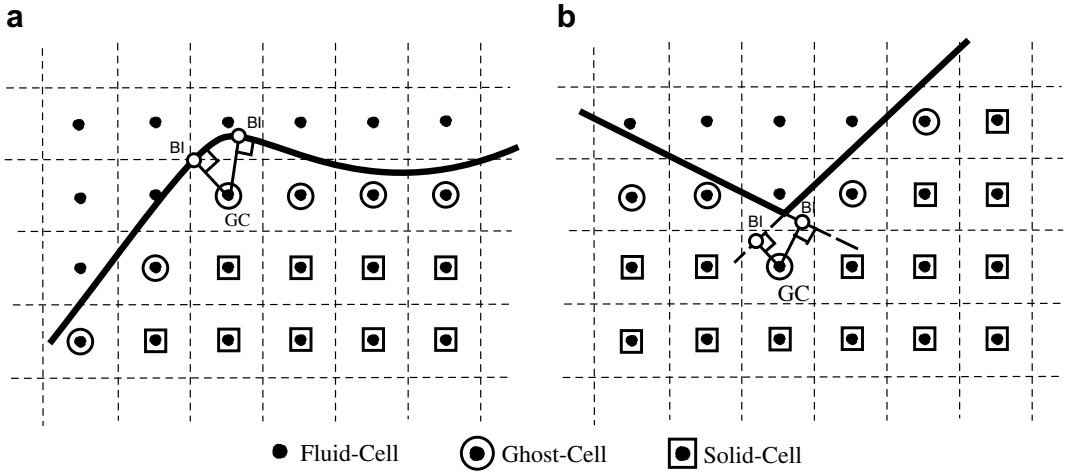

Fig. 6. 2D schematic showing two degenerate situations that can be encountered in the identification of the body-intercept point for a ghost-cell. (a) Case where there are two possible body-intercept points and (b) case where there is no body-intercept point detected on the body.

BI can lead to an excessively large interpolation stencil for the ghost-cell and can severely deteriorate the iterative convergence of the governing equations.

To avoid these problems we have adopted an approach whereby we first determine surface element vertex which is closest to the ghost-cell. The vertex closest to a node can be determined uniquely and therefore no complex logic is needed for this step. Next, we identify the set of surface elements that share this vertex and search for a normal-intercept among these elements. In cases where multiple normal-intercepts are found, the body-intercept point is chosen to be the normal-intercept point that has the shortest intercept. For cases where no normal-intercepts are found on the surface, we first repeat the search over a larger region of the surface surrounding the closest vertex. If the search is still unsuccessful, we revert back to first set of surrounding elements and search for the point on this set of elements that is closest to the ghost-cell, keeping in mind that this closest point could even be on the edge or vertex of an element. This procedure although somewhat complex, increases significantly the robustness of the BI identification scheme and allows us to perform simulations with very complex geometries. Note also that this complexity can be viewed as one of the costs of retaining a sharp-interface method since these issues would typically not arise in diffuse interface methods. However, in our view, the cost is well worth the ability of retaining a sharp-interface treatment especially given the vortex dominated flows that are of interest to us.

Once the BI and the corresponding IP have been identified, a trilinear interpolant of the following form is used to express the value of a generic variable (say ϕ) in the region between the eight nodes surrounding the image-point:

$$\phi(x_1, x_2, x_3) = C_1 x_1 x_2 x_3 + C_2 x_1 x_2 + C_3 x_2 x_3 + C_4 x_1 x_3 + C_5 x_1 + C_6 x_2 + C_7 x_3 + C_8 \tag{14}$$

The eight unknown coefficients can be determined in terms of the variable values of the eight surrounding nodes

$$\{C\} = [V]^{-1}\{\phi\} \tag{15}$$

where

$$\{C\}^T = \{C_1, C_2, \ldots, C_8\} \tag{16}$$

is the vector containing the eight unknown coefficients and

$$\{\phi\}^T = \{\phi_1, \phi_2, \ldots, \phi_8\} \tag{17}$$

are the values of the variables at the eight surrounding points. Furthermore, $[V]$ is the Vandermonde matrix [43] corresponding to the trilinear interpolation scheme shown in Eq. (14) and has the form

$$[V] = \begin{bmatrix} x_1x_2x_3|_1 & x_1x_2|_1 & x_1x_3|_1 & x_2x_3|_1 & x_1|_1 & x_2|_1 & x_3|_1 & 1 \\ x_1x_2x_3|_2 & x_1x_2|_2 & x_1x_3|_2 & x_2x_3|_2 & x_1|_2 & x_2|_2 & x_3|_2 & 1 \\ \vdots & \vdots & \vdots & \vdots & & & & \\ x_1x_2x_3|_8 & x_1x_2|_8 & x_1x_3|_8 & x_2x_3|_8 & x_1|_8 & x_2|_8 & x_3|_8 & 1 \end{bmatrix} \tag{18}$$

where the subscripts in the above equation are identifiers of the eight surrounding nodes. Once the coefficients are determined from Eq. (15), use of Eq. (14) at the image-point gives a final expression for the variable at the image-point of the form

$$\phi_{\mathrm{IP}} = \sum_{i=1}^{8} \beta_i \phi_i + \mathrm{T.E.} \tag{19}$$

In the above equation, β's depend on C's as well as the coordinates of the image-point. Since all of these depend only on the geometry of the immersed boundary and the grid, β's can be determined as soon as the immersed boundary and grid are specified. The expression for the leading order truncation error for the above interpolant has been derived in the appendix where it is shown that T.E. $= \mathrm{O}(\Delta^2)$ where the grid spacing is $(\mathrm{O}(\Delta))$.

In the above procedure, a situation may be encountered where one of the eight nodes surrounding the image-point is the ghost-node itself. In this case the row in Eq. (15) corresponding to the ghost-node is replaced by the boundary condition at the BI point. This ensures that the interpolation procedure for the ghost-node is well-posed without affecting the accuracy of the interpolation. Note that this procedure is completely consistent with the interpolation scheme and does not degrade its accuracy since the BI point lies within the cuboidal region for which the trilinear interpolant is constructed.

It may also be the case that the interpolation stencil for a given ghost-cell contains other ghost-cells (see for instance the left ghost-cell in Fig. 5). This situation does not pose any consistency issues although it does imply that some of the ghost-cell values are coupled to each other and a fully coupled solution procedure is required in order to solve for the flow variables at the ghost-cells. However in such situations we also necessarily have the condition that the ghost-node under consideration is the node that is closest to the body-intercept point (as can be seen in Fig. 5). Thus, the interpolation weight for this node has the largest relative magnitude and the coupled system to be solved for the ghost-cell is diagonally dominant. Consequently, we use a point Gauss–Seidel method for obtaining this solution and it is found to converge very rapidly (within about 10 iterations in most cases).

Following this, the value of variable at the ghost-cell (denoted by GC) is computed by using a linear approximation along the normal probe which incorporates the prescribed boundary condition at the boundary intercept. Thus, for Dirichlet boundary conditions that are employed for the velocity variables, the formula is

$$\phi_{\mathrm{BI}} = \frac{1}{2}(\phi_{\mathrm{IP}} + \phi_{\mathrm{GC}}) + \mathrm{O}(\Delta l^2) = \frac{1}{2}\left(\sum_{i=1}^{8} \beta_i \phi_i + \phi_{\mathrm{GC}}\right) + \mathrm{O}(\Delta^2) + \mathrm{O}(\Delta l^2) \tag{20}$$

where Δl is the length of the normal line segment extending from GC to IP. During the solution process, the above equation for the ghost-cell is written in the following implicit form

$$\phi_{\mathrm{GC}} + \sum_{i=1}^{8} \beta_i \phi_i = 2\phi_{\mathrm{BI}} \tag{21}$$

For the pressure Poisson equation, we need to impose Neumann boundary conditions on the immersed boundary and the following second-order central-difference, expression is written along the normal probe:

$$\left(\frac{\delta\phi}{\delta n}\right)_{\mathrm{BI}} = \frac{\phi_{\mathrm{IP}} - \phi_{\mathrm{GC}}}{\Delta l} + \mathrm{O}(\Delta l^2) = \frac{1}{\Delta l}\left(\sum_{i=1}^{8} \beta_i \phi_i - \phi_{\mathrm{GC}}\right) + \mathrm{O}(\Delta^2/\Delta l) + \mathrm{O}(\Delta l^2) \tag{22}$$

respectively and the following implicit expression is obtained for the ghost-cell

$$\phi_{GC} - \sum_{i=1}^{8} \beta_i \phi_i = -\Delta l \left(\frac{\delta \phi}{\delta n} \right)_{BI} \tag{23}$$

for this boundary condition.

Eqs. (21) and (23) are then solved in a fully coupled manner with the discretized governing Eqs. (2) and (10) for the neighboring fluid cells along with the trivial equation $\phi = 0$ for the internal solid cells. Given that $\Delta l = O(\Delta)$, the Dirichlet boundary conditions for the velocity Eq. (20) are prescribed to second-order accuracy and this along with the second-order accurate discretization of the fluid cells leads to local and global second-order for the velocity variables. The Neumann pressure boundary condition Eq. (22) is imposed to a nominally first-order accuracy which leads to a pressure-gradient field which is locally first-order but globally second-order. The pressure however, being an integral of the pressure gradient, is locally and globally second-order accurate. Furthermore, even though the pressure gradient is locally first-order near the boundary, it is multiplied by Δt during the velocity correction procedure Eq. (13) and since $\Delta t = O(\Delta)$ due to the CFL constraint, the velocity is expected to remain locally and globally second-order.

As described in Ghias et al. [14], the current ghost-cell methodology has some general similarities with the ghost-fluid method (GFM) of Gibou et al. [15] which also employs ghost-nodes to impose the boundary conditions on the immersed boundary. There are, however, also some key differences between the two methods that have an impact on the accuracy, robustness and efficiency of the two methods. First, GFM performs 1D interpolation along the Cartesian directions whereas the current method constructs the interpolation along the boundary-normal direction and this has implications for the implementation of Neumann boundary condition on the immersed boundary. For instance, consider middle ghost-cell shown in Fig. 5 for which the 1D, interpolation method would employ the nodal value to the north (ϕ_N) as well as the boundary condition on the immersed boundary at the location where the immersed boundary intersects the vertical line segment between the ghost-cell and north cell (this can be denoted as the body-intercept BI for this interpolation). For a Neumann boundary condition where $(\partial \Phi / \partial n)_{BI}$ is given, this interpolation would give $\phi_N - \phi_{GC} = \frac{\Delta x_2}{n_2} [(\partial \Phi / \partial n)_{BI} - n_1 (\partial \Phi / \partial x_1)_{BI} - n_3 (\partial \Phi / \partial x_2)_{BI}]$ where $\hat{n} = (n_1, n_2, n_3)$ is the unit normal at the BI location. The above expression however requires the derivatives of ϕ at BI along the x_1 and x_2 directions. These are not readily available and one would have to resort to additional interpolations and/or approximation to obtain these derivatives. In contrast, since the current method implements the interpolation along a direction normal to the boundary, the Neumann boundary condition is easily implemented using Eq. (23).

The second significant difference is the use of the normal probe and image-point in the current interpolation scheme which ensures that the interpolation is well behaved in the limit of the boundary-intercept point approaching the ghost-node. Again, using the same ghost-cell as in the previous paragraph, the 1D, linear interpolation scheme applied in the GFM for a Dirichlet boundary condition would give $\phi_{GC} = (\phi_{BI} + (\theta - 1)\phi_N)/\theta$ where θ is the weight factor of the linear interpolation scheme. Gibou et al. [15] point out that the above interpolation becomes ill-conditioned when the north-node is close to the BI points and $\theta \to 0$. In contrast, in the current scheme, we ensure that the body-intercept is always exactly midway between the ghost and image-point and this guarantees that the approximations in (21) and (23) remain well behaved even in the limit of vanishing probe length. In fact, in this limiting case, Eq. (21) automatically results in ϕ_{IP} and ϕ_{GC} both approaching ϕ_{BI} in a smooth manner for a Dirichlet boundary condition whereas for a Neumann boundary condition, ϕ_{GC} smoothly approaches ϕ_{IP}.

Finally, it should be noted that the choice of a collocated mesh scheme leads to considerable simplification since for a staggered mesh, four separate sets (three for the velocity component and one for pressure) of ghost-cells would have to be employed thereby quadrupling the effort and memory required for implementing the ghost-cell methodology.

2.2.3. Boundary motion

Boundary motion can be included into the above formulation with relative ease. In advancing the field equations from time level n to $n + 1$ in the case of a moving boundary, the first step is to move from its current

location to the new location. This is accomplished by moving the nodes of the surface triangles with a known velocity. Thus we employ the following equation to update the coordinates (X_i) of the surface element vertices

$$\frac{X_i^{n+1} - X_i^n}{\Delta t} = V_i^{n+\frac{1}{2}} \tag{24}$$

where V_i is the vertex velocity. The vertex velocity can either be prescribed or it can be computed from a dynamical equation if the body motion is coupled to the fluid. The next step is to determine the ghost-cells for this new immersed boundary location and recompute the body-intercepts, image-points and associated weights βs. Subsequently, the flow Eqs. (3)–(13) which are written in Eulerian form are advanced in time. The general framework described above can therefore be considered as Eulerian–Lagrangian, wherein the immersed boundaries are explicitly tracked as surfaces in a Lagrangian mode, while the flow computations are performed on a fixed Eulerian grid.

For sharp interface methods, one issue encountered with moving boundaries is the so-called "fresh-cell" problem [54,55]. This refers to the situation where a cell that is in the solid at one time-step, emerges into the fluid at the next time-step due to boundary motion. Fig. 7 shows a 2D schematic where boundary motion from time-level n to $n+1$ leads to the appearance of two fresh cells. Note that since the fluid flow simulations are limited by the CFL constraint, the boundary velocity is also subject to a similar constraint. Therefore at any given time-step the layer of fresh-cells can at most be one cell deep. Now considering the solution of Eq. (3) for a fresh cell, it can be seen that terms such as N_i^n, N_i^{n-1}, $\frac{\partial p^n}{\partial x_i}$ and D_i^n are not readily available. In the context of fluid flow simulations, Ye et al. [63] devised a simple and consistent methodology for this problem for their cut-cell based Cartesian grid method and we have adopted the same method for the current finite-difference based immersed boundary method.

Referring to Fig. 7, the value of the intermediate velocity u_i^* at time-level $n+1$ for the fresh-cell is obtained by interpolation from neighboring fluid nodes. The procedure adopted in order to perform this interpolation is consistent with the approach taken for the ghost-cell interpolation. As shown in Fig. 7, a normal intercept is extended from the fresh-cell node to the boundary and this intersects at the boundary-intercept "BI" point. An image-point "*IP*" corresponding to the boundary-intercept is then obtained and the eight (four in 2D) nodes surrounding the image-point identified. One of these nodes is necessarily the fresh-cell node itself. An interpolation stencil for the fresh-cell values is now obtained by performing a trilinear interpolation in the hexahedron defined by the seven-nodes surrounding the image point (all nodes except for the fresh-cell node) and the boundary-intercept point. In the 2D case shown in Fig. 7, this region of interpolation is a quadrilateral and the interpolation stencil for one fresh-cell is shown schematically in this figure.

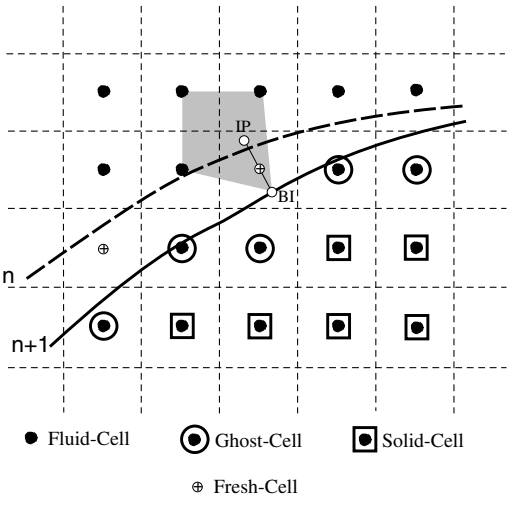

Fig. 7. Schematic showing the formation of fresh-cells due to boundary motion and the interpolation stencil (in grey) for one representative fresh-cell.

The above methodology is not only easy to implement in the context of the current ghost-cell interpolation scheme, it is also eminently consistent with it. This becomes clear if we note that any cell that is a fresh-cell at a given time-step was necessarily a ghost-cell at the previous time-step. Thus, the stencil that is employed for the fresh-cell consists mostly of nodes that were in the stencil for the node when it was a ghost-cell in this previous time-step. This allows for a smoother transition in the nodal value as the cell changes phase from solid to fluid. Furthermore in the special situation where the fluid node is exactly coincident with the immersed boundary, both the ghost-cell and fresh-cell interpolation schemes default automatically to the same boundary value.

Once the intermediate velocity is obtained, the pressure for the fresh-cell can be obtained as for the other cells by solving the pressure Poisson equation Eq. (10). The final cell-center and face velocities i.e. u_i^{n+1} and U_i^{n+1}, respectively, as well as the final pressure p^{n+1} are subsequently obtained by solving Eq. (13).

3. Results and discussion

In this section we assess the accuracy and fidelity of the solver and also demonstrate the solver's ability to handle highly complex boundaries. We first describe a grid convergence study which examines the accuracy of the solver for a prototypical flow. Following this, a number of two- (2D) and three-dimensional (3D) flows with stationary boundaries are simulated and results compared with established experimental and numerical data sets. Next simulations of flows with moving immersed bodies are conducted and results validated against other studies. Finally, we simulate flows with highly complex, non-canonical geometries in order to showcase the capabilities of the solver.

It should be pointed out that due to the explicit treatment of the convention term in Eq. (3), the time-step in the current simulations is limited by the CFL number criterion wherein

$$\left(\frac{|u_1|}{\Delta x_1}+\frac{|u_2|}{\Delta x_2}+\frac{|u_3|}{\Delta x_3}\right)\Delta t < \mathrm{CFL}_{\max}. \tag{25}$$

For the current solver, we find that $\mathrm{CFL}_{\max} \approx 0.5$ gives a stable solution and we use this criterion to choose the time-step size.

3.1. Grid convergence study

In addition to the second-order accurate spatial discretization used for the regular fluid cells, care has been taken to maintain a second-order accurate treatment in the imposition of the velocity boundary condition on the immersed boundary. Thus, we expect the solver to exhibit second-order global and local accuracy. The second-order accuracy for the cells in the vicinity of the immersed boundary is especially important for the resolution of thin boundary layers that develop on the immersed boundary for moderate to high Reynolds number flows. Here we examine the spatial accuracy of the solver for flow past a circular cylinder at $Re_d = U_\infty d/\nu = 100$ where d is the cylinder diameter, U_∞ is the free stream velocity and ν is the kinematic viscosity. For this test, we employ a uniform Cartesian grid on a $2d \times 2d$ computational domain size.

Since an exact solution for this case does not exist, we use the solution computed on a highly resolved 630×630 grid as a baseline for computing the truncation error. We choose a time-step of $0.0001 d/U_\infty$ and integrate the solution for 2000 time-steps. The resulting solution is shown in Fig. 8(a). The same flow is then computed on a hierarchy of grids (210×210, 126×126, a 90×90, and a 70×70) with the same time-step size as the 630×630 grid. The distribution of error magnitude in the u_1 velocity component for the 126×126 grid is shown in Fig. 8(b). As can be seen from this figure, the largest magnitudes of error in velocity are located around the cylinder. This implies that this error is a true measure of the error of the immersed boundary treatment and therefore, examination of this error provides an accurate view of the order of accuracy of the boundary treatment.

The L_1, L_2 and $L_{\max}$ norms of the error for a solution on a $N \times N$ grid can now be computed. It should be noted that on one end, the L_1 error-norm is a good measure of the global error whereas on the other, the $L_{\max}$ error-norm effectively captures the local error around the immersed boundary. Fig. 9 shows the variation of the L_1, L_2 and $L_{\max}$ error norms in the two velocity components for the three grids on a log–log plot. Also included in the plot is a line denoting second-order convergence. Both error norms show nearly second-order

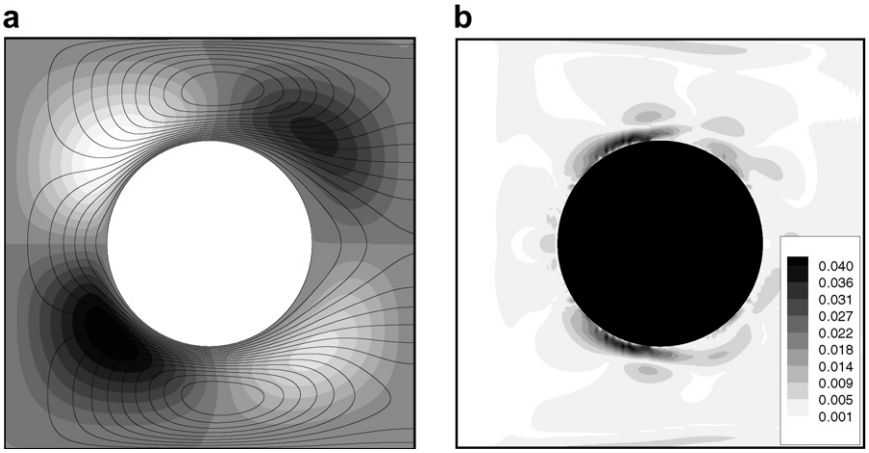

Fig. 8. (a) Contours of u_1 (line contours) and u_2 (greyscale contours) for numerical solution on the 630×630 grid. (b) Distribution of error in u_1 component of velocity on the 126×126 grid.

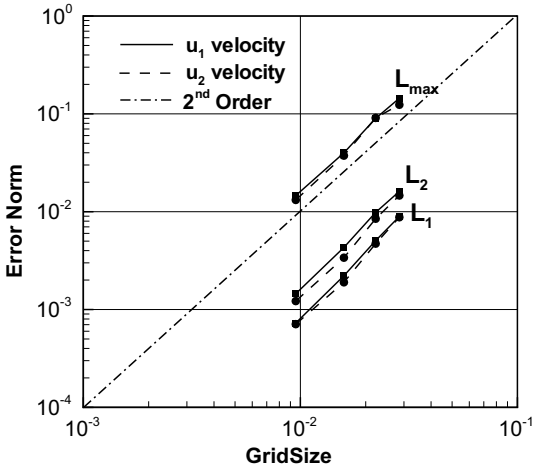

Fig. 9. L_1, L_2 and L_∞ norms of the error for the streamwise velocity u_1 and transverse velocity u_2 components versus the computational grid size.

convergence thereby confirming that the current immersed boundary solver is globally and locally second-order accurate.

3.2. Flow past a circular cylinder

The flow past a circular cylinder has become the *de-facto* standard for assessing the fidelity of Navier–Stokes solvers. Up to a Reynolds number of about 47 the flow is steady and symmetrical about the wake-centerline. At Reynolds numbers higher than this value, the flow becomes unstable to perturbations and leads to periodic Karman vortex shedding. The flow remains two-dimensional up to a Reynolds number of about 180 [62] beyond which the flow becomes intrinsically three-dimensional [29,62]. We have performed 2D simulations at Reynolds numbers of 40, 100, 300 and 1000 and compared the computed results with available numerical and experimental results.

Fig. 10 shows the grid used for the $Re_d = 1000$ simulations and as can be seen from the figure, we employ a non-uniform Cartesian grid wherein high resolution is provided to the region around the cylinder as well as the wake. For instance, the resolution for the $Re_d = 300$ and 1000 cylinder cases in the region around the cylinder

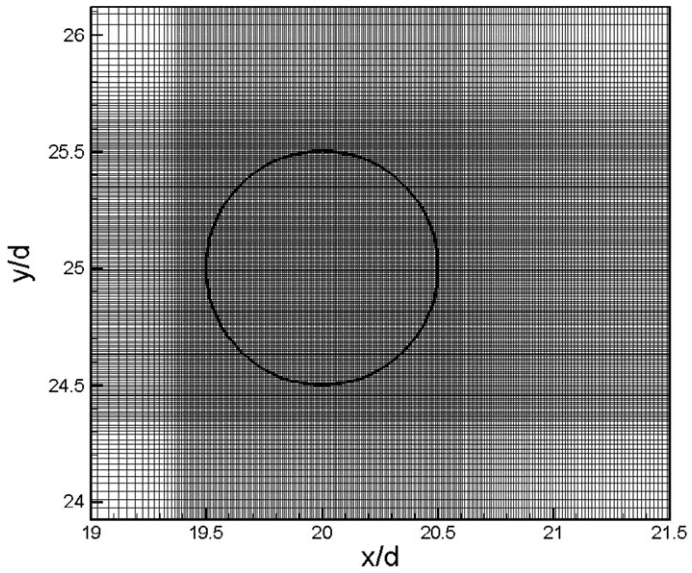

Fig. 10. Non-uniform grid employed in the vicinity of the circular cylinder for the $Re_d = 1000$ simulations.

are $\Delta x_1 = \Delta x_2 = 0.015d$ and $0.01d$, respectively. Large domains of size $40d \times 40d$ are employed to minimize domain confinement effects. Overall grid sizes for the $Re_d = 300$ and 1000 cylinder cases are 385×105 and 417×289, respectively. To provide some context to the grid employed in the current study, it should be noted that Marella et al. [26] who presented a Cartesian grid method, employed a 452×452 mesh on a $30d \times 30d$ in their $Re_d = 300$ cylinder simulations.

Fig. 11 presents spanwise vorticity contour plots for $Re_d = 300$ and 1000 at one time-instant and both plots indicate the presence of Karman vortex shedding. The vortex shedding leads to the development of time-varying drag and lift forces and in Fig. 12 we have plotted the temporal variation of the drag and lift coefficients, defined as $C_D = F_D/\left(\frac{1}{2}\rho U_\infty^2 d\right)$ and $C_L = F_L/\left(\frac{1}{2}\rho U_\infty^2 d\right)$ respectively where F_D and F_L are the drag and lift forces, respectively. It can be observed that for $Re_d = 300$ the vortex shedding reaches a stationary state at

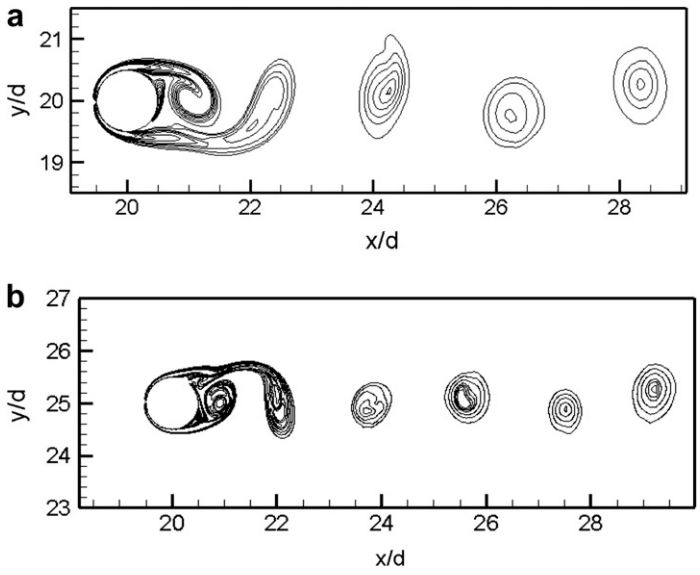

Fig. 11. Computed spanwise vorticity contour plots for (a) $Re_d = 300$ and (b) 1000 at one time-instant.

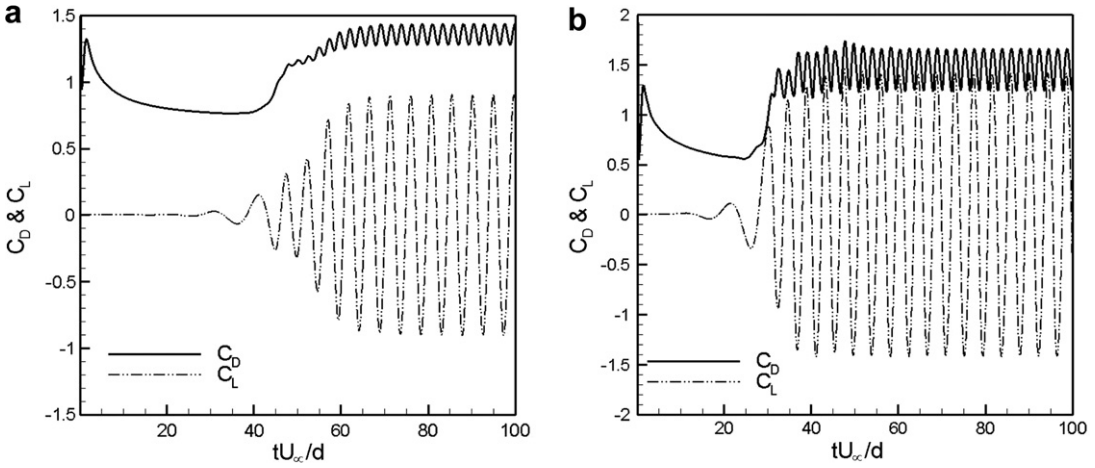

Fig. 12. Computed temporal variation of drag and lift coefficients for the (a) $Re_d = 300$ and (b) 1000 cases.

a non-dimensional time tU_∞/d of about 70 where as the $Re_d = 1000$ case attains this state at a non-dimensional time of about 60.

In order to validate the simulations we have computed a number of key quantities including mean drag and base pressure coefficient C_{p_b}, where pressure coefficient at any location is defined as $C_p = (p - p_\infty)/\frac{1}{2}\rho U_\infty^2$. The vortex shedding Strouhal number $St = fd/U_\infty$ where f is the vortex shedding frequency has also been computed from the temporal variation of the lift coefficient. All of these flow quantities are computed from data accumulated after the flow has reached a stationary state.

Fig. 13(a) shows the variation of Strouhal number with Reynolds number obtained from the current simulations. Also presented are the results from a number of established experimental and numerical studies and we find good agreement between the present study and past numerical studies. The agreement with the experiment of Williamson [61] is also good up to about $Re_d = 200$ beyond which the present study as well as other numerical studies deviate from the experiment. This is due to the fact that at these Reynolds numbers, the flow is intrinsically three-dimensional [62,29] and predictions from 2D simulations tend not to match experiments well in this regime.

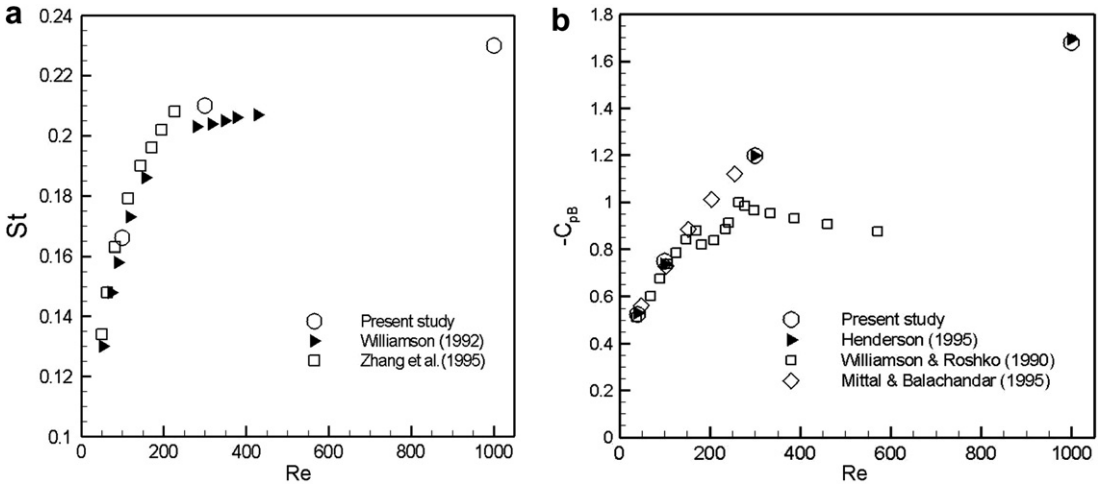

Fig. 13. Comparison of computed (a) vortex shedding Strouhal number (St) and (b) computed base suction coefficient ($-C_{p_b}$) with established computational and experimental results.

Fig. 13(b) shows a comparison of the mean base-suction pressure coefficient $-C_{p_b}$ compared to past 2D numerical simulations of Henderson [17] and Mittal and Balachandar [30] and experiments of Williamson and Roshko [60]. Note that the simulations of Henderson [17] employed a spectral element method whereas those of Mittal and Balachandar [30] employed a highly accurate spectral collocation method. It is found that the predictions from the current study match these previous numerical studies over the entire range of Reynolds numbers simulations here. As before, the match with experiments is quite good up to about $Re_d = 200$ beyond which intrinsic three-dimensional effects in the experiments lead to a significant mismatch.

Finally, in Table 1 we compare the mean drag coefficient predicted by the current solver with some other numerical studies that have conducted 2D simulations of this flow. The Reynolds number range varies from 40 to 100 and comparisons are made with highly accurate spectral element [17] and spectral collocation [31] as well as another sharp interface, immersed boundary solver [26]. In general, we find very good agreement with these other numerical simulations and this further confirms the accuracy of the current solver over a relatively wide range of Reynolds numbers.

3.3. Flow past an airfoil

In addition to circular cylinder simulations, we have also performed 2D simulations of flow past a NACA 0008 airfoil at two different angles-of-attack ($\alpha = 0^\circ$ and 4°) at chord-based Reynolds number (Re_c) of 2000 and 6000. These configurations have particular relevance for micro-aerial vehicles [36] where Reynolds numbers tend to be in the range from 10^2 to 10^4 and have been the subject of a numerical study by Kunz and Kroo [22]. In this previous study, the authors used a body-fitted 256×64 point C-grid with the outer radius placed at $15c$. The body-conformal grid allowed for the placement of about 25 grid cells across the boundary layer on the airfoil.

This flow is quite challenging for the current method since the region around the leading-edge of the airfoil has very small radius of curvature compared to the dominant length scale of the immersed body which is the airfoil chord. Thus, the resolution of the Cartesian grid which is body-non-formal, is driven by the need to resolve the geometry and corresponding flow near the airfoil leading edge. Consequently we employ a non-uniform 926×211 which gives about 12 points across the boundary layer on the airfoil. Furthermore, a domain size of $9c \times 12c$ where c is the chord length of the airfoil is employed. Both the grid and domain were chosen after a systematic grid refinement and domain dependence study [57]. The simulations require about 9 h of CPU time per chord flow time (c/U_∞) on a single processor of a 64-bit, 2.0 GHz AMD Opteron computer with 16 GB of local memory.

Fig. 14 shows a flow visualization for the $Re_c = 6000$ and $\alpha = 4^\circ$ case and the simulations indicate that the flow separates from the suction side of the airfoil. Fig. 15 presents the temporal variation of the drag and lift

Table 1
Comparison of computed mean drag coefficient with results from previous 2D cylinder simulations

	Re_d			
	40	100	300	1000
Present study	1.53	1.35	1.36	1.45
Henderson [17]	1.54	1.35	1.37	1.51
Marella et al. [26]	1.52	1.36	1.28	–
Mittal and Balachandar [30]	–	–	1.37	–

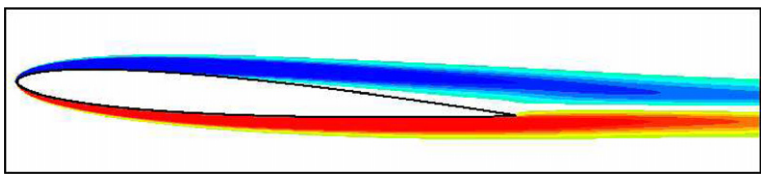

Fig. 14. Contour plot of spanwise vorticity at one time-instant for the NACA 0008 airfoil at $Re_c = 6000$ and $\alpha = 4^o$.

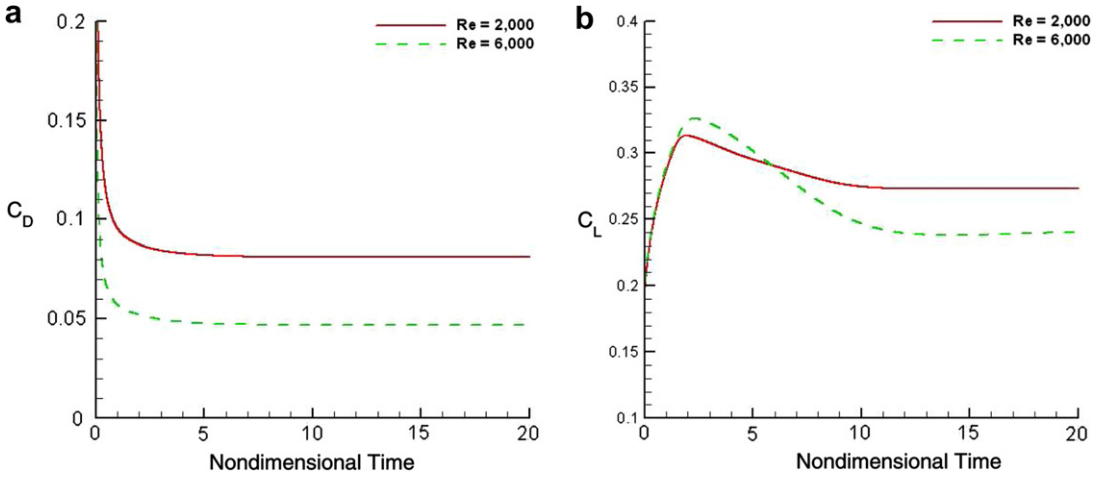

Fig. 15. Temporal variation of force coefficients for NACA 0008 airfoil at $\alpha = 4^\circ$ for $Re_c = 2000$ and 6000 (a) drag coefficient (b) lift coefficient.

coefficients. The simulations are run for a relatively large time duration of $tU_\infty/c = 20$ at which point the force on the foil reaches a nearly constant value. These lift and drag coefficients are compared with the numerical simulations of Kunz and Kroo [22] in Table 2 and we find that the current methodology provides a reasonably good prediction of these key quantities.

3.4. Flow past a sphere

Flow past a stationary sphere is a canonical flow that allows us to test the fidelity of the solver for three-dimensional flows. A number of experimental [8,48,39], and numerical studies [32,18,33] have examined this flow at low to moderate Reynolds numbers which are accessible via direct numerical simulation. Flow past a sphere is axisymmetric and steady below a Reynolds number (based on the diameter) of 210 [37]. Between Reynolds numbers of 210 and about 280, the flow is non-axisymmetric but steady and beyond that the flow is non-axisymmetric and unsteady.

In the current study we have performed simulations of flow past a stationary sphere with Reynolds numbers ranging from 100 to 350 and made qualitative as well as quantitative comparisons with established data. For the highest Reynolds number of $Re_d = 350$ we have employed a $192 \times 120 \times 120$ non-uniform grid with grid clustering provided around the sphere and in the near wake. Furthermore, the domain size used in these sphere simulations is $16d \times 15d \times 15d$. For comparison, Marella et al. [26] employed a $130 \times 110 \times 110$ mesh on a $15d \times 15d \times 15d$ domain with their Cartesian grid, $Re_d = 300$ sphere simulations. Johnson and Patel [18], who used a body-conformal spherical grid and a second-order upwind finite-difference method, employed a $101 \times 42 \times 101$ grid on a domain of size $15d$. On the other hand, Mittal [32] used a spectral collocation method with a spherical, body-fitted mesh and simulated the flow at $Re_d = 350$ on a $81 \times 80 \times 32$ grid. The current

Table 2
Comparison of computed steady-state lift and drag coefficient values for NACA 0008 airfoil at two angles-of-attack with the results obtained from simulations of Kunz and Kroo [22]

Study	Re_c							
	2000				6000			
	$\alpha = 0^\circ$		$\alpha = 4^\circ$		$\alpha = 0^\circ$		$\alpha = 4^\circ$	
	C_D	C_L	C_D	C_L	C_D	C_L	C_D	C_L
Present	0.078	–	0.081	0.273	0.044	–	0.047	0.240
Kunz and Kroo [22]	0.076	–	0.080	0.272	0.043	–	0.047	0.234

simulations were performed on a single processor of a 64-bit, 1.8 GHz AMD Opteron computer and the $Re_d = 350$ simulation required about 7 h of CPU time per time-unit (d/U_∞).

For $Re_d = 100$ and 150 the computed flow is steady and axisymmetric and Fig. 16(a) shows the streamline pattern on one plane of symmetry for $Re_d = 100$. For these axisymmetric flows, we can identify the center (coordinates (x_c, y_c)) of the flow recirculation pattern in the wake of the sphere. In Fig. 16(a) this location is denoted by a small white circle. We can also determine the length of the recirculation zone from the back of the sphere denoted by L_b. The values of these parameters for $Re_d = 40$ and 150 are compared with previous studies in Table 3 and found to be very much inline with these previous studies.

For $Re_d = 300$ and 350, the flow is non-axisymmetric and unsteady and Fig. 16(b) shows a visualization of the entrophy fields for $Re_d = 350$. As has been seen in past studies, the wake at these Reynolds numbers is found to be dominated by vortex loops which exhibit a planar symmetric topology [32]. These simulations are run for a long enough time period so as to reach a well established stationary state. Average quantities such mean force coefficients and vortex shedding Strouhal number are computed based on stationary state data. Fig. 17(a) shows the temporal variation of the drag and side force coefficient for $Re_d = 350$ case and these flow quantities clearly show that the flow has reached a stationary state. In Fig. 17(b) we compare the computed mean drag coefficient with a number of previous experimental and numerical studies and find that the values match quite well with these past studies as well as the correlation of Clift et al. [8]. The temporal variation of pressure in the wake has been used to estimate a vortex shedding Strouhal number and as can be seen in Table 3, this quantity also matches well with past studies.

3.5. Simulation of flow past suddenly accelerated bodies

All of the cases simulated so far have been with stationary immersed boundaries. In the current section we describe simulations of flow past moving bodies with the objective of demonstrating the fidelity and accuracy

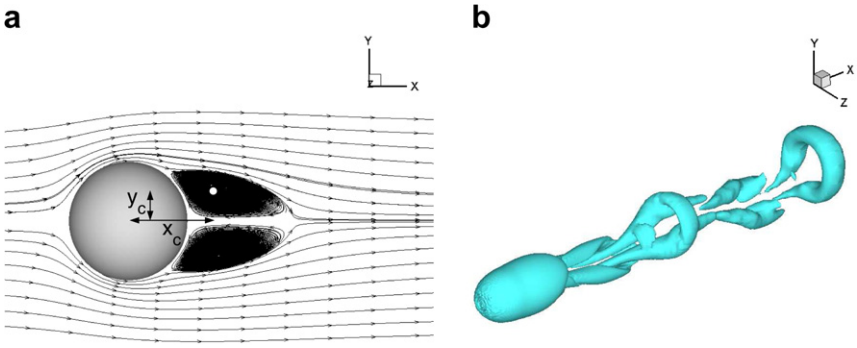

Fig. 16. (a) Computed streamline pattern on one plane of symmetry for $Re_d = 100$ sphere case. (b) Isosurface of entrophy at one time-instance for $Re_d = 350$ sphere case.

Table 3
Comparison of key computed results for flow past a sphere with other established experimental and computational studies

Study	Re_d							
	100			150			300	350
	x_c/d	y_c/d	L_b/d	x_c/d	y_c/d	L_b/d	St	St
Mittal [32]	–	–	0.87	–	–	–	–	0.14
Bagchi et al. [2]	–	–	0.87	–	–	–	–	0.135
Johnson and Patel [18]	0.75	0.29	0.88	0.32	0.29	1.2	0.137	–
Taneda [50]	0.745	0.28	0.8	0.32	0.29	1.2	–	–
Marella et al. [26]	–	–	0.88	–	–	1.19	0.133	–
Present	0.742	0.278	0.84	0.31	0.3	1.17	0.135	0.142

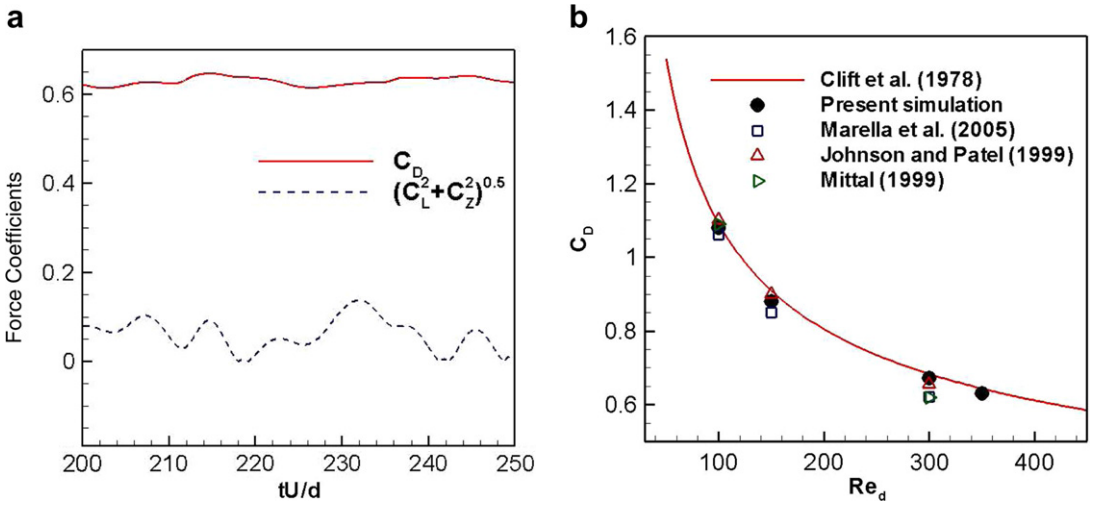

Fig. 17. (a) Temporal variation of drag and side force coefficients on a sphere in a uniform flow for $Re_d = 350$. (b) Comparison of computed mean drag coefficient with experimental and numerical data.

of the current solver for such flows. The focus is on bodies which instantaneously accelerate from zero to a finite velocity U_0 in a stagnant fluid. Accurate prediction of the temporal variation of the drag force and wake evolution requires that the thin vorticity layer that develops on the accelerating body be adequately resolved in time and space. Therefore, such flows offer a severe test of both the spatial and temporal accuracy of the method for moving boundaries.

3.5.1. Suddenly accelerated normal plate

This first case of a moving immersed boundary is of an infinitesimally thin finite flat plate of height h accelerating normal to its surface. The Reynolds numbers based $Re_h = U_0 h/\nu$ is chosen to be 126 and 1000 in order to match the simulations of Koumoutsakos and Shiels [21]. These authors employed a vortex particle method for simulating this flow which is particularly well suited for such flows since at least for early times, the vorticity is confined to a small subregion of the domain thereby easing the computational requirements for the simulations. The current simulation for the $Re_h = 1000$ case employs 481×161 grid with a minimum spacing around the plate of $\Delta x = \Delta y = 0.01h$. This grid was chosen based on a grid refinement study. For context, Koumoutsakos and Sheils [21] employed approximately half a million vortex particles in their simulations.

Fig. 18 shows the evolution of the separation bubble behind the plate at four different time-steps and these plots compare well with the corresponding Figs. 5 and 8 in the paper of Koumotsakos and Sheils [21]. Fig. 19 shows the temporal variation of the computed bubble length (which is the length of the region of reverse flow on the centerline behind the plate normalized by plate height) obtained from the current simulations. Also included in the plot are results from the experiments of Taneda and Honji [51] and simulations of Koumoutsakos and Shiels [21] and we find an excellent match between the three data sets. Note also that this case demonstrates the ability of the solver to handle infinitesimally thin (membraneous) bodies. Membraneous entities such as insect wings and fish fins abound in biology. Infinitesimally thin interfaces are also encountered in flows involving bubbles and drops and therefore, the ability to handle such geometries significantly enhances the operational envelope of the solver. The key issue in dealing with such boundaries is to allow for ghost-cells on *both sides* of the immersed boundary. Furthermore for such cases, a given ghost-node is also concurrently a fluid-node. In the current solver we have handled this through the use of auxiliary arrays that store ghost-cell nodal values separately from fluid values at a given ghost-cell. Pointers are used to access this auxiliary storage thereby simplifying the solution algorithm for such bodies.

3.5.2. Suddenly accelerated circular cylinder

Next we present results of simulated flow past suddenly accelerated circular cylinder. Flow is simulated at two different Reynolds numbers ($Re_d = U_0 d/\nu$) of 550 and 1000, both of which have been simulated using a

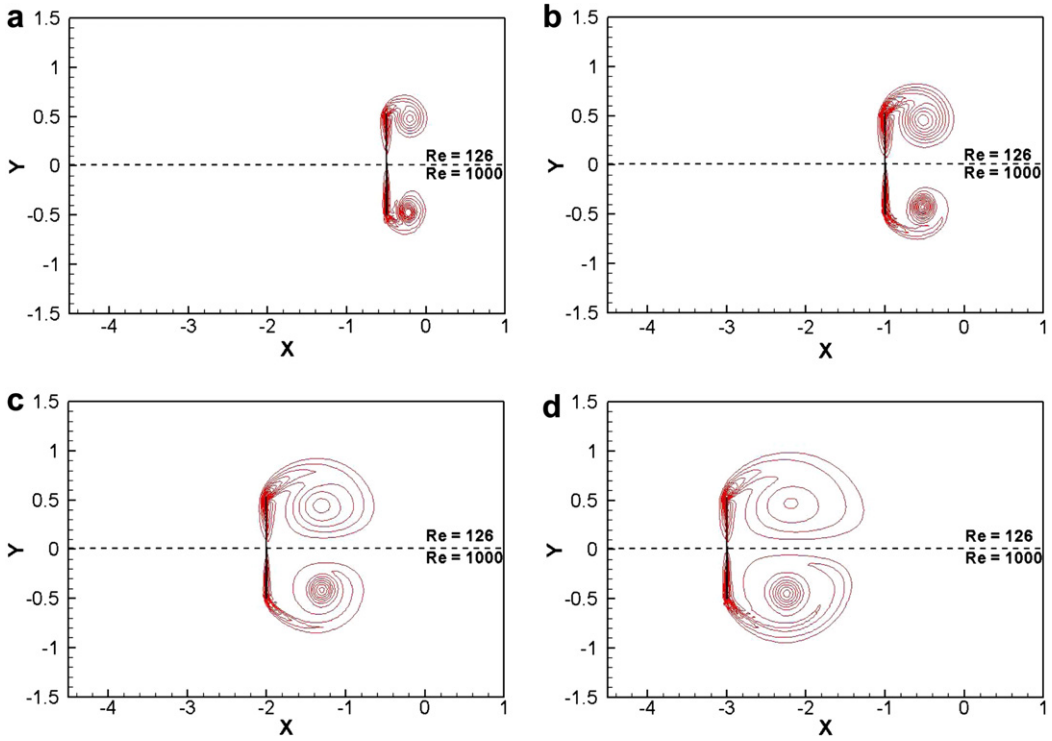

Fig. 18. Computed spanwise vorticity contours for a suddenly started normal flat-plate at four stages in the start-up process. Upper and lower halves of each figure correspond to $Re_h = 126$ and 1000 respectively. (a) $tU_0/h = 0.5$ (b) 1 (c) 2 and (d) 3.

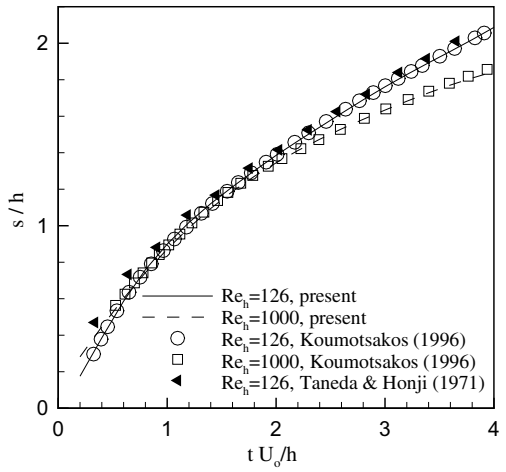

Fig. 19. Time evolution of computed bubble length behind flat plate at $Re_h = 126$ and 1000 compared to established experimental and computational results.

vortex particle method by Koumoutsakos and Leonard [20]. The current $Re_d = 1000$ simulation employs a 541×161 grid minimum grid spacing of $\Delta x = \Delta y = 0.01d$ in order to resolve the extremely thin boundary layers that develop on the cylinder surface. For comparison, the simulations of Koumoutsakos and Leonard [20] employed 300,000 vortex particles.

Fig. 20 shows four stages in the evolution of the vortex behind the cylinder at the two Reynolds numbers. These figures compare well with corresponding figures in the paper of Koumoutsakos and Leonard [20].

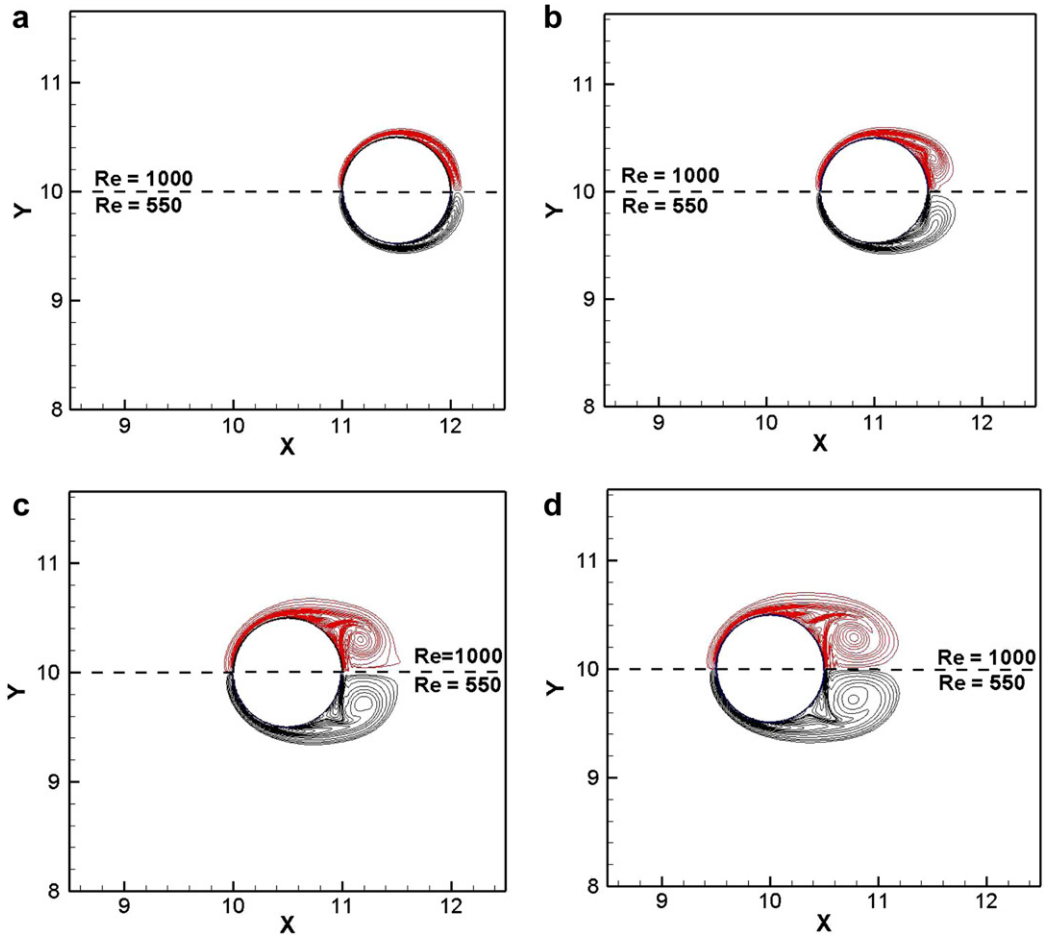

Fig. 20. Computed spanwise vorticity contours for a suddenly started cylinder at four stages in the start-up process. Upper and lower halves of each figure correspond to $Re_h = 1000$ and 550 respectively. (a) $tU_0/d = 0.5$ (b) 1 (c) 1.5 and (d) 2.

In Fig. 21 we have plotted the temporal variation of the computed drag coefficient along with available results from [20] for both cases. It is noted that the current immersed boundary calculations match the results of these previous calculations very well, thereby providing further confidence in the ability of the solver to simulate accurately the production and subsequent convection of vorticity from moving and rapidly accelerating bodies as well as the transient fluid-dynamic forces.

Also included in Fig. 21 is the temporal variation of force on a suddenly accelerated sphere at a Reynolds number of 550. For this simulation, we have employed a $541 \times 161 \times 161$ grid which is based on the grid used for the $Re_d = 550$ suddenly accelerated cylinder. Although there is no other computational or experimental data available for comparison, we provide this data so that it may be used as a benchmark for moving 3D bodies in the future.

3.6. Simulation of flow past complex moving bodies

With the solver validated systematically for two- and three-dimensional stationary and moving boundaries, we now turn to demonstrating the ability of the current solver to simulate flow with highly complex boundaries.

3.6.1. Fish pectoral fin hydrodynamics

The first case chosen is that of a fish pectoral fin. The particular fish which is the subject of this study is the bluegill sunfish, which has been studied extensively by Lauder and co-workers [24,35]. This fish was videotaped

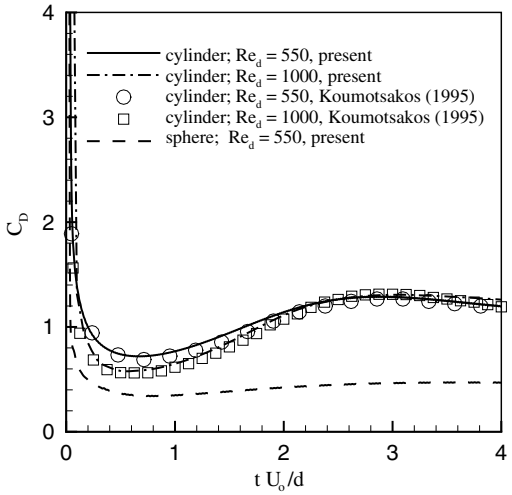

Fig. 21. Time evolution of computed drag coefficient for suddenly started cylinder at $Re_d = 550$ and 1000 compared to established experimental and computational results. Also included in the figure is the temporal variation of drag-coefficient for a suddenly started sphere at $Re_d = 550$.

swimming steadily in a current of water moving at a speed of about one body-length per second. In this particular swimming mode, the fish uses only its pectoral fins to produce propulsive force and is therefore a good case to study pectoral fin based (“labriform”) propulsion. Two high-speed, high-resolution video cameras are used simultaneously to record the motion of the fin and the videos are used to construct an accurate, three-dimensional, time-varying reconstruction of the fin kinematics. The fin is a thin membranous structure supported by slender bony rays and can therefore be modeled as a membrane in the simulations. The fin position at three stages in the stroke is shown in Fig. 22 and it can be seen that the fin undergoes significant deformation, both

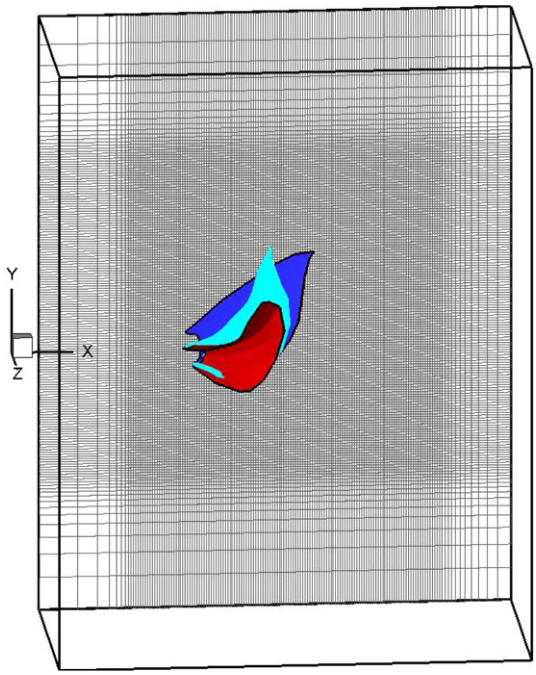

Fig. 22. Grid employed in the pectoral fin simulations and fin configuration at three stages in its motion.

active as well as flow-induced, as it moves through the fluid. All of these deformations are captured in the videos and incorporated with high fidelity into the reconstructed fin kinematics.

The stroke frequency of the fin is about 2 *Hz* and using the length of the longest ray which is 4 cm as the length scale and the freestream velocity of 0.16 m/s as the velocity scale, we estimate the Reynolds numbers ($Re = U_\infty L_S/\nu$ where L_S is the spanwise size of the fin) and Strouhal numbers ($St = fL_S/U_\infty$ where f is the stroke frequency) for this flow to be approximately 6300 and 0.54, respectively. In addition to the fin kinematics, these non-dimensional parameters are also matched in the simulation. Earlier simulations of this case at a lower Reynolds number of 1140 were carried out on a $153 \times 161 \times 113$ grid points [35]. However, the current higher Reynolds number simulation has been carried out on a finer $201 \times 193 \times 129$ wherein the region around the fin has a highest resolution with $\Delta \approx 0.012 L_s$. Grid refinement studies indicate that further refinement has no significant effect on either the vortex structures or the hydrodynamic forces [5]. The simulations were carried out on a single processor of a 64-bit, 2 GHz AMD Opeteron computer with 16 GB of local memory and required 176 h of CPU time per fin-beat cycle.

It is useful to compare and contrast the current simulation with that of Ramamurti et al. [44] which are quite similar in that they also simulated flow associated with a fish pectoral fin, albeit with a unstructured body-conformal method. The simulations of Ramamurti et al. [44] employed a sophisticated mesh generation and adaptive remeshing strategy wherein the grid in the vicinity of the fin was remeshed at every time-step and the entire mesh was remeshed over longer time-intervals. Every time this remeshing is done, the geometric parameters(shape-function derivatives, Jacobians, etc.) for all the modified elements have to be recomputed. In contrast, the current method does not require any remeshing algorithms and only the geometrical quantities (primarily the coefficients βs in Eq. (19)) in one layer of cells surrounding the immersed boundary have to be recomputed at each time-step. The simulations of Ramamurti et al. [44] employed 840×10^3 tetrahedral elements. The focus of the study was on the surface forces and flow in the immediate vicinity of the fin, and therefore, the flow in the wake was not resolved. Information regarding grid resolution around the fin as well as CPU time used is not provided. It should also be pointed out that the pectoral fin studied by Ramamurti et al. [44] was fairly stiff and did not display anywhere near the same degree deformation that is observed for the bluegill fin. Such a highly deforming geometry would be even more challenging for a body-conformal grid method since maintaining grid quality during remeshing and grid adaptation would be quite difficult.

Fig. 23 shows a sequence of flow visualizations at three stages in the fin stroke. The fish body is shown for reference purposes only and is not actually included in the simulations. Instead, the fin is put next to a flat-plate which is oriented in the $x - y$ plane. The plot shows streamlines as well as isosurface plots of the imaginary part of the complex eigenvalue of the velocity deformation tensor [49] denoted by Λ_i. The plots show the formation of a complex system of distinct vortices including a strong abduction tip-vortex from the fin. As this vortex system convects downstream it coalesces into a nearly spherical agglomeration of vortex structures. An ongoing study is examining the thrust production, energetics and associated flow mechanism and a parameter survey of this flow is also being undertaken. Detailed results from this study are to be presented in a future publication.

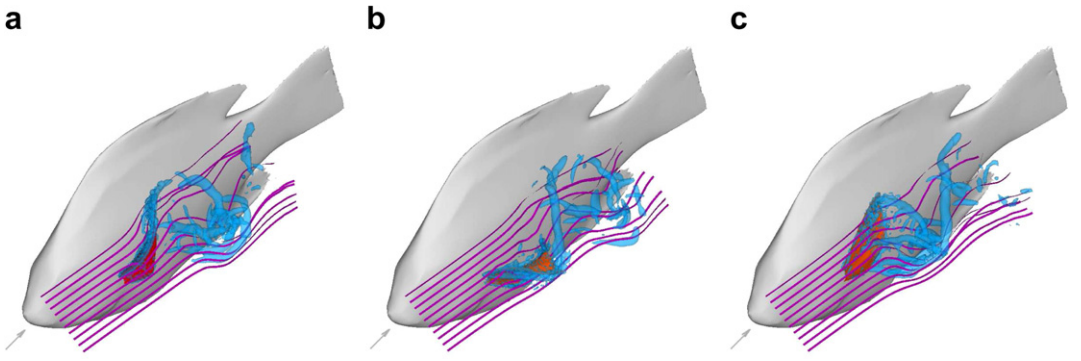

Fig. 23. Isosurfaces of Λ_i and corresponding streamlines at three stages in the pectoral fin stroke of the bluegill sunfish. (a) $t \times f = 1/3$ (b) $t \times f = 2/3$ (c) $t \times f = 1$. Body of the sunfish is shown for reference only and not included in the simulations.

3.6.2. Dragonfly flight aerodynamics

In this section, we present results from a simulation devised to examine aerodynamics of dragonfly flight with a significant level of complexity and realism including the role of wing–wing interaction and wing–body interaction. The results presented here are primarily intended to show the complexity of the flow for such a configuration and the ability of the solver to handle a case which includes a combination of multiple moving/stationary membranes and solid bodies. The dragonfly body and wing anatomy is based on images of a variegated meadowhawk (*Sympetrum corruptum*) which is a medium-sized dragonfly. Fig. 24(a) shows the final body configuration as well as the surface mesh for dragonfly model which consists of 8832 triangular elements. A Cartesian grid of size $161 \times 177 \times 113$ is used which provides high resolution around the dragonfly and the wake as shown in Fig. 24(b).

Actual kinematics of dragonflies in free-flight including, rolling amplitude, pitch angles, inclination of wing stroke-planes, wing beat frequency and phase-relation between the fore- and hind-wings, is available from a number of sources [38,58]. In the current model, we choose a relatively simple representation of the wing kinematics wherein each pair of wings undergoes a sinusoidal pitching-rolling motion where the roll axis is situated at the inner (closer to body) tip of the wing and the pitching is along a spanwise axis located at 10% wing chord. Furthermore, pitching leads the rolling motion by 90° in phase and the forewings also lead the hind-wings by 180° in phase. In the specific case presented here, the Reynolds number ($Re = U_\infty c/\nu$) based on the maximum chord length of the fore-wing is 320 and Strouhal number (defined as $St = Af/U_\infty$ where A is the peak-to-peak amplitude of the tip of the forewing, and f is the wing flapping frequency) is 0.5. Alias MAYA animation software [10] is used to incorporate the prescribed kinematics into the model and the animation is input into the IBM solver as a sequence of surface velocity files.

Fig. 25 shows the flow structures of the dragonfly case at three different stages in the flapping cycle. In Fig. 25(a) and (b) the two wing pairs are at the extreme positions of their flapping motion and are at the stage of reversing their respective motions. At these stages, the dominant features in the flow are the remnants of the

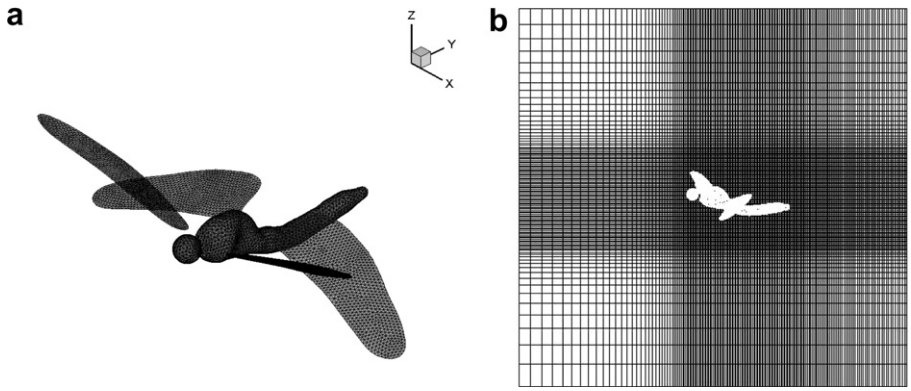

Fig. 24. (a) Surface mesh used to define the geometry of the dragonfly body and wings. (b) Two-dimensional view of the dragonfly model immersed in the fluid grid.

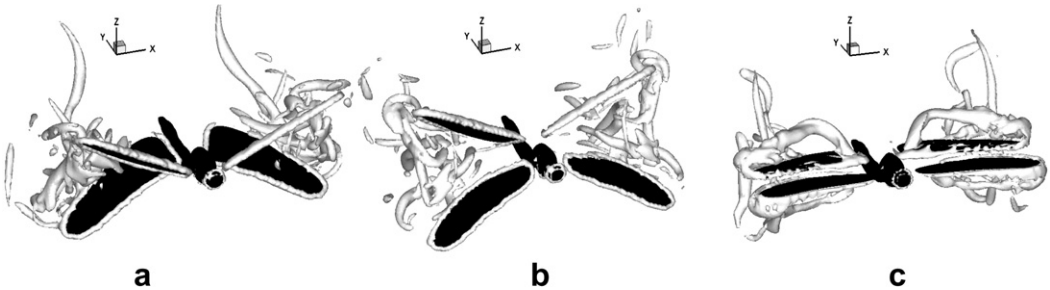

Fig. 25. Isosurfaces of Λ_i at three stages in the flapping cycle of a modeled dragonfly. (a) $t \times f = 0.25$ (b) $t \times f = 0.75$ (c) $t \times f = 1$.

tip and wake vortices created during the stroke. In contrast, Fig. 25(c) shows the wings at the end of the cycle at which stage both wings are at the center position with the forewings rolling upwards and the hind-wings rolling downwards. At this stage, the dominant vortices features in the flow are the detached leading-edge vortices which can be found on the lower(upper) surface of the fore(hind) wings. These vortices are stronger towards the wing-tips where wing velocity is the highest.

It should be pointed out that such a simulation that involves of multiple 3D moving/stationary solid and membraneous bodies would be a severe challenge for any body-conformal grid method. The large relative motion of the fore and hind-wings as they move past each other in opposite direction would in particular require a high level of sophistication in the remeshing and grid adaptation methodology. To our knowledge, a simulation with this degree of complexity has not been attempted before. The study of Wang and Sun [59] which examined the aerodynamic interaction between the fore and hind-wings of a dragonfly is one which comes closest to the current simulations in this regard. In their simulations, Wang and Sun [59] did not include the body of the dragonfly and only modeled the two wings as membraneous bodies. The computational method employed moving overset grid methodology wherein each wing had a single-block structured body-fitted mesh that moved with the wing. These body-fitted meshes were "overset" on a background, stationary Cartesian grid which covered the entire computational domain. The body-fitted overset mesh employed is subject to the usual advantages and disadvantages that come with this approach; whereas the body-fitted grid allows for more precise placement of high resolution grid in the region around immersed boundary, creation of a structured body fitted mesh is difficult for anything other than a relatively simple geometry. Furthermore in this approach, interpolation operators that interpolate the variables between the overset and stationary meshes have to be updated at each time-step and the simulation proceed by iterating till convergence between these two meshes at each time-step. In contrast, such simulations pose no particular challenge for the current immersed boundary method.

The dragonfly configuration presented here is being further refined with more realism injected into the wing kinematics from detailed experimental observations [58] and will then be used for a detailed investigation of dragonfly flight aerodynamics.

4. Conclusions

A highly versatile immersed boundary method for simulating incompressible flow past complex three-dimensional moving boundaries is described. The immersed boundary method is based on a discrete-forcing scheme that allows for a "sharp" representation of the immersed boundary. The immersed boundary method developed is closely coupled with a unstructured grid surface representation of the immersed boundary and salient feature of the IB methodology and the implications of using an unstructured mesh are discussed.

A variety of distinct two- and three-dimensional canonical flows are simulated and computed results compared with available data sets in order to establish the accuracy and fidelity of the current solver. Reynolds numbers for these flows range from $O(10^1)$ up to $O(10^3)$. Simulations are also conducted for flows with moving boundaries and we show that even for rapidly accelerated bodies, the current solver predicts accurately, the temporal variation of the fluid dynamic forces and the evolution of the vorticity field. Finally, we demonstrate the ability of the solver to handle flows with extremely complicated three-dimensional moving boundaries by showing selected results from a fish-pectoral fin simulation as well as simulation of a dragonfly in flight. These two cases show that the solver can handle complex, highly deformable membranous objects as well as multi-component bodies with membranous as well as non-membranous components in complex relative motion.

Acknowledgements

RM would like to acknowledge support for this work from ONR-MURI Grant N00014-03-1- 0897, AFOSR Grant F49550-05-1-0169 and NIH Grant R01-DC007125-0181. FM has been supported by the Department of Energy through the University of California under Subcontract B523819. The CT scan of the porpoise was provided by Dr. Frank Fish of Westchester University.

Appendix A. Truncation error analysis of trilinear interpolation

Fig. A.1 shows the image-point (IP) inside a cuboid with vertices defined by the eight surrounding grid nodes. For the situation shown in Fig. A.1, the value of ϕ inside the cuboidal regions can be expressed as

$$\phi = C_1 x_1' x_2' x_3' + C_2 x_1' x_2' + C_3 x_2' x_3' + C_4 x_1' x_3' + C_5 x_1' + C_6 x_2' + C_7 x_3' + C_8 + \text{T.E.} \tag{A.1}$$

where $x_i' = x_i/\Delta x_i$ (no summation intended) for $i = 1, 2, 3$. In this appendix, we obtain an expression for the truncation error (T.E.) of the above interpolation scheme. In the rescaled coordinates, the coefficients C in the above interpolant can be obtained in terms of the surrounding nodal values as follows:

$$\begin{Bmatrix} C_1 \\ C_2 \\ C_3 \\ C_4 \\ C_5 \\ C_6 \\ C_7 \\ C_8 \end{Bmatrix} = \begin{Bmatrix} (\phi_{111} + \phi_{100} + \phi_{010} + \phi_{001}) - (\phi_{000} + \phi_{110} + \phi_{011} + \phi_{101}) \\ (\phi_{000} + \phi_{110}) - (\phi_{100} + \phi_{010}) \\ (\phi_{000} + \phi_{011}) - (\phi_{010} + \phi_{001}) \\ (\phi_{000} + \phi_{101}) - (\phi_{100} + \phi_{001}) \\ \phi_{100} - \phi_{000} \\ \phi_{010} - \phi_{000} \\ \phi_{001} - \phi_{000} \\ \phi_{000} \end{Bmatrix} \tag{A.2}$$

Next, the nodal values at all the eight nodes are expanded in a Taylor series around the image-point located at $(x_1', x_2', x_3') = (a_1, a_2, a_3)$ as follows:

$$\phi_{\alpha\beta\gamma} = \phi_{\text{IP}} + (D_{\text{IP}}\phi) + \frac{1}{2!}(D_{\text{IP}} D_{\text{IP}}\phi) + \cdots \tag{A.3}$$

where D_{IP} corresponds to following differential operator evaluated at location IP:

$$D_{\text{IP}} \equiv \left[(\alpha - a_1)\frac{\partial}{\partial x_1'} + (\beta - a_2)\frac{\partial}{\partial x_2'} + (\gamma - a_3)\frac{\partial}{\partial x_3'} \right] \tag{A.4}$$

In the above equation α, β and γ can take values equal to 0 and 1 and thus the above expression can be used for any node by using the corresponding value of these indices. The above expressions for all the eight nodes can be substituted into the coefficients in (A.2). Following this, the coefficients are substituted into Eq. (A.1) evaluated at the image-point, i.e. $(x_1', x_2', x_3') = (a_1, a_2, a_3)$. After simplifying the resulting expression, we obtain the following leading order truncation error:

$$\text{T.E.} = f(a_1)\Delta x_1^2 + f(a_2)\Delta x_2^2 + f(a_3)\Delta x_3^2 + g(a_1, a_2)\Delta x_1 \Delta x_2 + g(a_1, a_3)\Delta x_1 \Delta x_3 + g(a_2, a_3)\Delta x_2 \Delta x_3 \tag{A.5}$$

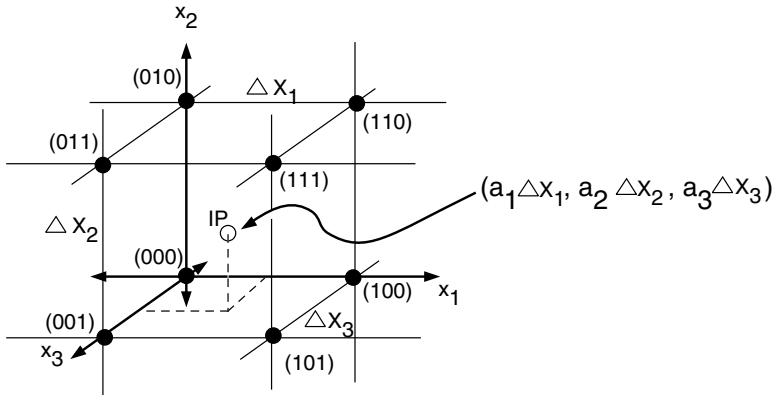

Fig. A.1. Cuboidal region formed by eight nodes surrounding the image-point (IP).

where $f(a_i) = \frac{a_i}{2}(1 - a_i)$ and $g(a_i, a_j) = 4a_i a_j(1 - a_i)(1 - a_j)$ (no summation intended). Finally, assuming that there is some Δ such that $\Delta x_1, \Delta x_2$ and $\Delta x_3 = O(\Delta)$, we conclude that T.E. $= O(\Delta^2)$.

References

[1] A.D. Anderson, C.J. Tannehill, R.H. Pletcher, Computational Fluid Mechanics and Heat Transfer, Hemisphere Publishing Corporation, New York, 1984.
[2] P. Bagchi, M.Y. Ha, S. Balachandar, Direct numerical simulation of flow and heat transfer from a sphere in a uniform cross flow, J. Fluids Eng. 123 (2001) 347–358.
[3] E. Balaras, Modeling complex boundaries using an external force field on fixed Cartesian grids in large-eddy simulations, Comput. Fluids 33 (2004) 375–404.
[4] M. Bozkurttas, H. Dong, V. Seshadri, R. Mittal, F. Najjar, Towards numerical simulation of flapping foils on fixed Cartesian grids, AIAA Paper 2005-0081, 2005.
[5] M. Bozkurttas, Hydrodynamic Performance of Fish Pectoral Fins with Application to Autonomous Underwater Vehicles, D.Sc. Thesis, Department of Mechanical and Aerospace Engineering, The George Washington University, 2007.
[6] H. Borouchaki, S.H. Lo, Fast Delaunay triangulation in three dimensions, Comput. Methods Appl. Mech. Eng. 128 (1995) 153–167.
[7] S.A. Canann, Y.C. Liu, A.V. Mobley, Automatic 3D surface meshing to address today's industrial needs, Finite Elem. Anal. Des. 25 (1997) 185–198.
[8] R. Clift, J.R. Grace, M.E. Weber, Bubbles Drops and Particles, Academic Press, New York, 1978.
[9] R.K.C. Cheng, Inside Rhinoceros 3, Onward Press, 2003.
[10] D. Derakhshani, Introducing Maya 8: 3D for Beginners, Maya Press, 2006.
[11] E.A. Fadlun, R. Verzicco, P. Orlandi, J. Mohd-Yusof, Combined immersed boundary finite-difference methods for three-dimensional complex flow simulations, J. Comput. Phys. 161 (2000) 35–60.
[12] F.N. Felten, T.S. Lund, Kinetic energy conservation issues associated with the collocated mesh scheme for incompressible flow, J. Comput. Phys. 215 (2) (2000) 465–484.
[13] R. Ghias, R. Mittal, T.S. Lund, A non-body conformal grid method for simulation of compressible flows with complex immersed boundaries, AIAA Paper 2004-0080, 2004.
[14] R. Ghias, R. Mittal, H. Dong, A sharp interface immersed boundary method for viscous compressible flows, J. Comput. Phys. 225 (2007) 528–553.
[15] F. Gibou, R.J. Fedkiw, L.T. Cheng, M. Kang, A second-order-accurate symmetric discretization of the Poisson equation on irregular domains, J. Comput. Phys. 176 (2002) 205–227.
[16] D. Goldstein, R. Handler, L. Sirovich, Modeling an no-slip flow boundary with an external force field, J. Comput. Phys. 105 (1993) 354–366.
[17] R.D. Henderson, Details of the drag curve near the onset of vortex shedding, Phys. Fluids 7 (9) (1995) 2102–2104.
[18] T.A. Johnson, V.C. Patel, Flow past a sphere up to a Reynolds number of 300, J. Fluid Mech. 378 (1999) 19–70.
[19] J. Kim, D. Kim, H. Choi, An immersed boundary finite-volume method for simulations of flows in complex geometries, J. Comput. Phys. 171 (2001) 132–150.
[20] P. Koumoutsakos, A. Leonard, High-resolution simulations of the flow around an impulsively started cylinder using vortex methods, J. Fluid Mech. 296 (1995) 1–38.
[21] P. Koumoutsakos, D. Shiels, Simulations of the viscous flow normal to an impulsively started and uniformly accelerated flat plate, J. Fluid Mech. 328 (1996) 177–227.
[22] P.J. Kunz, H. Kroo, Analysis and design of airfoils for use at ultra-low Reynolds numbers, in: T.J. Mueller (Ed.), Progress in Astronautics and Aeronautics, 2001, p. 35.
[23] M.C. Lai, C.S. Peskin, An immersed boundary method with formal second-order accuracy and reduced numerical viscosity, J. Comput. Phys. 160 (2000) 705–719.
[24] G.V. Lauder, P. Madden, R. Mittal, H. Dong, M. Bozkurttas, Locomotion with flexible propulsors: I. Experimental analysis of pectoral fin swimming in sunfish, Bioinsp. Biomim. 1 (2006) S25–S34.
[25] S. Majumdar, G. Iaccarino, P.A. Durbin, RANS solver with adaptive structured boundary non-conforming grids, Cent. Turbul. Res. Annu. Res. Briefs (2001) 353.
[26] S. Marella, S. Krishnan, H. Liu, H.S. Udaykumar, Sharp interface Cartesian grid method I: an easily implemented technique for 3D moving boundary computations, J. Comput. Phys. 210 (2005) 1–31.
[27] D.J. Mavriplis, An advancing front Delaunay triangulation algorithm designed for robustness, J. Comput. Phys. 117 (1) (1995) 90–101.
[28] D.J. Mavriplis, V. Venkatakrishnan A 3D agglomeration multigrid solver for the Reynolds-averaged Navier–Stokes equations on unstructured meshes, AIAA-1995-345, 1995.
[29] R. Mittal, S. Balachandar, Effect of three-dimensionality on the lift and drag of nominally two-dimensional cylinder, Phys. Fluids 7 (8) (1995) 1841–1865.
[30] R. Mittal, S. Balachandar, On the inclusion of three-dimensional effects in simulations of two-dimensional bluff-body wake flows, in: Symposium on Separated and Complex flows, ASME Summer meeting, Vancouver, 1997.
[31] R. Mittal, P. Moin, Suitability of upwind biased schemes for large-eddy simulation of turbulent flows, AIAA J. 35 (8) (1997) 1415–1417.

[32] R. Mittal, A Fourier–Chebyshev spectral collocation method for simulating flow past spheres and spheroids, Int. J. Numer. Meth. Fluids 30 (1999) 921–937.
[33] R. Mittal, J.J. Wilson, F.M. Najjar, Symmetry properties of the transitional sphere wake, AIAA J. 40 (3) (2002) 579–582.
[34] R. Mittal, G. Iaccarino, Immersed boundary methods, Ann. Rev. Fluid Mech. 37 (2005) 239–261.
[35] R. Mittal, H. Dong, M. Bozkurttas, G.V. Lauder, P. Madden, Locomotion with flexible propulsors: II. Computational modeling of pectoral fin swimming in sunfish, Bioinsp. Biomim. 1 (2006) S35–S41.
[36] T.J. Mueller, J.D. DeLaurier, Aerodynamics of small vehicles, Ann. Rev. Fluid Mech. 35 (2003) 89–111.
[37] R. Natarajan, A. Acrivos, The instability of the steady flow past spheres and disks, J. Fluid Mech. 254 (1993) 323–344.
[38] R.A. Norberg, Hovering flight of the dragonfly *Aeschna juncea* L kinematics and aerodynamics, in: T.Y. Wu, C.J. Brokaw, C. Brennen (Eds.), Swimming and Flying in Nature, Plenum Press, NewYork, 1975, p. 763.
[39] D. Ormieres, M. Provansal, Transition to turbulence in the wake of a sphere, Phys. Rev. Lett. 83 (1) (1999) 80–83.
[40] S. Osher R.P. Fedkiw, Level set methods – an overview and some recent results, J. Comput. Phys. 169 (2001) 463–502.
[41] C.S. Peskin, Flow patterns around the heart valves, J. Comput. Phys. 10 (1972) 252–271.
[42] L. Piegl, On NURBS: a survey, IEEE Comput. Graphics Appl. 11 (1991) 55.
[43] W.H. Press, B.P. Flannery, S.A. Teukolsky, W.T. Vetterling, Numerical Recipes in Fortran, Cambridge University Press, 1992.
[44] R. Ramamurti, W.C. Sandberg, R. Lohner, J.A. Walker, M.W. Westneat, Fluid dynamics of flapping aquatic flight in the bird wrasse: three-dimensional unsteady computations with fin deformation, J. Exp. Biol. 205 (2002) 2997–3008.
[45] J.W. Ruge, K. Stueben, Multigrid Methods, Society for Industrial and Applied Mathematics, Philadelhia, PA, 1987.
[46] K. Stuben, A review of algebraic multigrid, J. Comput. Appl. Math. 128 (2001) 281–309.
[47] E.M. Saiki, S. Biringen, Numerical simulation of a cylinder in uniform flow: application of a virtual boundary method, J. Comput. Phys 123 (1996) 450–465.
[48] H. Sakamoto, H. Haniu, The formation mechanism and shedding frequency of vortices from a sphere in uniform shear flow, J. Fluid Mech. 287 (1995) 151–171.
[49] J. Soria, B.J. Cantwell, Identification and classification of topological structures in free shear flows, in: J.P. Bonnet, M.N. Glauser (Eds.), Eddy Structure Identification in Free Turbulent Shear Flows, Kluwer Academic Publishers, Netherlands, 1994.
[50] S. Taneda, Experimental investigation of the wake behind a sphere at low Reynolds numbers, J. Phys. Soc. Jpn. 11 (10) (1956) 1104–1108.
[51] S. Taneda, H. Honji, Unsteady flow past a flat plate normal to the direction of motion, J. Phys. Soc. Jpn. 30 (1) (1971) 262–272.
[52] Y-H Tseng, J.H. Ferziger, A ghost-cell immersed boundary method for flow in complex geometry, J. Comput. Phys. 192 (2003) 593–623.
[53] H.S. Udaykumar, W. Shyy, M.M. Rao, Elafint: a mixed Eulerian–Lagrangian method for fluid flows with complex and moving boundaries, Int. J. Numer. Methods Fluids 22 (1996) 691–712.
[54] H.S. Udaykumar, R. Mittal, W. Shyy, Computational of solid–liquid phase fronts in the sharp interface limit on fixed grids, J. Comput. Phys. 153 (1999) 535–574.
[55] H.S. Udaykumar, R. Mittal, P. Rampunggoon, A. Khanna, A sharp interface Cartesian grid method for simulating flows with complex moving boundaries, J. Comput. Phys. 174 (2001) 345–380.
[56] J. Van Kan, A second-order accurate pressure-correction scheme for viscous incompressible flow, SIAM J. Sci. Stat. Comput. 7 (1986) 870–891.
[57] A. Vargas, A Numerical Investigation of the Aerodynamic Characteristics of a Dragonfly Wing Section, D.Sc. Thesis, Department of Mechanical and Aerospace Engineering, The George Washington University, 2006.
[58] J.M. Wakeling, C.P. Ellington, Dragonfly flight (2). Velocities, accelerations and kinematics of flapping flight, J. Exp. Biol. 200 (1997) 557–582.
[59] J.K. Wang, M. Sun, A computational study of the aerodynamics and forewing–hindwing interaction of a model dragonfly in forward flight, J. Exp. Biol. 208 (2005) 3785–3804.
[60] C.H.K. Williamson, A. Roshko, Measurement of base pressure in the wake of a cylinder at low Reynolds numbers, Z. Flugwiss. Weltraumforsch 14 (1990) 38.
[61] C.H.K. Williamson, The natural and forced formation of spot-like' vortex dislocations' in the transition of a wake, J. Fluid Mech. 243 (1992) 393–441.
[62] C.H.K. Williamson, Vortex dynamics in the cylinder wake, Ann. Rev. Fluid Mech. 8 (1996) 477–539.
[63] T. Ye, R. Mittal, H.S. Udaykumar, W. Shyy, An accurate Cartesian grid method for viscous incompressible flows with complex immersed boundaries, J. Comput. Phys. 156 (1999) 209–240.
[64] D. You, M. Wang, R. Mittal, P. Moin, Study of rotor tip-clearance flow using large eddy simulation, AIAA Paper 2003-0838, 2003.
[65] Y. Zang, R.L. Streett, J.R. Koseff, A non-staggered fractional step method for time-dependent incompressible Navier–Stokes equations in curvilinear coordinates, J. Comput. Phys. 114 (1994) 18–33.