Available online at www.sciencedirect.com
SCIENCE DIRECT®
Journal of Computational Physics 207 (2005) 457–492
JOURNAL OF COMPUTATIONAL PHYSICS
www.elsevier.com/locate/jcp

A hybrid Cartesian/immersed boundary method for simulating flows with 3D, geometrically complex, moving bodies

Anvar Gilmanov, Fotis Sotiropoulos *

School of Civil and Environmental Engineering, Georgia Institute of Technology, 790 Atlantic Drive, Atlanta, GA 30332-0355, USA

Received 6 August 2004; received in revised form 14 December 2004; accepted 26 January 2005
Available online 3 March 2005

Abstract

A numerical method is developed for solving the 3D, unsteady, incompressible Navier–Stokes equations in Cartesian domains containing immersed boundaries of arbitrary geometrical complexity moving with prescribed kinematics. The governing equations are discretized on a hybrid staggered/non-staggered grid layout using second-order accurate finite-difference formulas. The discrete equations are integrated in time via a second-order accurate dual-time-stepping, artificial compressibility iteration scheme. Unstructured, triangular meshes are employed to discretize complex immersed boundaries. The nodes of the surface mesh constitute a set of Lagrangian control points used to track the motion of the flexible body. At every instant in time, the influence of the body on the flow is accounted for by applying boundary conditions at Cartesian grid nodes located in the exterior but in the immediate vicinity of the body by reconstructing the solution along the local normal to the body surface. Grid convergence tests are carried out for the flow induced by an oscillating sphere in a cubic cavity, which show that the method is second-order accurate. The method is validated by applying it to calculate flow in a Cartesian domain containing a rigid sphere rotating at constant angular velocity as well as flow induced by a flapping wing. The ability of the method to simulate flows in domains with arbitrarily complex moving bodies is demonstrated by applying to simulate flow past an undulating fish-like body and flow past an anatomically realistic planktonic copepod performing an escape-like maneuver.

Keywords: Artificial compressibility; Dual time-stepping; Immersed boundaries; Fluid–structure interaction

* Corresponding author. Tel.: +1 404 894 4432; fax: +1 404 385 1131.
E-mail addresses: anvar.gilmanov@ce.gatech.edu (A. Gilmanov), fsotirop@ce.gatech.edu, fs30@ce.gatech.edu (F. Sotiropoulos).

0021-9991/$ - see front matter © 2005 Elsevier Inc. All rights reserved.
doi:10.1016/j.jcp.2005.01.020

1. Introduction

Many flows of technological and/or biological significance take place in multi-connected domains with complex, flexible, immersed boundaries. Typical examples range from flows in natural rivers with flexible vegetation, to flows in the human cardiovascular system, and flows past swimming and flying animals and insects. Simulation of the fluid/structure interaction that dominates the dynamics of these flows poses a formidable challenge to even the most advanced numerical techniques, and is currently at the forefront of ongoing work in computational fluid dynamics. In recent years, ''non-boundary conforming'' numerical methods are attracting attention in simulations of such problems due to their increased versatility as compared to boundary-fitted techniques based on the so-called arbitrary Lagrangian Eulerian (ALE) approach [1–4]. ALE methods are better suited for carrying out high Reynolds number simulations but, due to the need for the mesh to conform to the body at all times, they are inherently limited to problems with moderate body deformations. This limitation can be mitigated by solving the governing equations on a fixed Cartesian grid and accounting for the effect of a stationary or moving body, which no longer coincides with a grid surface, via proper treatment of the solution variables at grid cells in the vicinity of the boundary. This feature eliminates the tedious task of grid adaptation required in classical ''boundary conforming'' methods, allowing the simulation of flows with complex boundaries undergoing large deformations in a relatively straightforward manner.

A major challenge in the implementation of any non-boundary conforming methodology is the establishment of the relation between the Lagrangian coordinates of the body and the underlying Eulerian grid and the imposition of boundary conditions. In fact, the overall accuracy and applicability of such methods to complex three-dimensional flows depends critically on three important issues: (1) description of the topology of complex moving boundaries; (2) identification of the relation between the surface of the body and the underlying grid; (3) enforcement of proper boundary conditions. The first two issues, which are collectively referred to as the ''interface tracking'' problem, can be addressed either on an Eulerian or a Lagrangian frame of reference, leading to purely Eulerian or Lagrangian–Eulerian formulations, respectively. A review of different ''interface tracking'' methodologies can be found in [5] and some characteristic examples are given in [6–8]. In terms of the imposition of boundary conditions (item 3), non-boundary conforming methods can be classified in two major categories: *Cartesian* methods and *Immersed Boundary* methods.

In *Cartesian* (or *cut-cell*) methods a solid boundary is tracked as a sharp interface and the grid cells at the body interface are modified according to their intersections with the underlying Cartesian grid. Using proper interpolation strategies the flow variables on the resulting modified (irregular) cells can be computed according to the boundary conditions on the body. *Cartesian* methods allow a clear distinction between the solid and the fluid, by practically generating a boundary-fitted grid around the body, and are applicable to problems involving arbitrary deformations of a body (or an ensemble of bodies) relative to the fixed Eulerian grid. The first predecessor of modern-day *Cartesian* methods was proposed in the early work of Noh [9] and dubbed as coupled Eulerian–Lagrangian method (CEL). Recent successful applications to two-dimensional inviscid and viscous flow problems can be found in [10–13]. A difficulty encountered in the implementation of *Cartesian* methods arises from the large number of possible intersections between the fixed grid and the surface of the deforming boundary, leading to the formation of various irregular ''interface-cells''. Implementing the boundary conditions in such cells necessitates a large number of ''special treatments'', which could result in complex coding logistics. Furthermore, in complex configurations the unavoidable generation of irregularly shaped cells with very small volume can adversely impact the conservation and stability properties of the solver. To address this problem cell merging techniques have been proposed by Quirk [12] and Ye et al. [14]. The later formulation [14] was also extended to treat moving boundaries with good results for a variety of two-dimensional problems [15]. Cut-cell formulations applicable to three-dimensional problems with stationary immersed boundaries have been proposed in [16–18] and [19].

In the so-called *Immersed Boundary* methods the governing equations are also discretized on a fixed Cartesian grid. The effect of a stationary or moving boundary, however, is accounted for by introducing an external force field in the equations of motion, which is designed to ensure that the fluid satisfies the no-slip conditions on solid boundaries. The method was first introduced by Peskin in the early 70s, although some earlier work by Vieceli [21,22] in the extension of the Marker and Cell method to cases with arbitrarily shaped and moving boundaries also falls into this category of methods. Peskin and McQueen [23–26] applied the *Immersed Boundary* methodology to study blood flow in the human heart. In the above computations the vascular boundary was modelled as a set of elements linked by springs and a Lagrangian coordinate system was attached to track their location in space. The tracking information was then used to compute the spatial distribution of the external force field that was explicitly introduced in the governing equations at the fixed Eulerian grid nodes. To avoid numerical instabilities due to the stiffness of the problem, a delta function was used to distribute the forcing over 3−4 grid nodes in the vicinity of the boundary. Goldstein et al. [27] introduced an alternative formulation to treat stationary solid boundaries coupled with a highly accurate spectral method. They used a "feedback-forcing" approach which asymptotically enforces the desired boundary conditions on a solid boundary. Due to the global character of the spectral method, to avoid spurious oscillations near the immersed boundaries forcing was also spread over a few points using a function similar to the delta-function employed by Peskin [20]. Three-dimensional computations of turbulent flow in plane channels [27], and ridded channels [28] were in good agreement with reference data. More recently, Cortez and Minion [29] proposed a higher-order immerse boundary scheme using the smoothed blob projection to compute the force field and demonstrated better accuracy than the original immersed boundary method. Their method, however, also employs a discrete delta function and results in the smearing of the interface.

The spreading of the forcing function over several grid nodes is an inherent feature of immersed boundary formulations. This feature increases substantially the spatial resolution requirements and appears to be a major obstacle in the extension of the method to realistic Reynolds number technological and biological applications. A variant of the classical immersed boundary method that overcomes this limitation was proposed by LeVeque and Li [30,31] and was dubbed the immersed interface method (*IIM*). This method was initially applied to solve 2D elliptic equations [30] and Stokes flow problems with flexible boundaries [31] and more recently it was extended to solve the 2D, incompressible Navier–Stokes equations [32]. The *IIM* modifies the governing equations at grid nodes only in the immediate vicinity of the interface by adding forcing functions constructed to enforce a set of appropriate jump conditions at the interface. The method maintains sharp-interfaces between different phases and is second-order accurate.

A different approach that does not require the explicit addition of discrete forces to the governing equations was recently proposed by Mohd-Yusof [33] and Fadlun et al. [34]. Similarly to *Cartesian* methods and the *IIM*, this approach treats the solid boundary as a sharp interface. Rather than modifying the Eulerian grid cells in the vicinity of the boundary and applying boundary conditions exactly on the boundary, however, this method applies boundary conditions at the grid nodes closest to the solid boundary. The specific values of various flow variables at such near-boundary nodes are calculated by interpolating linearly along an appropriate grid line between the nearest interior node, where flow variables are available from the solution of the governing equations, and the point where the grid line intersects the solid boundary, where physical boundary conditions are known. This approach can be thought of as accounting for the presence of a solid boundary by introducing, albeit implicitly, a set of discrete body-forces at the grid nodes nearest to the boundary. In that sense, this method is conceptually related to the *IIM* but since it does not require the explicit derivation of jump conditions and the addition of body forces it has been more appropriately classified as a hybrid Cartesian/immersed boundary (*HCIB*) approach [34]. The *HCIB* approach is very promising for 3D flows and has been successfully applied to a variety of problems including large-eddy simulation (LES) of turbulent flow inside a motored IC piston/cylinder assembly [34,35] and the direct numerical simulation (DNS) of the flow in an impeller stirred tank [36].

A critical issue for the successful implementation of the *HCIB* approach in three-dimensional flows, lies in the choice of the solution reconstruction scheme near the immersed interface. In Fadlun et al. [34], a simple one-dimensional scheme was suggested where the solution is reconstructed along an arbitrarily selected grid line. The method is straightforward, second-order accurate, and works well for bodies that are largely aligned with the grid lines. In cases of complex bodies the choice of the reconstruction direction at several points in the flow field can be ambiguous. Multidimensional schemes can remove this limitation. Kim et al. [37] suggested a scheme, which uses a bilinear reconstruction procedure, which is reduced to a one-dimensional linear one when there are no available points in the vicinity of the boundary to support the two-dimensional stencil. Gilmanov et al. [38] recently introduced a more general scheme, which is applicable to complex three-dimensional boundaries without special treatments. In the method of Gilmanov et al. the immersed boundary is discretized with an unstructured triangular mesh and the reconstruction of the solution is performed along the well defined, normal to the body direction – which can be easily computed in a straightforward manner for a triangulated surface. The method [38] was applied to simulate laminar flow past a sphere for $Re \leqslant 300$. Gilmanov et al. reported nearly second-order spatial accuracy and excellent agreement with benchmark computations for this flow on body-fitted meshes. A method employing the same reconstruction idea but is applicable only to 2D geometries was recently applied successfully to carry out large-eddy simulation of turbulent flow in a plane channel with a wavy wall [39].

In this paper, we develop a new *HCIB* formulation applicable to three-dimensional flows with arbitrarily complex immersed boundaries moving with prescribed motion. Our methodology maintains a sharp fluid/body interface by discretizing the body surface using an unstructured, triangular mesh. The nodes of this mesh constitute a set of Lagrangian control points, which are used to track the motion and reconstruct the instantaneous shape of the moving immersed boundary. The reconstruction of the solution near the boundary is carried out using a method that relies on the key idea presented in [38], i.e., by interpolation along the normal to the body. Our method, however, differs substantially and in several aspects from that presented in [38]. The method of Gilmanov et al. is only applicable to stationary bodies of simple (convex) shape while the present approach is applicable to arbitrarily complex, moving bodies including bodies with multiple moving appendages. In the present approach the governing equations are discretized using a novel hybrid staggered/non-staggered grid arrangement (as compared to the non-staggered grid method employed in [38]), which enhances accuracy in flows with moving boundaries and simplifies the implementation of the solution reconstruction algorithm near complex immersed bodies. Finally, a quadratic interpolation approach is developed for reconstructing the solution near the body. This interpolation was found to improve accuracy in problems with moving boundaries as compared to the linear interpolation method used in [38].

A grid convergence study is carried out for flow induced by a rigid sphere undergoing translational motion in a closed cubic container to establish the order accuracy of the method and second-order convergence rate is demonstrated. The method is validated by applying it to two test cases. The first case is flow past a sphere rotating steadily in a fluid, which is at rest at large distance from the sphere. This problem is formulated and solved as a transient, moving-body problem with the sphere rotating at the prescribed angular velocity relative to the fixed Cartesian grid. The computed steady results compare well with benchmark numerical results [40]. The second case is flow induced by a flapping 3D wing undergoing complex and continuous translational and rotational motion in a confined region. Experiments for this case have been reported by Birch and Dickinson [41]. The calculated lift and drag forces time histories are shown to be in good agreement with the measurements. The method is also applied to simulate flow past an undulating fish-like body and a planktonic copepod of reasonably realistic anatomy performing an escape-like maneuver.

In Section 2, the governing equations are presented and the basic numerical method is described. In Section 3, the interface tracking algorithm and the implementation of boundary conditions on flexible, immersed boundaries are discussed. Results are presented in Section 4 and finally some concluding remarks are given in Section 5.

2. The numerical method

2.1. The governing equations and boundary conditions

We solve the unsteady, three-dimensional, incompressible Navier–Stokes formulated in primitive variable form. The governing equations are non-dimensionalized by a characteristic velocity U, a characteristic length L, and a characteristic time scale $1/f$ (where f is a characteristic frequency of the flow) and formulated in Cartesian coordinates, $\{x_i\}$, as follows:

$$\frac{\partial u_j}{\partial x_j} = 0, \tag{1}$$

$$St\frac{Du_i}{Dt} = St\frac{\partial u_i}{\partial t} + u_j\frac{\partial u_i}{\partial x_j} = -\frac{\partial p}{\partial x_i} + \frac{1}{Re}\frac{\partial^2 u_i}{\partial x_j \partial x_j}, \tag{2}$$

where $D(\,)/Dt$ is the Lagrangian derivative, u_i are the Cartesian velocity components, p is the pressure divided by the density, $St = fL/U$ and $Re = UL/\nu$ are the Strouhal and Reynolds numbers, respectively, and ν is the kinematic viscosity of the fluid.

For a flow domain containing a deformable solid immersed body, boundary conditions for the velocity vector and pressure field need to be prescribed on the dynamically evolving interface $\Gamma^b(t)$ between the fluid and the solid. Assume that the interface is discretized by a set of K material points, which lie on $\Gamma^b(t)$ at all times and can be described by their respective Lagrangian position vector $\mathbf{r}^k(t)$

$$\mathbf{r}^k(t) \in \Gamma^b(t)\ \forall\ t > 0, \quad \text{with} \quad \mathbf{r}^k(0) = \mathbf{r}^k_{\mathrm{o}}\ \forall\ k = 1, K, \tag{3}$$

where $\mathbf{r}^k_{\mathrm{o}}$ is the initial location of the kth material point on $\Gamma^b_{\mathrm{o}} \equiv \Gamma^b(0)$. Assuming that all points on $\Gamma^b(t)$ move with a known velocity $\mathbf{U}^k(t)$, the shape of $\Gamma^b(t)$ at time t can be obtained by solving the Lagrangian advection equations for all material points on the surface (for $k = 1, K$):

$$\frac{\partial \mathbf{r}^k}{\partial t} = \mathbf{U}^k(t), \quad \text{with} \quad \mathbf{r}^k(0) = \mathbf{r}^k_{\mathrm{o}}. \tag{4}$$

With the shape and location of the interface $\Gamma^b(t)$ known at time t, boundary conditions for the Eulerian fluid velocity vector $\mathbf{u}(\mathbf{r}, t)$ must be prescribed at all points on Γ^b as follows:

$$\mathbf{u}(\mathbf{r}^k(t), t) = \mathbf{U}^k(t)\ \forall\ k = 1, K. \tag{5}$$

This boundary condition establishes the link between the Eulerian description of the fluid motion on the fixed Cartesian domain and the Lagrangian description of the motion of the immersed boundary and must be enforced at all points on the interface at every instant in time.

Neumann boundary conditions for the pressure field [42] on $\Gamma^b(t)$ can be obtained by projecting the momentum equation (2) on the direction normal to the interface as follows:

$$-\mathbf{n}^k \cdot \nabla p = -\left(\frac{\partial p}{\partial n}\right)_{\text{at } \mathbf{r}=\mathbf{r}^k(t)} = \mathbf{n}^k \cdot \left[St\frac{D\mathbf{u}}{Dt} - \frac{1}{Re}\nabla^2\mathbf{u}\right]_{\text{at } \mathbf{r}=\mathbf{r}^k(t)}, \tag{6}$$

where $\mathbf{n}^k$ denotes the outward pointing, normal-unit-vector at point $\mathbf{r}^k(t)$. For sufficiently high Re and since the momentum equation is applied on the interface, the above equation can be written as follows:

$$-\left(\frac{\partial p}{\partial n}\right)_{\text{at } \mathbf{r}=\mathbf{r}^k(t)} = St\left[\mathbf{n}^k \cdot \frac{\partial \mathbf{U}^k(t)}{\partial t}\right]. \tag{7}$$

The term in the right-hand side of this equation is the acceleration of the material nodes on the interface and can be calculated from the known velocity of the interface. For a stationary or steadily moving body or for a deforming body with $St \ll 1$, the above condition simplifies to the well-known homogeneous boundary condition for the pressure field.

The above boundary conditions on the immersed interface need to be supplemented with appropriate inflow, outflow, and far-field boundary conditions, which are discussed in subsequent sections of this paper. Given a complete set of boundary conditions, the incompressible Navier–Stokes and continuity equations constitute a well-posed problem if the following integral constraint is satisfied:

$$\int_{\Gamma} \mathbf{u} \cdot \mathrm{d}\mathbf{S} + \sum_{k=1,N} \mathbf{U}^k(t) \cdot \mathbf{n}^k \Delta S^k = 0. \tag{8}$$

This equation expresses the global mass conservation in a domain Γ containing a flexible, immersed boundary $\Gamma^b(t)$.

2.2. The numerical method

In this section, we describe the numerical method for solving the governing equations in a Cartesian domain without flexible, immersed boundaries. We discretize the continuity and momentum equations on a hybrid staggered/non-staggered mesh arrangement using second-order accurate, finite-difference formulas. The divergence operator in the continuity equation and the pressure gradient and viscous terms in the momentum equations are approximated using three-point central, second-order accurate differencing. The convective terms in the momentum equations are discretized using the second-order accurate, upwind-biased QUICK scheme [43]. The discrete equations are integrated in time using a second-order accurate, dual-time-stepping artificial compressibility approach [44,45]. We begin the description of our numerical method by discussing the need for and describing the implementation of the staggered/non-staggered grid arrangement.

It is well known that when three-point, central differencing is used to discretize both the velocity divergence and the pressure gradient operators on a non-staggered grid the resulting discrete velocity field will be divergence-free but the pressure field will, in general, exhibit non-physical, odd-even oscillations [46,47]. Discretization techniques aimed at eliminating spurious decoupling of the pressure nodes have been proposed in the literature (see for example [46–48]) and successfully applied to a broad range of complex flows ([38,49,50]). Such methods introduce, either explicitly [46] or implicitly by the way the mass fluxes are constructed at cell interfaces [47], in the discrete continuity equation an artificial dissipation term, which is third-order accurate in space and involves fourth-order spatial derivatives of the pressure field. For flows with flexible immersed boundaries, however, such artificial dissipation terms could become very large in the vicinity of the body – especially for sufficiently high values of the Strouhal number – leading to unacceptably high errors in the discrete continuity equation. The reasons for this can be understood from Eq. (7), which shows that the pressure gradient in the immediate vicinity of the body is directly proportional to the Strouhal number. One can of course remedy this situation by refining the grid in the immediate vicinity of the body but this approach could require excessively large computational grids, especially in the context of Cartesian grid methods.

The above computational difficulties can be circumvented by adopting a staggered grid arrangement, which guarantees the satisfaction of the discrete continuity to machine zero while ensuring the smoothness of the discrete pressure field. Implementing a pure staggered grid discretization in a Cartesian grid approach, however, would be cumbersome due to need for special treatment of grid nodes near the body – see [19] for a recent extensive discussion on the difficulties encountered in the implementation of a staggered grid approach in a Cartesian method. In staggered grid formulations, for instance, the solid boundary is

placed at half grid nodes (velocity nodes). This practice eliminates the need for explicitly specifying a pressure boundary condition at the solid surface but complicates the implementation of boundary conditions for the velocity field. For example, the discretization of the convective and viscous terms in the momentum equations at velocity nodes immediately adjacent to the boundary requires knowledge of the velocity at fictitious nodes that lie outside of the computational domain [51]. In the context of the HCIB approach we develop herein, this requirement could complicate the reconstruction of the solution near the body. For that reason we develop a hybrid staggered/non-staggered grid approach, which eliminates the aforementioned difficulties by combining the versatility and ease of implementation of the non-staggered mesh with the superior accuracy of a staggered mesh approach.

Let u, v, and w be the three velocity components, i, j, and k the Cartesian grid indices in the x, y, and z directions, respectively, and im, jm, km the corresponding number grid nodes in each direction. As in a collocated, non-staggered grid approach both the pressure and velocity fields are defined and stored at the (i,j,k) nodes and the boundaries of the computational domain are located at $i = 1$ and im, $j = 1$ and jm, and $k = 1$ and km (see Fig. 1). The velocity components, however, are also defined at their respective half node locations as required in a standard staggered grid approach – i.e., u, v, and w are also defined at the $(i \pm 1/2,j,k)$, $(i,j \pm 1/2,k)$, and $(i,j,k \pm 1/2)$ nodes, respectively (Fig. 1). Similarly to the staggered grid approach, the continuity equation is satisfied at the (i,j,k) nodes while the momentum equations are satisfied at their respective half nodes. The key feature of our hybrid formulation is that even though we satisfy the discrete equations on a staggered grid arrangement, physical boundary conditions for the velocity components need to be prescribed only at the boundaries of the computational domain as defined in a collocated grid layout.

To illustrate the method consider the staggered-grid discretization of the governing equations formulated in dual time-stepping, artificial compressibility form, as follows:

$$\frac{P^{n+1,\ell+1}_{i,j,k} - P^{n+1,\ell}_{i,j,k}}{\Delta\tau} + \delta_x u^{n+1,\ell}_{i,j,k} + \delta_y v^{n+1,\ell}_{i,j,k} + \delta_z w^{n+1,\ell}_{i,j,k} = 0, \tag{9}$$

$$\begin{aligned}
&\frac{u^{n+1,\ell+1}_{i+1/2,j,k} - u^{n+1,\ell}_{i+1/2,j,k}}{\Delta\tau} + St\delta_t u^{n+1,\ell+1}_{i+1/2,j,k} + XM^{n+1,\ell}_{i+1/2,j,k} + \delta_x P^{n+1,\ell}_{i+1/2,j,k} = 0,\\
&\frac{v^{n+1,\ell+1}_{i,j+1/2,k} - v^{n+1,\ell}_{i,j+1/2,k}}{\Delta\tau} + St\delta_t v^{n+1,\ell+1}_{i,j+1/2,k} + YM^{n+1,\ell}_{i,j+1/2,k} + \delta_y P^{n+1,\ell}_{i,j+1/2,k} = 0,\\
&\frac{w^{n+1,\ell+1}_{i,j,k+1/2} - w^{n+1,\ell}_{i,j,k+1/2}}{\Delta\tau} + St\delta_t w^{n+1,\ell+1}_{i,j,k+1/2} + ZM^{n+1,\ell}_{i,j,k+1/2} + \delta_z P^{n+1,\ell}_{i,j,k+1/2} = 0,
\end{aligned} \tag{10}$$

where

$$\delta_x(\,)_{i,j,k} = \frac{(\,)_{i+1/2,j,k} - (\,)_{i-1/2,j,k}}{\Delta x},$$

$$\delta_t u^{n+1,\ell+1}_{i+1/2,j,k} = \frac{-3u^{n+1,\ell+1}_{i+1/2,j,k} + 4u^{n}_{i+1/2,j,k} - u^{n-1}_{i+1/2,j,k}}{2\Delta t},$$

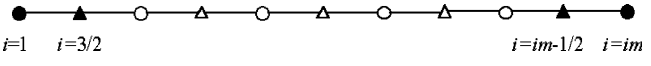

Fig. 1. Schematic depicting the hybrid staggered/non-staggered grid layout. Filled circles: non-staggered grid nodes where boundary conditions are specified; filled triangles: staggered grid nodes where boundary conditions are obtained via linear interpolation; open circles: nodes where the continuity equation is solved and the convective and viscous terms of the u-momentum equation are discretized; open triangles: nodes where the u-momentum equation is solved using interpolated values for the viscous and convective terms.

$\Delta\tau$ is the dual-time increment, Δt and Δx are the (physical) time and space increments, respectively, the superscripts ℓ and n indicate dual and physical time levels, respectively, and XM, YM, and ZM are discrete approximations of the convective and viscous terms in the x-, y-, and z-momentum equations, respectively. Note that in order to facilitate the presentation of the algorithm we have used in the above equations the Euler explicit scheme to integrate the equations in dual time – see subsequent discussion for more details on the dual-time integration scheme. Also note that the superscript $(n+1,\ell)$ denotes the ℓth dual-time iterate of the solution at the $n+1$ physical time step. At convergence, $(\,)^{n+1,\ell+1}$, $(\,)^{n+1,\ell} \to (\,)^{n+1}$ and the above dual time iteration converges to the solution of the incompressible Navier–Stokes equations at the $n+1$ physical time step. For the sake of simplicity, in what follows we adopt the simpler notation $(\,)^{\ell}$ to denote $(\,)^{n+1,\ell}$.

As we discussed above, the implementation of a pure staggered grid approach is complicated by the need to discretize the spatial derivatives of the velocity field contained in the XM, YM, and ZM terms at velocity nodes near the boundary. To remedy this situation we discretize these terms at the (i,j,k) nodes in exactly the same way as in a non-staggered grid formulation and then use interpolation to obtain their discrete values at the half nodes. Assuming for the time being that the velocity field is known at all (i,j,k) nodes at the ℓ iteration level (see below for the approach we adopt to calculate the velocity components at these nodes), we employ the second-order accurate, upwind-biased QUICK scheme [43] to discretize the convective terms along with three-point, central differencing for the viscous terms to compute $XM^{\ell}_{i,j,k}$, $YM^{\ell}_{i,j,k}$ and $ZM^{\ell}_{i,j,k}$ terms. Note that since the discretization of these terms involves only standard, non-staggered grid operators, the need for specifying boundary conditions at fictitious velocity nodes that lie outside of the computational domain is eliminated. After the convective and viscous terms have been discretized at the collocated grid nodes, we employ interpolation along the corresponding grid lines to calculate $XM^{\ell}_{i+1/2,j,k}$, $YM^{\ell}_{i,j+1/2,k}$, and $ZM^{\ell}_{i,j,k+1/2}$, which are required for solving the momentum equations (10).

Let us demonstrate the interpolation approach for $XM^{\ell}_{i+1/2,j,k}$, which is evaluated by interpolating along the x-direction (refer to Fig. 1 for notation). As discussed above, boundary conditions for all three velocity components are known at $i=1$ and $i=im$, and $XM^{\ell}_{i,j,k}$ has been discretized at all interior nodes $(2 \leqslant i \leqslant im-1)$. We will also assume for the time being that boundary conditions for the u velocity component are known at the staggered grid nodes $i = 1 + 1/2$ and $im - 1/2$ (see below for details) at the ℓ iteration level, and, thus, the x-momentum equations (10) are solved only for $2 \leqslant i \leqslant im-2$. The following third-order accurate, backward-biased, interpolation formula is used to evaluate $XM^{\ell}_{i+1/2,j,k}$:

$$XM^{\ell}_{i+1/2,j,k} = \frac{1}{8}\left(3XM^{\ell}_{i+1,j,k} + 6XM^{\ell}_{i,j,k} - XM^{\ell}_{i-1,j,k}\right). \tag{11}$$

Since $XM^{\ell}_{1,j,k}$ is not available, this formula works only for $i = 3$ to $im - 2$. To evaluate XM at the staggered node $i = 2 + 1/2$ the following forward biased version of the above interpolation formula is used (applied at $i = 2$):

$$XM^{\ell}_{i+1/2,j,k} = \frac{1}{8}\left(3XM^{\ell}_{i,j,k} + 6XM^{\ell}_{i+1,j,k} - XM^{\ell}_{i+2,j,k}\right). \tag{12}$$

The procedure for evaluating YM and ZM involves interpolation along the y- and z-directions of the grid in a manner that is a straightforward extension of the above interpolation procedure along the x-direction. With XM, YM, and ZM known and with prescribed boundary conditions for the staggered grid velocities $u_{3/2,j,k}$, $u_{im-1/2,j,k}$, $v_{i,3/2,k}$, $v_{i,jm-1/2,k}$, $w_{i,j,3/2}$, $w_{i,j,km-1/2}$, the discrete continuity and momentum equations (9) and (10) can be iterated to advance the solution to the next dual-time level and obtain the pressure and velocity components as in the staggered grid approach. After computing the velocity components at the half grid nodes at the $\ell+1$ level, by solving Eq. (10), we interpolate to define a new approximation of the velocity components at all interior collocated grid nodes as follows:

$$u^{\ell+1}_{i,j,k} = \frac{1}{8}\left(3u^{\ell+1}_{i+1/2,j,k} + 6u^{\ell+1}_{i-1/2,j,k} - u^{\ell+1}_{i-3/2,j,k}\right), \quad \text{for } i = im-1, \tag{13}$$

$$u^{\ell+1}_{i,j,k} = \frac{1}{8}\left(3u^{\ell+1}_{i-1/2,j,k} + 6u^{\ell+1}_{i+1/2,j,k} - u^{\ell+1}_{i+3/2,j,k}\right), \quad \text{for } 2 \leqslant i \leqslant im-2. \tag{14}$$

With the velocity field known at all interior collocated nodes, the physical boundary conditions of the problem at hand can be applied to update the velocity field at all boundary nodes of the collocated grid layout. Finally, as we mentioned above boundary conditions also need to be specified for the staggered grid velocities at the $(3/2,j,k)$, $(im-1/2,j,k)$, etc. nodes. Note that by specifying these boundary conditions we eliminate the need for explicit pressure boundary conditions, thus, retaining in our algorithm yet another desirable feature of the pure staggered grid approach. Simple, second-order accurate averaging is used to to update these boundary values. For example, $u^{\ell+1}_{3/2,j,k}$ is updated as follows:

$$u^{\ell+1}_{3/2,j,k} = \frac{1}{2}\left(u^{\ell+1}_{1,j,k} + u^{\ell+1}_{2,j,k}\right). \tag{15}$$

The overall procedure for advancing the solution from the n to the $n+1$ physical time step can, thus, be summarized as follows:

1. Use the solution at the n time level to initialize the velocity (both at the collocated and staggered grid nodes) and pressure fields at the $\ell = 0$ dual time level.
2. Using the QUICK scheme for the convective terms and central-differencing for the viscous terms, calculate $XM^{\ell}_{i,j,k}$, $YM^{\ell}_{i,j,k}$, and $ZM^{\ell}_{i,j,k}$ terms at all interior collocated nodes.
3. Using the third-order interpolation formulas given by Eqs. (11) and (12), evaluate $XM^{\ell}_{i+1/2,j,k}$, $YM^{\ell}_{i,j+1/2,k}$, and $ZM^{\ell}_{i,j,k+1/2}$.
4. Solve the governing equations (9) and (10) to advance the pressure and velocity fields to the $\ell+1$ iteration level.
5. Using the third-order interpolation formulas given by Eqs. (13) and (14), evaluate $u^{\ell+1}_{i,j,k}$, $v^{\ell+1}_{i,j,k}$, and $w^{\ell+1}_{i,j,k}$ at all interior nodes of the collocated grid.
6. Update the velocity field at all boundaries of the collocated grid by applying the proper boundary conditions at the inflow, outflow, far field, and solid boundaries of the computational domain.
7. Using linear interpolation, Eq. (15), update the velocity components at the boundary nodes of the staggered grid.
8. If convergence in dual time has been achieved (based on some appropriate error norm) go to step 1 to begin the dual iteration procedure for advancing the solution to the next physical time step. If not, continue the dual time iteration process at the current physical time step by returning to step 2.

To test the spatial resolution of the above hybrid staggered/non-staggered algorithm we compared its accuracy for the well-known lid-driven cavity problem with the accuracy of its pure staggered-grid counter-part. We found that both the hybrid and staggered grid algorithms are second-order accurate in space and on a given grid yield essentially identical results. Furthermore, both methods were able to reduce the discrete divergence of the velocity field to machine zero. We also examined the sensitivity of the solutions obtained with the hybrid algorithm to the spatial bias introduced by the linear interpolation schemes given by Eqs. (11)–(14). We found that slightly better results are obtained when we symmetrize the overall scheme (as in Eqs. (11)–(14)) by using forward-biased stencil to calculate the XM terms at the staggered nodes and backward-biased stencil to calculate the velocity components at the collocated grid nodes. Additional evidence of the accuracy of the hybrid formulation is presented in the subsequent sections of this paper.

The solution algorithm described above has been implemented using the pointwise implicit, four-stage Runge–Kutta algorithm described in [52] to integrate the governing equations in dual-time – instead of

the Euler explicit scheme given by Eqs. (1) and (2). The Runge–Kutta algorithm was enhanced with the pressure-based, implicit residual smoothing operator of [53] and local (dual) time stepping to accelerate its rate of convergence in dual time. Typically, 30–60 dual iterations are required per physical time step to reduce the residuals by approximately three orders of magnitude. Finally except inflow, where boundary conditions are known, and solid boundaries, where boundary conditions are specified as discussed in the subsequent section, all other boundaries in this work were handled by implementing in the dual time-stepping artificial compressibility algorithm, the non-reflective characteristic approach proposed by Thompson [54].

3. Treatment of flexible immersed boundaries

In this section, we discuss the numerical treatment of domains with complex, flexible, immersed boundaries, which move with prescribed motion. Our approach consists of the following two-step procedure: (1) Lagrangian tracking of the immersed boundary, which requires the precise description of the boundary shape and its relation to the fixed Cartesian grid and (2) local reconstruction of the solution near the immersed boundary such that Eqs. (5) and (6) are satisfied exactly on the body at every instant in time. The reconstruction algorithm we develop herein relies on the basic idea presented in [38] – i.e., the reconstruction of the solution along the local normal to the body. The algorithm of [38], however, is modified substantially in this work in order to make it applicable to bodies of arbitrary geometrical complexity and to further enhance its accuracy in fluid/structure interaction problems.

In the HCIB formulation proposed in [38] the immersed boundary is treated as a sharp interface. Boundary conditions are applied at nodes in the immediate vicinity of the immersed boundary (see Fig. 2) by reconstructing the solution along the well-defined normal to the body direction using information from interior nodes and the known boundary conditions on the body. To facilitate the reconstruction of the solution in the vicinity of arbitrarily complex immersed boundaries, the immersed boundary is discretized using

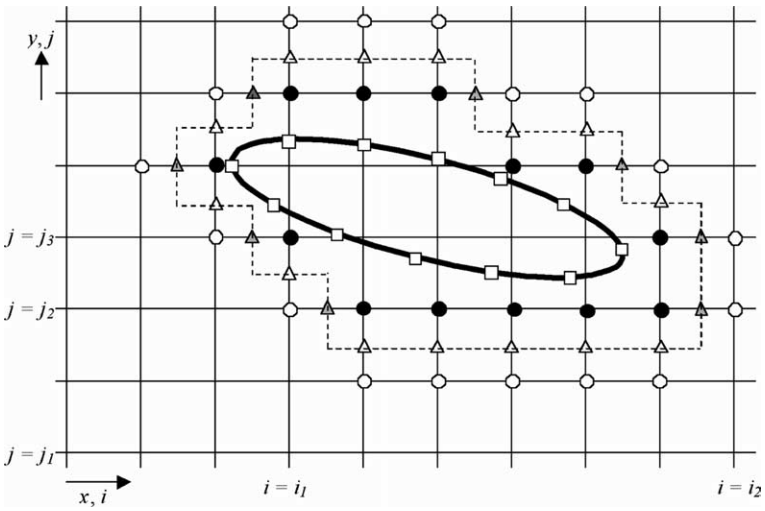

Fig. 2. 2D schematic of an immersed boundary in a Cartesian grid. Open squares mark the Lagrangian control points defining the immersed boundary. Filled circles indicate the IB nodes were the solution is reconstructed. Triangles mark the boundary nodes of the staggered grid layout (filled triangles: u velocity; open triangles: v velocity). Open circles indicate the last layer of nodes adjacent to IB nodes where the discrete continuity equation is satisfied.

an unstructured, triangular mesh with M triangular elements of size similar to the Cartesian grid spacing in the vicinity of the body. Before we proceed with a description of the algorithm, let us introduce the notation we use in this section. The position vector of the (i,j,k) Cartesian grid node ($i = 1$ to im, $j = 1$ to jm, and $k = 1$ to km) is $\mathbf{r}_{i,j,k}$, while the position vector of the centroid of the mth triangular element ($m = 1$ to M) on the interface is $\mathbf{r}_{m+1/2}$. Finally, $\mathbf{n}_{m+1/2}$ denotes the normal unit vector at the centroid of the mth triangular element, which can be readily computed after the interface mesh has been constructed.

3.1. Interface tracking

The first step in the implementation of the algorithm is to determine the location and shape of the immersed boundary. Since in this work we assume that the motion of the body is prescribed, this step is straightforward. The location of the body at every physical time step can be determined by solving Eq. (4) for each vertex of the unstructured, surface mesh. Assuming that the location of all vertices $(X_k, Y_k, Z_k)^n$ (for $k = 1$, K) at the n time step is known, their location at $n + 1$ can be determined via the following, second-order accurate integration scheme:

$$\mathbf{r}_k^{n+1} = \mathbf{r}_k^n + \mathbf{U}_k^{n+1/2}\Delta t, \tag{16}$$

where $\mathbf{U}_k$ is the prescribed velocity vector of the kth surface node.

3.2. Identification of immersed boundary nodes

Having established the shape and location of the interface, the next step is to identify the near-boundary nodes of the Cartesian collocated grid (''black'' nodes in Fig. 2) where the velocity vector needs to be reconstructed in order to obtain boundary conditions – we shall denote such nodes as immersed boundary (*IB*) nodes. In [38], we developed an algorithm for determining the *IB* nodes that is strictly applicable to geometrically simple, convex bodies – i.e., bodies that contain all lines connecting any two points on their surface. This algorithm needs to be generalized if the method is to be applied to simulate flows past complex aquatic animals, insects and birds, which in general consist of a main flexible body of concave shape with multiple flexible appendages, wings, fins, legs, etc., whose thickness is often too small to be resolved by the computational grid. Furthermore, since the location of the *IB* nodes changes at every time step any algorithm for determining them should be simple to implement and computationally very efficient. The algorithm described in the subsequent steps exhibits these features and can be applied to locate the *IB* nodes for bodies of arbitrary geometry.

First, we locate all Cartesian grid nodes that are in the immediate vicinity of the body. At this stage we do not distinguish between internal and external nodes to the body but rather seek to identify all *near-boundary* nodes located within a small search radius ds_0 from some region of the body. These nodes are marked with circles (open and filled) in Fig. 3. A Cartesian grid node will be designated as a *near-boundary* node if its position vector $\mathbf{r}_{nb}$ satisfies the following condition:

$$\min_{m=1,M} |\mathbf{r}_{nb} - \mathbf{r}_{m+1/2}| < ds_0, \tag{17}$$

where ds_0 is a prescribed search radius (see Fig. 3), which is set approximately equal to the minimum Cartesian mesh spacing in the vicinity of the body. After the above search has been completed a total of NB Cartesian grid nodes (i.e., $nb = 1$, NB) have been identified.

In the second step of the algorithm we separate the NB near-boundary nodes into nodes internal and external (*IB* nodes) to the body as follows. For every near-boundary Cartesian grid node nb, we first identify all triangular elements $\mathbf{r}_{m+1/2}$ located within a sphere of radius ds_0 centered at node nb (see Fig. 3) and for these triangular elements we examine the sign of the scalar product $\mathbf{n}_{m+1/2} \cdot (\mathbf{r}_{nb} - \mathbf{r}_{m+1/2})$.

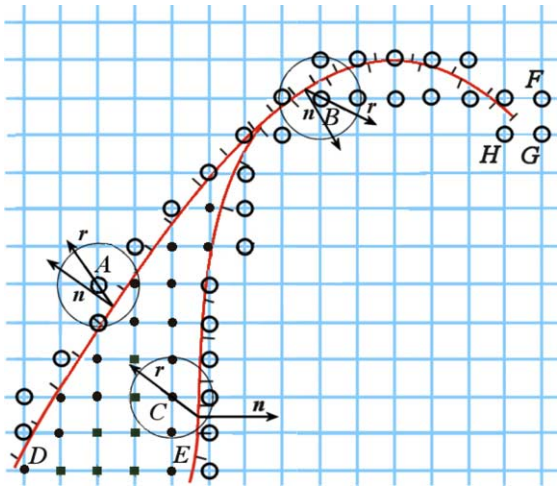

Fig. 3. Schematic illustrating the search algorithm for identifying the IB nodes for a complex immersed boundary. Circles (open and filled) are the near-boundary nodes that are identified in the first step of the search procedure. Open and filled circles mark the IB and internal near-boundary nodes, respectively, as classified in the second step of the algorithm. Square nodes are nodes internal to the body located at least one node away from the interface (i.e., internal, non-near-boundary nodes).

If $\mathbf{n}_{m+1/2} \cdot (\mathbf{r}_{nb} - \mathbf{r}_{m+1/2}) > 0$ *for at least one boundary node* $r_{m+1/2}$ *within the local search neighborhood*, then the Cartesian grid node $\mathbf{r}_{nb}$ is external to the body and, thus, an *IB*-node (see nodes *A* or *B* in Fig. 3).

If $\mathbf{n}_{m+1/2} \cdot (\mathbf{r}_{nb} - \mathbf{r}_{m+1/2}) < 0$ *for all triangular elements* $r_{m+1/2}$ *within the local search neighborhood*, then the Cartesian grid node $\mathbf{r}_{ns}$ is internal to the body (see node *C* in Fig. 3).

After all *near-boundary* Cartesian grid nodes have been classified as either *IB* or internal to the body, all other grid nodes interior to the body (marked with squares in Fig. 3) can be easily identified as those located along a grid line connecting two interior *near-boundary* nodes at opposite sides of the body (such as the nodes between nodes *D* and *E* in Fig. 3).

The above algorithm is general and, thus, applicable to arbitrarily complex bodies. Its capabilities are demonstrated in subsequent sections of this paper where it is successfully applied to simulate flow past an undulating fish-like body as well as flow past a reasonably realistic model of a planktonic copepod.

3.3. Reconstruction of the velocity field

After the *IB* nodes have been determined, boundary conditions need to be specified for the velocity field at all *IB* nodes at the new physical time level $n + 1$. Since the immersed boundary is tracked as a sharp interface, Dirichlet boundary conditions for the velocity field are known at all nodes of the unstructured surface mesh (Eq. (5). Let Φ_m^{n+1} denote any one of the three components of $\mathbf{U}_m$ on the immersed boundary and $\varphi^{n+1,\ell+1}$ denote the corresponding velocity component at a Cartesian grid node. With reference to Fig. 4, the boundary conditions for $\varphi^{n+1,\ell+1}$ at the *IB* node *b* are determined as follows.

According to the notation of this figure, *IB* node *b* is associated with the triangular element with normal $\mathbf{n}$ on the interface mesh. Since $\mathbf{n}^{n+1}$ is known, we can construct the line that passes through node *b* and is parallel to $\mathbf{n}^{n+1}$. This line intersects the surface element at point *a* (the projection of node *b* on the immersed boundary) and the Cartesian grid element defined by nodes α–β–γ–δ at point *c*. Since Φ^{n+1} is known at all vertices of the surface element from (4) to (6), Φ_a^{n+1} can be computed by linear interpolation among the vertices of the triangular surface element

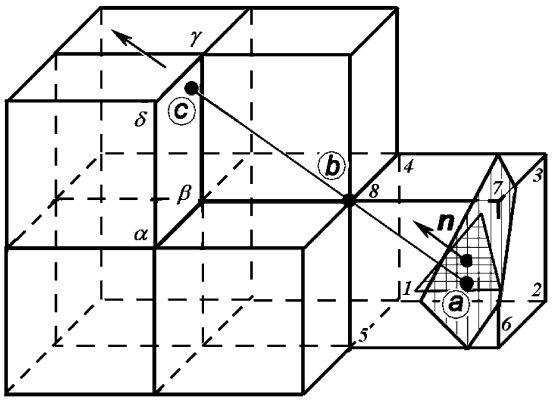

Fig. 4. Schematic depicting the reconstruction of the solution at an immersed boundary node b by interpolating along the local normal to the surface of the body.

$$\Phi_a^{n+1} = \left(\sum_{k=1,3} \Phi_k^{n+1} \Big/ s_k\right) \Big/ \left(\sum_{k=1,3} 1 \Big/ s_k\right), \tag{18}$$

where $1 \leqslant k \leqslant 3$ are the three vertices of the considered triangular element, and s_k is the distance between a and k-vertex

$$s_k = \sqrt{(x_a - x_k)^2 + (y_a - y_k)^2 + (z_a - z_k)^2}. \tag{19}$$

Similarly, $\varphi_c^{n+1,\ell+1}$ can be obtained by interpolating (using an interpolation formula similar to (18) among the internal Cartesian grid nodes α–β–γ–δ, where the solution is already known at the previous dual time step (quantities $(\,)^{n+1,\ell}$ are already known at all grid nodes).

With Φ_a^{n+1} and $\varphi_c^{n+1,\ell+1}$ known, the velocity components $\varphi_b^{n+1,\ell+1}$ at the *IB* node b can now be computed via interpolation. We employ the following quadratic interpolation procedure, which was found to enhance the overall accuracy of the algorithm in problems with moving boundaries relative to the linear interpolation procedure used in [38]. Let s be the arc-length variable that measures length along the normal from the surface node a (i.e., $s_a = 0$) The velocity component φ is assumed to vary in a quadratic manner with s as follows:

$$\varphi(s) = C_1 s^2 + C_2 s + C_3, \tag{20}$$

where C_1, C_2, and C_3 are coefficients to be determined. There is a total of four unknowns, the three coefficients C_i and $\varphi_b^{n+1,\ell+1}$, which can be fully determined by solving the following system of linear equations:

$$\begin{aligned}
&\varphi(0) = \Phi_a^{n+1} = C_3,\\
&\varphi(s_b) = \varphi_b^{n+1,\ell+1} = C_1 s_b^2 + C_2 s_b + C_3\\
&\varphi(s_c) = \varphi_c^{n+1,\ell+1} = C_1 s_c^2 + C_2 s_c + C_3\\
&(\mathrm{d}\varphi/\mathrm{d}s)_{s=s_b}^{n+1,\ell+1} = 2C_1 s_b + C_2.
\end{aligned} \tag{21}$$

To close the above system of equations, we calculate the derivative at $s = s_b$ via linear interpolation between its values at the mid-points of the segments ab and bc – where $\mathrm{d}\varphi/\mathrm{d}s$ can be calculated with second-order accurate central differencing – as follows:

$$\left(\frac{\mathrm{d}\varphi}{\mathrm{d}s}\right)_{s=s_b} = \alpha\frac{\varphi_b^{n+1,\ell+1} - \Phi_a^{n+1}}{\Delta s_{ab}} + (1-\alpha)\frac{\varphi_c^{n+1,\ell+1} - \varphi_b^{n+1,\ell+1}}{\Delta s_{bc}}, \tag{22}$$

where $\alpha = \Delta s_{bc}/\Delta s_{ac}$, and Δs is a distance between specified points. Eqs. (21) and (22) form a closed system of four unknowns and four equations, which can be solved to determine $\varphi_b^{n+1,\ell+1}$.

It is important to note that at some *IB* nodes located in regions where the curvature of the body is changing rapidly in space, the projection onto the surface of the body may not be uniquely defined or even exist (see nodes *F*, *G*, and *H* in the schematic shown in Fig. 3). At such nodes, boundary conditions for the velocity components are reconstructed by interpolating along the line defined by the *IB* node and the nearest node on the surface of the body.

In the above algorithm, the determination of the normal vectors to the body and the interpolation procedure for computing Φ_a^{n+1} for all *IB* nodes need to be carried out only once per physical time step. The velocity boundary conditions at the *IB* nodes, however, are enforced in an iterative manner and, thus, need to be updated during every dual time iteration. With boundary conditions for the velocity field at all *IB* nodes prescribed at the $\ell + 1$ dual time level, the iterative scheme described in the previous section can be applied to advance the solution at all interior nodes in dual time. During the iterative process, nodes interior to the immersed boundary are blanked out and do not participate in the calculation.

3.4. The problem with "freshly cleared" nodes

An important issue that requires careful consideration when applying a sharp-interface method to fluid/structure interaction problems arises when the motion of the immersed boundary relative to the fixed Cartesian grid exposes into the fluid a grid node, which at the previous time step was in the interior of the body [15]. To illustrate the problem, consider a grid node that was in the interior of the body at time level n but after the body is displaced during the $n + 1$ time level it emerges in the fluid. The numerical difficulty in this situation stems from the lack of physically realistic values for u^n and u^{n-1}, which are required for satisfying the momentum equations (10) at such a node. In the cut-cell formulation of Udaykumar et al. [15] this issue of *freshly cleared cells* was addressed using a cell-merging formulation in conjunction with quadratic interpolation among neighboring grid nodes in the fluid. Because of the dual-time stepping scheme we adopt in this work and the fact that the boundary conditions at *IB* nodes are enforced iteratively at the $n + 1$ time level, this issue does not pose any particular difficulty in our algorithm as long as the previously solid node first emerges into the fluid as an *IB* node (see Fig. 5). This condition can be easily enforced by selecting the

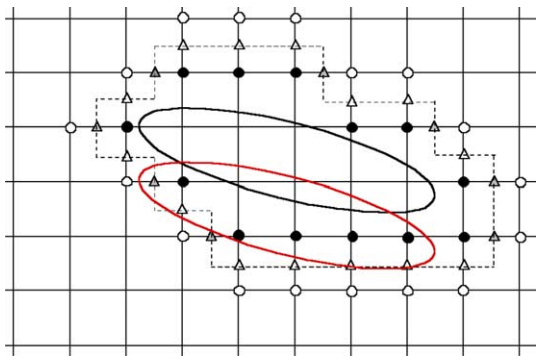

Fig. 5. 2D schematic of an immersed boundary moving relative to the fixed Cartesian grid. Black line: body location at time t. Red line: body location at time $t + \Delta t$. The classification of the various nodes is according to the notation introduced in Fig. 2 and is based on the location of the body at time t. (For interpretation of the references to color in this figure legend, the reader is referred to the web version of this article.)

physical time step to ensure that the body never transverses an entire computational cell within one time step. In other words, the following Courant-like condition needs to be satisfied for all n:

$$\Delta t \leqslant \frac{h}{\max\limits_{m=1,M}\left(\left|U_m^n\right|,\left|V_m^n\right|,\left|W_m^n\right|\right)},$$

where U, V, and W are the Cartesian components of the immersed boundary velocity vector $\mathbf{U}$, and h is the minimum grid spacing in the vicinity of the immersed boundary. Since in this work the motion of the immersed boundary is prescribed, whether the time increment satisfies this condition or not needs to be checked once at the start of the computation. It is important to note that in all subsequently discussed computations the above condition turns out to be less restrictive than the Courant condition for stability of the dual time-stepping algorithm and, thus, it does not impose any additional stability burden on the overall algorithm.

4. Results and discussion

4.1. Order of accuracy: flow due to an oscillating sphere in a cavity

To demonstrate the second-order accuracy of the method we carry out a grid refinement study for a test problem, which is the three-dimensional analogue of the problem used by Udaykumar et al. [15] for the convergence study of their method. A rigid sphere of diameter $D = 1$ is placed in a closed cubic box with side $H = 2$ filled with an incompressible, viscous fluid, which is initially at rest. Flow is induced by oscillating the sphere back and forth along the horizontal (x) direction. The motion is initiated impulsively at $t = 0$ and the location of the sphere is prescribed as follows:

$$x_k(t) = x_k^0 + x_0(1-\cos(2\pi t)), \quad y_k(t) = y_k^0, \quad z_k(t) = z_k^0 \qquad 1 \leqslant k \leqslant K, \tag{23}$$

where $x_0 = 0.125D$ is the amplitude of oscillation, and $\mathbf{r}_k^0$, $\mathbf{r}_k$ are the initial and current locations of the k-th node of the surface of sphere. The Reynolds number for this flow is based on the sphere diameter and the maximum sphere velocity and is set equal to $Re = 20$.

Four, uniformly spaced, successively finer mesh sizes are used for error analysis, with 20^3, 40^3, 80^3, and 160^3 grid points, respectively, and the finest mesh solution is considered to be the 'exact' solution. On all grids the same small physical time step ($\Delta t = 0.005$ was employed in order to emphasize the spatial resolution of the method, as was also done in [15]). For all grids, the simulation was initiated impulsively and was continued for one complete period. At the end of the period, the L_∞ and L_q norms of the error the u-velocity component are calculated as follows:

$$\varepsilon_N^\infty = \max_{i=1,N^3}\left|u_i^{(N)} - u_i^e\right|, \quad \varepsilon_N^q = \left[\frac{1}{N^3}\sum_{i=1}^{N^3}\left|u_i^{(N)} - u_i^e\right|^q\right]^{1/q}, \tag{24}$$

where ε_N^∞ and ε_N^q are the infinity and qth error norms, $u_i^{(N)}$ is the u-velocity component at the ith node of the N^3 mesh, and u^e is the 'exact' velocity field calculated on the 160^3 grid.

The results of the grid refinement study are summarized in Fig. 6(a) which shows the variation of the L_∞, L_1 and L_2 norms of the error with grid spacing in a log-log scale. The lines with slope one and two are also shown for reference. Fig. 6(b) also shows the instantaneous streamlines and pressure field at the moment when the error was calculated. It is evident from Fig. 6(a) that the method converges at a rate which is close to second order. A small deviation from strict second-order slope is observed only in the convergence of the L_∞ error norm but this should be attributed to the lack of a true exact solution of the governing equations –

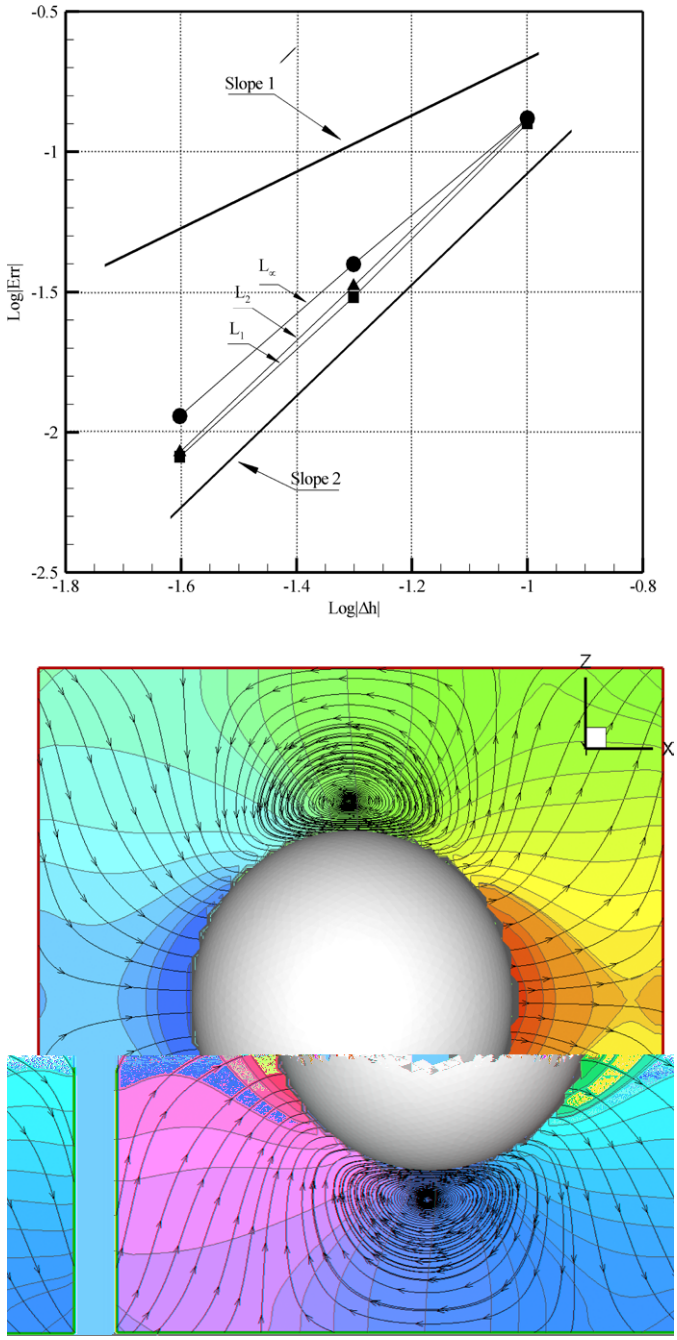

Fig. 6. Grid convergence study for flow in a closed cube filled with a horizontally oscillating sphere. (Top) Convergence of the L_∞, L_1 and L_2 norms of the error for the velocity field. The solution on the 160^3 grid is considered to be the exact solution. (Bottom) Instantaneous streamlines and pressure contours on the finest mesh. The slope of the L_∞ line is 1.77 while the slope for both the L_1 and L_2 lines is essentially equal to 2.

similar conclusion was also reached by Udaykumar et al. [15]. The results shown in this figure, however, make a strong case that our method is at the very least nearly second-order accurate.

To further demonstrate the accuracy of our method we also use the following Richardson-estimation procedure used by many authors to estimate the accuracy of a numerical solutions (see [55,56]). Let f^N denote the numerical solution on the N^3 mesh. Assume that the discrete solution is a γ-order approximation to its accurate value f^{exact}, i.e.

$$f^N = f^{\text{exact}} + C(h_N)^\gamma,$$

where C is a function not dependent on h_N is the uniform spacing of the N^3 mesh. Assuming that the flow changes gradually and it has no singularity points, it can be shown that

$$\gamma = \frac{\log\left(\left\|f^N - f^{N/2}\right\| / \left\|f^{N/2} - f^{N/4}\right\|\right)}{\log 2},$$

where $\|\,\|$ denotes an error norm (L_∞, L_1 or L_2). If $\gamma \simeq 2$ the solution is second-order accurate. We apply the above procedure for $N = 81$ (using solutions obtained on meshes 21^3, 41^3, and 81^3) and $N = 161$ (using solutions on meshes 41^3, 81^3, and 161^3), respectively, to calculate γ for successively refined meshes. For each case, we use all three norms to compute the error and the results are summarized in Table 1. Note that this error-estimation procedure is approximate and its output should be expected to asymptote to the rate of convergence of the numerical solution as the mesh is refined. In that sense, the results shown in Table 1 further reinforce those shown in Fig. 6(a) and support our assertion about the second-order accuracy of our method.

4.2. Validation test cases

4.2.1. Flow due to a steadily rotating sphere

For the first validation test case, we simulate flow induced by a sphere of radius R_0 rotating at constant angular velocity Ω about a diameter directed along the z-axis in an incompressible, viscous fluid of density ρ and kinematic viscosity ν, which is at rest at large distances from the sphere. The Reynolds number for this flow is defined as $Re = \Omega R_0^2/\nu$. For Reynolds numbers in the range $Re = 1$–100, benchmark solutions for this problem have been reported by Dennis et al. [40] who solved numerically the steady, axisymmetric Navier–Stokes equations in spherical, polar coordinates using a vorticity-streamfunction formulation. Even though within this range of Reynolds number this flow exhibits steady and axisymmetric solutions, we solve the full three-dimensional and unsteady problem with the sphere starting to rotate impulsively from rest relative to the stationary Cartesian grid. Such level of modeling allows us to validate the accuracy of the basic flow solver and gain some confidence about the correct implementation of the numerical infrastructure we have developed for handling flows in domains with moving immersed boundaries. A more stringent validation case of the moving-body aspects of our formulation, however, will be provided in the subsequent section.

Table 1
Rate of convergence γ calculated for different error norms and two sets of meshes

Norm	Grids	
	21^3, 41^3, 81^3	41^3, 81^3, 161^3
L_∞	1.57	1.78
L_1	2.72	2.32
L_2	2.08	1.99

The Cartesian computational domain is a $(10R_0)^3$ cube discretized with a 100^3 grid. The surface of the sphere is discretized with an unstructured, triangular mesh consisting of 7700 elements. The sphere is placed at the center of the cubical domain and starts to rotate impulsively at $t = 0$ with constant rotational velocity Ω about the z-axis (see Fig. 7). The Cartesian grid is uniform in all three directions within a $(2.5R_0)^3$ cube containing the sphere, with grid spacing $\Delta x = \Delta y = \Delta z = 0.03R_0$. Hyperbolic tangent stretching is used to stretch the grid in the remaining of the computational domain. As discussed above, all results have been obtained by integrating the 3D, unsteady Navier–Stokes in time from $t = 0$, when the sphere starts rotating from rest, until a steady-state and axisymmetric solution is achieved using 100 time steps per rotation period ($\Delta t = 0.01T$, where $T = 2\pi/\Omega$). At every physical time step, the sphere is displaced rotationally within the fixed Cartesian mesh by solving Eq. (4) to update the location of every Lagrangian control node on the surface. The velocity of the sphere at each Lagrangian node k is defined by prescribing the surface Cartesian components U, V, and W as follows:

$$U_k = -\Omega Y_k, \quad V_k = \Omega X_k \quad \text{and} \quad W_k = 0$$

which correspond to steady, counter-clockwise rotation of the sphere around the z-axis.

Calculations have been carried out for three Reynolds numbers, $Re = 20$, 50, and 100 on the same computational grid. Grid sensitivity studies showed that this grid resolution (100^3) is adequate for obtaining grid independent results. Characteristic-based, non-reflective boundary conditions [54] were applied at all six boundaries of the Cartesian domain. Sensitivity studies carried out to assess the potential effect of

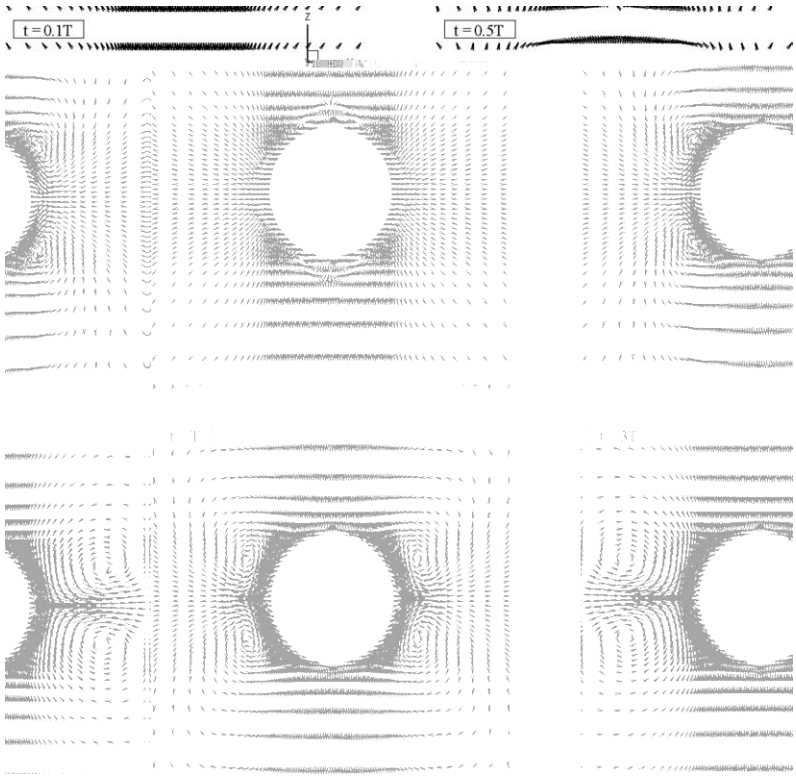

Fig. 7. Flow induced by a sphere rotating steadily relative to the fixed Cartesian grid. Instantaneous snapshots of velocity vectors at the $y = 0$ diametral plane depicting the early stages of the flow evolution toward steady state for $Re = 100$. Time is measured from the start of the impulsive rotational acceleration and T is the rotation period of the sphere.

the extent of the computational domain on the accuracy of the computed flow fields showed that $(10R_0)^3$ cube domain is the best compromise between accuracy and computational efficiency (see comments below).

Several snapshots of instantaneous velocity vectors at the $y = 0$ diametral plane for the $Re = 100$ case are shown in Fig. 7 to document the early stages of the evolution of the flow from the beginning impulsive

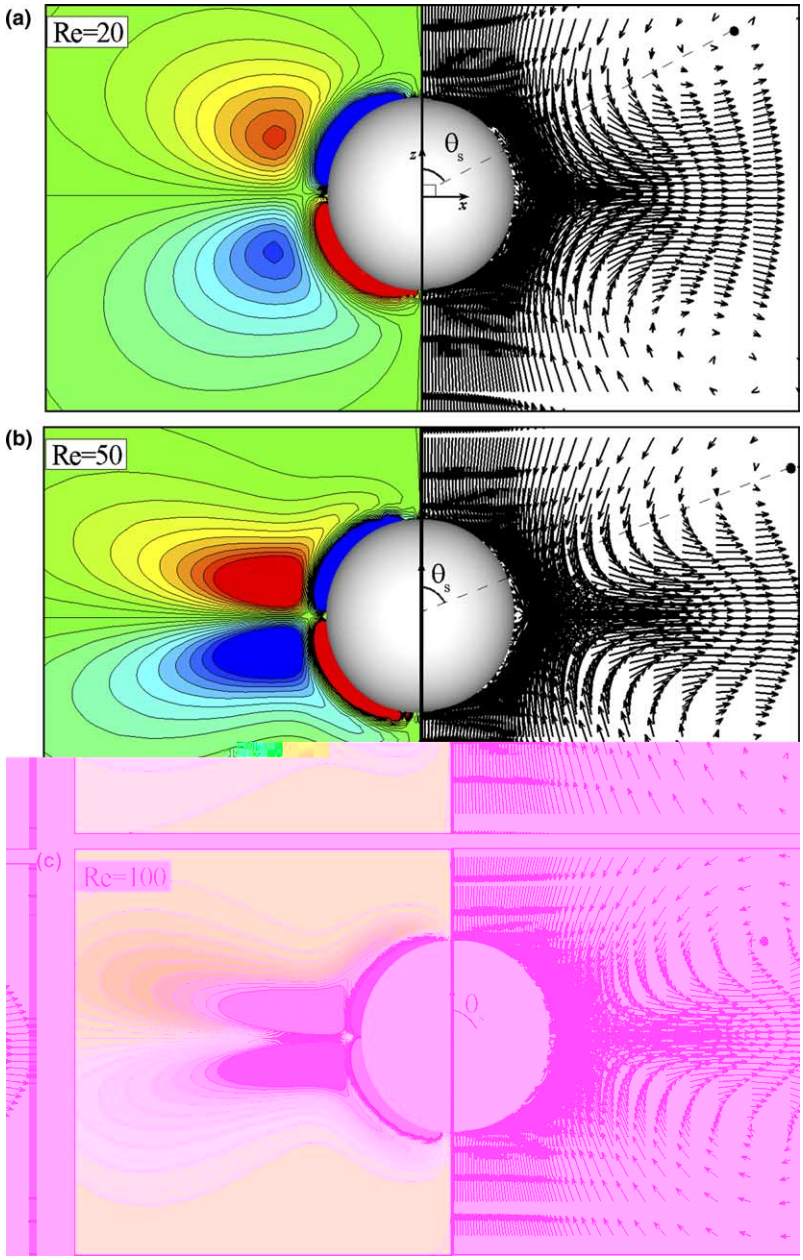

Fig. 8. Steady-state velocity vectors and ω_y vorticity contours at the $y = 0$ diametral plane for a steadily rotating sphere: (a) $Re = 20$; (b) $Re = 50$; and (c) $Re = 100$. Vorticity levels: from -0.3 to 0.3 by 0.03.

rotation of the sphere to the final axisymmetric steady state, which for this *Re* is shown in Fig. 8. The imbalance between the component of the centrifugal force in the tangential to the sphere direction and the pressure gradient, which is directed in the direction normal to the sphere, gives rise to a flow directed from the two poles toward the equator. The flow from the northern and southern hemispheres collide at the equator forming an equatorial jet that is directed radially outward. To satisfy conservation of mass and since the sphere rotates in an unbounded domain with the fluid at infinity being at rest, the streamlines have to close to form two toroidal vortex rings located symmetrically with respect to the equator. During early times the two rings form near the poles but subsequently are advected by the flow toward the equator where they begin to move away from the sphere in the radial direction. In qualitative agreement with earlier experimental [57] and numerical [40] studies, our computations show that the inflow to the poles is slower and spread over a larger region whereas the equatorial jet is narrow and very fast. It is also worth noting that the computed flow remains perfectly axisymmetric at all instants in time.

The computed steady state flow patterns at the *y*-plane of symmetry for all three Reynolds numbers are shown in Fig. 8. These results are in excellent agreement with those reported by Dennis et al. [40]. As the Reynolds number increases, the polar inflow region tends to become slower and spread over a larger area

Table 2
Comparison of the measured [40] and calculated angle θ_s

Re	θ_{Dennis}	θ_{calc}	ε (%)
20	62.6	61.6	1.6
50	69.4	68.2	1.7
100	73.8	72.1	2.3

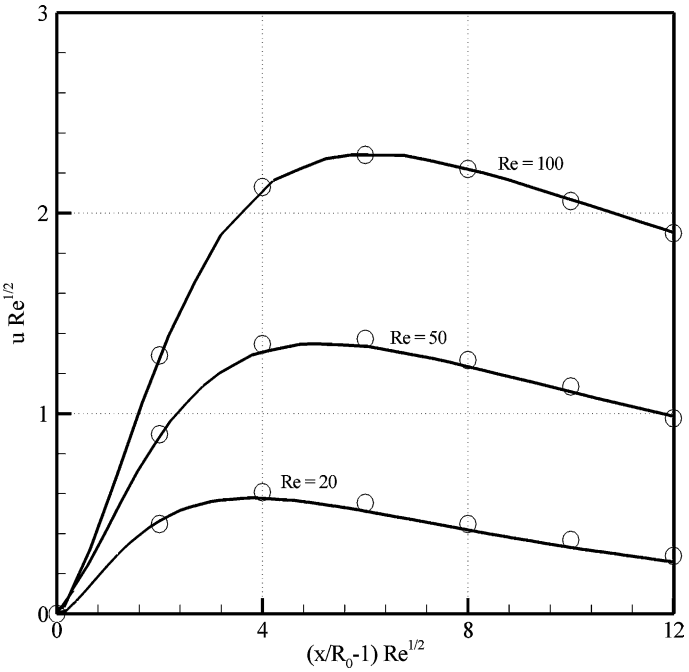

Fig. 9. Radial velocity profile along the equator for *Re* = 20,50, and 100. Lines: present computations; points: Benchmark data from [40]. The radius of the sphere is R_0.

while the equatorial jet becomes narrower and considerably faster. The extent of the polar inflow region is quantified in terms of the angle θ_s between the z-axis and the radius that passes through the center of the toroidal vortex ring (see Fig. 8 for definition). In Table 2, we compare our results for θ_s with those reported in [40] and the agreement is very good.

Additional evidence establishing the accuracy of our method is provided in Fig. 9, which compares our computations with the results of Dennis et al. [40] in terms of the radial velocity profiles along the equator for all three Reynolds numbers. The agreement between the two numerical solutions is very good.

4.2.2. Flow due to a flapping wing

This test case provides a far more stringent test of the fluid/structure interaction algorithm as it involves flow generated by a continuously flapping, three-dimensional wing. The experiments were carried out by Birch and Dickinson [41] who constructed a dynamically scaled robotic insect with wings cut in the shape of a *Drosophila* wing (see also [58,59]). The robot was placed in a rectangular tank filled with mineral oil and its motion was controlled by a set of digitally controlled motors that allowed inputting into the robot and testing a broad range of kinematical scenarios (see [41] for details).

The geometry of the wing was provided by Dickinson [60] and its 3D, plan and cross-sectional views are shown in Fig. 10. The cross-section of the wing is an ellipse with 12% thickness of the chord length. The surface of the wing is discretized with a triangular mesh with 1332 elements (Fig. 10).

We carry out Navier–Stokes calculations using the kinematical scenario considered in [41], which involves both translational and rotational motion of the wing. The configuration and positions of the wing at equal spaced instants in time for one half of the flapping cycle (up-stroke) are shown of Fig. 11. The coordinate system (x, y, z) is fixed to the wing with the z-axis normal to the stroke plane. The wing is flapped through an angle ψ of amplitude with an angle of attack α at mid-stroke. The motion is rather complex and

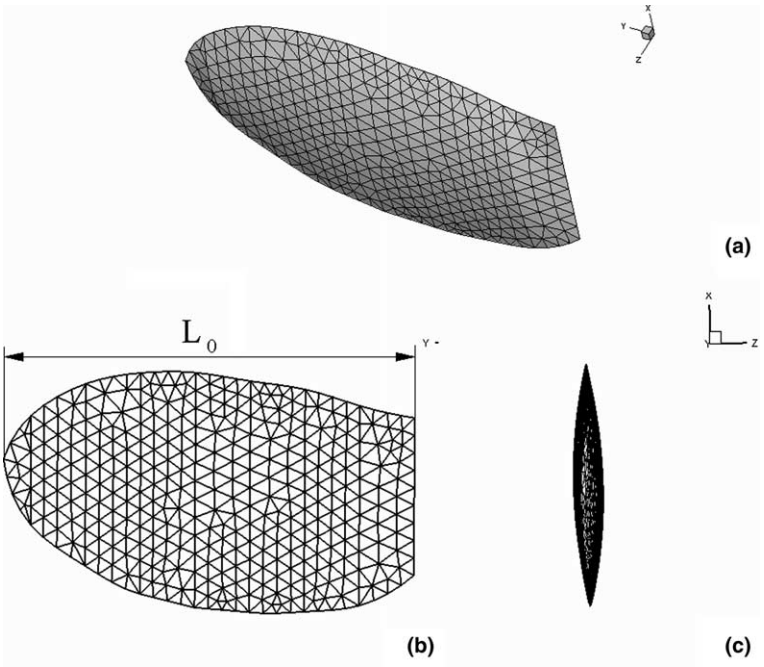

Fig. 10. The robotic insect wing of Birch and Dickinson [41]. Unstructured triangular surface mesh. (a) Three-dimensional view; (b) planar view; and (c) cross-sectional view.

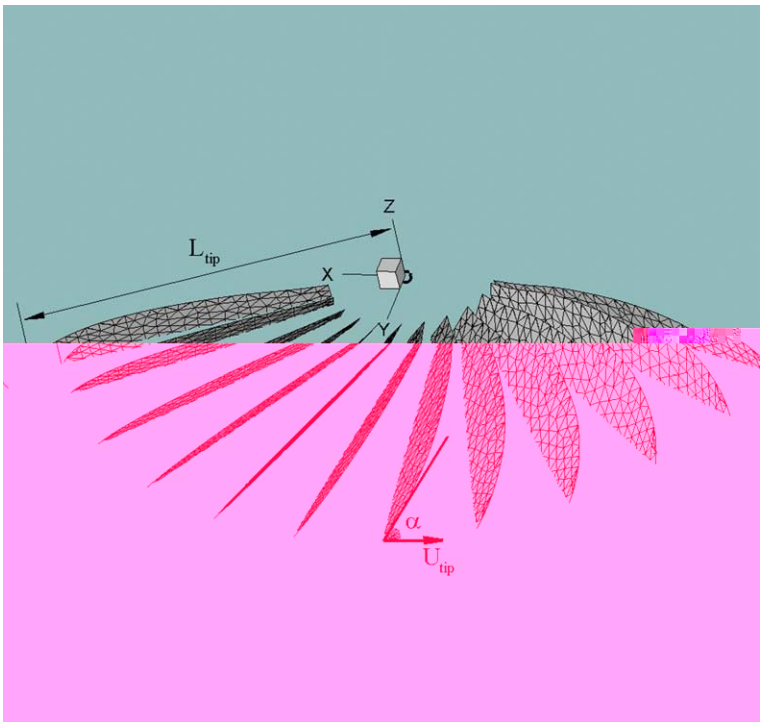

Fig. 11. The robotic insect wing of Birch and Dickinson [41]. Coordinate system and instantaneous location and orientation of the wing at equally spaced time intervals throughout the up-stroke.

one wing stroke can be divided into four stages: two translational phases, during upstroke (counter-clockwise rotation) and downstroke (clockwise rotation), respectively, when the wing undergoes azimuthal rotation around the z-axis with constant tip-velocity u_t and at fixed angle of attack α; and two rotational phases when the direction of azimuthal rotation is reversed while the wing simultaneously undergoes rotation along its long (spanwise) axis. With reference to Fig. 12, the following parameters are introduced in order to completely define the wing kinematics: T is the stroke period (upstroke plus downstroke); $\Delta\tau_r$ is duration of the rotational phase; $\Delta\tau_t$ is the duration of wing deceleration/acceleration; and τ_0 is the time when the rotational phase begins. All times are given as a percentage of the stroke period T.

During the deceleration at the end of the stroke and the acceleration at the beginning of the stroke, the wing tip velocity u_t is prescribed as

$$u_t^+ = u_0^+ \cos[\pi(\tau - \tau_1)/\Delta\tau_t], \quad \tau_1 \leqslant \tau \leqslant \tau_1 + \Delta\tau_t, \tag{25}$$

where $u_t^+ = u_t/U_0$, $u_0^+ = U_{tip}/U_0$, $\tau = t^*U_0/L_0$, where L_0 is the length of the wing (Fig. 10(b)), U_0 is the velocity at the point of the radius of the second moment of wing area that is determined by $r_0 = \sqrt{\oint_S r^2\,dS/S}$, t^* is the dimensional time, τ_1 is the time instant when the deceleration near the end of stroke starts (see Fig. 12) and $\tau_1 + \Delta\tau_t$ is the time at which the acceleration at the beginning of the next stroke ends. The angle of attack α is constant during upstroke and downstroke except at the near the beginning and the end of the stroke. As stroke reversal is approached, α changes with time such that the angular velocity $\dot{\alpha}$ is given by

$$\dot{\alpha}^+ = 0.5\dot{\alpha}_0^+\{1 - \cos[2\pi(\tau - \tau_2)/\Delta\tau_r]\}, \quad \tau_2 \leqslant \tau \leqslant \tau_2 + \Delta\tau_r, \tag{26}$$

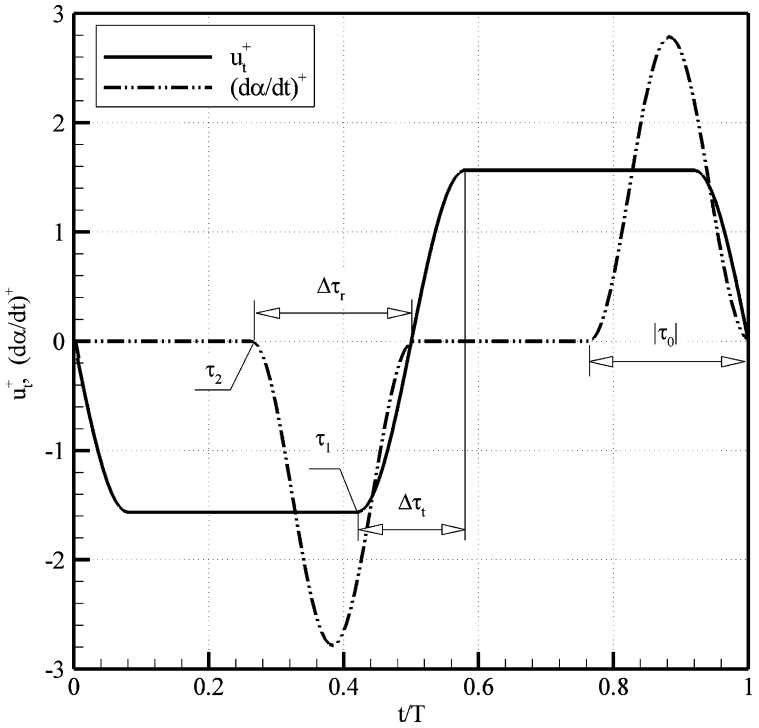

Fig. 12. The robotic insect wing of Birch and Dickinson [41]. Prescribed wing kinematics for a complete stroke. Solid line: Translational velocity. Dash line: rotational velocity.

where $\dot{\alpha}^+ = \dot{\alpha}L_0/U_0$, $\dot{\alpha}_0^+$ is a constant, $\tau_2 = \tau_1 + \Delta\tau_t/2 + \tau_0$ is the non-dimensional time at which the rotation starts, U_0 is the reference velocity, so $Re = L_0U_0/\nu$. We should mention that u_0^+ and $\dot{\alpha}_0^+$ has opposite sign during opposite stroke reversals.

We carry out simulations for the following set of parameters: $\psi = 160°$, $\alpha = 45°$, $\tau_0 = -0.24$ T, $\Delta\tau = 0.24$ T, $\Delta\tau_t = 0.16$ T, $L_0 = 0.19$ m, $U_0 = 0.166$ m/s, $U_{tip} = 0.26$ m/s. These parameters where extracted from the kinematics programmed into the robot shown in Fig. 11 of [41] paper and other data reported in that paper. At the bottom and top of the box non-reflecting boundary conditions were used, while non-slip boundary conditions were prescribed on all other boundaries, which were placed at the same distance from the wing as the walls of the tank used in the experiment of [41].

Calculations were carried out on two grids with 40^3 and 80^3 grid nodes, respectively. For both grids a physical time step of $\Delta t = 0.01$ was employed and the calculations were continued for four periods. The results reported herein are those obtained for the last stroke. Birch and Dickinson [41] reported measured time-series of lift and drag forces acting on the wing throughout its motion. To compare our simulations with their data we calculate the force acting on the wing by integrating the pressure and viscous stresses over the wing surface as follows:

$$F_i = \int_\Sigma [-p\delta_{ij} + \tau_{ij}]n_j \, d\Sigma, \tag{27}$$

where p is the pressure, τ_{ij} is component of the stress tensor, δ_{ij} is delta function, Σ is the wing surface. The calculated forces are compared with the measurements of [41] in Fig. 13. As seen in the figure, even the coarse mesh is adequate for capturing the essential features of the measured force histories with reasonable accuracy.

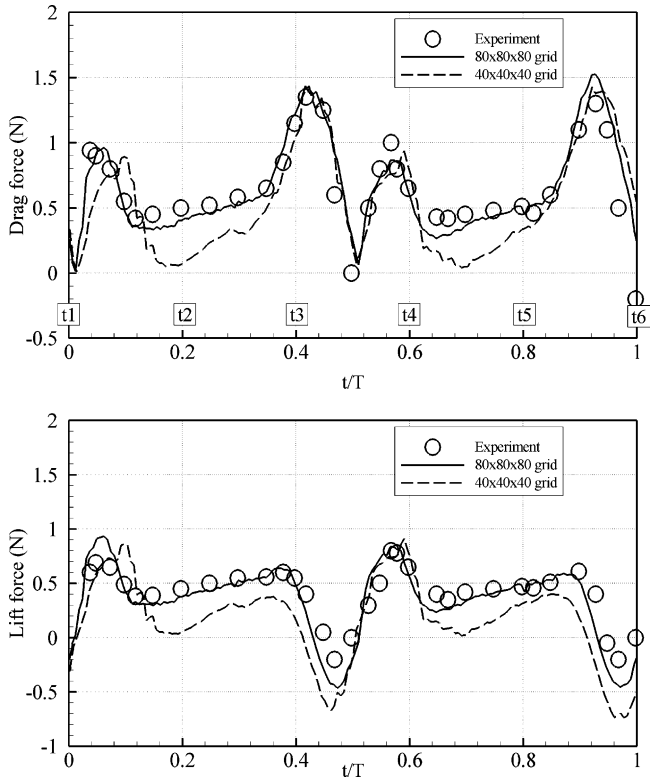

Fig. 13. The robotic insect wing of Birch and Dickinson [41]. Measured and computed drag (top) and lift (bottom) forces (in Newtons) for one complete stroke. Solid line: 80^3 mesh; dash line: 40^3 mesh; circles: measurements from [41].

The fine mesh prediction, however, is in very good agreement with the measurements both qualitatively and quantitatively. Further grid refinement did not change the results in any appreciable way. To illustrate the complexity of the flow induced by the flapping wing, Fig. 14 shows several snapshots of instantaneous streamlines at a z = const plane at various instants in time are marked in Fig. 13 during the wing stroke.

4.3. Simulation of flow past aquatic animals

In this section, we seek to demonstrate the applicability of the method to simulate flows induced by arbitrarily complex, flexible immersed boundaries and underscore its potential as a powerful simulation tool for biofluids problems. We consider two cases: (1) flow induced by a mackerel fish performing lateral undulations and (2) flow past a reasonably realistic model of a planktonic copepod, including the body and all its major legs and appendages, performing an escape-like maneuver. For both cases, the shapes of the bodies and their kinematics are based on biological observations and data reported in the literature.

4.3.1. Numerical simulation of fish-like swimming

We simulate the flow induced by a fish-like, flexible, three-dimensional object of length L moving in a straight line with constant axial velocity U and undulating in the lateral direction with a characteristic frequency f. Lengths are scaled with L, velocities are scaled with U, and time is scaled with f^{-1}. The coordinate system (x, y, z) is that fixed to the moving body and is related to the inertial coordinate system (X, Y, Z) as follows: $x = X + Ut$, $y = Y$, and $z = Z$.

The shape of the three-dimensional body is based on measurements obtained from a real mackerel fish. The fish was frozen and sliced in several fillets. The cross-sectional dimensions of each fillet were carefully measured and the cross-sections were then stacked together along the fish main axis to construct the

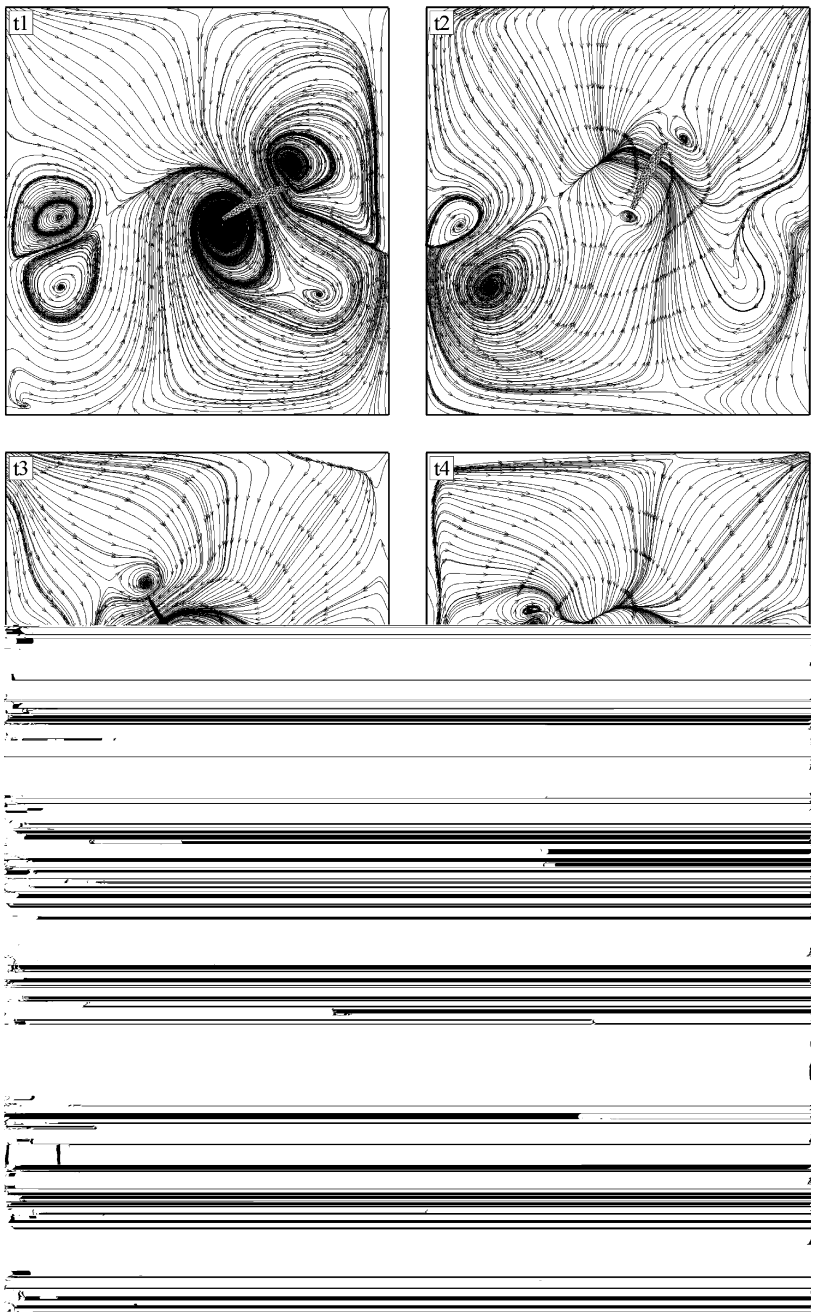

Fig. 14. The robotic insect wing of Birch and Dickinson [41]. Instantaneous streamlines at the z=constant mid-plane for one complete stroke. Specific time instants are defined in Fig. 13.

three-dimensional body. With the exception of the caudal fin (fish tail) all other fins are not modeled. The body surface is discretized with 6670 triangular elements and is shown in Fig. 15.

Fish-like swimming motion is prescribed by specifying the lateral displacement of the fish backbone as a function of time. We consider biologically inspired kinematics mimicking body and caudal fin (BFC) locomotion, which is the most frequently encountered swimming mode in fishes [61]. In the BFC mode fish swim by bending their body into a backward-traveling undulatory wave that extends all the way to their caudal fin. To mimic such motion, we prescribe the lateral displacement of the fish body in terms of a traveling wave of varying amplitude [62] as follows:

$$y(x,t) = a(x)\sin(kx - \omega t), \tag{28}$$

where $a(x)$ is the wave amplitude that is assumed to vary non-linearly along the fish body, the $k = 2\pi/\lambda$ is the wave number, corresponding to wave length λ, and the circular frequency of oscillation ω. The wave amplitude is a quadratic function of x

$$a(x) = a_0 + a_1 x + a_2 x^2, \tag{29}$$

where the constants $a_0 = 0.02$, $a_1 = -0.08$, and $a_2 = 0.16$ are selected to ensure that $a(x)$ becomes maximum at the tail. The precise form of the function $a(x)$ is shown in Fig. 16 and is taken from [2].

The following non-dimensional numbers are important in fish-like locomotion: (1) the Reynolds number, which is typically based on the fish length and swimming speed, $Re = UL/\nu$, (2) the Strouhal number,

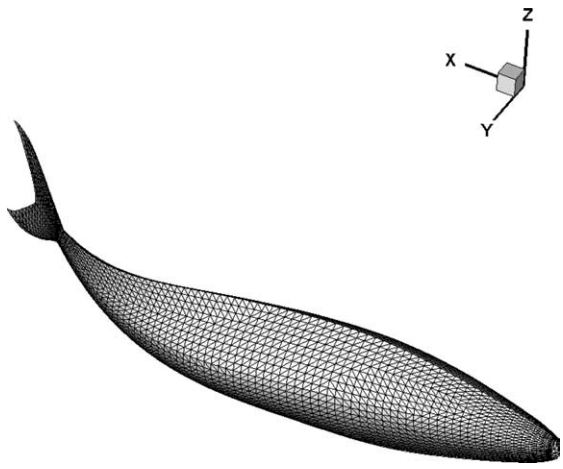

Fig. 15. Three dimensional view of mackerel body.

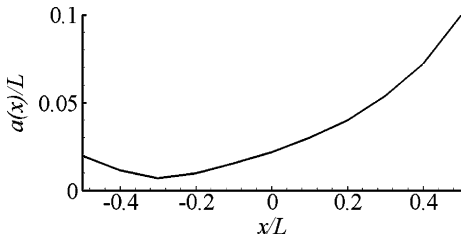

Fig. 16. Prescribed variation of the amplitude of the undulatory motion $a(x)$ (see Eq. (29)) with distance along the axis of the fish-like body. From [2].

which is based on the mean lateral excursion of the caudal fin at the trailing edge $A = 2a_{\max}$, where $a_{\max} = a(0.5)$, and the tailbeat frequency $f = \omega/2\pi$: $St = 2fa_{\max}/U = a_{\max}\omega/(\pi U)$; and (3) the slip ratio, which is the ratio of the swimming speed to the speed at which the undulatory wave travels down the body ($V = \omega/k$), Slip $= U/V = U/(\omega/k)$.

Laboratory experiments with waving flat plates [63] and actively swimming fish [64] have shown that the structure of fish wakes depends critically on the velocity slip. For slip greater than one, the wake is a classical Karman street consisting of two rows of vortices with their jet-like common flow directed toward the fish, thus, resulting in a net drag force. For slip near unity the wake consists of a single row of vortices with laterally oriented jets and zero axial force. Finally, for slip less than one, i.e., when the body wave travels faster than the flow, a reverse Karman street develops in the wake with two rows of vortices rotating in such a manner that their common-flow jet is directed away from the fish. Reverse Karman street results in a thrust force and is the primary mechanism via which fish produce propulsive force using their caudal fin (Triantafyllou and co-workers [62], Müller et al. [65]). All these experiments have been performed at Reynolds numbers high enough ($Re > 10^4$) for viscous effects not to be the primary consideration – Strouhal number and slip being the dominant parameters. For that reason, essentially all aspects of fish-like locomotion have been well reproduced by the potential flow simulations of Triantafyllou and co-workers [62,66].

To demonstrate the capability of our method to reproduce the effect of slip on the wake structure of an undulating fish-like body, we carry out a series of inviscid simulations for various Slip velocities. The viscous terms in the governing equations are set equal to zero, only the no-flux boundary condition is imposed on the fish body while the tangential velocity component on the fish body is calculated by linear extrapolation from interior nodes. All other features of the numerical method remain the same. The Cartesian computational domain is a $7L \times 2L \times L$ cube and is discretized with a 100^3 grid. The fish is placed $2L$ downstream from the inlet plane in the axial direction and is centered in the vertical and transverse directions. Uniform grid with spacing $h = 0.026$ is used in a $1.5L \times 1.5L \times 0.5L$ box, which encloses the fish at all times, and stretched grid is used in the remaining domain. Fig. 17 shows calculated instantaneous streamlines at the same instant within the period for three slip ratios. In qualitative agreement with the experimental observations, the simulated wake is a classical (drag) Karman street for Slip = 1.1 and consists of a single row of vortices for Slip = 1.0. For Slip = 0.6, however, a reverse Karman street emerges with a thrust component. To quantify the effect of the Slip parameter on the fish wake structure, we have calculated the thrust coefficient C_T for all three simulated cases. The thrust coefficient is defined as follows:

$$C_T = \frac{F_x}{0.5\rho U^2 S},$$

where F_x is the mean axial force, which is calculated by integrating the calculated pressure field over the fish surface and S is the wetted area of the fish. The calculated values of the thrust coefficient are $C_T = -6 \times 10^{-3}$, 1.5×10^{-3}, and 4×10^{-3} for Slip = 1.1, 1.0, and 0.6, respectively. The negative sign for the Slip = 1.1case indicates a net drag force on the body, which is consistent with the structure of the wake shown in Fig. 17.

Reverse Karman street for Slip < 1 can also be reproduced numerically with a full viscous simulation. Such undertaking, however, is far more challenging computationally as it requires very fine meshes in the vicinity of the body to accurately resolve the body boundary layer. Fig. 18 shows the results of a viscous simulation for $Re = 3000$, Slip = 0.6, and $St = 0.5$. Calculations have been carried out on two meshes, the coarse, ''inviscid'' mesh used in the previous calculations and a much finer mesh with more than 3.0 million nodes ($210 \times 120 \times 120$). For the fine mesh the grid spacing in the uniform-grid box surrounding the body is $h = 0.008L$. Fig. 18 compares the coarse and fine mesh flowfields at the same instant in the period in terms of instantaneous streamlines and ω_z contours. The calculated streamline patterns on both meshes are very similar in the vicinity of the body but significant quantitative differences are noticeable in the predicted

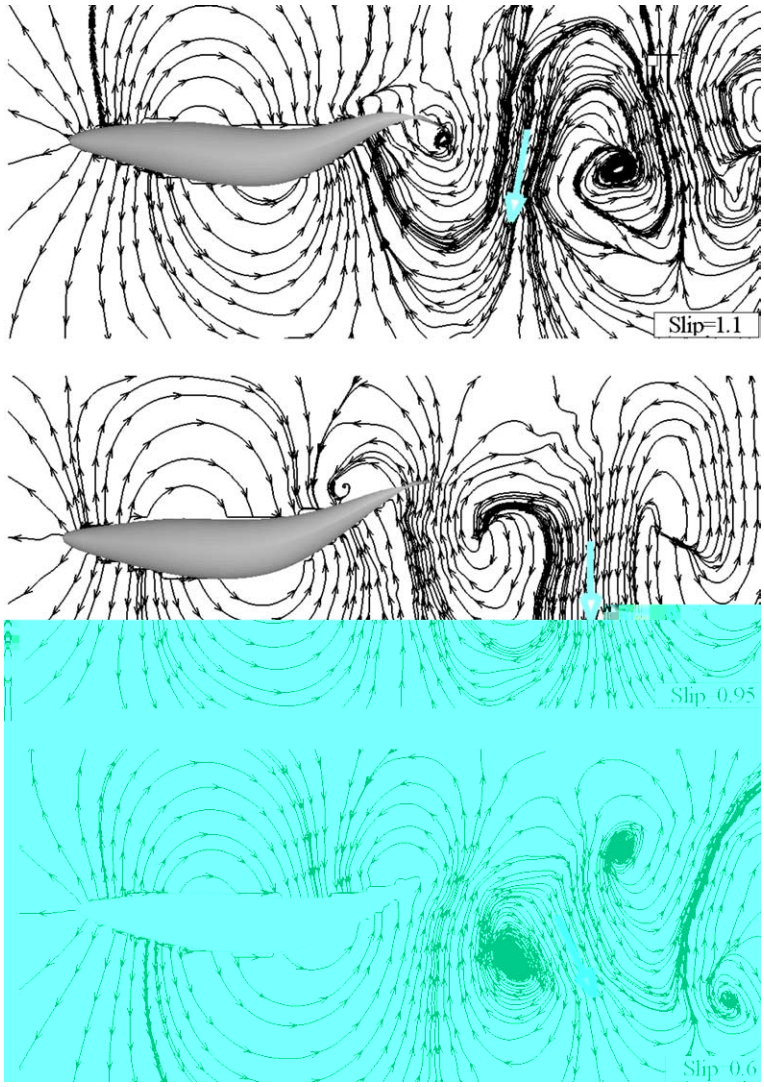

Fig. 17. Inviscid flow past an undulating mackerel. Instantaneous streamlines for various slip ratios. (Top) Slip = 1.1; (middle) Slip = 1.0; (bottom) Slip = 0.6. Arrow indicates the general direction of the wake flow.

vorticity field. On the coarse mesh the fish boundary layers are thicker, more diffuse, and produce much less vorticity than on the fine mesh. The quantitative difference between the two predictions is particularly evident in the immediate vicinity of the tail where much stronger vorticity is calculated on the fine mesh. The significant quantitative differences in the resolution of the near-body boundary layer are seen to result in quantitatively different wake structures. On the coarse mesh the wake is a classical (drag) Karman street while on the fine mesh the undulatory body motion is able to pump fluid away from the fish and produce thrust.

The results presented in this section demonstrate the capability of the method to capture well known physics of fish wakes and in particular the very important dependence of the wake structure on the slip parameter. The results also underscore the difficulties in performing simulations with a Cartesian method

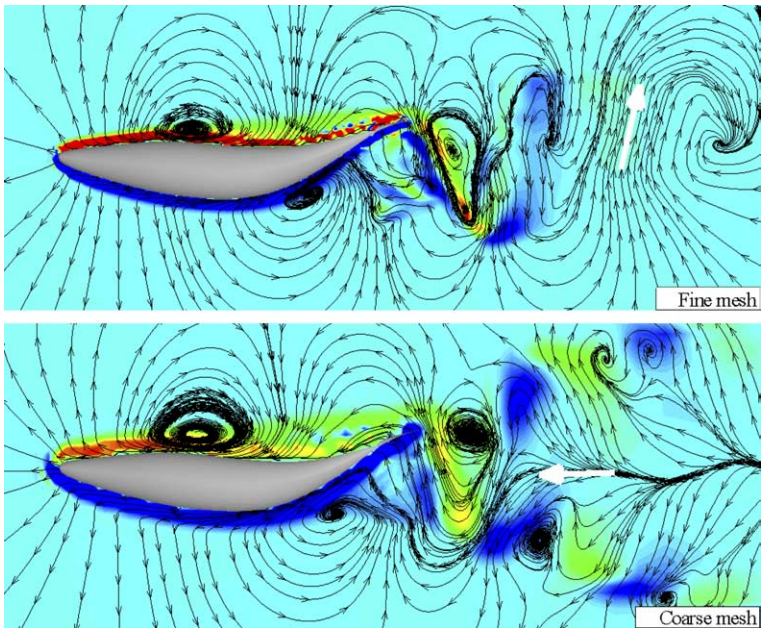

Fig. 18. Viscous flow past an undulating mackerel for $Re = 3000$ and Slip = 0.6. Instantaneous streamlines and ω_z vorticity contours. (Top) Fine mesh ($210 \times 120 \times 120$) solution showing reverse Karman street in the wake; (bottom) coarse mesh (100^3). Arrow indicates the general direction of the wake flow.

at intermediate (or transitional) Reynolds numbers – where the term transitional refers here to the transition from the viscous to the inertial flow regimes.

4.3.2. Flow past a planktonic copepod

Planktonic micro-crustaceans, such as copepods, are among the most abundant animals in the planet. They constitute a major source of food for fish while their feeding habits play an important role in controlling phytoplankton population growth. Thus, planktonic micro-crustaceans play a critical role in the balance of the oceanic ecosystem and for that reason they are being studied intensely by biologists. Much of the recent work with copepods has focused on visualizing and quantifying the flow patterns they generate as they move through water in order to characterize the hydrodynamic signatures they generate (Malkiel et al. [67], Yen and Strickland [68]). This is because it has long been hypothesized that copepods are able to distinguish an attractive mate from a lunging predator by sensing their respective hydrodynamic signatures in the form of coherent vortical structures. Due to their very small size (of the order of 1 mm), copepods typically operate in a low Reynolds number environment, with Reynolds numbers based on their length and swimming speed ranging from order 10^0 to 10^2. Yet they can generate very complex flowfields because of their intricate anatomy, which is dominated by multiple moving appendages (antennules, legs, tail, etc.) – see Fig. 19. Their complex geometry makes the numerical simulation of the flows they induce particularly challenging. To date the only attempt to simulate such flows using a quasi-steady CFD approach has been reported by Jiang et al. [69,70] who considered a simplified copepod model and collectively accounted for most of its multiple appendages by imposing a force field in the vicinity of the copepod body. In this section, we report a small sample of results from the first attempt to simulate copepod flows from first principles. That is, by modeling individual appendages, prescribing their kinematics from biological observations, and simulating the complex flowfield the combined motion of these appendages generate.

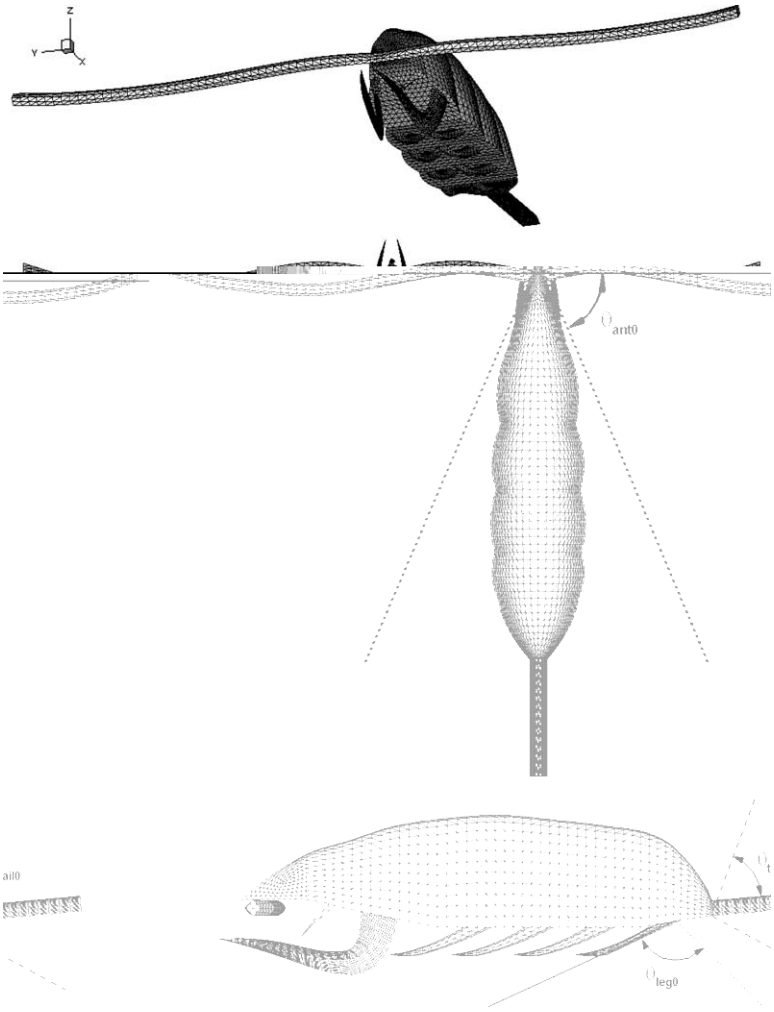

Fig. 19. Viscous flow past a planktonic copepod. Three-dimensional views of the unstructured surface mesh along with definitions of the various angles used to define the kinematics of the prescribed motion. Appendages that move are the long, horizontally oriented antennules at the front of the body (see middle figure), the four pairs of legs (see bottom side view) and the tail (see bottom figure). The lines indicate the envelope of motion for each appendage.

The copepod model we simulate is shown in Fig. 19, which depicts various three-dimensional views of the unstructured surface mesh. The model includes the two long antennules (the defining feature of the copepod anatomy), the two front maxillae, the four pairs of legs and the tail (uresome) and has been constructed using input from biological observations and high-resolution digital photography [71]. We carry out viscous flow calculations for a set of kinematics, which corresponds to what biologist refer to as an escape maneuver typically performed when a copepod is attempting to escape a predator. During this maneuver the animal attempts to maximize hydrodynamic thrust and deploys sequentially all of its major appendages, first the two antennules, followed by the uresome, and then the legs, with the rear pair being deployed first. Maximum angular amplitudes of antennules $\theta_{ant0} = 75°$, of tail $\theta_{tail0} = 80°$, and of all pairs of legs $\theta_{leg0} = 90°$ are shown in Fig. 19. Fig. 20 shows the kinematics of the appendages as prescribed in our

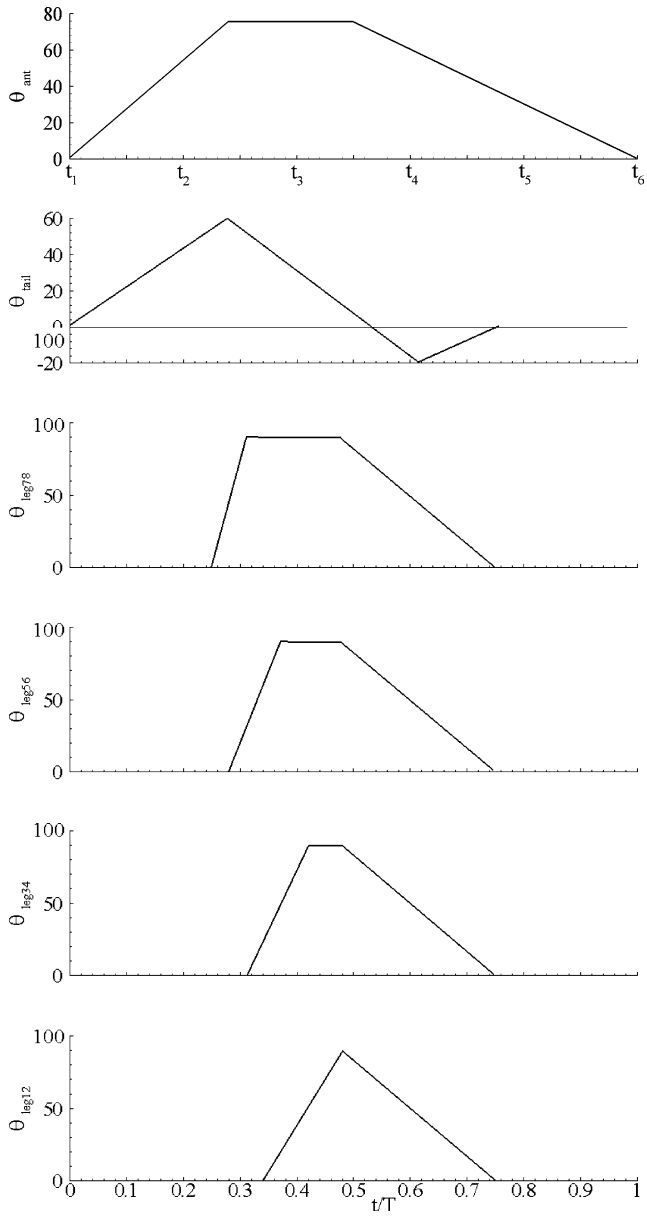

Fig. 20. Viscous flow past a planktonic copepod. Prescribed kinematics for the antennules, legs, and tail. The leg pairs are marked as 12, 34, 56, and 78 from the front to the rear of the animal, respectively. For definitions of the various angles see Fig. 20.

numerical simulation in terms of the temporal evolution of the various angles of rotation θ_{ant}, θ_{tail}, θ_{leg12} (front legs), θ_{leg34}, θ_{leg56}, θ_{leg78} (rear legs). The details of the kinematics is based on biological observations of live animals [71].

The surface mesh was discretized with 11432 triangles and simulations were carried out in a Cartesian grid with 100^3 nodes. The computational domain a $6L \times 6L \times 4L$ box with the body placed exactly at its center (where L is the streamwise length of the body). Uniform grid spacing with $h = 0.05L$ is used within

a region $2L \times 2L \times 1L$ centered around the body. The fluid in the computational domain is initially stagnant and the copepod begins to deploy its appendages impulsively at $t = 0$. The copepod is not allowed to move and, thus, the simulation corresponds to the motion of a tethered animal, as frequently done in laboratory experiments. Non-reflective boundary conditions were prescribed at all boundaries of the computational domain. Fig. 21 shows instantaneous streamlines and pressure contours at the vertical plane of symmetry of the animal at various instants in time. For reference, the body shape with the current location of the various appendages is also included in the figure. Even for this very complex body shape and motion, the numerical method had no difficulty obtaining converged solutions at each time step for several simulated periods of motion.

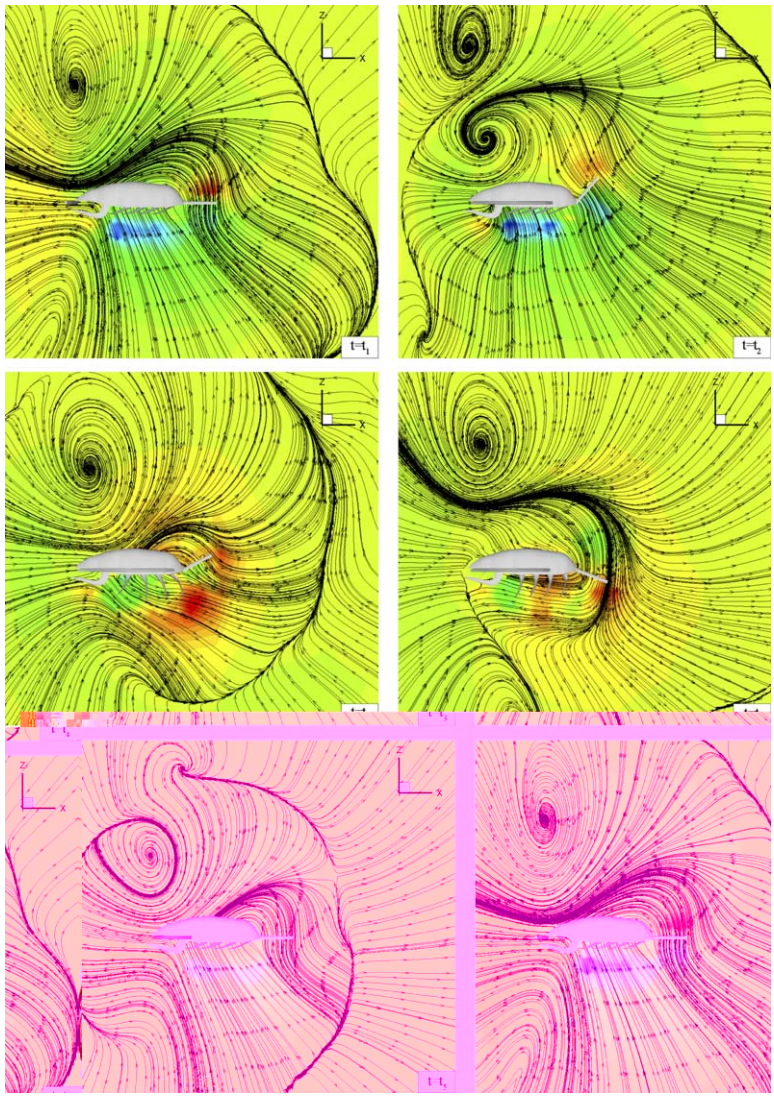

Fig. 21. Viscous flow past a planktonic copepod. Instantaneous streamlines and pressure contours at the vertical plane of symmetry of the copepod. The various instants in time, t_1 to t_6, are marked in Fig. 20. For reference, the instantaneous location of the animal appendages is also shown in each figure.

The periodic deployment of the various appendages in the manner described in Fig. 20 results in a very complex, albeit periodic flowfield as indicated in the sequence of images shown in Fig. 21. At the start of the period (Fig. 21(t_1)), the flow pattern around the body is dominated by the clockwise eddy at the upper half of the flow, which was created at the end of the period – compared Fig. 21(t_1) and (t_6). This eddy causes the flow in the vicinity of the body to be directed upstream and generates a drag force. As the antennules begin to push fluid backward, like two long paddles, a counter-clockwise, start-up eddy is generated near the body, which pushes the clockwise eddy away and begins to generate a pocket of downstream-directed thrust flow at the front of the animal (see Fig. 21(t_2)). The thrust-jet is further enhanced as the tail and legs are deployed (see Fig. 21(t_3) and (t_4)). As the various appendages begin to return to their initial positions, however, the thrust flow is pushed away from the body by a clockwise rotating eddy in the front part of the body, which reverses the thrust jet into a drag flow. The formation of the clockwise eddy has been completed at the end of the period when all appendages have returned to their initial positions (Fig. 21(t_6)). Note that the results shown in this figure suggest that while escaping the copepod produces bursts of thrust with each upstroke, which would propel the animal forward and possibly allow it to coast because of its inertia during the drag-inducing downstroke. Detailed simulations of copepod flows aimed at quantifying the effect of body kinematics and Reynolds numbers on the flow patterns and locomotive forces are currently under way and the results will be reported in a future communication.

5. Summary and conclusion

We have developed a novel hybrid Cartesian/immersed boundary method for simulating three-dimensional flows in domains with arbitrarily complex, flexible immersed boundaries moving with prescribed motion. The immersed boundary is treated as a sharp interface and the solution in its vicinity is reconstructed using interpolation along the local normal to the body. To facilitate the calculation of the normal, we discretize the body with an unstructured, triangular grid. The use of unstructured grid greatly enhances the generality of the method as it allows the modeling of arbitrarily complex, three-dimensional immersed boundaries and eliminates ambiguities encountered when the reconstruction is carried out by interpolating along grid lines. We also developed a hybrid staggered/non-staggered discretization approach for discretizing the governing equations. The hybrid approach retains all desirable features of the pure staggered grid arrangement but greatly simplifies the discretization of the equations and the implementation of boundary conditions near flexible immersed boundaries.

A grid convergence study was carried out, which showed that the method is second-order accurate. Validation studies were also reported for flow induced by a sphere rotating steadily in a fluid that is at rest sufficiently far from the sphere as well as for flow induced by a three-dimensional flapping wing in a confined domain. For both cases good agreement with benchmark solutions and laboratory measurements was obtained.

To demonstrate the applicability of the method to unsteady flows with flexible, three-dimensional immersed boundaries, we simulated flow past an undulating fish-like body and flow past a reasonably realistic model of a planktonic copepod. For the fish problem, our simulations captured the well known dependence of the wake structure on the slip velocity, yielding wakes with reverse Karman street for slip ratios less than unity. The copepod simulations demonstrated the ability of the method to simulate flow induced by bodies with multiple moving appendages. To the best of our knowledge this is the first time that a sharp-interface method has been successfully applied to simulate flows past 3D moving bodies of such geometrical intricacy. Therefore, our work underscores the promise of our method as a powerful numerical simulation tool for a broad range of biofluids applications

So far we have applied the method to flows in the low to moderate Reynolds number regime. Extension to higher Reynolds numbers hinges on the main limitation of all methods of this type, namely the need for

very fine grid resolutions in the vicinity of the immersed boundary. This difficulty can be addressed by combining the present method with local grid embedding and adaptive grid refinement techniques (see, for example, the recent work by [18,72,73]). Our interface tracking and solution reconstruction techniques are, at least in principle, applicable in conjunction with such resolution enhancing strategies but this important issue will be left as topic for future research.

Acknowledgments

This work was supported by NSF Career Grant 98-75691, by NIH Grant RO1-HL-07262, and by a grant from the Energy Efficiency and Renewable Energy Office of the US Department of Energy, Wind and Hydropower Technologies Office. We are grateful to Jeanette Yen for helping us with the construction of the copepod numerical model and for many helpful discussions. We also thank Michael Dickinson for providing us with the geometry of the robotic fly wing.

References

[1] C.W. Hirt, A.A. Amsden, J.L. Cook, An arbitrary Lagrangian–Eulerian computing method for all flow speeds, J. Comput. Phys. 14 (3) (1974) 227–253.
[2] H. Liu, K. Kawachi, A numerical study of undulatory swimming, J. Comput. Phys. 155 (2) (1999) 223–247.
[3] R. Ramamurti, W.C. Sandberg, A three-dimensional computational study of the aerodynamic mechanisms of insect flight, J. Exp. Biol. (2002) 1507–1518.
[4] R. Ramamurti, W.C. Sandberg, R. Löhner, J.A. Walker, M.W. Westneat, Fluid dynamics of flapping aquatic flight in the bird wrasse: thre-dimensional unsteady computations with fin deformation, J. Exp. Biol. 205 (2002) 2997–3008.
[5] R. Scardovelli, S. Zaleski, Direct numerical simulation of free-surface and interfacial flow, Annu. Rev. Fluid Mech. 31 (567–603) (1999).
[6] J. Glimm, O.M. Bryan, R. Menikoff, D. Sharp, Front tracking applied to Rayleigh–Taylor instability, SIAM J. Sci. Stat. Comput. 7 (1986) 230–251.
[7] S.O. Unverdi, G. Tryggvason, A front-tracking method for viscous, incompressible, multi-fluid flows, J. Comput. Phys. 100 (1992) 25–37.
[8] H.S. Udaykumar, H.-C. Kan, W. Shyy, R. Tran-Son-Tay, Multiphase dynamics in arbitrary geometries on fixed Cartesian grids, J. Comput. Phys. 137 (366–405) (1997).
[9] W.F. Noh, CEL: a time-dependent, two-space-dimensional, Coupled Eulerian–Lagrange Code, Methods in Computational Physics, Academic Press, New York, 1964, pp. 117–179.
[10] D. De Zeeuw, K.G. Powell, An adaptively refined Cartesian mesh solver for the Euler equations. 91-1542-cp, AIAA Paper, 1991.
[11] S.A. Bayyuk, K.G. Powell, B. van Leer. A simulation technique for 2-D unsteady inviscid flows around arbitrary moving and deforming bodies of arbitrary geometry. AIAA Paper 93-3391-CP, 1993.
[12] J.J. Quirk, An alternative to unstructured grids for computing gas dynamics flows around arbitrarily complex two-dimensional bodies, Comput. Fluids 23 (1994) 125–142.
[13] 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.
[14] 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.
[15] 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.
[16] A.S. Almgren, J.B. Bell, P. Colella, T. Marthaler, A Cartesian grid projection method for the incompressible Euler equations in complex geometries, SIAM J. Sci. Comput. 18 (5) (1997).
[17] H. Johansen, P. Colella, A Cartesian grid embedded boundary method for Poisson's equation on irregular domains, J. Comput. Phys. 147 (1998) 60–85.
[18] S. Popinet, Gerris: a tree-based adaptive solver for the incompressible Euler equations in complex geometries, J. Comput. Phys. 190 (2) (2003) 572–600.
[19] M.P. Kirkpatrick, S.W. Armfield, J.H. Kent, A representation of curved boundaries for the solution of the Navier–Stokes equations on a staggered three-dimensional Cartesian grid, J. Comput. Phys. 184 (1) (2003) 1–36.

[20] C.S. Peskin, Flow patterns around heart valves: a numerical method, J. Comput. Phys. 10 (1972) 252–271.
[21] J.A. Vieceli, A method for including arbitrary external boundaries in the MAC incompressible fluid computing technique, J. Comput. Phys. 4 (1969) 543–551.
[22] J.A. Vieceli, A computing method method for incompressible flows bounded by moving walls, J. Comput. Phys. 8 (1971) 119–143.
[23] C.S. Peskin, Numerical analysis of blood flow in the heart, J. Comput. Phys. 25 (1977) 220–252.
[24] C.S. Peskin, D.M. McQueen, Numerical analysis of blood flow in the heart, J. Comput. Phys. 25 (1980) 220–252.
[25] C.S. Peskin, D.M. McQueen, Cardiac fluid dynamics, Crit. Rev. Biomed. Eng. 20 (1992) 451–459.
[26] C.S. Peskin, D.M. McQueen, Shared-memory parallel vector implementation of the immersed boundary method for the computation of blood flow in the beating mammalian heart, J. Supercomput. 11 (1997) 213–236.
[27] D. Goldstein, R. Handler, L. Sirovich, Modeling a no-slip flow boundary with an external force field, J. Comput. Phys. 105 (1993) 354–366.
[28] D. Goldstein, R. Handler, L. Sirovich, Direct numerical simulation of turbulent flow over a modeled riblet covered surface, J. Fluid Mech. 302 (1995) 333.
[29] R. Cortez, M.L. Minion, The blob projection method for immersed boundary problems, J. Comput. Phys. 161 (2000) 428.
[30] R.J. LeVeque, Z. Li, The immersed interface method for elliptic equations with discontinuous coefficients and singular sources, SIAM J. Numer. Anal. 31 (1994) 1019.
[31] R.J. LeVeque, Z. Li, Immersed interface method for Stokes flow with elastic boundaries or surface tension, SIAM J. Sci. Comput. 18 (1997) 709.
[32] Zhilin Li, Ming-Chih Lai, The immersed interface method for the Navier–Stokes equations with singular forces, J. Comput. Phys. 171 (2001) 822–842.
[33] J. Mohd-Yusof, Combined immersed boundaries/b-splines methods for simulations of flows in complex geometries. Ctr annual research briefs, Stanford University, NASA Ames, 1997.
[34] 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.
[35] R. Verzicco, J. Mohd-Yusof, P. Orlandi, D. Haworth, Large-eddy simulation in complex geometric configurations using body forces, AIAA J. 38 (2000) 427.
[36] R. Verzicco, G. Iaccarino, M. Fatica, P. Orlandi, Flow in an impeller stirred tank using an immersed boundary method. Ctr annual research briefs, NASA Ames/Stanford University, 2000.
[37] J. Kim, D. Kim, H. Choi, An immersed-boundary finite-volume method for simulations of flow in complex geometries, J. Comput. Phys. 171 (2001) 132–150.
[38] A. Gilmanov, F. Sotiropoulos, E. Balaras, A general reconstruction algorithm for simulating flows with complex 3D immersed boundaries on Cartesian grids, J. Comput. Phys. 191 (2003) 660–669.
[39] E. Balaras, Modeling complex boundaries using an external force field on fixed Cartesian grids in large-eddy simulations, Comput. Fluids 33 (3) (2004) 375–404.
[40] S.C.R. Dennis, S.N. Singh, D.B. Ingham, The steady flow due to a rotating sphere at low and moderate Reynolds numbers, Phys. Fluids 101 (1980) 257–279.
[41] James M. Birch, Michael H. Dickinson, The influence of wing-wake interactions on the production of aerodynamic forces in flapping flight, J. Exp. Biol. 206 (2003) 2257–2272.
[42] P.M. Gresho, R.L. Sani, On pressure boundary conditions for the incompressible Navier–Stokes equations, Int. J. Numer. Methods Fluids 7 (1990) 11–46.
[43] B.P. Leonard, Boundary higher-order upwind multidimensional finite-volume convection–diffusion algorithms, Comput. Meth. Appl. Mech. Eng. 19 (1979) 59–98.
[44] J. Chorin, A.A numerical method for solving incompressible viscous flow problems, J. Comput. Phys. 2 (1) (1967) 12–26.
[45] Y.W. Soh, J.W. Goodrich, Unsteady solution of incompressible Navier–Stokes equations, J. Comput. Phys. 79 (1988) 113–134.
[46] F. Sotiropoulos, S. Abdallah, The discrete continuity equation in primitive variable solutions of incompressible-flow, J. Comput. Phys. 95 (1) (1991) 212–227.
[47] J. Ferziger, M. Peric, Computational Methods for Fluid Dynamics, Springer, Berlin, 1999.
[48] S.W. Armfield, Ellipticity, accuracy, and convergence of the discrete Navier–Stokes equations, J. Comput. Phys. 114 (2) (1994) 176–184.
[49] F. Sotiropoulos, S. Abdallah, A primitive variable method for the solution of external, 3-D incompressible, viscous flows, J. Comput. Phys. 103 (2) (1992) 336–349.
[50] T.A. Johnson, V.C. Patel, Flow past a sphere up to a Reynolds number of 300, J. Fluid Mech. 378 (10) (1999).
[51] R. Peyret, Th.D. Taylor, Computational Methods for Fluid Flow, Springer, Berlin, 1986.
[52] F. Sotiropoulos, Y. Ventikos, Transition from bubble-type vortex breakdown to columnar vortex in a confined swirling flow, Int. J. Heat Fluid Fl 19 (5) (1998) 446–458.
[53] F. Sotiropoulos, G. Constantinescu, Pressure-based residual smoothing operators for multistage pseudo-compressibility algorithms, J. Comput. Phys. 133 (1) (1997) 129–145.

[54] K.W. Thompson, Time-dependent boundary conditions for hyperbolic systems. II, J. Comput. Phys. 89 (2) (1990) 439–461.
[55] J. Van Kan, A second-order accurate pressure-correction scheme for viscous incompressible flow, SIAM J. Sci. Stat. Comput. 7 (1986) 870–891.
[56] J.B. Bell, P. Colella, H.M. Glaz, A second-order projection method for the incompressible Navier–Stokes equations, J. Comput. Phys. 85 (1989) 257–283.
[57] F.P. Bowden, R.G. Lord, The aerodynamic resistance to a sphere rotating at high speed, Proc. R. Soc. Lond. Ser. 271 (143) (1963).
[58] Michael H. Dickinson, F.O. Lehman, S. Sane, Wing rotation and the aerodynamic basis of insect flight, Science 284 (1999) 1954–1960.
[59] S.P. Sane, Michael H. Dickinson, The control flight force by a flapping wing: lift and drag production, J. Exp. Biol. 204 (2001) 2607–2626.
[60] Michael H. Dickinson, Private communication, 2004.
[61] M. Sfakiotakis, D.M. Lane, D.J. Bruce, Review of fish swimming modes for aquatic locomotion, IEEE J. Oceanic Eng. 24 (2) (1999) 237–252.
[62] Q. Zhu, M.J. Wolfgang, D.K.P. Yue, M.S. Triantafyllou, Three-dimensional flow structures and vorticity control in fish-like swimming, J. Fluid Mech. 486 (1) (2002) 1–28.
[63] H. Hertel, Structure, Form and Movement (Biology and Technology), Reinhold, New York, 1966.
[64] Ulrike K. Müller, Eize J. Stamhuis, John J. Videler, Riding the waves: the role of the body wave in undulatory fish swimming, J. Integr. Comput. Biol. 42 (2002) 981–987.
[65] U.K. Müller, B.L.E. van den Heuvel, E.J. Stamhuis, Fish foot prints: morphology and energetics of the wake behind a continuously swimming mullet (chelon labrosus risso), J. Exp. Biol. 200 (22) (1997) 2893–2906.
[66] M.S. Triantafyllou, G.S. Triantafyllou, D.K.P. Yue, Hydrodynamics of fishlike swimming, Annu. Rev. Fluid Mech. 32 (2000) 33.
[67] Edwin Malkiel, Jian Sheng, Joseph Katz, Rudi Strickler, The three-dimensional field generated by a feeding calanoil copepod meashured using digital holography, J. Exp. Biol. 206 (2003) 3657–3666.
[68] David M. Fields, Jeanette Yen, Fluid mechanosensory stimulation of behaviour from a planktonic marine copepod, euchaeta rimana bradford, J. Plankton Res. 24 (8) (2002) 747–755.
[69] Houshuo Jiang, Charles Meneveau, Thomas R. Osborn, Numerical study of the feeding current around a copepod, J. Plankton Res. 21 (8) (1999) 1391–1421.
[70] Houshuo Jiang, Thomas R. Osborn, Charles Meneveau, Hydrodynamic interaction between two copepods: a numerical study, J. Plankton Res. 24 (3) (2002) 235–253.
[71] Jeanette Yen, Private communication, 2003.
[72] A.M. Roma, C.S. Peskin, M.J. Berger, An adaptive version of the immersed boundary method, J. Comput. Phys. 153 (2) (1999).
[73] A.N. Gilmanov, Application of dynamically adaptive grids to the analysis of flows with a multiscale structure, Comput. Math. Math. Phys. 41 (2) (2001) 289–303.